

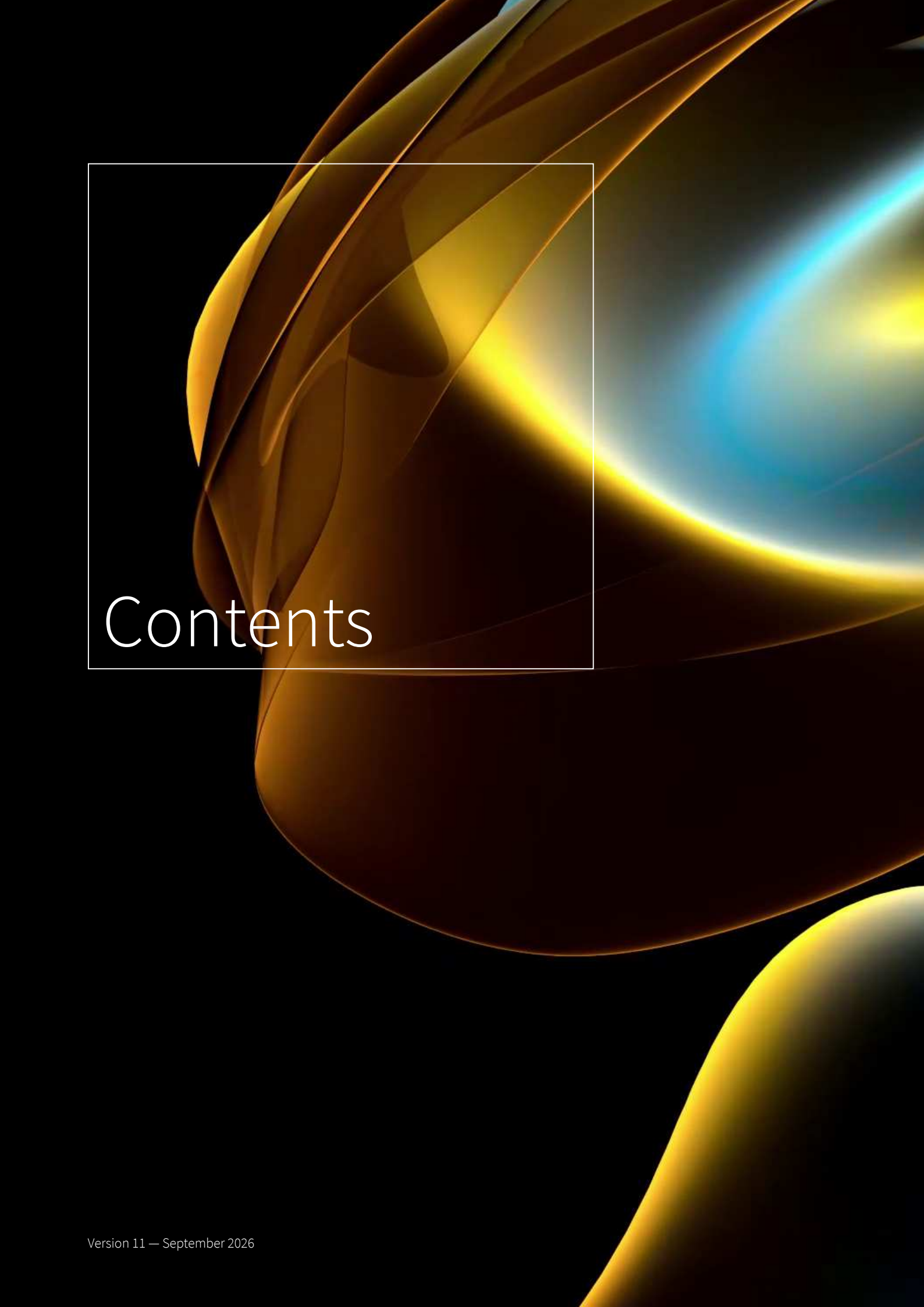
```
// Create and name new table object
var oTbl = doc.objects().create(ObjectType.Table);
oTbl.name = "Table 1";
// Set number of rows and columns of a VIVA table
oTbl.rowCount = 5;
oTbl.columnCount = 3;
// Set position and size of the table of a VIVA table
oTbl.tableWidthType = WidthType.Manual;
oTbl.pagemex = "10mm";
oTbl.pagey = "10mm";
oTbl.width = "120mm";
oTbl.height = "150mm";
// Colorize interior of the VIVA table
oTbl.brush = Color.Green;
oTbl.shade = 20;

// Add the table to a current VIVA document page
doc.currentPage.add(oTbl);

// Get a table cell and change its color
var tblCell = oTbl.cell(1, 1);
if( tblCell ) {
    tblCell.brush = Color.Red;
    tblCell.shade = 70;
}

// Retrieve content type of the cell
if (tblCell.contentType == ContentType.Custom)
    |   messageBox.information("Content Type", "Custom");
if (tblCell.contentType == ContentType.Text)
    |   messageBox.information("Content Type", "Text");
if (tblCell.contentType == ContentType.Picture)
    |   messageBox.information("Content Type", "Picture");
```



The background of the page is an abstract composition of flowing, organic shapes in shades of gold, yellow, and blue against a dark background. A white rectangular box is positioned on the left side, containing the word 'Contents'.

# Contents

# Introduction

## Scripting Features

|                                |   |
|--------------------------------|---|
| Variable declaration.....      | 4 |
| Object creation.....           | 4 |
| Special Variable Types.....    | 4 |
| UnitValueString.....           | 4 |
| LineStyleString.....           | 5 |
| FontName.....                  | 5 |
| ThreeStates.....               | 5 |
| Exception Processing.....      | 5 |
| Error Status.....              | 6 |
| Functions and Recursion.....   | 6 |
| Public Commands.....           | 7 |
| Command: include.....          | 7 |
| Command: quit.....             | 7 |
| Command: pause.....            | 7 |
| Command: repaint.....          | 7 |
| Command: execute.....          | 8 |
| Command: convertUnitValue..... | 8 |
| Command: dump.....             | 8 |

## Application, Document Structure Objects

|                                               |    |
|-----------------------------------------------|----|
| Main object “application”.....                | 12 |
| Class: Document.....                          | 13 |
| Class: Page.....                              | 19 |
| Class: DocumentPage.....                      | 21 |
| Class: AliasPage.....                         | 22 |
| Alias pages and their objects in scripts..... | 23 |
| Class: Spread.....                            | 24 |
| Class: AliasSpread.....                       | 26 |
| Class: Layer.....                             | 27 |

## Object Types and Structure

|                                |    |
|--------------------------------|----|
| Class: Objects.....            | 30 |
| Class: Object.....             | 32 |
| Class: GraphicObject.....      | 36 |
| Class: LineObject.....         | 37 |
| Class: FrameObject.....        | 40 |
| Class: TextObject.....         | 42 |
| Class: TextContent.....        | 44 |
| Class: PictureObject.....      | 47 |
| Class: PictureContent.....     | 48 |
| Class: TextChains.....         | 51 |
| Class: TextChain.....          | 51 |
| Class: TableObject.....        | 52 |
| Class: TableRow.....           | 54 |
| Class: TableColumn.....        | 55 |
| Class: TableCell.....          | 57 |
| Class: TableSeparator.....     | 60 |
| Class: GroupObject.....        | 62 |
| Class: ObjectAlias.....        | 64 |
| Class: TextObjectAlias.....    | 64 |
| Class: PictureObjectAlias..... | 64 |

## FormattedText Objects

|                           |     |
|---------------------------|-----|
| Class: FormattedText..... | 70  |
| Class: TextRanges.....    | 101 |
| Class: TextRange.....     | 101 |
| Class: TextCursor.....    | 102 |

## Object shapes

|                         |     |
|-------------------------|-----|
| Class: Point.....       | 106 |
| Class: PointList.....   | 107 |
| Class: BoundingBox..... | 109 |

|                    |     |
|--------------------|-----|
| Class: Matrix..... | 109 |
|--------------------|-----|

## Brushes and Colors

|                        |     |
|------------------------|-----|
| Introduction.....      | 114 |
| Class: Brushes.....    | 114 |
| Class: Brush.....      | 114 |
| Class: ColorBrush..... | 115 |
| Class: BlendBrush..... | 116 |
| Class: Stop.....       | 117 |
| Class: Rgb.....        | 117 |
| Class: Cmyk.....       | 118 |
| Class: Hsv.....        | 118 |
| Class: Lab.....        | 118 |
| Examples.....          | 119 |

## StyleSheets

|                                   |     |
|-----------------------------------|-----|
| Class: StyleSheets.....           | 124 |
| Class: StyleSheet.....            | 125 |
| Class: CharacterStyleSheet.....   | 126 |
| Class: ParagraphStyleSheet.....   | 127 |
| Class: LayoutStyleSheet.....      | 130 |
| Class: PictureStyleSheet.....     | 130 |
| Class: GraphicStyleSheet.....     | 131 |
| Class: TableStyleSheet.....       | 133 |
| Class: RowColumnStyle.....        | 134 |
| Class: RowColumnStyleAlt.....     | 134 |
| Class: TableRowStyleSheet.....    | 135 |
| Class: TableColumnStyleSheet..... | 135 |
| Class: TableCellStyleSheet.....   | 136 |

## Document and Page Settings

|                              |     |
|------------------------------|-----|
| Class: DocumentSettings..... | 140 |
|------------------------------|-----|

|                                |     |
|--------------------------------|-----|
| Class: PageSettings.....       | 141 |
| Class: ChapterInfo.....        | 142 |
| Class: DocumentStatistics..... | 142 |
| Class: SearchPaths.....        | 143 |

## Preferences

|                                      |     |
|--------------------------------------|-----|
| Class: EmbeddedFontsPreferences..... | 148 |
| Class: LanguageInfo.....             | 148 |
| Class: LanguagePreferences.....      | 148 |
| Class: LayoutPreferences.....        | 149 |
| Class: ObjectPreferences.....        | 149 |
| Class: PagePreferences.....          | 150 |
| Class: PreflightPreferences.....     | 150 |
| Class: TextPreferences.....          | 151 |

## Helped Data Objects

|                                       |     |
|---------------------------------------|-----|
| Class: Annotation.....                | 158 |
| Class: Attributes.....                | 159 |
| Class: AutomaticCharacterSpacing..... | 159 |
| Class: AutomaticCharacterWidth.....   | 159 |
| Class: AutomaticWordSpacing.....      | 160 |
| Class: BaselineGrid.....              | 160 |
| Class: BaselinePosition.....          | 160 |
| Class: ColorOverlay.....              | 161 |
| Class: ColumnLines.....               | 161 |
| Class: Comment.....                   | 162 |
| Class: DataTable.....                 | 162 |
| Class: DropCaps.....                  | 163 |
| Class: Enumerations.....              | 164 |
| Class: FontSize.....                  | 166 |
| Class: GraphicFootnotes.....          | 166 |
| Class: Guide.....                     | 167 |

|                                             |     |
|---------------------------------------------|-----|
| Class: HeaderFooter.....                    | 167 |
| Class: Hyphenation.....                     | 167 |
| Class: ChangePictureColorSpaceSettings..... | 168 |
| Class: CharacterBackground.....             | 168 |
| Class: CharacterFrame.....                  | 169 |
| Class: CharacterKerning.....                | 170 |
| Class: CharacterPositionCorrection.....     | 171 |
| Class: CharacterRule.....                   | 171 |
| Class: Image.....                           | 172 |
| Class: InnerGlow.....                       | 172 |
| Class: LayoutColumnsInfo.....               | 173 |
| Class: LayoutFootnotes.....                 | 174 |
| Class: LineStyle.....                       | 174 |
| Class: LineNumbers.....                     | 175 |
| Class: Link.....                            | 175 |
| Class: Note.....                            | 176 |
| Class: OpenType.....                        | 176 |
| Class: OptimizePictureSettings.....         | 178 |
| Class: OuterGlow.....                       | 178 |
| Class: OutlineOptions.....                  | 179 |
| Class: PackageSettings.....                 | 179 |
| Class: ParagraphBackground.....             | 180 |
| Class: ParagraphFrame.....                  | 181 |
| Class: ParagraphRule.....                   | 182 |
| Class: PDFExportSettings.....               | 183 |
| Class: PreviewInstructions.....             | 184 |
| Class: Reflection.....                      | 185 |
| Class: Runaround.....                       | 185 |
| Class: Shadow.....                          | 186 |
| Class: SpellLang.....                       | 187 |
| Class: StrikethroughOptions.....            | 187 |
| Class: TablesAbbreviation.....              | 188 |

|                                |     |
|--------------------------------|-----|
| Class: TablesBibliography..... | 188 |
| Class: TablesContents.....     | 188 |
| Class: TablesIndex.....        | 189 |
| Class: TablesPicture.....      | 189 |
| Class: TablesRunningTitle..... | 189 |
| Class: Tabulator.....          | 190 |
| Class: Tabulators.....         | 190 |
| Class: TaggedPdfOptions.....   | 191 |
| Class: TextEndnotes.....       | 191 |
| Class: TextFootnotes.....      | 192 |
| Class: TransparencyStop.....   | 192 |
| Class: Transparency.....       | 192 |
| Class: UnderlineOptions.....   | 193 |
| Class: Variable.....           | 194 |
| Class: WidowsOrphans.....      | 194 |

## User Interface Objects

|                              |     |
|------------------------------|-----|
| Class: MessageBox.....       | 198 |
| Class: Directory.....        | 198 |
| Class: File.....             | 199 |
| Class: TextFile.....         | 200 |
| Class: FileDialog.....       | 203 |
| Class: ProgressReporter..... | 204 |
| Class: Dialog.....           | 204 |
| Class: Label.....            | 206 |
| Class: CheckBox.....         | 207 |
| Class: ComboBox.....         | 207 |
| Class: GroupBox.....         | 209 |
| Class: RadioButton.....      | 209 |
| Class: LineEdit.....         | 210 |
| Class: TextEdit.....         | 211 |
| Class: Button.....           | 211 |

# Enumeration Objects Index

## Index



# Introduction

# Introduction

JScripter is a VivaDesigner tool that allows you to manipulate documents using a simplified version of the JavaScript programming language. This document describes JScripter's available functions and how to use them.

## Examples

The document contains many examples that demonstrate the principles of the scripting language. Do not assume that all of the examples are complete and ready to run. They are intended to illustrate the feature mentioned in the text. To experiment with the examples in the real script, you may need to add code that is important for running the code fragment but not for explaining the appropriate feature.

## Legend

The documentation uses various fonts to differentiate the text's various meanings. The following legend defines the text's meaning and the font used for it:

| Font                  | Meaning of the text                              |
|-----------------------|--------------------------------------------------|
| Normal font           | General description                              |
| <code>Courier</code>  | Script examples                                  |
| <b>Courier Bold</b>   | Script commands                                  |
| <i>Courier Italic</i> | Names of objects and classes, Jscripter keywords |



# Scripting Features

JScripter manipulates objects of various classes. JScripter uses a common syntax that is well known from other scripting languages.

## Variable declaration

The declaration of a variable is realized by the keyword `var` followed by the variable name. The type of variable is specified by the type of value that is assigned to the variable.

```
var variableName1 = 10; // numeric variable
var variableName2 = new Rgb(0,0,0); // object variable
var variableName3 = [1,2,3,4,5]; // array of numbers
var variableName4 = [new Rgb(0,0,0), new Rgb(25,255,255)]; // array of objects
```

## Object creation

An object can be created by calling a constructor:

```
var myRgb = new Rgb(255, 100, 25);
```

or via the object provider (i.e. a function which creates an object inside and returns it as a result of the function call):

```
var myObjGroup = doc.objects().create(Object.Group);
```

The parameter of the function `create()` is the name of the class of the created object.

## Special Variable Types

### UnitValueString

The type *UnitValueString* defines the distance (length, width, space, ...) with a unit. If the *UnitValueString* value is returned without a unit, the "pt" unit is presumed. The following units are supported:

Base: "dtp"

Metric: "mm", "cm", "m", "in", "dz" (decimal inch), "q" (quart)

Didot system: "dd", "c"

Pica system: "pt", "p"

The conversion function `convertUnitValue` is used for the conversion of a value to another unit.

#### Example 1:

```
"12mm", "10pt", "2in", ...
```

#### Example 2:

```
textObject.pagex = textObject.pagex + "+10mm";
```

```
textObject.pagey = "(5mm * 2) + 22pt + 1cm";
```

```
var res = convertUnitValue("125", "m", true, "pt");
```

## LineStyleString

The *LineStyleString* is a string type from a defined list of values that are created from substrings: *Solid, Dash, Dot, Line: SolidLine, DashLine, DotLine, DashDotLine, DashDotDotLine*

Or the options: '*Pattern 0*' to '*Pattern 17*'.

## FontName

The *FontName* is composed of the family name (i.e. "Arial") and font face (i.e. "Italic", "Bold") separated by a hyphen.

### Examples:

"Arial-Regular", "Times New Roman-Bold Italic"

## ThreeStates

The type *ThreeStates* takes three logical values. They are available as the global constants on (1), off (0) and undef (-1):

*on* - it is set

*off* - it is unset

*undef* - it is not defined

## Exception Processing

Some methods generate exceptions when an error occurred. The error handling code has the following structure:

```
try {
    block of code to try
}
catch(err) {
    block of code to handle errors
}
finally {
    block of code to be executed regardless of the try / catch result
}
```

The **try** statement lets you test a block of code for errors.

The **catch** statement lets you handle the error.

The **finally** statement lets you execute code, after try and catch, regardless of the result.

### Examples:

```
try {
    var cb = doc.brushes.addColor("[Green]");
    // Color named [Green] exists than an exception occurs!
} catch (err) {
    messageBox.information("Error:", "Error Desc: " + err.name + ', ' +
err.message);
} finally {
```

```
        messageBox.warning("Warning:", "Finally after exception!");  
    }
```

## Error Status

Each object has a property ***errorStatus*** which is set after calling each method or property. The property ***errorStatus*** returns an error that occurred when the method or the property was processed. The method or the property returns default value in case of the error.

The property ***errorStatus*** returns the following values of the enumeration **ErrorStatus**:

***Ok*** – calling the property or the method has been successfully done without the error

***IsNotProcessed*** – calling the property or the method has not been done

***IsNotValid*** – the result value is not valid. The default value has been returned.

***IsNotSet*** – the property is not set yet. The value is returned when the attribute is not set at all. For example, in the case of creating new stylesheet profile.

***IsNotUsable*** – the property or the method is not usable in the context.

***IsNotAccessToFileSystem*** – the property or the method is not accessible because the file system is protected.

### Example:

```
var brushes = application.currentDocument.brushes;  
var cb = brushes.find("MyColor");  
if (brushes.errorStatus != ErrorStatus.Ok)  
    messageBox.warning("Error", "Status: " + brushes.errorStatus);
```

## Functions and Recursion

Functions can be used for writing more complicated scripts. The function can be called with input parameters and can return a result. The functions can be saved in separate files and can be included in the script using the 'include' command. The functions can be called recursively.

### Examples:

```
function function1(parameter1, parameter2, ...)  
{  
    ... function body  
  
    if( ... condition)  
        // recursively calling  
        function1(par1, par2, ...);  
    else  
        // function returns result  
        return result;  
}  
  
function function2(text)  
{  
    ... function body  
    function1( par1, par2, ...);  
}
```

```
}
```

## Public Commands

### Command: include

The command allows the inclusion of the content of another script file. The content is processed and all functions from this file are accessed after processing. The include command must be inserted in the file before the script code and be processed before the script is running. The scripts are searched for in two directories: application and user scripts directories.

#### Using:

---

```
include ( String fileName )
```

where:

*fileName* is the path of the script file relative to the application or user scripts directory, e.g. "js\_library.js" or "Library/js\_library.js" for a file in a subdirectory. Upper and lower case are not distinguished.

Each file is included only once, even if several scripts include it. A file that is not found is skipped without an error message.

### Command: quit

Quits the running of the script prematurely.

#### Using:

---

```
quit ( ) ;
```

### Command: pause

Suspends the running of the script for a time in milliseconds. The document window is repainted before the pause.

#### Using:

---

```
pause ( long time ) ;
```

where:

*time* is the number of milliseconds to wait

### Command: repaint

Repaints the whole window with the document.

#### Using:

```
repaint ( ) ;
```

## Command: execute

The command runs an external program. The first parameter is a program path, the following parameters (at most five) are passed to the program. The script does not wait for the program to finish. The command is only available when VivaDesigner runs with a user interface.

Using:

---

```
execute (String programPath, String arg1, String arg2, String arg3, String arg4, String arg5)
```

## Command: convertUnitValue

Converts the value of the *UnitValueString* type from one unit to another.

Using:

---

```
convertUnitValue(String value, String toUnit, bool withUnit=false, String fromUnit = "pt")
```

where:

**value** is the value that is converted. If the value contains a unit, this unit is used and fromUnit is ignored.

**toUnit** – result unit name ("mm", "in", "pt", ...)

**fromUnit** – source unit name ("mm", "in", "pt", ...). Optional, the default value is "pt"

**withUnit** – flag, true = the unit will be added to the result after value. Optional, the default value is false.

The function returns an empty string if the value or one of the units is not valid.

## Command: dump

Returns the content of a value as text. Arrays and objects are returned in JSON notation; other values are converted to a string. The command is useful for inspecting values in a message box.

Using:

---

```
dump(value)
```

### Global objects: application, messageBox

The object **application** (class Application) gives access to the application and its documents. The object **messageBox** (class MessageBox) shows message boxes; it is only available when VivaDesigner runs with a user interface.

---

|                    |                    |             |
|--------------------|--------------------|-------------|
| <b>application</b> | <i>Application</i> | <i>Read</i> |
|--------------------|--------------------|-------------|

---

|                   |                   |             |
|-------------------|-------------------|-------------|
| <b>messageBox</b> | <i>MessageBox</i> | <i>Read</i> |
|-------------------|-------------------|-------------|

### Properties: appScriptsPath, userScriptsPath

Return the full paths to the application scripts or to the user's scripts. Both paths are searched automatically for scripts.

---

|                       |               |             |
|-----------------------|---------------|-------------|
| <b>appScriptsPath</b> | <i>String</i> | <i>Read</i> |
|-----------------------|---------------|-------------|

|                        |               |             |
|------------------------|---------------|-------------|
| <b>userScriptsPath</b> | <i>String</i> | <i>Read</i> |
|------------------------|---------------|-------------|

## Examples

content of the file js\_library.js:

```
include("js_library_2.js");

function myFunction(a, b)
{
    return (a * b) - 10;
}
```

content of base script file:

```
include("js_library.js");
include("js_library_1.js");
pause(2000);

messageBox.information("Result", myFunction(11, 22));
var ds = application.currentDocument.documentSettings;
messageBox.information("Msg:",
convertUnitValue(ds.columnGutter, "m", true, "pt"));

repaint();
quit();

messageBox.information("Both Scripts Paths",
appScriptsPath + "\n" + userScriptsPath);
```





# Application, Document Structure Objects

# Application, Document Structure Objects

This chapter describes the hierarchy of object classes in relation to the application and documents, spreads, layers and pages in the application to understand the structure of object relationships, and shows examples of how to use them.

## Main object “application”

VIVA Designer has just one built-in root object application. This object always exists. This object must be used for contained documents, pages, spreads etc.

### Properties:

|                                                                                                                                                                                                                                                                                                                                                                                                                                             |                 |             |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|-------------|
| <b>identifier</b>                                                                                                                                                                                                                                                                                                                                                                                                                           | <i>String</i>   | <i>Read</i> |
| Returns the identifier of the application, e.g. “Viva Technology/Viva Designer”.                                                                                                                                                                                                                                                                                                                                                            |                 |             |
| <b>filePath</b>                                                                                                                                                                                                                                                                                                                                                                                                                             | <i>String</i>   | <i>Read</i> |
| Returns full path to the VivaDesigner.exe file.                                                                                                                                                                                                                                                                                                                                                                                             |                 |             |
| <b>absolutePath</b>                                                                                                                                                                                                                                                                                                                                                                                                                         | <i>String</i>   | <i>Read</i> |
| Returns the full path of the folder that contains the application file.                                                                                                                                                                                                                                                                                                                                                                     |                 |             |
| <b>documentCount</b>                                                                                                                                                                                                                                                                                                                                                                                                                        | <i>int</i>      | <i>Read</i> |
| Returns number of opened documents in the application                                                                                                                                                                                                                                                                                                                                                                                       |                 |             |
| <b>currentDocument</b>                                                                                                                                                                                                                                                                                                                                                                                                                      | <i>Document</i> | <i>Read</i> |
| Returns current Document object or <i>null</i> if no object exists.                                                                                                                                                                                                                                                                                                                                                                         |                 |             |
| <b>dictionariesPath</b>                                                                                                                                                                                                                                                                                                                                                                                                                     | <i>String</i>   | <i>Read</i> |
| Returns the path to the folder with the dictionaries of the application.                                                                                                                                                                                                                                                                                                                                                                    |                 |             |
| <b>language</b>                                                                                                                                                                                                                                                                                                                                                                                                                             | <i>String</i>   | <i>Read</i> |
| Returns the language of the program interface.                                                                                                                                                                                                                                                                                                                                                                                              |                 |             |
| <b>documentOpenFormats</b>                                                                                                                                                                                                                                                                                                                                                                                                                  | <i>Array</i>    | <i>Read</i> |
| <b>documentImportFormats</b>                                                                                                                                                                                                                                                                                                                                                                                                                | <i>Array</i>    | <i>Read</i> |
| <b>documentSaveFormats</b>                                                                                                                                                                                                                                                                                                                                                                                                                  | <i>Array</i>    | <i>Read</i> |
| <b>documentExportFormats</b>                                                                                                                                                                                                                                                                                                                                                                                                                | <i>Array</i>    | <i>Read</i> |
| Return the document formats registered in the application: formats that can be opened, imported, saved and exported. Each entry is an object with the properties <i>name</i> (name of the format) and <i>extensions</i> (array of file extensions without “.”, e.g. [ <i>"idml"</i> ]). Opening a file of an open or import format creates a document; save and export formats can be written with <i>save()</i> and <i>exportToXML()</i> . |                 |             |

### Methods:

|                                                                                                |                 |
|------------------------------------------------------------------------------------------------|-----------------|
| <b>documentAt</b> ( <i>int index</i> )                                                         | <i>Document</i> |
| Returns the document specified by its index or <i>null</i> if the document doesn’t exist.      |                 |
| <i>index</i> : index of required document. The value must be in the range 0–(documentCount–1). |                 |

---

**newDocument** (*DocumentSettings settings, bool show*) *Document*

Creates and returns a new Document object. Returns *null* if a problem occurs when creating.

*settings*: Parameters of the new document.

*show*: Use true or false to show or hide the new document.

---

**newDocument** () *Document*

Creates and returns a new document using default document parameters The new document is set as a current document. Returns *null* if problem occurs.

---

**openDocument** (*String filePath, bool show, Object parameters = {}*) *Document*

Opens a document specified by its file path. Returns the opened Document object or *null* if document cannot be opened. An exception is caused when the document is open already.

*filePath*: Path to the opened document.

*show*: Use true or false to show or hide the new document.

*parameters*: Optional import parameters for import filters that support them (e.g. IDML). If parameters are given, the settings dialog of the import filter is not shown.

---

**setQrCodeToImage** (*String url, String qrPictureId*) *bool*

Creates a QR code for the text *url* and places it into the picture object with the ID *qrPictureId* (see the property *qrId* of the class PictureContent). Returns false if one of the parameters is empty.

---

#### Examples:

```
var docCount = application.documentCount;
var docCurr = application.currentDocument;
var docFirst = application.documentAt (0);
```

## Class: Document

Objects of the class *Document* are only objects which can be owned by the object *application*. One document object represents the whole document with its pages, spreads, texts, pictures, graphic objects etc., which is stored in one file.

#### Properties:

| <b>pageMode</b>                                                                                                                                                                                 | <i>PageMode</i> | <i>Read/Write</i> |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|-------------------|
| Contains information about the the page mode in the document.                                                                                                                                   |                 |                   |
| <b>documentPageCount</b>                                                                                                                                                                        | <i>int</i>      | <i>Read</i>       |
| Contains the number of document pages. See the chapter “Class: DocumentPage” for more info about document pages. Returns 0 if an error occurs.                                                  |                 |                   |
| <b>aliasPageCount</b>                                                                                                                                                                           | <i>int</i>      | <i>Read</i>       |
| Contains the number of Alias pages. See the chapter “Class: AliasPage” for more info about Alias pages. Returns 0 if an error occurs. A facing Alias page (PageType.FacingPages) counts as one. |                 |                   |
| <b>documentSpreadCount</b>                                                                                                                                                                      | <i>int</i>      | <i>Read</i>       |
| Contains the number of document spreads. See the chapter “Class: Spread” for more info about document spreads. Returns 0 if an error occurs.                                                    |                 |                   |

---

|                                                                                                                                                                                                                                                                                               |                                 |                   |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------|-------------------|
| <b>aliasSpreadCount</b>                                                                                                                                                                                                                                                                       | <i>int</i>                      | <i>Read</i>       |
| Contains the number of Alias facing pages/spreads. Returns 0 if an error occurs.                                                                                                                                                                                                              |                                 |                   |
| <b>layerCount</b>                                                                                                                                                                                                                                                                             | <i>int</i>                      | <i>Read</i>       |
| Contains the number of layers. See the chapter “Class: Layer” for more info about document layers. Returns 0 if an error occurs.                                                                                                                                                              |                                 |                   |
| <b>brushes</b>                                                                                                                                                                                                                                                                                | <i>Brushes</i>                  | <i>Read</i>       |
| Keeps an object of the class Brushes, which is a repository of all available brushes. New brushes can be added, unnecessary ones can be removed. See chapter “Class: Brushes” for more info about brushes.                                                                                    |                                 |                   |
| <b>styleSheets</b>                                                                                                                                                                                                                                                                            | <i>StyleSheets</i>              | <i>Read</i>       |
| Keeps an object of the class StyleSheets, which is a repository of all available style sheets. New style sheets can be added, unnecessary ones can be removed. See chapter “Class: StyleSheets” for more info about style sheets.                                                             |                                 |                   |
| <b>textCursor</b>                                                                                                                                                                                                                                                                             | <i>TextCursor</i>               | <i>Read</i>       |
| Keeps an object of the class TextCursor. This object contains information about the current position of the text cursor and/or the current selection. It also gives opportunity to set a new cursor position. See the chapter “Class: TextCursor” for more information about the text cursor. |                                 |                   |
| <b>textPreferences</b>                                                                                                                                                                                                                                                                        | <i>TextPreferences</i>          | <i>Read/Write</i> |
| Keeps an object of the class TextPreferences. See the chapter “Class: TextPreferences” for more information about text preferences.                                                                                                                                                           |                                 |                   |
| <b>preflightPreferences</b>                                                                                                                                                                                                                                                                   | <i>PreflightPreferences</i>     | <i>Read/Write</i> |
| Keeps on object of the class PreflightPreferences. See the chapter “Class: PreflightPreferences” for more information about preflight preferences.                                                                                                                                            |                                 |                   |
| <b>languagePreferences</b>                                                                                                                                                                                                                                                                    | <i>LanguagePreferences</i>      | <i>Read/Write</i> |
| Keeps on object of the class LanguagePreferences. See the chapter “Class: LanguagePreferences” for more information about language preferences.                                                                                                                                               |                                 |                   |
| <b>objectPreferences</b>                                                                                                                                                                                                                                                                      | <i>ObjectPreferences</i>        | <i>Read/Write</i> |
| <b>layoutPreferences</b>                                                                                                                                                                                                                                                                      | <i>LayoutPreferences</i>        | <i>Read/Write</i> |
| <b>pagePreferences</b>                                                                                                                                                                                                                                                                        | <i>PagePreferences</i>          | <i>Read/Write</i> |
| <b>embeddedFontsPreferences</b>                                                                                                                                                                                                                                                               | <i>EmbeddedFontsPreferences</i> | <i>Read/Write</i> |
| <b>documentStatistics</b>                                                                                                                                                                                                                                                                     | <i>DocumentStatistics</i>       | <i>Read/Write</i> |
| Keeps an object of the class DocumentStatistics. See the chapter “Class: DocumentStatistics” for more information.                                                                                                                                                                            |                                 |                   |
| <b>searchPaths</b>                                                                                                                                                                                                                                                                            | <i>SearchPaths</i>              | <i>Read/Write</i> |
| Keeps an object of the class SearchPaths. See the chapter “Class: SearchPaths” for more information about search paths.                                                                                                                                                                       |                                 |                   |
| <b>filePath</b>                                                                                                                                                                                                                                                                               | <i>String</i>                   | <i>Read</i>       |
| Contains the full file path including the file name of the document.                                                                                                                                                                                                                          |                                 |                   |
| <b>currentPage</b>                                                                                                                                                                                                                                                                            | <i>DocumentPage</i>             | <i>Read</i>       |

Returns a DocumentPage object, which represents the page with the current cursor.

|                                                                                                            |                         |                   |
|------------------------------------------------------------------------------------------------------------|-------------------------|-------------------|
| <b>currentSpread</b>                                                                                       | <i>Spread</i>           | <i>Read</i>       |
| Returns the current spread. See the chapter “Class: Spread” for more information about document spreads.   |                         |                   |
| <b>currentLayer</b>                                                                                        | <i>Layer</i>            | <i>Read/Write</i> |
| Returns the current document layer. See the chapter “Class: Layer” for more information about layers.      |                         |                   |
| <b>currentText</b>                                                                                         | <i>FormattedText</i>    | <i>Read</i>       |
| Returns a FormattedText object which represents the text where the cursor is currently placed.             |                         |                   |
| <b>currentPictureObject</b>                                                                                | <i>PictureObject</i>    | <i>Read</i>       |
| Returns the picture object that contains the current picture selection, or null if no picture is selected. |                         |                   |
| <b>pageWidth</b>                                                                                           | <i>float</i>            | <i>Read</i>       |
| <b>pageHeight</b>                                                                                          | <i>float</i>            | <i>Read</i>       |
| Width and height of the document pages in points (pt).                                                     |                         |                   |
| <b>documentSettings</b>                                                                                    | <i>DocumentSettings</i> | <i>Read/Write</i> |
| <b>toggleColumn</b>                                                                                        | <i>int</i>              | <i>Read/Write</i> |

#### Methods:

|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |                     |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------|
| <b>objects</b> ( <i>WhichSpreads spreads = Document, WhichObjects objects = All, ObjectType objectType = All</i> )                                                                                                                                                                                                                                                                                                                                                                                                                                                          | <i>Objects</i>      |
| Returns an object of the class Objects. This object is a repository and an object provider of all document objects of the classes TextObject, PictureObject, TableObject, GraphicObject, LineObject, GroupObject. See examples of how to use the object provider in your JScript program.<br><br><i>spread</i> : filter for searching for objects, the default is WhichSpreads:Document.<br><i>objects</i> : filter for searching for objects, the default is WhichObjects:All.<br><i>objectType</i> : filter for searching for object type. See the enumeration ObjectType |                     |
| <b>objectCount</b> ( <i>WhichSpreads spreads = Document, WhichObjects objects = TopLevel</i> )                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | <i>int</i>          |
| Returns the number of objects that match the filters <i>spreads</i> and <i>objects</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |                     |
| <b>getObject</b> ( <i>int index, WhichSpreads spreads = Document, WhichObjects objects = TopLevel</i> )                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | <i>Object</i>       |
| Returns the object at the position <i>index</i> in the list of objects that match the filters, or null if the index is out of range.                                                                                                                                                                                                                                                                                                                                                                                                                                        |                     |
| <b>textChains</b> ( <i>WhichSpreads whichSpreads = Document, WhichObjects whichObjects = All</i> )                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | <i>TextChains</i>   |
| Returns the text chains (linked text objects) of the document. See the chapter “Class: TextChains”.                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                     |
| <b>documentPageAt</b> ( <i>int index</i> )                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | <i>DocumentPage</i> |
| Returns the DocumentPage object at the position which is specified by the parameter index. Returns the required DocumentPage or <i>null</i> if the page doesn't exist.<br><br><i>index</i> : Index of the required document page. The value must be in range 0 – (pageCount – 1).                                                                                                                                                                                                                                                                                           |                     |

---

**addDocumentPage**(*int count*, *PageSettings settings* = *null*,  
*AliasPage assignToAliasPage* = *null*) *DocumentPage*

The method adds new pages to the document. Returns the first created *DocumentPage* object or *null* if no page is created.

*count*: Number of added pages. The default value is 1.

*settings*: page settings. See *PageSetting* object.

*assignToAliasPage*: assigned *AliasPage*. See *AliasPage* description. If several pages are added, only the first new page gets this *AliasPage*.

---

**aliasPageAt** (*int index*) *AliasPage*

Returns an *AliasPage* object at the position which is specified by the parameter *index*. Returns the required *AliasPage* or *null* if the page doesn't exist.

*index*: Index of the required *AliasPage*. The value must be in the range 0 – (*aliasPageCount* – 1).

For facing *Alias* pages, the method returns the left page. The right page is available via *aliasSpreadAt(index).pageAt(1)*.

---

**aliasPageByName** (*String name*) *AliasPage*

Returns an *AliasPage* object with the name which is specified by parameter *name*.

*name*: Name of the required *AliasPage*.

Returns the required *AliasPage* or *null* if the page doesn't exist.

---

**addAliasPage** (*String name*, *PageType type* = *SinglePage*) *AliasPage*

The method adds a new *AliasPage* to the document. Returns the added *AliasPage*. For facing *Alias* pages the left page is returned.

*name*: Name of the new *AliasPage*.

*type*: *PageType* (See enumeration type *PageType*).

---

**gotoPage** (*DocumentPage or AliasPage page*) *bool*

Makes the specified page current. Returns true in case of success.

*page*: Target page. This can be *DocumentPage* or *AliasPage*.

---

**removeDocumentPages** (*int fromPageNumber*, *int count*) *bool*

Removes several document pages from the document. Returns true in case of success.

*fromPageNumber*: Index of the first page removed. The index is 0-based. The default value is 0.

*count*: Number of removed pages. The default value is 1.

---

**documentSpreadAt** (*int index*) *Spread*

Returns the *Spread* object at the position which is specified by parameter *index*.

*index*: Index of the required spread. The value must be in range 0 – (*documentSpreadCount* – 1).

---

**aliasSpreadAt** (*int index*) *AliasSpread*

Returns the *Alias spread* object at the position specified by parameter *index*.

*index*: Index of the required spread. The value must be in the range 0 – (*aliasSpreadCount* – 1).

|                                                                                                                                                                                                          |               |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|
| <b>addLayer()</b>                                                                                                                                                                                        | <i>Layer</i>  |
| Adds a new document layer. Returns the new layer.                                                                                                                                                        |               |
| <b>layerAt(int index)</b>                                                                                                                                                                                | <i>Layer</i>  |
| Gets a layer which is specified by the term index.                                                                                                                                                       |               |
| <i>index</i> : Index of the required layer. The value must be in range 0 – (layerCount – 1).                                                                                                             |               |
| <b>moveLayer(Layer layer, int beforeIndex)</b>                                                                                                                                                           | <i>int</i>    |
| Moves the specified layer to a position which is given by the index parameter. Returns -1 if the layer cannot be moved.                                                                                  |               |
| <i>layer</i> : The layer which has to be moved.                                                                                                                                                          |               |
| <i>beforeIndex</i> : Index of the layer before which the given layer has to be moved. The value, which is equal to the layerCount, moves the layer to the end of the list of layers.                     |               |
| <b>closeAndSave()</b>                                                                                                                                                                                    | <i>void</i>   |
| Closes the document. Before closing, the system asks a user if he wants to save the document before it will be closed.                                                                                   |               |
| <b>close()</b>                                                                                                                                                                                           | <i>void</i>   |
| Closes the document immediately. There is no request to save the document. If the document was changed and unsaved all changes are lost.                                                                 |               |
| <b>save(String filePath)</b>                                                                                                                                                                             | <i>bool</i>   |
| Save the document with the name 'filePath'.                                                                                                                                                              |               |
| <b>exportToXML(String filePath, Object parameters = {})</b>                                                                                                                                              | <i>bool</i>   |
| Exports the document with the export filter that matches the file extension of <i>filePath</i> . Returns true in case of success.                                                                        |               |
| <i>parameters</i> : Optional parameters for export filters that support them. Example for IDML: { <i>showDialog: false</i> } exports without the options dialog, using the options last confirmed in it. |               |
| <b>exportToPDF(PDFExportSettings settings)</b>                                                                                                                                                           | <i>bool</i>   |
| <b>package(String outputFolderPath, PackageSettings settings)</b>                                                                                                                                        | <i>bool</i>   |
| <b>setUserProperty(String ident, String data)</b>                                                                                                                                                        | <i>void</i>   |
| Set string data as user property with name <i>ident</i> . The data are "base64" encoded into the document file to the part "<uni:userInfos> <uni:base64Entry>"                                           |               |
| <b>userProperty(String ident)</b>                                                                                                                                                                        | <i>String</i> |
| Get user property string with name <i>ident</i> . The data are 'base64'encoded into the document file.                                                                                                   |               |
| Returns empty string if no item is found.                                                                                                                                                                |               |
| <b>containsUserProperty(String ident)</b>                                                                                                                                                                | <i>bool</i>   |
| Check if user property string with name <i>ident</i> exists                                                                                                                                              |               |
| <b>userXMLProperty(String ownerId, String entryId)</b>                                                                                                                                                   | <i>String</i> |
| Get user property string from the document file from the part '<uni:userInfos> <uni:xmlEntry>'                                                                                                           |               |

|                                                                                                    |             |
|----------------------------------------------------------------------------------------------------|-------------|
| <b>setUserXMLProperty</b> ( <i>String ownerId, String entryId, String data</i> )                   | <i>void</i> |
| Set string data as user property to the document file to the part '<uni:userInfos> <uni:xmlEntry>' |             |
| <b>updateView</b> ()                                                                               | <i>void</i> |
| Updates the view of all documents.                                                                 |             |
| <b>changed</b> ()                                                                                  | <i>void</i> |
| Marks the document as changed.                                                                     |             |
| <b>copyObjectPreferences</b> ( <i>Document sourceDocument</i> )                                    | <i>void</i> |
| Copy the object preferences from the source document.                                              |             |
| <b>updateCustomPropertis</b> ( <i>Document sourceDocument</i> )                                    | <i>void</i> |
| Update the custom properties of the object preferences from the source document.                   |             |

**Example 1:**

```
// Open and show an existing document
var doc = application.openDocument("C:/Documents/Example1.desd", true);
if (doc) {
    // Show file name from which the file was opened
    messageBox.information("Opened Document", doc.filePath);

    // Add 1 new empty page to the document
    var newPage = doc.addDocumentPage();
    // Manipulate the new page here:
    // Create a new text object and add it to the new page
    var objTxtFrm = doc.objects().create(ObjectType.Text);
    newPage.add(objTxtFrm);

    // Close the document with the 'Save' request
    doc.closeAndSave();
}
```

**Example 2:**

```
var doc = application.currentDocument;
if (doc) {
    for (var k = 1; k < 20; k++) {
        //get the count of selected objects
        var objs = doc.objects(WhichSpreads.All, WhichObjects.Selected);
        var arrSelLen = objs.length;

        //... do something

        repaint();
        pause(1000);
    }
}
```

**Example 3:**

```
doc.setUserProperty("IdentName", "IdentDATA");
if (doc.containsUserProperty("IdentName"))
    messageBox.information("Msg2:", doc.userProperty("IdentName"));
```

## Class: Page

The object of class *Page* represents one page.

### Properties:

|                                                                                                                  |               |                   |
|------------------------------------------------------------------------------------------------------------------|---------------|-------------------|
| <b>physicalNumber</b>                                                                                            | <i>int</i>    | <i>Read</i>       |
| Contains the physical page number. Page numbers start from 1 in contrast to the page index, which starts from 0. |               |                   |
| <b>isAliasPage</b>                                                                                               | <i>bool</i>   | <i>Read</i>       |
| Returns true if the page is an Alias page. Otherwise it returns false.                                           |               |                   |
| <b>spread</b>                                                                                                    | <i>Spread</i> | <i>Read</i>       |
| Returns a <i>Spread</i> object that belongs to a Page object.                                                    |               |                   |
| <b>offsetOnSpreadX</b>                                                                                           | <i>float</i>  | <i>Read</i>       |
| Returns the X coordinate of the page offset on the spread.                                                       |               |                   |
| <b>offsetOnSpreadY</b>                                                                                           | <i>float</i>  | <i>Read</i>       |
| Returns the Y coordinate of the page offset on the spread.                                                       |               |                   |
| <b>layoutPositionRow</b>                                                                                         | <i>int</i>    | <i>Read/Write</i> |
| <b>layoutPositionColumn</b>                                                                                      | <i>int</i>    | <i>Read/Write</i> |
| <b>guideCount</b>                                                                                                | <i>int</i>    | <i>Read</i>       |
| Returns the number of guides on the page.                                                                        |               |                   |

### Methods:

**add(Object obj)** *void*  
 Adds an object to the page. The object can be of the class *TextObject*, *PictureObject*, *TableObject*, *GraphicObject*, *LineObject*, *GroupObject*.

*obj*: inserted object of the appropriate class

**objects(WhichObjects type = TopLevel, ObjectType objectType = All)** *Objects*  
 Returns an object of the class *Objects*. This object is a repository and an object provider of all page objects of the classes *TextObject*, *PictureObject*, *TableObject*, *GraphicObject*, *LineObject* and *GroupObject*. On a document page with an Alias page, the list also contains the alias objects of the objects from the Alias page (types *ObjectType.TextAlias*, *PictureAlias* and *Alias*). See “Alias pages and their objects in scripts”.

*type*: Type of objects that are provided by the obtained “Objects” object. See the enumeration type *WhichObjects* at the end of this chapter.

*objectType*: See the enumeration *ObjectType*.

**objectCount(WhichObjects objects = TopLevel)** *int*  
**getObject(int index, WhichObjects objects = TopLevel)** *Object*

**objectCount()** returns the number of objects on the page, **getObject()** the object at the position *index* (or null). Alias objects are included, see **objects()**.

---

**deleteObjects()** *int*

Removes all objects from inside the page. Returns the number of removed objects.

---

**deleteSelectedObjects()** *int*

Removes selected objects from inside the page. Returns the number of removed objects.

---

**preview(String x, String y, String width, String height, PreviewInstructions instructions, String fileName, int quality)** *bool*

---

**preview(String width, String height, String fileName, int quality)** *bool*

Renders a preview image of the page area *x, y, width, height* and saves it to the file *fileName*. The file format is taken from the file extension (e.g. .png, .jpg). The short form renders the area starting at the top left corner of the page.

**instructions:** Optional rendering instructions (scale, alignment, cropping, frame), or null.

**quality:** Image quality 0–100 for formats that support it; other values use the default quality.

Returns true if the file was written.

---

**setUserProperty(String ident, String data)** *void*

Set string data as user property with name *ident*. The data are 'base64'encoded into the document file to the part '<uni:userInfos> <uni:base64Entry>'

---

**userProperty(String ident)** *String*

Get user property string with name *ident*. The data are 'base64'encoded into the document file to the part '<uni:userInfos> <uni:base64Entry>'

Returns empty string if no item is found.

---

**containsUserProperty(String ident)** *bool*

Check if user property string with name *ident* exists.

---

**userXMLProperty(String ownerId, String entryId)** *String*

Returns the custom part of the document description in the XML document.

---

**setUserXMLProperty(String ownerId, String entryId, String data)** *void*

Sets the custom part of the document description in the XML document.

---

**addHorizontalGuide(String position, Layer layer)** *int*

Returns ID of the guide

---

**addVerticalGuide(String position, bool fitToPage, Layer layer)** *int*

Returns ID of the guide

---

**removeGuide(int id)** *bool*

---

**guide(int id)** *Guide*

Returns the guide with the ID *id*. The returned object only has the property *isNull*, which is true if no guide with this ID exists.

---

`clearGuides()`


---

`void`

## Class: DocumentPage

The object of the class **DocumentPage** represents one particular page of the appropriate document. The class extends the root class **Page** and adds another properties, which are specific for this type of object. JScript can add one or more new pages, remove pages or move them to another position in the document.

The page is a container of other objects which are instances of the classes *TextObject*, *PictureObject*, *TableObject*, *GraphicObject*, *LineObject* and *GroupObject*.

### Properties:

---

|                      |            |             |
|----------------------|------------|-------------|
| <b>logicalNumber</b> | <i>int</i> | <i>Read</i> |
|----------------------|------------|-------------|

---

Contains the logical page number. The page numbers start from 1 in contrast with the page index, which starts from 0.

---

|              |               |             |
|--------------|---------------|-------------|
| <b>label</b> | <i>String</i> | <i>Read</i> |
|--------------|---------------|-------------|

---

Contains textual label of the page. Normally it is same like logical number.

---

|                    |                    |                   |
|--------------------|--------------------|-------------------|
| <b>chapterInfo</b> | <i>ChapterInfo</i> | <i>Read/Write</i> |
|--------------------|--------------------|-------------------|

---

Sets or gets the chapter info object. For more information about the chapter info object see the chapter “Class: ChapterInfo” in this documentation.

---

|                  |                  |             |
|------------------|------------------|-------------|
| <b>aliasPage</b> | <i>AliasPage</i> | <i>Read</i> |
|------------------|------------------|-------------|

---

Contains a reference to the Alias page. If the page has no Alias page, the value is *null*. The objects of the Alias page are returned by `aliasPage.objects()`.

### Methods:

---

|                     |                                            |             |
|---------------------|--------------------------------------------|-------------|
| <b>setAliasPage</b> | <i>(AliasPage page, bool applyContent)</i> | <i>bool</i> |
|---------------------|--------------------------------------------|-------------|

---

Set the Alias page of the document page. Returns true in case of success.

**page:** AliasPage which is set as a template of the document page. For facing Alias pages, the left or right half is chosen automatically to match the document page. page must not be null.

**applyContent:** If true, an alias object is placed on the document page for each object of the Alias page (see “Alias pages and their objects in scripts”). If false, only the assignment is changed.

---

|                 |             |
|-----------------|-------------|
| <b>remove()</b> | <i>bool</i> |
|-----------------|-------------|

---

Removes a document page from the document. Returns true in case of success.

### Examples:

```
var doc = application.currentDocument;
if (doc) {
    // Create a new object, which is then inserted to a page
    var obj = doc.objects().create(ObjectType.Text);
    // Configure the object
    with (obj) {
        name = "Text Frame 1";
    }
}
```

```
        form = FrameForm.Ellipse;
        lineWidth = 12;
        pagex = "20mm";
        pagey = "100mm";
        height = "40mm";
        width = "170mm";
    }
    // create page setting object
    var pgs = new PageSettings();
    pgs.insertPosition = PageInsertPosition.AfterCurrentPage;
    pgs.afterPageNumber = 1;
    pgs.copyObjects = false;
    pgs.copyOnlySelectedObjects = false;
    pgs.copyAsAlias = false;
    pgs.linkToTextChain = false;
    // add a new page to the document
    var newPage = doc.addDocumentPage(1, pgs);
    // ... or a function call with default parameters can be used
    //var newPage = doc.addDocumentPage();
    if (newPage) {
        // Insert the new object to the new page
        newPage.add(obj);

        // Show logical number of the page
        messageBox.information("Page Number", newPage.logicalNumber);
    }
    // Get last page of the document
    var lastPage = doc.documentPageAt(doc.documentPageCount - 1);
    if (lastPage)
        // Remove all objects from the last page
        lastPage.deleteObjects();
}
```

## Class: AliasPage

The class **AliasPage** represents a kind of page template. The class extends the root class **Page** and adds another properties, which are specific for this type of object. The author of the document can create one or more Alias pages and assign them to any document page. Each document page may have 0 or 1 Alias page. The Alias page is useful for the creation of page elements which are repeated on several or all document pages. The main advantage of Alias pages is the possibility of editing these common page elements in one operation. All changes are of course automatically visible on all pages that use the appropriate Alias page.

### Properties:

| name                                                                                                                                                       | String   | Read/Write |
|------------------------------------------------------------------------------------------------------------------------------------------------------------|----------|------------|
| The property is used to read or set name of the Alias page. The name belongs to the Alias spread, so both pages of a facing Alias page have the same name. |          |            |
| pageType                                                                                                                                                   | PageType | Read       |
| Returns PageType.FacingPages if the Alias page belongs to a facing Alias spread, otherwise PageType.SinglePage.                                            |          |            |

**Methods:****remove()** *bool*

Removes the Alias page. For facing Alias pages, both pages (the whole Alias spread) are removed. Returns true in case of success.

## Alias pages and their objects in scripts

If an Alias page is assigned to a document page with its content, VivaDesigner places an alias object on the document page for each object of the Alias page; the originals stay on the Alias page. The method **objects()** of the document page therefore returns the independent objects of the page together with these alias objects (ObjectType.TextAlias, PictureAlias, Alias). There is no separate method for the objects from the Alias page.

Alias object and object from the Alias page are not the same: every object of the class ObjectAlias is an alias object, including those created with Object > Alias or **createAlias()**. An object from the Alias page is an alias object whose original lies on the Alias page: its property **owner** returns the original, and **owner.page.isAliasPage** is true.

The Alias page of a document page is available via **aliasPage**, its own objects via **aliasPage.objects()**; **setAliasPage(aliasPage, applyContent)** assigns another one. Alias objects can only be rotated, not scaled or skewed. Independent objects are moved with **pagex/pagey, moveOnPageTo()** or **moveOnSpreadTo()** (see the chapter “Object Types and Structure”).

**Example:**

```
var doc = application.currentDocument;
if (doc) {
    var pg = doc.currentPage;
    var objs = pg.objects();
    var list = "";
    for (var i = 0; i < objs.length; i++) {
        var obj = objs.at(i);
        var t = obj.type;
        if (t == ObjectType.TextAlias || t == ObjectType.PictureAlias ||
            t == ObjectType.Alias) {
            var orig = obj.owner;
            if (orig && orig.page && orig.page.isAliasPage)
                list = list + "From the Alias page: " + orig.name;
            else
                list = list + "Alias object: " + obj.name;
            if (t == ObjectType.TextAlias)
                list = list + " - " + obj.content.formattedText.plainText();
        }
        else
            list = list + "Independent object: " + obj.name;
        list = list + "\n";
    }
    messageBox.information("Objects on page " + pg.logicalNumber, list);

    // The Alias page itself and its objects
    var ap = pg.aliasPage;
```

```
if (ap) {  
    var n = ap.objects().length;  
    messageBox.information("Alias page " + ap.name, n + " objects");  
}  
}
```

## Class: Spread

A spread is an area in which document pages are located. Each document has one or more spreads. Each page is located in one of these spreads. A spread may contain more than 1 page. Each object located on a page is also located on the appropriate spread. The object is considered as being on the page if at least 50% of its area lies on the page surface. Otherwise the object is not on the page but is still on the spread.

### Properties:

---

|                  |            |             |
|------------------|------------|-------------|
| <b>pageCount</b> | <i>int</i> | <i>Read</i> |
|------------------|------------|-------------|

Contains the number of pages located on the appropriate spread.

---

|                   |            |             |
|-------------------|------------|-------------|
| <b>guideCount</b> | <i>int</i> | <i>Read</i> |
|-------------------|------------|-------------|

Returns the number of guides on the spread.

---

### Methods:

---

**add**(*Object obj*, *Object afterObj = null*) *void*  
Add an object to the spread.

*obj*: Object which is added to the spread.

*afterObj*: Object after which the new object is inserted in the stacking order. Without *afterObj*, the object is placed on top, on the current layer.

---

**objects**(*WhichObjects type = TopLevel*, *ObjectType objectType = All*) *Objects*  
Returns an object of the class *Objects*. This object is a repository and an object provider of all spread objects of the classes *TextObject*, *PictureObject*, *TableObject*, *GraphicObject*, *LineObject* and *GroupObject*.

*type*: Type of objects which are provided by the “obtained Objects” object. See the enumeration type *WhichObjects* at the end of this chapter.

*objectType*: See the enumeration *ObjectType*.

---

|                                                               |            |
|---------------------------------------------------------------|------------|
| <b>objectCount</b> ( <i>WhichObjects objects = TopLevel</i> ) | <i>int</i> |
|---------------------------------------------------------------|------------|

---

|                                                                                |               |
|--------------------------------------------------------------------------------|---------------|
| <b>getObject</b> ( <i>int index</i> , <i>WhichObjects objects = TopLevel</i> ) | <i>Object</i> |
|--------------------------------------------------------------------------------|---------------|

---

**pageAt**(*int index*) *DocumentPage* or *AliasPage*  
Returns a *DocumentPage* object at the position which is identified by the index.

*index*: Index of the required page. Its value must be in the range 0 – (pageCount – 1);

Returns the required page or *null* if the required page doesn’t exist.

---

|                                            |                                         |
|--------------------------------------------|-----------------------------------------|
| <b>objectPage</b> ( <i>Object object</i> ) | <i>DocumentPage</i> or <i>AliasPage</i> |
|--------------------------------------------|-----------------------------------------|

Returns a *DocumentPage* object in which the specified object is located.

---

**object:** The object for which the container page searches.

Returns the container page or *null* if the object is not located in any page.

---

**setUserProperty**(*String ident, String data*) *void*

Set string data as user property with name *ident*. The data are 'base64'encoded into the document file to the part '<uni:userInfos> <uni:base64Entry>'

---

**userProperty**(*String ident*) *String*

Get user property string with name *ident*. The data are 'base64'encoded into the document file to the part '<uni:userInfos> <uni:base64Entry>'

Returns empty string if no item is found.

---

**containsUserProperty**(*String ident*) *bool*

Check if user property string with name *ident* exists.

---

**userXMLProperty**(*String ownerId, String entryId*) *String*

Get user property string from the document file from the part '<uni:userInfos> <uni:xmlEntry>'

---

**setUserXMLProperty** (*String ownerId, String entryId, String data*) *void*

Set string data as user property to the document file to the part '<uni:userInfos> <uni:xmlEntry>'

---

**addHorizontalGuide**(*String position, Layer layer*) *int*

Adds a horizontal guide at the given position on the given layer. Returns the ID of the guide or -1.

---

**addVerticalGuide**(*String position, VerticalGuideZone zone, Layer layer*) *int*

Adds a vertical guide at the given position on the given layer. **zone:** see the enumeration VerticalGuideZone. Returns the ID of the guide or -1.

---

**guide**(*int id*) *Guide*

Returns the guide with the ID *id*. The returned object only has the property *isNull*.

---

**removeGuide**(*int id*) *bool*

---

**clearGuides**() *void*

Remove one guide or all guides of the spread.

### Examples:

```
var doc = application.currentDocument;
if (doc) {
    // Show number of spreads in a current document
    messageBox.information("Spread Count", doc.documentSpreadCount);

    // Get 1st spread of the document
    var spr = doc.documentSpreadAt(0);

    // Create 2 various objects and add them to the spread
    var o1 = doc.objects().create(ObjectType.Table);
    o1.name = "Table";
    spr.add(o1);

    var o2 = doc.objects().create(ObjectType.Text);
```

```
o2.name = "Text";
spr.add(o2);

// Enumerate objects in the spread
var objs = spr.objects(WhichObjects.All);
var objList = "";
for (var idx = 0; idx < objs.length; idx++) {
    var obj = objs.at(idx);
    objList = objList + obj.name + " (" + obj.type + ")\n";
}

// Show list of objects in the spread
messageBox.information("Spread Objects", objList);
}
```

## Class: AliasSpread

The class *AliasSpread* represents a kind of spread template. The class extends the root class *Spread* and adds other properties which are specific for this type of object. It is an area in which an Alias page is located. Each Alias page is located in some Alias spread.

### Properties:

---

| name | String | Read/Write |
|------|--------|------------|
|------|--------|------------|

Reads or sets the name of the Alias spread.

### Methods:

---

| remove() | bool |
|----------|------|
|----------|------|

Removes the Alias spread (including nested Alias pages) from the appropriate document. Returns true in case of success.

### Examples:

```
var doc = application.currentDocument;
if (doc) {

    // Get number of Alias spreads
    messageBox.information("Alias Page Count", doc.aliasPageCount);

    // Add new Alias pages with single resp. facing Alias pages
    var ap1 = doc.addAliasPage("AP1", PageType.SinglePage);
    var ap2 = doc.addAliasPage("AP2", PageType.FacingPages);

    var aList = "";
    for (i = 0; i < doc.aliasPageCount; i++) {
        var apx = doc.aliasPageAt(i);
        if (apx) {
            var ax = apx.spread;
            ax.name = "A" + i;
            aList = aList + "Alias Spread [" + i + "] = " + ax.name + "\n";
        }
    }
    // Show Alias spread list
}
```

```

    messageBox.information("Alias Spreads", aList);
}

```

## Class: Layer

Each document has one or possibly more layers. They work like transparent pieces of film. Each visible object on a page and/or on a spread is located in one of the document layers. Objects in a higher layer cover objects in a lower layer. Any layer can be hidden or displayed at any time. A layer can be locked to protect it against changes. Options for a layer can exclude it from printing, etc.

### Properties:

|                                                                                                                                                                                                                                                          |                   |                   |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|-------------------|
| <b>name</b>                                                                                                                                                                                                                                              | <i>String</i>     | <i>Read/Write</i> |
| Contains the name of the layer. By default the name is "Layer X", where X is an integer number.                                                                                                                                                          |                   |                   |
| <b>hidden</b>                                                                                                                                                                                                                                            | <i>bool</i>       | <i>Read/Write</i> |
| The value is true or false if the layer is visible or hidden.                                                                                                                                                                                            |                   |                   |
| <b>locked</b>                                                                                                                                                                                                                                            | <i>bool</i>       | <i>Read/Write</i> |
| The value is true or false if the layer is locked or unlocked.                                                                                                                                                                                           |                   |                   |
| <b>printable</b>                                                                                                                                                                                                                                         | <i>bool</i>       | <i>Read/Write</i> |
| The value is true or false if the layer is printable or not printable.                                                                                                                                                                                   |                   |                   |
| <b>keepRunaround</b>                                                                                                                                                                                                                                     | <i>bool</i>       | <i>Read/Write</i> |
| <b>brush</b>                                                                                                                                                                                                                                             | <i>ColorBrush</i> | <i>Write</i>      |
| Sets the color of the layer from a ColorBrush. The color is used for the frames of the objects on this layer when guides are shown (option "Show Guides"), which allows you to distinguish objects on different layers. To read the color, use colorRgb. |                   |                   |
| <b>colorRgb</b>                                                                                                                                                                                                                                          | <i>Rgb</i>        | <i>Read/Write</i> |
| Set/get the color of a layer.                                                                                                                                                                                                                            |                   |                   |

### Methods:

|                                                                                                                                        |                |
|----------------------------------------------------------------------------------------------------------------------------------------|----------------|
| <b>objects(ObjectType objectType = All)</b>                                                                                            | <i>Objects</i> |
| objectType: See the enumeration ObjectType                                                                                             |                |
| Returns a list of objects which are assigned to the layer.                                                                             |                |
| <b>orderedObjects()</b>                                                                                                                | <i>Objects</i> |
| Returns the objects of the layer in their stacking order.                                                                              |                |
| <b>objectCount()</b>                                                                                                                   | <i>int</i>     |
| <b>getObject(int index)</b>                                                                                                            | <i>Object</i>  |
| <i>objectCount()</i> returns the number of objects on the layer, <i>getObject()</i> the object at the position <i>index</i> (or null). |                |
| <b>remove()</b>                                                                                                                        | <i>bool</i>    |
| Removes the layer from the appropriate document. Returns true in case of success.                                                      |                |

**Examples:**

```
var doc = application.currentDocument;
if (doc) {
    // Check the number of layers in the new document
    messageBox.information("Layer Count", doc.layerCount);

    // Get the first layer and configure it.
    var lay1 = doc.layerAt(0);
    lay1.name = "Background";
    lay1.brush = new ColorBrush(Color.Blue);
    lay1.printable = false; // Don't print background layer.

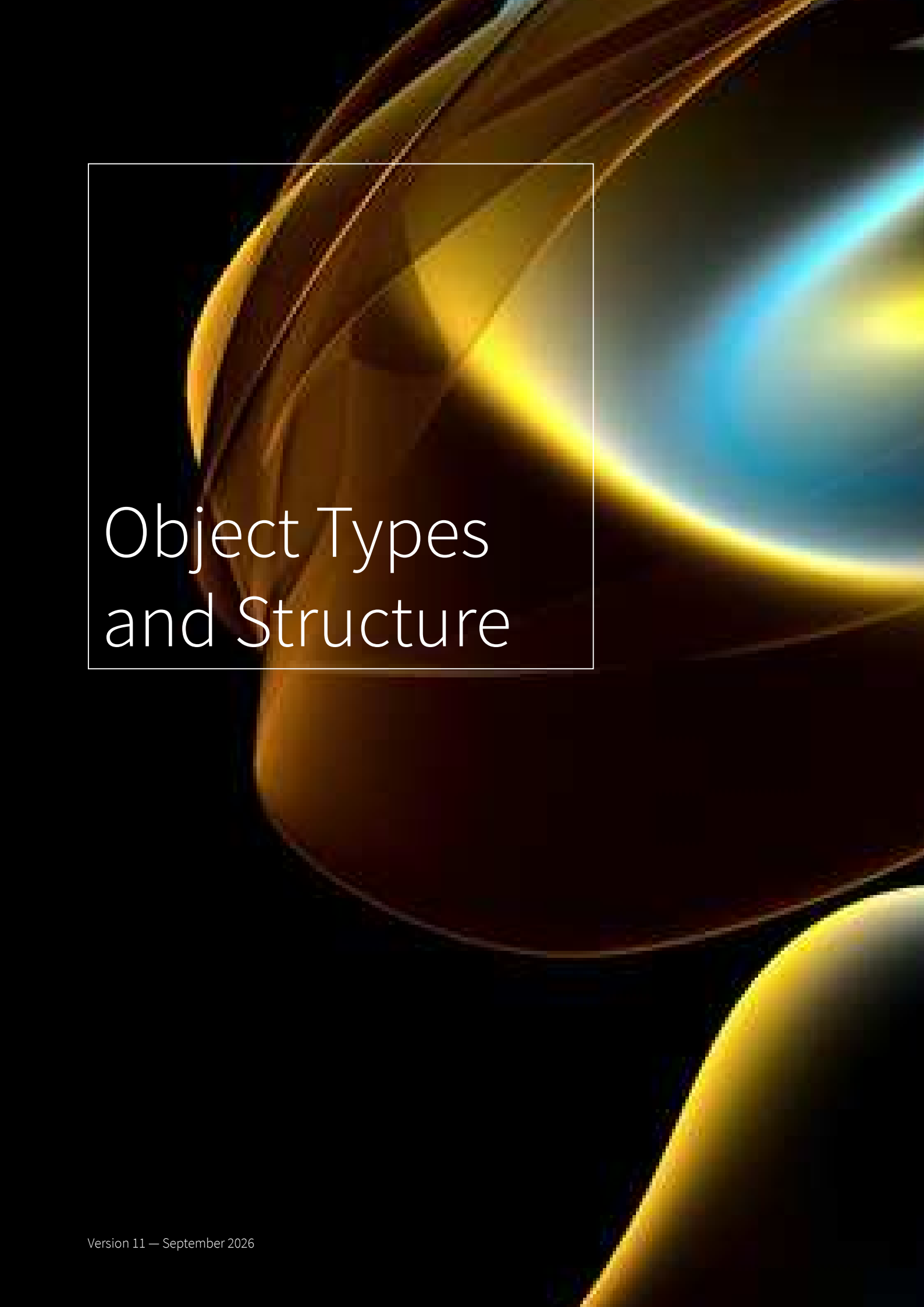
    // Add the 2nd layer and configure it
    var lay2 = doc.addLayer();
    lay2.name = "Layer 1";
    lay2.brush = new ColorBrush(Color.Red);

    // Make the layer current. Then all objects which are added to current
    // page are placed on this layer.
    doc.currentLayer = lay2;

    // Work with current layer here
    ...

    // Add another layer and move it between 2 existing layers.
    var lay3 = doc.addLayer();
    lay3.name = "Layer 2";
    lay3.brush = new ColorBrush(Color.Green);

    doc.moveLayer(lay2, 2);
}
```



# Object Types and Structure

# Object Types and Structure

This chapter describes the hierarchy of all graphic object classes that can be used on the page or on the spread in the document.

Document pages can contain objects of the classes *FrameObject*, *LineObject*, *TextObject*, *PictureObject*, *TableObject* and *GroupObject*. The root of all these classes is the class *Object*. The root class cannot be instantiated but shares properties and methods with its subclasses. Alias objects are represented by the classes *ObjectAlias*, *TextObjectAlias* and *PictureObjectAlias*. The complete tree of the page element classes is as follows:

```
Object
|--- GroupObject
|--- GraphicObject
|   |--- LineObject
|   |--- FrameObject
|       |--- TextObject
|       |--- PictureObject
|       |--- TableObject
|--- ObjectAlias
|   |--- TextObjectAlias
|   |--- PictureObjectAlias
```

The classes *Object* and *GraphicObject* cannot be instantiated. They only share their properties and methods with subclasses.

In scripts, some of these classes are available under other names, which are used with *instanceof*: *TextFrame* (*TextObject*), *PictureFrame* (*PictureObject*), *TableFrame* (*TableObject*), *GraphicFrame* (*FrameObject*), *Line* (*LineObject*) and *Group* (*GroupObject*).

The following chapters describe elements which can be inserted into *DocumentPages*, *Spreads*, *AliasPages* and *AliasSpreads*.

## Class: Objects

The class *Objects* represents a collection of objects which are attached to another container object. For example some document pages may contain several text objects, picture objects or table objects. A method *objects()* of the class *DocumentPage* returns the collection of all objects inside the page. The object of the class *Objects* is also a kind of object factory, which allows you to create new objects (text, pictures etc.). See examples to understand better how to use objects of the class *Objects*.

The object list is current at the time when it is created. The object list must be refreshed when the object is created or deleted in the source object container or the property *isOnline* must be set to *TRUE*.

The collection is indexed from 0 to length-1.

### Properties:

|                                                                           |             |                   |
|---------------------------------------------------------------------------|-------------|-------------------|
| <b>length</b>                                                             | <i>int</i>  | <i>Read</i>       |
| Contains a number of objects which are members of this object collection. |             |                   |
| <b>isOnline</b>                                                           | <i>bool</i> | <i>Read/Write</i> |

If the property is set to `TRUE`, the object list is synchronized with the source object container every time. If the object is deleted or added, the change is reflected to the object list.

The default is ***FALSE***.

### Methods:

---

**create** (*ObjectType type*)                      *Object*

Creates a new object of the specified type.

*type*: Type of the created object. See the enumeration type `ObjectType` at the end of this chapter to see all available object types. The ***Alias*** objects cannot be created.

Returns an object of required type.

---

**at** (*int index*)                      *Object*

Get object at the specified position inside the object collection.

*index*: Position of the required object. The value must be in the range from 0 to (length – 1).

Returns the object at the specified position or ***null*** if the object is not found (e.g. index equals or is higher than the length).

---

**refresh** ()                      *void*

The object is synchronized with the object in the source container.

---

### Example 1:

```
var doc = application.currentDocument;
if (doc) {
    var objs = doc.objects();
    messageBox.information("Object Count", objs.length);
    // Create 1st object
    var obj1 = objs.create(ObjectType.Text);
    obj1.name = "Obj 1";
    doc.currentPage.add(obj1);
    messageBox.information("Object Count", objs.length);
    // Create 2nd object
    var obj2 = objs.create(ObjectType.Picture);
    obj2.name = "Obj 2";
    doc.currentPage.add(obj2);
    messageBox.information("Object Count", objs.length);

    // Get 1st object in the current page
    var objX = doc.currentPage.objects().at(0);
    if( objX != null )
        messageBox.information("1st Object", objX.name);
    else
        messageBox.information("1st Object", "NOT FOUND");
}
```

### Example 2:

```
var doc = application.currentDocument;
```

---

```
if (doc) {
    var pg = doc.currentPage;
    var objs = pg.objects(WhichObjects.All);
    if (objs) {
        var textObject = null;
        var pictureObject = null;

        messageBox.information("Length", objs.length);

        for (var index = 0; index <= objs.length - 1 ; index++) {
            var obj = objs.at(index);
            if (obj instanceof TextFrame) {
                obj.content.insertPlainText(index, 0);
                if (!textObject)
                    textObject = obj;
            }
            else if (obj instanceof PictureFrame) {
                pictureObject = obj;
            }
        }
    }
}
```

## Class: Object

The class *Object* is a root class of page elements and provides the following properties and methods.

### Properties:

|                                                                                                                                                                                                                                                                                                                                                            |                        |                   |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------|-------------------|
| <b>name</b>                                                                                                                                                                                                                                                                                                                                                | <i>String</i>          | <i>Read/Write</i> |
| Set or get the name of the object.                                                                                                                                                                                                                                                                                                                         |                        |                   |
| <b>type</b>                                                                                                                                                                                                                                                                                                                                                | <i>ObjectType</i>      | <i>Read</i>       |
| Contains the current type of the object. See enumeration items at the end of this chapter.                                                                                                                                                                                                                                                                 |                        |                   |
| <b>id</b>                                                                                                                                                                                                                                                                                                                                                  | <i>String</i>          | <i>Read</i>       |
| Unique ID of the object in the document.                                                                                                                                                                                                                                                                                                                   |                        |                   |
| <b>typeString</b>                                                                                                                                                                                                                                                                                                                                          | <i>String</i>          | <i>Read</i>       |
| Internal type name of the object; for alias objects it contains e.g. "Object/TextAlias".                                                                                                                                                                                                                                                                   |                        |                   |
| <b>pagex</b>                                                                                                                                                                                                                                                                                                                                               | <i>UnitValueString</i> | <i>Read/Write</i> |
| X-coordinate of the position of the top left corner of the object inside the container page. If the object is not on any page, the value is <i>null</i> and the setting of a new value is ignored. The value is returned in points (pt) without a unit; a new value may contain a unit, e.g. "20mm". Objects inside a group cannot be positioned this way. |                        |                   |
| <b>pagey</b>                                                                                                                                                                                                                                                                                                                                               | <i>UnitValueString</i> | <i>Read/Write</i> |

Y-coordinate of position of the top left corner of the object inside the container page. If the object is not on any page, the value is *null* and the setting of a new value is ignored. The value is returned in points (pt) without a unit; a new value may contain a unit, e.g. "20mm". Objects inside a group cannot be positioned this way.

|                                                                                                                                                                                                                                                    |                        |                                              |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------|----------------------------------------------|
| <b>spreadx</b>                                                                                                                                                                                                                                     | <i>UnitValueString</i> | <i>Read/Write</i>                            |
| X-coordinate of position of the top left corner of the object inside the container spread. The value is returned in points (pt) without a unit; a new value may contain a unit, e.g. "20mm". Objects inside a group cannot be positioned this way. |                        |                                              |
| <b>spready</b>                                                                                                                                                                                                                                     | <i>UnitValueString</i> | <i>Read/Write</i>                            |
| Y-coordinate of position of the top left corner of the object inside the container spread. The value is returned in points (pt) without a unit; a new value may contain a unit, e.g. "20mm". Objects inside a group cannot be positioned this way. |                        |                                              |
| <b>printable</b>                                                                                                                                                                                                                                   | <i>bool</i>            | <i>Read/Write</i>                            |
| Set to true or false to make the object printable or not printable. <i>If the object is not suitable for printing, simply set it to false.</i>                                                                                                     |                        |                                              |
| <b>protected</b>                                                                                                                                                                                                                                   | <i>bool</i>            | <i>Read/Write</i>                            |
| If this property is set to true, the object content cannot be modified or deleted.                                                                                                                                                                 |                        |                                              |
| <b>visible</b>                                                                                                                                                                                                                                     | <i>bool</i>            | <i>Read/Write</i>                            |
| If this property is set to false, the object is hidden.                                                                                                                                                                                            |                        |                                              |
| <b>locked</b>                                                                                                                                                                                                                                      | <i>bool</i>            | <i>Read/Write</i>                            |
| If this property is set to true, the object size and position cannot be changed.                                                                                                                                                                   |                        |                                              |
| <b>rotation</b>                                                                                                                                                                                                                                    | <i>float</i>           | <i>Read/Write &lt;-360, 360&gt; [degree]</i> |
| Rotation of the object in degrees. A positive value means rotation in an anticlockwise direction. A negative value means rotation in a clockwise direction.                                                                                        |                        |                                              |
| <b>skew</b>                                                                                                                                                                                                                                        | <i>float</i>           | <i>Read/Write &lt;-75, 75&gt; [degree]</i>   |
| Defines the skewing of the object in degrees. A positive value means skewing the object to the left. A negative value means skewing the object to the right.                                                                                       |                        |                                              |
| <b>mirrored</b>                                                                                                                                                                                                                                    | <i>bool</i>            | <i>Read/Write</i>                            |
| If this property is set to true, the object is mirrored according to the horizontal axis.                                                                                                                                                          |                        |                                              |
| <b>selected</b>                                                                                                                                                                                                                                    | <i>bool</i>            | <i>Read/Write</i>                            |
| If this property is set to true, the object is selected.                                                                                                                                                                                           |                        |                                              |
| <b>guide</b>                                                                                                                                                                                                                                       | <i>bool</i>            | <i>Read/Write</i>                            |
| If this property is set to true, the object is set as a (magnetic) guide object.                                                                                                                                                                   |                        |                                              |
| <b>topLevel</b>                                                                                                                                                                                                                                    | <i>bool</i>            | <i>Read</i>                                  |
| If this property is set to true, the object is a top level object.                                                                                                                                                                                 |                        |                                              |
| <b>matrix</b>                                                                                                                                                                                                                                      | <i>Matrix</i>          | <i>Read/Write</i>                            |

The matrix defines the transformation of the object. Transformation affects the properties: *Rotation*, *Skew*, *Scaling*, *Mirroring* and *Position/Offset* (x/y). If the object is added to a page or a spread, the matrix parameters are applied to the spread.

|                                                                                                   |                                         |                   |
|---------------------------------------------------------------------------------------------------|-----------------------------------------|-------------------|
| <b>note</b>                                                                                       | <i>String</i>                           | <i>Read/Write</i> |
| Set/get a note for the object.                                                                    |                                         |                   |
| <b>page</b>                                                                                       | <i>DocumentPage</i> or <i>AliasPage</i> | <i>Read</i>       |
| Returns the page object where the object is placed.                                               |                                         |                   |
| <b>spread</b>                                                                                     | <i>Spread</i> or <i>AliasSpread</i>     | <i>Read</i>       |
| Returns the spread object where the object is placed.                                             |                                         |                   |
| <b>pageBoundingBox</b>                                                                            | <i>BoundingBox</i>                      | <i>Read</i>       |
| Returns the object bounding box in the page coordinates or <i>null</i> if the page doesn't exist. |                                         |                   |
| <b>spreadBoundingBox</b>                                                                          | <i>BoundingBox</i>                      | <i>Read</i>       |
| Returns the object bounding box in the spread coordinates.                                        |                                         |                   |
| <b>groupObject</b>                                                                                | <i>Object</i>                           | <i>Read</i>       |
| Returns the group object where the object is placed or <i>null</i> if the group doesn't exist.    |                                         |                   |
| <b>runaround</b>                                                                                  | <i>Runaround</i>                        | <i>Read/Write</i> |
| Set/get the runaround settings of the object. See the chapter "Class: Runaround".                 |                                         |                   |
| <b>layer</b>                                                                                      | <i>Layer</i>                            | <i>Read/Write</i> |
| Set/get the layer on which the object is placed.                                                  |                                         |                   |
| <b>taggedPdfOptions</b>                                                                           | <i>TaggedPdfOptions</i>                 | <i>Read/Write</i> |
| Tagging options of the object for the tagged PDF export.                                          |                                         |                   |

#### Methods:

|                                                                                                                                                                |               |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|
| <b>setUserProperty</b> ( <i>String ident</i> , <i>String data</i> )                                                                                            | <i>void</i>   |
| Set string data as user property with name <i>ident</i> . The data are 'base64' encoded into the document file to the part '<uni:userInfos> <uni:base64Entry>' |               |
| <i>ident</i> : Name of the property.                                                                                                                           |               |
| <i>data</i> : Value set to the property.                                                                                                                       |               |
| <b>userProperty</b> ( <i>String ident</i> )                                                                                                                    | <i>String</i> |
| Get value from the named property. The value is set to the property by an earlier call of the method <i>setUserProperty(ident, data)</i> .                     |               |
| <i>ident</i> : Name of the property.                                                                                                                           |               |
| Returns textual value of the property. If the property doesn't exist, the returned value is an empty string.                                                   |               |
| <b>copyUserProperties</b> ( <i>Object sourceObject</i> )                                                                                                       | <i>void</i>   |
| The user property data are copied from the source object.                                                                                                      |               |

|                                                                                                                                                                                      |                    |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------|
| <b>userXMLProperty</b> ( <i>String ownerId, String entryId</i> )                                                                                                                     | <i>String</i>      |
| Get user property string from the document file from the part '<uni:userInfos> <uni:xmlEntry>'                                                                                       |                    |
| <b>setUserXMLProperty</b> ( <i>String ownerId, String entryId, String data</i> )                                                                                                     | <i>void</i>        |
| Set string data as user property to the document file to the part '<uni:userInfos> <uni:xmlEntry>'                                                                                   |                    |
| <b>containsUserProperty</b> ( <i>String ident</i> )                                                                                                                                  | <i>bool</i>        |
| Checks if the object contains a specified user property.                                                                                                                             |                    |
| <i>ident</i> : Name of the checked property.                                                                                                                                         |                    |
| Returns true if the user property exists, otherwise it returns false.                                                                                                                |                    |
| <b>remove</b> ( )                                                                                                                                                                    | <i>bool</i>        |
| Removes the object from the appropriate page and destroys it. Returns true in case of success.                                                                                       |                    |
| <b>createAlias</b> ( )                                                                                                                                                               | <i>ObjectAlias</i> |
| Creates an alias object of the object (like Object > Alias). Returns a TextObjectAlias, PictureObjectAlias or ObjectAlias, or null.                                                  |                    |
| <b>fullCopy</b> ( )                                                                                                                                                                  | <i>Object</i>      |
| Copies the object and add it to the spread with right position on the page. Returns new object. Returns null for alias objects.                                                      |                    |
| <b>copy</b> ( )                                                                                                                                                                      | <i>Object</i>      |
| Copies the object. Returns new object only. Returns null for alias objects.                                                                                                          |                    |
| <b>customProperty</b> ( <i>String ident</i> )                                                                                                                                        | <i>String</i>      |
| Get value of the string type of the custom property. Name of the feature in the VivaDesigner is 'User Defined Properties'. Returns textual value of the property.                    |                    |
| <i>ident</i> : Name of the property.                                                                                                                                                 |                    |
| <b>customPropertyBoolean</b> ( <i>String ident</i> )                                                                                                                                 | <i>bool</i>        |
| Get value of the checkbox type of the custom property. The name of the feature in the VivaDesigner is 'User Defined Properties'. Returns boolean value (TRUE/FALSE) of the property. |                    |
| <i>ident</i> : Name of the property.                                                                                                                                                 |                    |
| <b>setCustomProperty</b> ( <i>String ident, String data</i> )                                                                                                                        | <i>void</i>        |
| <b>setCustomProperty</b> ( <i>String ident, bool data</i> )                                                                                                                          | <i>void</i>        |
| Set the value of the 'User Defined Properties' (String or boolean).                                                                                                                  |                    |
| <b>changeOrder</b> ( <i>ChangeOrderType type</i> )                                                                                                                                   | <i>bool</i>        |
| Changes the order of object on the page.                                                                                                                                             |                    |
| <b>preview</b> ( <i>String x, String y, String width, String height, PreviewInstructions instructions, String fileName, int quality</i> )                                            | <i>bool</i>        |
| <b>preview</b> ( <i>String width, String height, String fileName, int quality</i> )                                                                                                  | <i>bool</i>        |

Renders a preview image of the object area *x*, *y*, *width*, *height* and saves it to the file *fileName*. The file format is taken from the file extension (e.g. .png, .jpg). The short form renders the area starting at the top left corner.

**instructions:** Optional rendering instructions (scale, alignment, cropping, frame), or null.

**quality:** Image quality 0–100 for formats that support it; other values use the default quality.

Returns true if the file was written.

---

**moveOnPageTo** (*Point p*, *MoveReferencePoint ref* = *BBoxTopLeft*, *Page pg* = *null*) *bool*

Moves the object so that its reference point lies at the position *p* on the page *pg*. If *pg* is null, the page on which the object is located is used. Returns true in case of success.

**ref:** *MoveReferencePoint.BBoxTopLeft* (default, top left corner of the bounding box) or *MoveReferencePoint.OriginTopLeft* (origin of the object).

---

**moveOnPageTo** (*String x*, *String y*, *MoveReferencePoint ref* = *BBoxTopLeft*, *Page pg* = *null*) *bool*

Same as above with separate coordinates *x* and *y*. The coordinates may contain a unit, e.g. “20mm”; values without a unit are points (pt).

---

**moveOnSpreadTo** (*Point p*, *MoveReferencePoint ref* = *BBoxTopLeft*) *bool*

Moves the object so that its reference point lies at the position *p* on the spread. Returns true in case of success.

---

**moveOnSpreadTo** (*String x*, *String y*, *MoveReferencePoint ref* = *BBoxTopLeft*) *bool*

Same as above with separate coordinates *x* and *y*. The coordinates may contain a unit, e.g. “20mm”; values without a unit are points (pt).

---

**move** (*Point point*) *bool*

Moves the object to the absolute position *point* on the spread (it does not move the object by an offset). Same as *moveOnSpreadTo(point)* with the reference point *BBoxTopLeft*.

---

**getAttributes** () *Attributes*

---

**setAttributes** (*Attributes attrs*) *void*

---

**copyAttributes** (*Object fromObj*) *boolean*

## Class: GraphicObject

The class *GraphicObject* extends the root class *Object* and adds other properties that are specific for this type of object. This class also cannot be instantiated.

### Properties:

|                                                                                                                                            |               |                   |
|--------------------------------------------------------------------------------------------------------------------------------------------|---------------|-------------------|
| <b>dropShadow</b>                                                                                                                          | <i>Shadow</i> | <i>Read/Write</i> |
| Defines the parameters of the object’s drop shadow. See class <i>Shadow</i> in this documentation to see details of the shadow parameters. |               |                   |
| <b>innerShadow</b>                                                                                                                         | <i>Shadow</i> | <i>Read/Write</i> |

Defines the parameters of the object's inner shadow. See class *Shadow* in this documentation to see details of the shadow parameters.

|                                                                                                                                                                            |                          |                   |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------|-------------------|
| <b>innerGlow</b>                                                                                                                                                           | <i>InnerGlow</i>         | <i>Read/Write</i> |
| Defines the parameters of the object's inner glow. See class <i>InnerGlow</i> in this documentation to see details of the glow parameters.                                 |                          |                   |
| <b>outerGlow</b>                                                                                                                                                           | <i>OuterGlow</i>         | <i>Read/Write</i> |
| Defines the parameters of the object's outer glow. See class <i>OuterGlow</i> in this documentation to see details of the glow parameters.                                 |                          |                   |
| <b>colorOverlay</b>                                                                                                                                                        | <i>ColorOverlay</i>      | <i>Read/Write</i> |
| <b>reflection</b>                                                                                                                                                          | <i>Reflection</i>        | <i>Read/Write</i> |
| <b>transparency</b>                                                                                                                                                        | <i>Transparency</i>      | <i>Read/Write</i> |
| <b>styleSheet</b>                                                                                                                                                          | <i>GraphicStyleSheet</i> | <i>Read/Write</i> |
| <b>styleSheetByName</b>                                                                                                                                                    | <i>String</i>            | <i>Read/Write</i> |
| Get/Set the style sheet which is applied to define the appearance of the graphic object. The style sheet can be defined by name or by an object <i>GraphicStyleSheet</i> . |                          |                   |

## Class: LineObject

This class defines the features of a line which an author can place on any page. The *LineObject* is derived from the classes *GraphicObject* and *Object*. It means that this kind of object has the same properties and methods as the superclasses and adds line specific properties and methods.

The line can be a simple line, or it can have line ends such as arrows or bullets. It can be a straight line or a curved line, etc. See following properties to see what it is possible to do with this line object.

### Properties:

|                                                                                                                                                                                                                                                                                         |                        |                   |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------|-------------------|
| <b>form</b>                                                                                                                                                                                                                                                                             | <i>LineForm</i>        | <i>Read/Write</i> |
| This property defines the type of the line. See enumeration type <i>LineForm</i> at the bottom of this chapter.                                                                                                                                                                         |                        |                   |
| <b>lineLength</b>                                                                                                                                                                                                                                                                       | <i>UnitValueString</i> | <i>Read/Write</i> |
| Length of the line in points (pt) or millimeters (mm). This property is ignored if the form is <i>LineForm.PolyLine</i> . A polyline (multiple line) is then defined by the <i>properties pageGeometryPoints</i> and <i>spreadGeometryPoints</i> .                                      |                        |                   |
| <b>lineAngle</b>                                                                                                                                                                                                                                                                        | <i>float</i>           | <i>Read/Write</i> |
| Angle of the line in degrees. For a constrained line, the value should be a multiple of 45. This property is ignored if the form is <i>LineForm.PolyLine</i> . A polyline (multiple line) is then defined by the <i>properties pageGeometryPoints</i> and <i>spreadGeometryPoints</i> . |                        |                   |
| <b>lineColor</b>                                                                                                                                                                                                                                                                        | <i>ColorBrush</i>      | <i>Read/Write</i> |
| The line color is defined by <i>ColorBrush</i> . See the chapter "Class: ColorBrush" in this documentation.                                                                                                                                                                             |                        |                   |
| <b>lineShade</b>                                                                                                                                                                                                                                                                        | <i>float</i>           | <i>Read/Write</i> |

Defines the shade of the line color as a percentage. The maximum value is 100.0 (100%). The value 0 means that the line is white, but not transparent. The value -1 means that the line is fully transparent.

| <b>lineOpacity</b> | <i>float</i> | <i>Read/Write</i> |
|--------------------|--------------|-------------------|
|--------------------|--------------|-------------------|

Defines the line opacity. The maximum value 100.0 (i.e. 100%) means that the line has no translucence/opacity. The value 0 means that the line is fully transparent.

| <b>lineWidth</b> | <i>UnitValueString</i> | <i>Read/Write</i> |
|------------------|------------------------|-------------------|
|------------------|------------------------|-------------------|

Defines line width.

| <b>lineStyle</b> | <i>String</i> | <i>Read/Write</i> |
|------------------|---------------|-------------------|
|------------------|---------------|-------------------|

Defines the style of the line. Available values are: "Solid" (solid line), "Pattern 1" (dashed), "Pattern 2" (dashed with shorter gaps), "Pattern 3" (dashdot line), "Pattern 4" (dotted, squared dots), "Pattern 5" (dashed with rounded ends of dashes), "Pattern 6" (dashed with shorter gaps and rounded ends of dashes), "Pattern 7" (dashdot line rounded ends of dashes), "Pattern 8" (dotted, circular dots), "Pattern 9" (dashed with short dashes), "Pattern 10" (dashdot line with short dashes), etc. ... up to "Pattern 19" as combination of longer or shorter dashes, circular or squared dots.

| <b>pageGeometryPoints</b> | <i>PointList</i> | <i>Read/Write</i> |
|---------------------------|------------------|-------------------|
|---------------------------|------------------|-------------------|

This property is applied if the form property is *LineForm.PolyLine*. It is used to place the end points of polyline (multiple line) segments. See the chapter Point and PointList to see how to work with the points.

| <b>spreadGeometryPoints</b> | <i>PointList</i> | <i>Read/Write</i> |
|-----------------------------|------------------|-------------------|
|-----------------------------|------------------|-------------------|

This property is applied if the form property is *LineForm.PolyLine*. It is used to place the end points of poly-line (multiple line) segments. See the chapter Point and PointList to see how to work with the points.

| <b>lineStart</b> | <i>LineCap</i> | <i>Read/Write</i> |
|------------------|----------------|-------------------|
|------------------|----------------|-------------------|

The start of the line may have a normal end or can be terminated by an arrowhead, an arrow end/tail or a bullet point. See the available values of the enumeration type *LineCap*.

| <b>lineEnd</b> | <i>LineCap</i> | <i>Read/Write</i> |
|----------------|----------------|-------------------|
|----------------|----------------|-------------------|

The end of the line may have a normal end or can be terminated by an arrowhead, an arrow end/tail or a bullet point. See the available values of the enumeration type *LineCap*.

## Methods:

| <i>remove()</i> | <i>bool</i> |
|-----------------|-------------|
|-----------------|-------------|

Removes the object from the appropriate page and destroys it. Returns true in case of success.

## Examples:

```
var doc = application.currentDocument;
if (doc) {
    // Create simple line
    var line1 = doc.objects().create(ObjectType.Line);
    if (line1) {
        with(line1) {
            name = "Line 1";
            // Normal simple line
            form = LineForm.FreeLine;
        }
    }
}
```

```
    pagex = "20mm";
    pagey = "200.5pt";
    lineLength = "170pt";
    lineAngle = 30;
    lineWidth = "10pt";
    // Dashed line
    lineStyle = "Pattern 2";
    // Find (create if not found) a blue brush
    var brushBlue = doc.brushes.find("Blue");
    if( brushBlue == null ) {
        brushBlue = doc.brushes.addColor("Blue");
        brushBlue.rgb = new Rgb(0, 0, 255);
    }
    // Set the line color
    lineColor = brushBlue;
    // Set opacity of the line to be by 50% transparent
    lineOpacity = 50.0;
}
// Add the line to a current page
doc.currentPage.add(line1);
}
// Create poly line
var line2 = doc.objects().create(ObjectType.Line);
if (line2) {
    with(line2) {
        name = "Line 2";

        // Poly-line
        form = LineForm.PolyLine;
        // Assign array of points
        // a) simple points => the poly-line consists of 3 straight
        // lines
        pageGeometryPoints = [[193.58, 360.119], [192.638, 247.249],
            [280.689, 350.379], [310.689, 99.379]];

        // b) 2nd point is created by control and sizing points =>
        // the final poly-line consists of curved lines
        pageGeometryPoints = [[193.58, 360.119],
            [150.600, 200.00, "B"], [196.638, 255.249],
            [210.600, 400.00, "A"], [280.689, 350.379],
            [310.689, 99.379]];

        // Dotted line, 3mm wide
        lineStyle = "Pattern 8";
        lineWidth = "3mm";
    }
    // Add the line to a current page
    doc.currentPage.add(line2);
}
}
```

---

## Class: *FrameObject*

The class *FrameObject* is another subclass of *GraphicObject*. The basic feature of the frame object is that it can have a rectangular, oval or polygonal shape, can be framed and filled by a color or brush.

Optionally the frame object can be filled by a text or picture. If the frame object is rectangular, the content may also be a table.

The class *FrameObject* is a superclass for *TextObject*, *PictureObject* and *TableObject*.

### Properties:

|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                        |                   |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------|-------------------|
| <b>width</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | <i>UnitValueString</i> | <i>Read/Write</i> |
| Defines the width of the frame object.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                        |                   |
| <b>height</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | <i>UnitValueString</i> | <i>Read/Write</i> |
| Defines the height of the frame object.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |                        |                   |
| <b>form</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | <i>FrameForm</i>       | <i>Read/Write</i> |
| Defines the shape of the object. The value is one of the items of the enumeration type <i>FrameForm</i> . A description of this can be found at the end of this chapter.                                                                                                                                                                                                                                                                                                                                                                                                                                             |                        |                   |
| <b>pageGeometryPoints</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | <i>PointList</i>       | <i>Read/Write</i> |
| The property is taken into consideration if the form property is <i>FrameForm.Polygon</i> . It is used to place the end points of polygon segments.                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |                        |                   |
| <b>spreadGeometryPoints</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | <i>PointList</i>       | <i>Read/Write</i> |
| The property is taken into consideration if the form property is <i>FrameForm.Polygon</i> . It is used to place the end points of polygon segments.                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |                        |                   |
| <b>frameWidth</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | <i>UnitValueString</i> | <i>Read/Write</i> |
| Defines the frame width (stroke).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |                        |                   |
| <b>frameStyle</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | <i>String</i>          | <i>Read/Write</i> |
| Defines the line style of the frame. Available values are: "Solid" (solid line), "Pattern 1" (dashed), "Pattern 2" (dashed with shorter gaps), "Pattern 3" (dashdot line), "Pattern 4" (dotted, squared dots), "Pattern 5" (dashed with rounded ends of dashes), "Pattern 6" (dashed with shorter gaps and rounded ends of dashes), "Pattern 7" (dashdot line rounded ends of dashes), "Pattern 8" (dotted, circular dots), "Pattern 9" (dashed with short dashes), "Pattern 10" (dashdot line with short dashes), etc. ... up to "Pattern 19" as combination of longer or shorter dashes, circular or squared dots. |                        |                   |
| <b>frameColor</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | <i>ColorBrush</i>      | <i>Read/Write</i> |
| The line/frame color is defined by the term <i>ColorBrush</i> . See the chapter "Class: <i>ColorBrush</i> " in this documentation.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                        |                   |
| <b>frameShade</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | <i>float</i>           | <i>Read/Write</i> |
| Defines the line/frame color shade as a percentage. The maximum value is 100.0 (100%). The value 0 means that the line is white, but not transparent. The value -1 means that the line/frame is fully transparent.                                                                                                                                                                                                                                                                                                                                                                                                   |                        |                   |
| <b>frameOpacity</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | <i>float</i>           | <i>Read/Write</i> |

Defines the opacity of the line/frame. The value 100.0 (i.e. 100%) means that the frame has no translucence/opacity. The value 0 means that the frame is fully transparent.

---

|                    |                        |                   |
|--------------------|------------------------|-------------------|
| <b>frameRadius</b> | <i>UnitValueString</i> | <i>Read/Write</i> |
|--------------------|------------------------|-------------------|

Defines the radius of this rounding. A rectangular frame object can have rounded corners.

---

|                  |              |                   |
|------------------|--------------|-------------------|
| <b>fillColor</b> | <i>Brush</i> | <i>Read/Write</i> |
|------------------|--------------|-------------------|

The frame object is filled by using the term *Brush*. Examples are shown below.

---

|                  |              |                   |
|------------------|--------------|-------------------|
| <b>fillShade</b> | <i>float</i> | <i>Read/Write</i> |
|------------------|--------------|-------------------|

Defines the shade of the fill color. The maximum value is 100.0 (i.e. 100%). The value 0 means that the fill is white, but not transparent. The value -1 means that the fill is fully transparent.

---

|                    |              |                   |
|--------------------|--------------|-------------------|
| <b>fillOpacity</b> | <i>float</i> | <i>Read/Write</i> |
|--------------------|--------------|-------------------|

Defines the opacity of the fill color. The value 100.0 (i.e. 100%) means that the fill has no translucence/opacity. The value 0 means that the fill is fully transparent.

---

|                       |              |                   |
|-----------------------|--------------|-------------------|
| <b>fillBlendAngle</b> | <i>float</i> | <i>Read/Write</i> |
|-----------------------|--------------|-------------------|

Defines the angle of the color blend, if the frame is filled by a color blend instead of a solid color.

---

|                      |              |                   |
|----------------------|--------------|-------------------|
| <b>objectOpacity</b> | <i>float</i> | <i>Read/Write</i> |
|----------------------|--------------|-------------------|

Defines the overall opacity of the frame object. The value 100.0 (i.e. 100%) means that the object (fill and frame) has no translucence/opacity. The value 0 means that the object (fill and frame) is fully transparent.

#### Methods:

---

|                 |             |
|-----------------|-------------|
| <i>remove()</i> | <i>bool</i> |
|-----------------|-------------|

Removes the object from the appropriate page and destroys it. Returns true in case of success.

#### Examples:

```
var doc = application.currentDocument;
if (doc) {
    // Create simple frame object
    var frmO = doc.objects().create(ObjectType.Graphic);
    if (frmO) {
        with(frmO) {
            name = "Frame Object 1";
            // Rectangular frame object
            form = FrameForm.Rectangle;
            // position and size of the frame object
            pageGeometryBounds = [50.15, 201.65, 200, 300];

            // Rounded corners (2.0 mm)
            cornerRadius = 2.0;

            // Find (create if not found) a blue brush for border
            var brushBlue = doc.brushes.find("Blue");
            if( brushBlue == null ) {
                brushBlue = doc.brushes.addColor("Blue");
                brushBlue.rgb = new Rgb(0, 0, 255);
            }
        }
    }
}
```

```
// Set style of the border: color, line width, pattern
lineColor = brushBlue;
lineWidth = "3pt";
lineStyle = "Pattern 4";

// Find (create if not found) a red brush for interior
var brushRed = doc.brushes.find("Red");
if( brushRed == null ) {
    brushRed = doc.brushes.addColor("Red");
    brushRed.rgb = new Rgb(255, 0, 0);
}

// Set color of the frame interior
fillColor = brushRed;
fillShade = 100.0;
// Set overall opacity of the frame object to 50%
baseOpacity = 50.0;
}
// Add the frame object to a current page
doc.currentPage.add(frmo);
}
```

## Class: TextObject

The TextObject represents an area that is filled by formatted text content. The formatted text can contain not only text, but also pictures and tables. The content of the text object is accessible via its property content. The position, size, shape, border, background and other features of the text object are defined by the values of properties which are derived from the class FrameObject. See the description of the FrameObject in the chapter "Class: FrameObject". The position is defined by the properties pagex, pagey, spreadx and spready (class Object), the size by width and height (class FrameObject).

### Properties:

| <b>content</b>                                                                                                                                                                                                                                                                                                                            | <i>TextContent</i> | <i>Read</i>       |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------|-------------------|
| Returns the content of the text object. The TextContent object holds the settings of the text area (columns, column lines, indents, baseline grid, header and footer); the text itself with all character and paragraph formats is available via content.formattedText. See the chapters "Class: TextContent" and "Class: FormattedText". |                    |                   |
| <b>individualAliasContent</b>                                                                                                                                                                                                                                                                                                             | <i>bool</i>        | <i>Read/Write</i> |
| If true, the alias objects of this object (e.g. on document pages that use the Alias page) can have individual content. If false, they show the content of the original.                                                                                                                                                                  |                    |                   |
| <b>contentOpacity</b>                                                                                                                                                                                                                                                                                                                     | <i>float</i>       | <i>Read/Write</i> |
| <b>textChain</b>                                                                                                                                                                                                                                                                                                                          | <i>TextChain</i>   | <i>Read</i>       |
| The property returns the textchain of texts.                                                                                                                                                                                                                                                                                              |                    |                   |

**Methods:****connectTo** (*TextObject object*)      *bool*

Links the text object to the text chain of the given text object. Returns true in case of success.

**remove()**      *bool*

Removes the object from the appropriate page and destroys it. Returns true in case of success.

**Examples:**

```
var doc = application.currentDocument;
if (doc) {
    // Create new text object
    var objT = doc.objects().create(ObjectType.Text);
    if (objT) {
        with (objT) {
            // Configure the shape to be an ellipse
            form = FrameForm.Ellipse;

            // Set the position (x,y) and the width and height
            pagex = "25mm";
            pagey = "45.5pt";
            width = "150mm";
            height = "120mm";

            // Position and size can also be set using the property
            pageGeometryBounds
            //   pageGeometryBounds = ["25mm", "45.5pt", "80mm", "120mm"];
            // Create red border, yellow interior and set opacity
            lineWidth = "1.5mm";
            lineColor = Color.Red;
            fillColor = Color.Yellow;
            fillShade = 50;
            fillOpacity = 70;
        }
    }
    // Insert the text object on a current page
    doc.currentPage.add(objT);

    // Access the content of the text object
    with (objT.content) {
        // The text content will be arranged in 2 columns
        columnCount = 2;
        columnDistance = "10mm";
        // Set content position inside the text object
        leftIndent = "5mm";
        topIndent = "25mm";
        rightIndent = "5mm";
        bottomIndent = "35mm";

        // Add plain text to the content object
        insertPlainText("AABBBCCC", 10);
    }
}
```

```
// Get the object which processes text formatting
// inside the text object content.
var ft = formattedText;
// Insert new text at the 1st position before
// previously inserted plain text.
ft.insertPlainText("This is a sample formatted text! ", 0);
ft.setFontSize(20, 2, 5);

// Create a picture object
var objPicture = doc.objects().create(ObjectType.Picture);
// Configure the size and content
objPicture.width = "20mm";
objPicture.height = "40mm";
objPicture.content.importPicture("anypicture.jpg", 1);
// Insert the picture into the formatted text
if(objPicture)
    ft.setObject(objPicture, 3);

// Create a table object and insert it also to the text
var objTable = doc.objects().create(ObjectType.Table);
if(objTable)
    ft.setObject(objTable, 10);
}
```

## Class: TextContent

The class *TextContent* describes the content of a text object (see class *TextObject*). The text object consists of a frame (closed) object (see class *FrameObject*) and text content. The frame object implements the area for the content and can be a rectangle/square, ellipse/circle or polygon/bézier object, can be filled with any fill color or blend and can be surrounded by any border/ frame. The text content of the text object is implemented by the class *TextContent*.

### Properties:

|                                                                                                                                    |                        |                   |
|------------------------------------------------------------------------------------------------------------------------------------|------------------------|-------------------|
| <b>columnCount</b>                                                                                                                 | <i>int</i>             | <i>Read/Write</i> |
| Defines the number of columns in a text object. The default value is 1.                                                            |                        |                   |
| <b>columnOffset</b>                                                                                                                | <i>UnitValueString</i> | <i>Read/Write</i> |
| Defines the gap or gutter between columns. The former name columnDistance still works.                                             |                        |                   |
| <b>leftIndent</b>                                                                                                                  | <i>UnitValueString</i> | <i>Read/Write</i> |
| <b>topIndent</b>                                                                                                                   | <i>UnitValueString</i> | <i>Read/Write</i> |
| <b>rightIndent</b>                                                                                                                 | <i>UnitValueString</i> | <i>Read/Write</i> |
| <b>bottomIndent</b>                                                                                                                | <i>UnitValueString</i> | <i>Read/Write</i> |
| Define left, top, right and bottom distance between the text and a “virtual” rectangle surrounding whole frame of the text object. |                        |                   |
| <b>columnLines</b>                                                                                                                 | <i>ColumnLines</i>     | <i>Read/Write</i> |
| The property sets the columnLines parameters.                                                                                      |                        |                   |

|                                                                                                                                                                                                           |                         |                   |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------|-------------------|
| <b>textRange</b>                                                                                                                                                                                          | <i>TextRange</i>        | <i>Read</i>       |
| Use this method to get the parameters of the text inside the appropriate text object. See class <i>TextRange</i> to see all available parameters for the text.                                            |                         |                   |
| <b>formattedText</b>                                                                                                                                                                                      | <i>FormattedText</i>    | <i>Read</i>       |
| Get an object which allows you to read and manipulate the formatted text inside the text object. See class <i>FormattedText</i> to see all text properties and methods.                                   |                         |                   |
| <b>baselinePosition</b>                                                                                                                                                                                   | <i>BaselinePosition</i> | <i>Read/Write</i> |
| Set/get the <i>BaselinePosition</i> object. See Class: BaselinePosition.                                                                                                                                  |                         |                   |
| <b>baselineDescenders</b>                                                                                                                                                                                 | <i>bool</i>             | <i>Read/Write</i> |
| Descenders are the part of a character that hang down below the baseline, such as “p” or “y”. When this function is activated the last line of the text may now fit in the object when it did not before. |                         |                   |
| <b>linkableWithOriginal</b>                                                                                                                                                                               | <i>bool</i>             | <i>Read/Write</i> |
| <b>baselineGrid</b>                                                                                                                                                                                       | <i>BaselineGrid</i>     | <i>Read/Write</i> |
| <b>header</b>                                                                                                                                                                                             | <i>HeaderFooter</i>     | <i>Read/Write</i> |
| <b>footer</b>                                                                                                                                                                                             | <i>HeaderFooter</i>     | <i>Read/Write</i> |
| <b>footnotes</b>                                                                                                                                                                                          | <i>GraphicFootnotes</i> | <i>Read/Write</i> |
| <b>headerFormattedText</b>                                                                                                                                                                                | <i>FormattedText</i>    | <i>Read</i>       |
| <b>footerFormattedText</b>                                                                                                                                                                                | <i>FormattedText</i>    | <i>Read</i>       |

**Methods:**

**plainText** (*TextRange textRange*) *String*

Get simple plain text from the text object. The text doesn't contain any formatting parameters or any object that was inserted in the text.

*textRange*: The parameter is used to define the beginning and end of the requested text.

Returns the required text as a String.

**plainText** (*int begin, int end*) *String*

Get simple plain text from the text object. The text doesn't contain any formatting parameters or any object that was inserted inside the text.

*Begin*: Beginning of the requested text. Value is 0-based, i.e. first letter has index 0.

*end*: Position behind the last character of requested text.

Returns the requested text as a String.

**insertPlainText** (*String text, int index = 0*) *void*

Insert plain text to the requested position in the existing text.

*text*: Inserted text.

*index*: Index of the character before which the inserted text is placed. The index refers to the whole text of the text chain.

|                                                      |                   |
|------------------------------------------------------|-------------------|
| <b>getAttributes</b> ()                              | <i>Attributes</i> |
| <b>setAttributes</b> ( <i>Attributes attrs</i> )     | <i>void</i>       |
| <b>copyAttributes</b> ( <i>TextContent fromObj</i> ) | <i>bool</i>       |

**Examples:**

```

var doc = application.currentDocument;
if (doc) {
    // Create new text object
    var objT = doc.objects().create(ObjectType.Text);
    if (objT) {
        with (objT) {
            // Configure its shape as a rectangle
            form = FrameForm.Rectangle;
            pageGeometryBounds = ["15mm", "35.5pt", "180mm", "120mm"];
            // Create red border, yellow interior and set opacity
            lineWidth = "1mm";
            lineColor = Color.Red;
        }
    }
    // Insert the text object on a current page
    doc.currentPage.add(objT);

    // Access the content of the text object
    with (objT.content) {
        // Set content position inside the text object
        leftIndent = "5mm";
        topIndent = "25mm";
        rightIndent = "5mm";
        bottomIndent = "15mm";

        // The text content will be arranged as 3 columns with a gutter of 8
mm
        columnCount = 3;
        columnDistance = "8mm";
        // Configure the vertical line separators between the columns
        columnLineColor = Color.Green;
        columnLineShade = 50;
        columnLineWidth = "1.5mm";
        columnLineTopIndent = "4mm";
        columnLineBottomIndent = "4mm";
        columnLineTopAnchor = TextColumnLineAnchor.Baseline;
        columnLineBottomAnchor = TextColumnLineAnchor.ObjectFrame;
        columnLineStyle = "Pattern 8";
        showColumnLines = true;

        // Add plain text to the content object
        insertPlainText("This is plain text.", 10);

        // Get object which processes text formatting
        // inside the text object content.
        var ft = formattedText;

```

```
// Insert a new text at the 1st position before
// previously inserted plain text.
ft.insertPlainText("This is formatted text.\n\n", 0);
ft.setFontSize(20, 8, 22);
// Get plain text from the text object
var txt = plainText(8, 22);
messageBox.information("Text", txt);
}
}
```

## Class: PictureObject

The *PictureObject* represents an object which is filled by a picture. The picture itself is accessible via its property *content*. The position, size, shape, border, background and other features of the picture object are defined by values of properties that are derived from the class *FrameObject*. See the description of the *FrameObject* in the chapter “Class: FrameObject”. The position is defined by the properties *pageX*, *pageY*, *spreadX* and *spreadY* (class *Object*), the size by *width* and *height* (class *FrameObject*).

### Properties:

| <b>content</b>                                                                                                                                                                                                                      | <i>PictureContent</i> | <i>Read</i>       |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------|-------------------|
| This property is the entry point for image properties. See examples in this chapter to see how to use and manipulate the image. Also see the chapter “Class: PictureContent” for detailed information about all picture properties. |                       |                   |
| <b>individualAliasContent</b>                                                                                                                                                                                                       | <i>bool</i>           | <i>Read/Write</i> |
| If true, the alias objects of this object (e.g. on document pages that use the Alias page) can have individual content. If false, they show the content of the original.                                                            |                       |                   |
| <b>contentOpacity</b>                                                                                                                                                                                                               | <i>float</i>          | <i>Read/Write</i> |
| Set/Get the opacity of the imported picture in percent                                                                                                                                                                              |                       |                   |

### Methods:

| <b>remove()</b>                                                                                | <i>bool</i> |
|------------------------------------------------------------------------------------------------|-------------|
| Removes the object from the appropriate page and destroys it. Returns true in case of success. |             |

### Examples:

```
var doc = application.currentDocument;
if (doc) {
    var objPic = doc.objects().create(ObjectType.Picture);
    if (objPic) {
        with (objPic) {
            // Set rectangular shape of the picture object
            form = FrameForm.Rectangle;
            // Set the position (x,y) and the width and height
            pageGeometryBounds = ["35mm", "220.2pt", "130mm", "85mm"];
            // Rotate the object by 15 degrees
            rotation = 15.0;

            // Import a picture to the picture object
        }
    }
}
```

```

content.importPicture("c:\\picture.pdf");
// Read or manipulate the picture using property "content"
with (content) {
    opacity = 80;
    rotation = 10;
    skew = 0;
    verticalScale = 100;
    horizontalScale = 150;
    verticalMirrored = false;
    horizontalMirrored = false;
    // set the picture coordinates
    horizontalOffset = "10mm";
    verticalOffset = "25.5pt";
    // following properties are for reading only
    filePath;
    resolutionX;
    resolutionY;
    width;
    height;
}
}
doc.currentPage.add(objPic);
}
}

```

## Class: PictureContent

The class *PictureContent* describes the content of the picture object (see the class *PictureObject*).

### Properties:

| <b>filePath</b>                                                                                    | <i>String</i>          | <i>Read</i>       |
|----------------------------------------------------------------------------------------------------|------------------------|-------------------|
| Returns the full path of the imported image to the object or <i>null</i> if picture doesn't exist. |                        |                   |
| <b>horizontalOffset</b>                                                                            | <i>UnitValueString</i> | <i>Read/Write</i> |
| Set/get the X position of the top-left corner of the imported image.                               |                        |                   |
| <b>verticalOffset</b>                                                                              | <i>UnitValueString</i> | <i>Read/Write</i> |
| Set/get the Y position of the top-left corner of the imported image.                               |                        |                   |
| <b>width</b>                                                                                       | <i>float</i>           | <i>Read</i>       |
| Get the width of the imported image in points [pt].                                                |                        |                   |
| <b>height</b>                                                                                      | <i>float</i>           | <i>Read</i>       |
| Get the height of the imported image in points [pt].                                               |                        |                   |
| <b>resolutionX</b>                                                                                 | <i>float</i>           | <i>Read</i>       |
| Get the X resolution of the imported image in ppi. (points per inch)                               |                        |                   |
| <b>resolutionY</b>                                                                                 | <i>float</i>           | <i>Read</i>       |
| Get the Y resolution of the imported image in ppi. (points per inch)                               |                        |                   |

## Object Types and Structure | Class: PictureContent

|                                                                                                                                                                            |                          |                   |                      |                                        |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------|-------------------|----------------------|----------------------------------------|
| <b>verticalScale</b>                                                                                                                                                       | <i>float</i>             | <i>Read/Write</i> |                      |                                        |
| Set/get the vertical scale of the imported image in percent. [%]                                                                                                           |                          |                   |                      |                                        |
| <b>horizontalScale</b>                                                                                                                                                     | <i>float</i>             | <i>Read/Write</i> |                      |                                        |
| Set/get the horizontal scale of the imported imageimage in percent. [%]                                                                                                    |                          |                   |                      |                                        |
| <b>rotation</b>                                                                                                                                                            | <i>float</i>             | <i>Read/Write</i> |                      |                                        |
| Set/get the rotation of the imported image in degree at interval (-360,360).                                                                                               |                          |                   |                      |                                        |
| <b>skew</b>                                                                                                                                                                | <i>float</i>             | <i>Read/Write</i> |                      |                                        |
| Set/get the skew of the imported image in degree at interval (-75,75).                                                                                                     |                          |                   |                      |                                        |
| <b>verticalMirrored</b>                                                                                                                                                    | <i>bool</i>              | <i>Read/Write</i> |                      |                                        |
| Set/get a flag if the imported image is mirrored vertically.                                                                                                               |                          |                   |                      |                                        |
| <b>horizontalMirrored</b>                                                                                                                                                  | <i>bool</i>              | <i>Read/Write</i> |                      |                                        |
| Set/get a flag if the imported image is mirrored horizontally.                                                                                                             |                          |                   |                      |                                        |
| <b>clippingEnabled</b>                                                                                                                                                     | <i>bool</i>              | <i>Read/Write</i> |                      |                                        |
| Set/get a flag if the imported image can be clipped if clipping for the imported picture is activated.                                                                     |                          |                   |                      |                                        |
| <b>clippingTight</b>                                                                                                                                                       | <i>bool</i>              | <i>Read</i>       |                      |                                        |
| Switches on clipping tightness for the imported image.                                                                                                                     |                          |                   |                      |                                        |
| <b>clippingTolerance</b>                                                                                                                                                   | <i>int</i>               | <i>Read</i>       |                      |                                        |
| Returns the clipping tolerance (0–255) of the imported image. It is set with setClipping().                                                                                |                          |                   |                      |                                        |
| <b>colorMode</b>                                                                                                                                                           | <i>ColorMode</i>         | <i>Read/Write</i> | <i>value limits:</i> | <i>enumeration type</i>                |
| Set/get the color mode.                                                                                                                                                    |                          |                   |                      |                                        |
| <b>colorSpace</b>                                                                                                                                                          | <i>ColorSpace</i>        | <i>Read</i>       |                      |                                        |
| Get the color space.                                                                                                                                                       |                          |                   |                      |                                        |
| <b>halftoneAngle</b>                                                                                                                                                       | <i>float</i>             | <i>Read/Write</i> | <i>unit: degree</i>  | <i>value limits: &lt;-360, 360&gt;</i> |
| Set/get the halftone angle.                                                                                                                                                |                          |                   |                      |                                        |
| <b>halftoneScreen</b>                                                                                                                                                      | <i>float</i>             | <i>Read/Write</i> | <i>unit: lpi</i>     | <i>value limits: &lt;0,20000&gt;</i>   |
| Set/get the halftone screen.                                                                                                                                               |                          |                   |                      |                                        |
| <b>styleSheet</b>                                                                                                                                                          | <i>PictureStyleSheet</i> | <i>Read/Write</i> |                      |                                        |
| <b>styleSheetByName</b>                                                                                                                                                    | <i>String</i>            | <i>Read/Write</i> |                      |                                        |
| Set/get the style sheet which is applied to define the appearance of the picture object. The style sheet can be defined by name or by an object <i>PictureStyleSheet</i> . |                          |                   |                      |                                        |
| <b>isEmpty</b>                                                                                                                                                             | <i>bool</i>              | <i>Read</i>       |                      |                                        |
| <b>isEmbedded</b>                                                                                                                                                          | <i>bool</i>              | <i>Read</i>       |                      |                                        |
| <b>qrId</b>                                                                                                                                                                | <i>String</i>            | <i>Read</i>       |                      |                                        |

ID of the picture object for `application.setQrCodeToImage()`.

| <b>image</b> | <i>Image</i> | <i>Read/Write</i> |
|--------------|--------------|-------------------|
|--------------|--------------|-------------------|

Set/get the picture as an Image object. See the chapter “Class: Image”.

| <b>preview</b> | <i>Image</i> | <i>Read</i> |
|----------------|--------------|-------------|
|----------------|--------------|-------------|

Returns the preview of the picture as an Image object.

### Methods:

| <b>importPicture</b> ( <i>String filePath, int index = 0</i> ) | <i>bool</i> |
|----------------------------------------------------------------|-------------|
|----------------------------------------------------------------|-------------|

Import the image to the object from the file *filePath*. The index parameter is the page number (zero based) in the imported PDF document.

Returns TRUE if the image is imported successfully.

| <b>setClipping</b> ( <i>int tolerance, bool tight</i> ) | <i>void</i> |
|---------------------------------------------------------|-------------|
|---------------------------------------------------------|-------------|

Set the parameters *tolerance* and *tight* to clip the imported image.

The parameter *tolerance* is in intervals of <0,255>

| <b>movePicture</b> ( <i>PictureHorizontalPosition hor, PictureVerticalPosition ver</i> ) | <i>void</i> |
|------------------------------------------------------------------------------------------|-------------|
|------------------------------------------------------------------------------------------|-------------|

Set horizontal and vertical parameters for moving the image.

| <b>scalePicture</b> ( <i>ScalePictureType type, bool center = true</i> ) | <i>void</i> |
|--------------------------------------------------------------------------|-------------|
|--------------------------------------------------------------------------|-------------|

Set parameters *type* and *center* flag for moving the image.

| <b>optimizePicture</b> ( <i>OptimizePictureSettings settings</i> ) | <i>bool</i> |
|--------------------------------------------------------------------|-------------|
|--------------------------------------------------------------------|-------------|

Optimize the image. Returns *TRUE* if success.

| <b>changeColorSpace</b> ( <i>ChangePictureColorSpaceSettings settings</i> ) | <i>bool</i> |
|-----------------------------------------------------------------------------|-------------|
|-----------------------------------------------------------------------------|-------------|

Change the picture color space.

| <b>getAttributes</b> () | <i>Attributes</i> |
|-------------------------|-------------------|
|-------------------------|-------------------|

| <b>setAttributes</b> ( <i>Attributes attrs</i> ) | <i>void</i> |
|--------------------------------------------------|-------------|
|--------------------------------------------------|-------------|

| <b>copyAttributes</b> ( <i>PictureContent fromObj</i> ) | <i>bool</i> |
|---------------------------------------------------------|-------------|
|---------------------------------------------------------|-------------|

| <b>copyContentFrom</b> ( <i>PictureContent fromObj</i> ) | <i>bool</i> |
|----------------------------------------------------------|-------------|
|----------------------------------------------------------|-------------|

Copies the picture from another picture content. Returns true in case of success.

| <b>saveToFile</b> ( <i>String fullPath, String format = null, int quality = -1</i> ) | <i>void</i> |
|--------------------------------------------------------------------------------------|-------------|
|--------------------------------------------------------------------------------------|-------------|

Saves the picture to a file. Without *format*, the format is taken from the file extension.

| <b>imageData</b> ( <i>String format, int quality = -1</i> ) | <i>String</i> |
|-------------------------------------------------------------|---------------|
|-------------------------------------------------------------|---------------|

Returns the picture as Base64 encoded image data in the given format (e.g. “PNG”).

| <b>setImageData</b> ( <i>String data, String format = null</i> ) | <i>void</i> |
|------------------------------------------------------------------|-------------|
|------------------------------------------------------------------|-------------|

Replaces the picture with Base64 encoded image data.

## Class: TextChains

The class holds list of TextChain objects.

### Properties:

**length** *int* *Read*

### Methods:

**at** (*int index*) *TextChain*

### Example:

```
var chns = doc.textChains(WhichSpreads.All, WhichObjects.All);
if (chns)
{
    doc.textChains(WhichSpreads.All,
WhichObjects.All).at(0).text.appendPlainText("Text");
    var text = doc.textChains(WhichSpreads.All,
WhichObjects.All).at(0).text.plainText();

    // or ...

    for (var i=0; i<chns.length; i++)
    {
        var chn = chns.at(0);
        var formatText = chn.text;
        var plainText = formatText.plainText();

        // ...
    }
}
```

## Class: TextChain

The class “TextChain” holds a list of text objects that are linked and contain one text that continues to the next and following text objects.

### Properties:

---

|                  |            |             |
|------------------|------------|-------------|
| <b>areaCount</b> | <i>int</i> | <i>Read</i> |
|------------------|------------|-------------|

Returns the number of chained text objects.

---

|             |                      |             |
|-------------|----------------------|-------------|
| <b>text</b> | <i>FormattedText</i> | <i>Read</i> |
|-------------|----------------------|-------------|

Returns the object FormattedText with the whole text that is placed in linked objects.

### Methods:

---

**areaAt** (*int index*) *TextContent*

Returns the text area (TextContent) from its index position in the text chain.

---

**areaIndex** (*TextContent area*) *int*

Returns an index of the TextContent area in the text chain.

---

**equals** (*TextChain obj*) *bool*

---

Returns **TRUE** if the text chains are identical. They have same ID.

## Class: TableObject

The class **TableObject** implements a table which is drawn into the frame object. The position, size, shape, border, background and other features of the table object are defined by the values of properties derived from the class **FrameObject**. See description in the chapter “Class: FrameObject”. The position is defined by the properties `pagex`, `pagey`, `spreadx` and `spready` (class **Object**), the size by width and height (class **FrameObject**).

### Properties:

|                                                                                                                                                                                                                                                                                                                                                                         |                        |                   |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------|-------------------|
| <b>tableHeight</b>                                                                                                                                                                                                                                                                                                                                                      | <i>UnitValueString</i> | <i>Read/Write</i> |
| Defines table object height.                                                                                                                                                                                                                                                                                                                                            |                        |                   |
| <b>tableWidth</b>                                                                                                                                                                                                                                                                                                                                                       | <i>UnitValueString</i> | <i>Read/Write</i> |
| Defines table object width.                                                                                                                                                                                                                                                                                                                                             |                        |                   |
| <b>tableWidthType</b>                                                                                                                                                                                                                                                                                                                                                   | <i>WidthType</i>       | <i>Read/Write</i> |
| Defines how the width of the table object is calculated. Permitted values are specified by the enumeration type <i>WidthType</i> . See the description at the end of this chapter.                                                                                                                                                                                      |                        |                   |
| <b>rowCount</b>                                                                                                                                                                                                                                                                                                                                                         | <i>int</i>             | <i>Read/Write</i> |
| Set/get the number of table rows.                                                                                                                                                                                                                                                                                                                                       |                        |                   |
| <b>columnCount</b>                                                                                                                                                                                                                                                                                                                                                      | <i>int</i>             | <i>Read/Write</i> |
| Set/get the number of table columns.                                                                                                                                                                                                                                                                                                                                    |                        |                   |
| <b>horizontalSeparatorAbove</b>                                                                                                                                                                                                                                                                                                                                         | <i>bool</i>            | <i>Read/Write</i> |
| Sets the horizontal separator lines over the vertical separator lines and is the default setting for all new table objects. It only makes sense to use it when the color of the horizontal separator lines is different from the color of the vertical separator lines. The vertical separator is set over the horizontal separator when this property is set to false. |                        |                   |
| <b>styleSheet</b>                                                                                                                                                                                                                                                                                                                                                       | <i>TableStyleSheet</i> | <i>Read/Write</i> |
| <b>styleSheetByName</b>                                                                                                                                                                                                                                                                                                                                                 | <i>String</i>          | <i>Read/Write</i> |
| Set/get the style sheet that is applied to define the appearance of the table object. The style sheet can be defined by name or by an object <i>TableStyleSheet</i> .                                                                                                                                                                                                   |                        |                   |

### Methods:

|                                                                                                                        |                    |
|------------------------------------------------------------------------------------------------------------------------|--------------------|
| <b>column</b> ( <i>int index</i> )                                                                                     | <i>TableColumn</i> |
| Returns an object that represents the whole table column.                                                              |                    |
| <i>index</i> : Index of the required column. The value must be in the range from 0 to ( <code>columnCount</code> – 1). |                    |
| Returns a <i>TableColumn</i> object or <b>null</b> if the column doesn't exist.                                        |                    |
| <b>row</b> ( <i>int index</i> )                                                                                        | <i>TableRow</i>    |
| Returns an object that represents the whole table row.                                                                 |                    |

**index:** Index of the required row. The value must be in the range from 0 to (rowCount – 1).

Returns a TableRow object or *null* if the row doesn't exist.

---

**cell** (*int row, int column*)      *TableCell*

Returns an object that represents one particular table cell.

**row:** Index of a row in which the required cell is located. The value must be in the range from 0 to (rowCount – 1).

**column:** Index of a column in which the required cell is located. The value must be in the range from 0 to (columnCount – 1).

Returns a TableCell object or *null* if the cell doesn't exist.

---

**separator** (*int row, int column, TableSeparatorPosition position*)      *TableSeparator*

Returns selected table separator. i.e. parameters of a line of the appropriate cell frame. See the specification of table separator in chapter "Class: TableSeparator".

**row:** Row index of the required table cell.

**column:** Column index of the required table cell.

**position:** Parameter defines which part of the cell border the function must return.

Returns the required TableSeparator object or *null* if the separator is not found.

---

**setFillStyle** (*Brush brush, float shade, float opacity, float blendAngle*) *void*

Defines the fill parameters of the table.

**brush:** Color of the table object fill.

**shade:** The parameter defines the color shade of the table fill color. The maximum value is 100.0 (100%). The value 0 means that the table fill is white, but not transparent. The value -1 means that the table fill is fully transparent.

**opacity:** Defines the opacity of the table fill color. The value 100.0 (i.e. 100%) means that the fill has no translucence/opacity. The value 0 means that the fill is fully transparent.

**blendAngle:** If the table object is filled by a color blend instead of a solid color, this parameter defines the angle of the color blend.

---

**remove()**      *boolean*

Removes table object.

---

### Examples:

```
var doc = application.currentDocument;
if (doc) {
    // Create and name new table object
    var oTbl = doc.objects().create(ObjectType.Table);
    oTbl.name = "Table 1";
    // Set number of rows and columns
    oTbl.rowCount = 5;
    oTbl.columnCount = 3;
    // Set position and size of the table
    oTbl.tableWidthType = WidthType.Manual;
```

```
oTbl.pagex = "10mm";
oTbl.pagey = "10mm";
oTbl.width = "120mm";
oTbl.height = "150mm";
// Colorize interior of the table
oTbl.brush = Color.Green;
oTbl.shade = 20;

// Add the table to a current page
doc.currentPage.add(oTbl);

// Get a table cell and change its color
var tblCell = oTbl.cell(1, 1);
if( tblCell ) {
    tblCell.brush = Color.Red;
    tblCell.shade = 70;
}

// Retrieve content type of the cell
if (tblCell.contentType == ContentType.Custom)
    messageBox.information("Content Type", "Custom");
if (tblCell.contentType == ContentType.Text)
    messageBox.information("Content Type", "Text");
if (tblCell.contentType == ContentType.Picture)
    messageBox.information("Content Type", "Picture");
if (tblCell.contentType == ContentType.Table)
    messageBox.information("Content Type", "Table");
}
```

## Class: TableRow

The class **TableRow** represents the settings for one table row. The row settings take effect on all the table cells in the table row, unless an individual cell or cells is/are assigned attributes that take priority over the row settings, such as fill color or shade.

### Properties:

|                                                                                                                                                   |                        |                   |
|---------------------------------------------------------------------------------------------------------------------------------------------------|------------------------|-------------------|
| <b>height</b>                                                                                                                                     | <i>UnitValueString</i> | <i>Read/Write</i> |
| Defines the table row height. See the property heightType to see how this property value is interpreted.                                          |                        |                   |
| <b>heightType</b>                                                                                                                                 | <i>HeightType</i>      | <i>Read/Write</i> |
| Defines how to interpret value in property height. See the enumeration type <i>HeightType</i> at the end of this chapter to see available values. |                        |                   |
| <b>blendAngle</b>                                                                                                                                 | <i>float</i>           | <i>Read/Write</i> |
| Defines the angle of the color blend if the table row is filled by a color blend instead of a solid color,                                        |                        |                   |
| <b>shade</b>                                                                                                                                      | <i>float</i>           | <i>Read/Write</i> |

Defines the shade of the table row fill color. The maximum value is 100.0 (i.e. 100%). The value 0 means that the table row is white, but not transparent. The value -1 means that the table row is fully transparent. If the row fill color is a blend instead of a solid color, then only the values 100% or Transparent (-1) are relevant.

|              |              |                   |
|--------------|--------------|-------------------|
| <b>brush</b> | <i>Brush</i> | <i>Read/Write</i> |
|--------------|--------------|-------------------|

The table row color is defined by the term *Brush*. See the chapter “Class: Brush” in this documentation.

|                         |               |                   |
|-------------------------|---------------|-------------------|
| <b>styleSheetByName</b> | <i>String</i> | <i>Read/Write</i> |
|-------------------------|---------------|-------------------|

|                   |                           |                   |
|-------------------|---------------------------|-------------------|
| <b>styleSheet</b> | <i>TableRowStyleSheet</i> | <i>Read/Write</i> |
|-------------------|---------------------------|-------------------|

Set/get the style sheet applied to define the appearance of the table row object. The style sheet can be defined by name or by an object *TableRowStyleSheet*.

### Methods:

**setFillStyle** (*Brush brush, float density, float opacity, float blendAngle*) *void*

Defines the filling parameters of the table row.

*brush*: Color of the table row.

*shade*: The parameter defines the shade of the table row fill color. The maximum value is 100.0 (100%). The value 0 means that the table row is white, but not transparent. The value -1 means that the table row is fully transparent. If the row fill color is a blend instead of a solid color, then only the values 100% or Transparent (-1) are relevant.

*opacity*: The parameter defines the opacity of the table row. The value 100.0 (i.e. 100%) means that the row has no translucence/opacity. The value 0 means that the table row is fully transparent.

*blendAngle*: Defines the angle of the color blend pf the table row is filled by a color blend instead of a solid color.

|                                                   |                   |
|---------------------------------------------------|-------------------|
| <b>getAttributes</b> ()                           | <i>Attributes</i> |
| <b>setAttributes</b> ( <i>Attributes attrs</i> )  | <i>void</i>       |
| <b>copyAttributes</b> ( <i>TableRow fromObj</i> ) | <i>bool</i>       |

### Examples:

See examples in the chapters “Class: TableObject” and “Class: TableCell”.

## Class: TableColumn

The class *TableColumn* represents the settings for one table column. The column settings take effect on all the table cells in one table column unless an individual cell or cells is/are assigned attributes that take priority over the column settings, such as fill color or shade..

### Properties:

|              |                        |                   |
|--------------|------------------------|-------------------|
| <b>width</b> | <i>UnitValueString</i> | <i>Read/Write</i> |
|--------------|------------------------|-------------------|

Defines the table column width. See the property *widthType* to see how this property value is interpreted.

|                  |                  |                   |
|------------------|------------------|-------------------|
| <b>widthType</b> | <i>WidthType</i> | <i>Read/Write</i> |
|------------------|------------------|-------------------|

Defines the interpretation of the value in the property width. See the enumeration type *WidthType* at the end of this chapter to see available values for this property.

|                   |              |                   |
|-------------------|--------------|-------------------|
| <b>blendAngle</b> | <i>float</i> | <i>Read/Write</i> |
|-------------------|--------------|-------------------|

Defines the angle of the color blend if the table column is filled by a color blend instead of a solid color.

|              |              |                   |
|--------------|--------------|-------------------|
| <b>shade</b> | <i>float</i> | <i>Read/Write</i> |
|--------------|--------------|-------------------|

Defines the shade of the table column fill color. The maximum value is 100.0 (100%). The value 0 means that the table column is white, but not transparent. The value -1 means that the table column is fully transparent. If the column fill color is a blend instead of a solid color, then only the values 100% or Transparent (-1) are relevant.

|              |              |                   |
|--------------|--------------|-------------------|
| <b>brush</b> | <i>Brush</i> | <i>Read/Write</i> |
|--------------|--------------|-------------------|

The table column color is defined by the term *Brush*. See the chapter “Class: Brush” in this documentation.

|                         |               |                   |
|-------------------------|---------------|-------------------|
| <b>styleSheetByName</b> | <i>String</i> | <i>Read/Write</i> |
|-------------------------|---------------|-------------------|

|                   |                              |                   |
|-------------------|------------------------------|-------------------|
| <b>styleSheet</b> | <i>TableColumnStyleSheet</i> | <i>Read/Write</i> |
|-------------------|------------------------------|-------------------|

Set/get style sheet that is applied to define the appearance of the table column object. The style sheet can be defined by name or by an object *TableColumnStyleSheet*.

### Methods:

|                                                                                            |             |
|--------------------------------------------------------------------------------------------|-------------|
| <b>setFillStyle</b> ( <i>Brush brush, float density, float opacity, float blendAngle</i> ) | <i>void</i> |
|--------------------------------------------------------------------------------------------|-------------|

Sets filling parameters of the table column.

*brush*: Color of the table column.

*shade*: The parameter defines the color shade of the table column fill color. The maximum value is 100.0 (100%). The value 0 means that the table column is white, but not transparent. The value -1 means that the table column is fully transparent.

*opacity*: The parameter defines the opacity of the table column. The value 100.0 (i.e. 100%) means that the column has no translucence/opacity. The value 0 means that the table column is fully transparent. If the column fill color is a blend instead of a solid color, then only the values 100% or Transparent (-1) are relevant.

*blendAngle*: Defines the angle of the color blend if the table column is filled by a color blend instead of solid color.

|                         |                   |
|-------------------------|-------------------|
| <b>getAttributes</b> () | <i>Attributes</i> |
|-------------------------|-------------------|

|                                                  |             |
|--------------------------------------------------|-------------|
| <b>setAttributes</b> ( <i>Attributes attrs</i> ) | <i>void</i> |
|--------------------------------------------------|-------------|

|                                                      |             |
|------------------------------------------------------|-------------|
| <b>copyAttributes</b> ( <i>TableColumn fromObj</i> ) | <i>bool</i> |
|------------------------------------------------------|-------------|

### Examples:

See examples in chapters “Class: TableObject” and “Class: TableCell”.

## Class: TableCell

The class *TableCell* represents one particular table cell. The cell is accessed in a table via row and column index using method `cell(row, column)`. Each cell can contain another table or text or picture, or may have the content setting “None”. Cell properties such as fill color take preference over the fill color properties of columns and rows.

### Properties:

|                                                                                                                                                                                                                                                                                                                                                   |                            |                   |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------|-------------------|
| <b>blendAngle</b>                                                                                                                                                                                                                                                                                                                                 | <i>float</i>               | <i>Read/Write</i> |
| Defines the angle of the color blend if the table cell is filled by a color blend instead of solid color.                                                                                                                                                                                                                                         |                            |                   |
| <b>shade</b>                                                                                                                                                                                                                                                                                                                                      | <i>float</i>               | <i>Read/Write</i> |
| Defines the shade of the table cell fill color. The maximum value is 100.0 (i.e. 100%). The value 0 means that the table cell is white, but not transparent. The value -1 means that the table cell is fully transparent. If the cell fill color is a blend instead of a solid color, then only the values 100% or Transparent (-1) are relevant. |                            |                   |
| <b>brush</b>                                                                                                                                                                                                                                                                                                                                      | <i>Brush</i>               | <i>Read/Write</i> |
| The table cell color is defined by the term <i>Brush</i> . See the chapter “Class: Brush” in this documentation.                                                                                                                                                                                                                                  |                            |                   |
| <b>leftIndent</b>                                                                                                                                                                                                                                                                                                                                 | <i>UnitValueString</i>     | <i>Read/Write</i> |
| <b>topIndent</b>                                                                                                                                                                                                                                                                                                                                  | <i>UnitValueString</i>     | <i>Read/Write</i> |
| <b>rightIndent</b>                                                                                                                                                                                                                                                                                                                                | <i>UnitValueString</i>     | <i>Read/Write</i> |
| <b>bottomIndent</b>                                                                                                                                                                                                                                                                                                                               | <i>UnitValueString</i>     | <i>Read/Write</i> |
| Define the indent of the cell content from the left, top, right and bottom boundaries of the table cell.                                                                                                                                                                                                                                          |                            |                   |
| <b>horizontalAlignment</b>                                                                                                                                                                                                                                                                                                                        | <i>HorizontalAlignment</i> | <i>Read/Write</i> |
| Horizontal alignment of an object inside the table cell when the cell content is “None” and an object is imported into the cell. See enumeration type <i>HorizontalAlignment</i> at the end of this chapter to see all allowed values for this property.                                                                                          |                            |                   |
| <b>verticalAlignment</b>                                                                                                                                                                                                                                                                                                                          | <i>VerticalAlignment</i>   | <i>Read/Write</i> |
| Vertical alignment of an object inside the table cell when the cell content is “None” and an object is imported into the cell. See enumeration type <i>VerticalAlignment</i> at the end of this chapter to see all allowed values for this property.                                                                                              |                            |                   |
| <b>contentRotation</b>                                                                                                                                                                                                                                                                                                                            | <i>ContentRotation</i>     | <i>Read/Write</i> |
| The cell content can be rotated by 0°, 90°, 180° or 270°. This property defines the rotation angle and applies to all cell content including imported objects when the cell content is “None” and an object is imported into the cell. See enumeration type <i>ContentRotation</i> to see available values for this property.                     |                            |                   |
| <b>rowSpan</b>                                                                                                                                                                                                                                                                                                                                    | <i>int</i>                 | <i>Read/Write</i> |
| Any table cell can be merged with one or more adjacent cells in one table column. The property <i>rowSpan</i> defines the number of merged cells, i.e. number of rows from which the cells are merged to one cell. A single cell has <i>rowSpan</i> = 1. In the VivaDesigner Table Dialog, the option is “Extend Cell”.                           |                            |                   |
| <b>columnSpan</b>                                                                                                                                                                                                                                                                                                                                 | <i>int</i>                 | <i>Read/Write</i> |

Any table cell can be merged with one or more adjacent cells in one table row. The property `columnSpan` specifies number of merged cells, i.e. number of columns, from which the cells are merged to one cell. A single cell has `columnSpan = 1`. In the VivaDesigner Table Dialog, the option is “Extend Cell”.

|                                                                                                                                                                                                                                                                                                       |                            |                   |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------|-------------------|
| <b>contentType</b>                                                                                                                                                                                                                                                                                    | <i>ContentType</i>         | <i>Read</i>       |
| Defines the cell content type. See enumeration type <i>ContentType</i> at the end of this chapter to see allowed content types.                                                                                                                                                                       |                            |                   |
| <b>contentObject</b>                                                                                                                                                                                                                                                                                  | <i>Object</i>              | <i>Read/Write</i> |
| Set/get the content object of the table cell. It can be a picture object, text object or another table object. The cell content type is not assigned to the inserted object.                                                                                                                          |                            |                   |
| <b>contentExpandableObject</b>                                                                                                                                                                                                                                                                        | <i>Object</i>              | <i>Write</i>      |
| Set/get the content object of the table cell. It can be a picture object, text object or another table object. The cell content type is assigned to the inserted object and the object parameters are affected. See the property <code>contentType</code> to see details about expected content type. |                            |                   |
| <b>row</b>                                                                                                                                                                                                                                                                                            | <i>int</i>                 | <i>Read</i>       |
| <b>column</b>                                                                                                                                                                                                                                                                                         | <i>int</i>                 | <i>Read</i>       |
| Row and column index of the cell.                                                                                                                                                                                                                                                                     |                            |                   |
| <b>valid</b>                                                                                                                                                                                                                                                                                          | <i>bool</i>                | <i>Read</i>       |
| Returns false if the cell object does not refer to an existing cell.                                                                                                                                                                                                                                  |                            |                   |
| <b>styleSheetByName</b>                                                                                                                                                                                                                                                                               | <i>String</i>              | <i>Read/Write</i> |
| <b>styleSheet</b>                                                                                                                                                                                                                                                                                     | <i>TableCellStyleSheet</i> | <i>Read/Write</i> |
| Set/get the style sheet that is applied to define the appearance of the table cell object. The style sheet can be defined by name or by an object <i>TableCellStyleSheet</i> .                                                                                                                        |                            |                   |

#### Methods:

|                                                                                                                                                                                                                                                                                                                                                                  |             |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------|
| <b>removeContent()</b>                                                                                                                                                                                                                                                                                                                                           | <i>void</i> |
| Resets the table cell content.                                                                                                                                                                                                                                                                                                                                   |             |
| <b>setFillStyle(<i>Brush brush, float shade, float opacity, float blendAngle</i>)</b>                                                                                                                                                                                                                                                                            | <i>void</i> |
| Sets the cell filling parameters.                                                                                                                                                                                                                                                                                                                                |             |
| <i>brush</i> : Table cell fill color.                                                                                                                                                                                                                                                                                                                            |             |
| <i>shade</i> : Defines the shade of the table cell fill color. The maximum value is 100.0 (i.e. 100%). The value 0 means that the table cell is white, but not transparent. The value -1 means that the table cell is fully transparent. If the cell fill color is a blend instead of a solid color, then only the values 100% or Transparent (-1) are relevant. |             |
| <i>opacity</i> : Defines the table cell opacity. The value 100.0 (i.e. 100%) means that the cell has no translucence/opacity. The value 0 means that the cell is fully transparent.                                                                                                                                                                              |             |
| <i>blendAngle</i> : Defines the angle of the color blend if the table cell is filled by a color blend instead of solid color.                                                                                                                                                                                                                                    |             |

---

**setIndents** (*UnitValueString left, UnitValueString top, UnitValueString right, UnitValueString bottom*) *void*

Sets the indent of the content from left, top, right and bottom boundaries of the table cell.

---

**getAttributes** () *Attributes*

---

**setAttributes** (*Attributes attrs*) *void*

---

**copyAttributes** (*TableCell fromObj*) *bool*

### Examples:

```
var doc = application.currentDocument;
if (doc) {
    // Create example table
    var oTbl = doc.objects().create(ObjectType.Table);
    oTbl.name = "Examples: Tables";

    // Define number of rows and columns
    oTbl.rowCount = 4;
    oTbl.columnCount = 3;

    // Size of the table. The width is fixed.
    oTbl.pagex = "10mm";
    oTbl.pagey = "10mm";
    oTbl.tableWidthType = WidthType.Manual;
    oTbl.width = "120mm";
    oTbl.height = "150mm";

    // Insert the table to a current page
    doc.currentPage.add(oTbl);

    // Get table row
    var oTblRow = oTbl.row(2);
    if( oTblRow ) {
        // Minimum height of the row is 20mm
        oTblRow.heightType = HeightType.Minimum;
        oTblRow.height = "20mm";
        // Create a blend brush for the table row.
        // First check if the brush exists. If a brush with existing
        // name is created, an error occurs.
        var brBl = doc.brushes.find("TblRowBrush");
        if( brBl == null ) {
            brBl = doc.brushes.addBlend("TblRowBrush");
            brBl.type = BlendType.Linear;
            brBl.setStopAt(0, new Stop(Color.Red, 70, 40, 0, 85));
            brBl.setStopAt(1, new Stop(Color.White, 100, 55, 100, 23));
            var stop3 = new Stop(Color.Green, 50, 55, 60, 25);
            brBl.appendStop(stop3)
        }
        // Set the brush to the table row.
        oTblRow.brush = brBl;
        // Don't forget to make it visible. By default
        // the background of the row is transparent.
```

```
oTblRow.shade = 100;
}
// Prepare a graphic object for one of the cells
var oFrm = doc.objects().create(ObjectType.Graphic);
with (oFrm) {
    // Ellipse with blue border and yellow interior
    form = FrameForm.Ellipse;
    lineWidth = 8;
    pagex = "0mm";
    pagey = "0mm";
    width = "30mm";
    height = "20mm";
    lineColor = Color.Blue;
    fillColor = Color.Yellow;
    lineShade = 80;
    fillShade = 70;
}
// Get one of the cells and insert the ellipse there
var oTblCell = oTbl.cell(1,0);
if( oTblCell ) {
    oTblCell.contentObject = oFrm;
}
// Prepare the text object for another table cell
var oTxt = doc.objects().create(ObjectType.Text);
with(oTxt) {
    pagex = "25mm";
    pagey = "45.5pt";
    width = "30mm";
    height = "20mm";
    content.insertPlainText("Text in a cell", 0);
}
// Get a table cell
oTblCell = oTbl.cell(1,2);
if( oTblCell ) {
    // Define indents inside the cell
    oTblCell.leftIndent = 10;
    oTblCell.topIndent = 5;
    oTblCell.rightIndent = 10;
    oTblCell.bottomIndent = 5;
    // Rotate the text by 90 degrees
    oTblCell.contentRotation = ContentRotation.Angle90;
    // Insert the text in the cell
    oTblCell.contentObject = oTxt;
}
}
```

## Class: TableSeparator

The class *TableSeparator* describes the parameters of one particular separator line of a table cell. See the method *separator()* of the class *Table*, which returns an object representing the required separator.

**Properties:**

|                                                                                                                                                                                                                    |                   |                   |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|-------------------|
| <b>visible</b>                                                                                                                                                                                                     | <i>bool</i>       | <i>Read/Write</i> |
| Set this property to true to make the separator visible or to false to hide the separator.                                                                                                                         |                   |                   |
| <b>shade</b>                                                                                                                                                                                                       | <i>float</i>      | <i>Read/Write</i> |
| Defines the color shade of the separator color. The maximum value is 100.0 (100%). The value 0 means that the separator is white, but not transparent. The value -1 means that the separator is fully transparent. |                   |                   |
| <b>brush</b>                                                                                                                                                                                                       | <i>ColorBrush</i> | <i>Read/Write</i> |
| The color of the separator is defined by the term <i>ColorBrush</i> . See chapter “Class: ColorBrush” in this documentation.                                                                                       |                   |                   |

**Methods:**

**setFrameStyle** (*ColorBrush brush, float shade, float opacity, String lineStyle, UnitValueString lineWidth*) *void*

This method implements the formatting of the cell separator in a complex way.

*brush*: Defines the the separator color. Blends are not permitted.

*shade*: Defines the shade of the separator color. The maximum value is 100.0 (100%). The value 0 means that the separator is white, but not transparent. The value -1 means that the separator is fully transparent.

*opacity*: The parameter defines the opacity of the separator. The value 100.0 (i.e. 100%) means that the separator has no translucence/opacity. The value 0 means that the separator is fully transparent.

*lineStyle*: The parameter defines the style of the line used to draw the separator. Available values are: “Solid” (solid line), “Pattern 1” (dashed), “Pattern 2” (dashed with shorter gaps), “Pattern 3” (dashdot line), “Pattern 4” (dotted, squared dots), “Pattern 5” (dashed with rounded ends of dashes), “Pattern 6” (dashed with shorter gaps and rounded ends of dashes), “Pattern 7” (dashdot line rounded ends of dashes), “Pattern 8” (dotted, circular dots), “Pattern 9” (dashed with short dashes), “Pattern 10” (dashdot line with short dashes), etc. up to “Pattern 19” as combination of longer or shorter dashes, circular or squared dots.

*lineWidth*: Defines the line width of the separator. Values without a unit are points (pt).

|                                                         |                   |
|---------------------------------------------------------|-------------------|
| <b>getAttributes</b> ()                                 | <i>Attributes</i> |
| <b>setAttributes</b> ( <i>Attributes attrs</i> )        | <i>void</i>       |
| <b>copyAttributes</b> ( <i>TableSeparator fromObj</i> ) | <i>bool</i>       |

**Examples:**

```
var doc = application.currentDocument;
if (doc) {
    // Create example table
    var oTbl = doc.objects().create(ObjectType.Table);
    oTbl.name = "Examples: Tables";
    // Define number of rows and columns
    oTbl.rowCount = 2;
    oTbl.columnCount = 3;
}
```

```
// Size of the table. The width is fixed.
oTbl.pagex = "10mm";
oTbl.pagey = "10mm";
oTbl.tableWidthType = WidthType.Manual;
oTbl.width = "100mm";
oTbl.height = "80mm";

// Insert the table to a current page
doc.currentPage.add(oTbl);

// Make the left cell separator red
var oSep = oTbl.separator(1, 1, TableSeparatorPosition.Left);
oSep.visible = true;
oSep.brush = Color.Red;
oSep.shade = 100;
// Make the right separator blue, thick and using non-solid line
oSep = oTbl.separator(1, 1, TableSeparatorPosition.Right);
oSep.visible = true;
oSep.setFrameStyle(Color.Blue, 80, 50, "Pattern 1", 5);
}
```

## Class: GroupObject

The group object is an invisible object that only groups several page or spread objects and allows some manipulation on all grouped objects, particularly moving, resizing and rotation.

### Properties:

---

|               |            |             |
|---------------|------------|-------------|
| <b>length</b> | <i>int</i> | <i>Read</i> |
|---------------|------------|-------------|

The property contains a number of grouped objects.

### Methods:

---

|                |                                      |                |
|----------------|--------------------------------------|----------------|
| <b>objects</b> | <i>(ObjectType objectType = All)</i> | <i>Objects</i> |
|----------------|--------------------------------------|----------------|

Get the set of objects which are grouped in this group object. See chapter "Class: Objects" for all information about the objects collection class.

**objectType:** See the enumeration ObjectType.

---

|                      |            |
|----------------------|------------|
| <b>objectCount()</b> | <i>int</i> |
|----------------------|------------|

---

|                  |                    |               |
|------------------|--------------------|---------------|
| <b>getObject</b> | <i>(int index)</i> | <i>Object</i> |
|------------------|--------------------|---------------|

---

|                  |             |
|------------------|-------------|
| <b>ungroup()</b> | <i>void</i> |
|------------------|-------------|

Release all objects that are grouped in this group object.

---

|                    |                                             |             |
|--------------------|---------------------------------------------|-------------|
| <b>insertAfter</b> | <i>(Object obj, Object afterObj = null)</i> | <i>void</i> |
|--------------------|---------------------------------------------|-------------|

Insert a new object in the group object.

**obj:** Newly inserted object.

**afterObj:** The object after which the new object is inserted. If afterObj is null, the object is added to the group.

---

**insertAt(Object obj, int pos)**      *void*

Insert a new object in the group object.

*obj*: Newly inserted object.*pos*: Position where the new object is inserted.

---

**add(Object obj)**      *void*

Add a new object to the group object. The new object is appended to the end of the list of previously grouped objects.

*obj*: Newly inserted object.

---

**remove()**      *bool*

Removes group object.

**Examples:**

```
var doc = application.currentDocument;
if (doc) {
    var objs = doc.objects();

    // Create several objects
    var obj1 = objs.create(ObjectType.Text);
    obj1.name = "Obj1";
    obj1.pageGeometryBounds = [20, 20, 100, 100];

    var obj2 = objs.create(ObjectType.Graphic);
    obj2.name = "Obj2";
    obj2.pageGeometryBounds = [60, 60, 150, 150];

    var obj3 = objs.create(ObjectType.Text);
    obj3.name = "Obj3";
    obj3.pageGeometryBounds = [210, 210, 50, 50];

    // Add the objects to the current page
    var page = doc.currentPage;
    page.add(obj1);
    page.add(obj2);
    page.add(obj3);
    messageBox.information("Objects in Page", "Count = " +
page.objects().length
        + "\n(no group exist)");
    // Create group object
    var group = objs.create(ObjectType.Group);
    group.name = "First Group";
    group.pagex = "50mm";
    group.pagey = "150mm";
    page.add(group);

    messageBox.information("Objects in Page", "Count = " +
page.objects().length
        + "\n(including new group)");
    // Add 2 objects to the group
```

---

```
group.add(obj2); // add to the end
group.insertAt(obj3, 0); // insert to 1st position
messageBox.information("Objects in Group", "Count = " + group.length);
}
```

## Class: ObjectAlias

The class *ObjectAlias* represents an alias object: a linked copy of another object (the original). Alias objects are created with *Object > Alias*, with the method *createAlias()*, or automatically for the objects of an assigned Alias page (see “Alias pages and their objects in scripts” in the chapter “Application, Document Structure Objects”). Alias objects can only be rotated, not scaled or skewed; *copy()* and *fullCopy()* return null.

### Properties:

---

| <b>owner</b>                                                                                                          | <i>Object</i> | <i>Read</i> |
|-----------------------------------------------------------------------------------------------------------------------|---------------|-------------|
| Returns the original object of the alias object. For objects from an Alias page, <i>owner.page</i> is the Alias page. |               |             |

### Methods:

---

| <b>remove()</b>                                            | <i>bool</i> |
|------------------------------------------------------------|-------------|
| Removes the alias object. Returns true in case of success. |             |

## Class: TextObjectAlias

The class *TextObjectAlias* extends the root class *ObjectAlias* and adds other properties that are specific for this type of object.

### Properties:

---

| <b>content</b>                                                                                                           | <i>TextContent</i> | <i>Read</i> |
|--------------------------------------------------------------------------------------------------------------------------|--------------------|-------------|
| Content of the original text object (see class <i>TextContent</i> ). The text is read via <i>content.formattedText</i> . |                    |             |

---

| <b>textChain</b>                        | <i>TextChain</i> | <i>Read</i> |
|-----------------------------------------|------------------|-------------|
| Text chain of the original text object. |                  |             |

### Methods:

---

| <b>remove()</b>                                            | <i>bool</i> |
|------------------------------------------------------------|-------------|
| Removes the alias object. Returns true in case of success. |             |

## Class: PictureObjectAlias

The class *PictureObjectAlias* extends the class *ObjectAlias*. It represents an alias object of a picture object.

### Properties:

---

| <b>content</b> | <i>PictureContent</i> | <i>Read</i> |
|----------------|-----------------------|-------------|
|----------------|-----------------------|-------------|

---

Content of the original picture object (see class PictureContent).

**Methods:**

---

**remove()** *bool*

Removes the alias object. Returns true in case of success.







The background features a dynamic, abstract composition of flowing, organic shapes in shades of gold, yellow, and blue against a dark, almost black, backdrop. The shapes appear to be layered and moving, creating a sense of depth and motion. A bright, glowing blue and yellow light source is visible on the right side, casting a soft glow across the scene.

# FormattedText Objects

# FormattedText Objects

## Class: FormattedText

Text with colored letters, with various fonts, with chapters, indents, lists etc. is called “formatted text”. The object of the class *FormattedText* is a repository of formatted text and a tool for manipulating it. The class *FormattedText* is nested in the class *TextObject*. Use the property *TextObject.content* and its property *formattedText* (*textObject.content.formattedText*) to access the text of the text object.

Note the difference between the two levels: *TextObject.content* (class *TextContent*) holds the settings of the text frame itself, such as columns, column gutter and the indents between frame and text. All character and paragraph formats (font, size, color, alignment, paragraph indents, style sheets etc.) are set through *content.formattedText*, i.e. through the methods of this class. *content.formattedText* returns the text of the whole text chain, not only the part that is visible in this text object.

### Properties:

|                                                                                                                   |            |             |
|-------------------------------------------------------------------------------------------------------------------|------------|-------------|
| <b>length</b>                                                                                                     | <i>int</i> | <i>Read</i> |
| The property contains a number of characters, including blank spaces and invisible characters (e.g. line breaks). |            |             |

|                                                      |             |                   |
|------------------------------------------------------|-------------|-------------------|
| <b>notesVisible</b>                                  | <i>bool</i> | <i>Read/Write</i> |
| Use this property to show or hide notes in the text. |             |                   |

### Methods: (text creation)

|                                                         |               |
|---------------------------------------------------------|---------------|
| <b>plainText</b> ( <i>TextRange range</i> )             | <i>String</i> |
| <b>plainText</b> ( <i>int begin=0, int end=length</i> ) | <i>String</i> |

Get the plain text or its fragment without any formatting.

*begin*: Index of first character. The default value is 0.

*end*: Index of the position behind the last requested character. The default value is the length of the text.

*range*: The parameter (its properties *begin* and *end*) defines the fragment.

Returns the required text or its fragment between positions *begin* (including) and *end* (excluding).

|                                                                                                                                         |
|-----------------------------------------------------------------------------------------------------------------------------------------|
| <b>insertPlainText</b> ( <i>String text, int pos = 0, TypographicQuotesMode quotesMode = TypographicQuotesOff</i> )<br><i>TextRange</i> |
|-----------------------------------------------------------------------------------------------------------------------------------------|

Insert new plain text in the formatted text object.

*text*: Inserted text.

*pos*: The position where the text has to be inserted. If not specified the default value 0 is taken.

*quotesMode*: typographic quotes mode

Returns a *TextRange* object which is related to the newly inserted text.

|                                                                                                                   |
|-------------------------------------------------------------------------------------------------------------------|
| <b>appendPlainText</b> ( <i>String text, TypographicQuotesMode mode = TypographicQuotesOff</i> ) <i>TextRange</i> |
|-------------------------------------------------------------------------------------------------------------------|

Append new plain text to the end of the existing formatted text.

*text*: Added text.

Returns a TextRange object which is related to the newly inserted text.

---

**insertHtml**(*String text, int pos=0*) *TextRange*

---

Insert HTML-formatted text in the formatted text object. The HTML formatting is converted to text formats.

*text*: HTML code of the inserted text.

*pos*: The position where the text has to be inserted. The default value is 0.

Returns a TextRange object which is related to the newly inserted text, or *null* if nothing was inserted.

---

**charAsString**(*int pos=0*) *String*

---

**charAsCode** (*int pos=0*) *int*

---

Get the character located at the specified position in the formatted text.

*pos*: Position of the requested character in the formatted text.

Returns the required character. If the parameter *pos* is higher than text length minus 1, the method returns value 0.

---

**insertCharacter**(*Char character, int pos=0, TypographicQuotesMode mode = TypographicQuotesOff*) *void*

---

Insert the character in the text.

*character*: Inserted character. The character can be set as a character ('a') or as a code (0x0061 ).

*pos*: The position where the text has to be inserted. The default value is 0.

---

**object**(*int pos=0*) *Object*

---

Get an object that is inserted to the formatted text at the specified position.

*pos*: Position in the formatted text. The default value is 0.

Returns an object at the specified position or *null* if there is no object at this position.

---

**setObject**(*Object obj, int pos=0*) *void*

---

Insert an object at the specified position in the formatted text.

*obj*: Object that is inserted at the specified position.

*pos*: Target position in the formatted text. The default value is 0.

---

**formattedText**(*int begin=0, int end=length*) *FormattedText*

---

**formattedText**(*TextRange range*) *FormattedText*

---

Get fragment of formatted text.

*begin*: Index of the first letter of the requested text fragment. The default value is 0.

*end*: Index of the position behind the last requested character. The default value is the length of the text.

*range*: The parameter (its properties *begin* and *end*) defines the fragment of affected text.

Returns the specified fragment of the formatted text.

---

**insertFormattedText**(*FormattedText text, int pos=0*) *TextRange*

---

Insert a new fragment of formatted text to this formatted text.

*text*: Inserted formatted text.

*pos*: Target position for the inserted formatted text.

Returns a text range object which represents the inserted text fragment inside the target formatted text.

---

|                                        |                   |
|----------------------------------------|-------------------|
| <b>annotation</b> ( <i>int pos=0</i> ) | <i>Annotation</i> |
|----------------------------------------|-------------------|

---

Get an annotation from a specified position in the formatted text.

*pos*: Position in the formatted text. The default value is 0.

Returns an object of the class *Annotation* if a annotation is found at the specified position. Otherwise it returns *null*. See the chapter “Class: Annotation” for detailed information about annotation object.

---

|                                                                                                  |             |
|--------------------------------------------------------------------------------------------------|-------------|
| <b>insertAnnotation</b> ( <i>String text, int begin=0, int end=len, RubyPosition pos=Above</i> ) | <i>void</i> |
|--------------------------------------------------------------------------------------------------|-------------|

---

---

|                                                                                         |             |
|-----------------------------------------------------------------------------------------|-------------|
| <b>insertAnnotation</b> ( <i>String text, TextRange range, RubyPosition pos=Above</i> ) | <i>void</i> |
|-----------------------------------------------------------------------------------------|-------------|

---

Inserts a new annotation in the formatted text and assign it to the specified text fragment.

*text*: Text of the new annotation.

*begin*: Index of the first letter of the annotated text fragment. The default value is 0.

*end*: Index of the position behind the last character of the annotated text fragment. If it is not specified, the annotation extends up to the last character of the text (excluding it).

*pos*: Position of the inserted annotation. See the enumeration type *RubyPosition* for information about possible annotation positions.

*range*: The parameter (its properties *begin* and *end*) defines the fragment of annotated text.

*pos*: Position of the inserted annotation. See enumeration type *RubyPosition* for information about possible annotation positions.

---

|                                     |                |
|-------------------------------------|----------------|
| <b>comment</b> ( <i>int pos=0</i> ) | <i>Comment</i> |
|-------------------------------------|----------------|

---

Get a comment from specified position of the formatted text.

*pos*: Position in the formatted text. The default value is 0.

Returns an object of the class *Comment*, if any comment is found at the specified position. Otherwise it returns *null*. See the chapter “Class: Comment” for detailed information about comment object.

---

|                                                            |                  |
|------------------------------------------------------------|------------------|
| <b>insertComment</b> ( <i>Comment comment, int pos=0</i> ) | <i>TextRange</i> |
|------------------------------------------------------------|------------------|

---

Insert a new comment in the specified position in the formatted text.

*comment*: New comment that is inserted in the specified position in the formatted text. See the chapter “Class: Comment” for all information for creating a new comment object.

*pos*: Target position for the new inserted comment.

Returns a text range object that describes the new inserted comment inside the target formatted text.

---

|                                  |             |
|----------------------------------|-------------|
| <b>note</b> ( <i>int pos=0</i> ) | <i>Note</i> |
|----------------------------------|-------------|

---

Get a note from specified position of the formatted text.

*pos*: Position in the formatted text. Default value is 0.

Returns an object of the class *Note* if a note is found at the specified position. Otherwise it returns *null*. See chapter “Class: Note” to get detailed information about note object.

---

**insertNote**(*NoteType noteType=Footnote, int pos=0*) *Note*

---

Insert a new note to the specified position in the formatted text.

*noteType*: New note inserted to the specified position in the formatted text. See chapter “Class: Note” to get all information about the creation a new note object.

*pos*: Target position for new inserted note.

---

**positionCorrection**(*int pos=0*) *CharacterPositionCorrection*

---

Get the current value of the character position correction.

*pos*: Position of the inspected character in the formatted text.

Returns an object of the class *CharacterPositionCorrection* that contains the position correction. If no correction is applied, the returned object’s horizontal and vertical correction is set to 0. See the chapter “Class: CharacterPositionCorrection” for information about correction values.

---

**setPositionCorrection**(*CharacterPositionCorrection cor, int pos=0*) *void*

---

Set the small horizontal and vertical shifts of particular characters to correct their position in the text.

*cor*: The parameters define the position correction. See the chapter “Class: CharacterPositionCorrection” for information about setting correction values.

*pos*: Position of the corrected character inside the formatted text.

---

**characterFunction**(*int pos=0*) *CharacterFunction*

---

Get a code that provides information about the function of the character located at the specified position in the formatted text.

*pos*: Position of the inspected character in the formatted text.

Returns one of the values of the enumeration type *CharacterFunction*. See the chapter “Enumeration Objects Index” for information about all available function codes.

---

**setCharacterFunction**(*CharacterFunction function, int pos=0*) *TextRange*

---

Insert a special function character at a specified position in the formatted text.

*function*: Special character code. See the enumeration type *CharacterFunction* in the chapter “Enumeration Objects Index” for complete information about the available function characters.

The value *CharacterFunction.NormalCharacter* doesn’t insert any character. The value is useful only for the function *characterFunction()*.

*pos*: Target position in the formatted text.

Returns a text range object that represents the inserted special character in the target formatted text.

---

**variable**(*int pos=0*) *Variable*

---

Get the variable (e.g. page number, date, running title) located at the specified position in the formatted text.

*pos*: Position of the variable in the formatted text. The default value is 0.

Returns an object of the class Variable. Its property *type* contains the type name of the variable; it is empty if there is no variable at this position.

---

|                                                               |                  |
|---------------------------------------------------------------|------------------|
| <b>insertVariable</b> ( <i>Variable variable, int pos=0</i> ) | <i>TextRange</i> |
|---------------------------------------------------------------|------------------|

---

Insert a variable at the specified position in the formatted text.

*variable*: The variable to be inserted.

*pos*: Target position in the formatted text. The default value is 0.

Returns a text range object that represents the inserted variable.

To insert a variable such as a page number, get an existing variable from a text with *variable(pos)* and insert it with *insertVariable()*.

---

|                                                           |             |
|-----------------------------------------------------------|-------------|
| <b>removeRange</b> ( <i>int begin=0, int end=length</i> ) | <i>void</i> |
|-----------------------------------------------------------|-------------|

---

|                                               |             |
|-----------------------------------------------|-------------|
| <b>removeRange</b> ( <i>TextRange range</i> ) | <i>void</i> |
|-----------------------------------------------|-------------|

---

Remove text or its fragment from the formatted text.

*begin*: Index of first letter of removed text fragment. The default value is 0.

*end*: Index of the position behind the last removed character. The default value is the length of the text.

*range*: The parameter (its properties begin and end) defines the fragment of removed text.

---

|                 |             |
|-----------------|-------------|
| <b>clear</b> () | <i>void</i> |
|-----------------|-------------|

---

Remove the whole text from the text object.

---

|                |                      |
|----------------|----------------------|
| <b>copy</b> () | <i>FormattedText</i> |
|----------------|----------------------|

---

The command creates a new instance of the formatted text.

---

|                                                                                             |                  |
|---------------------------------------------------------------------------------------------|------------------|
| <b>find</b> ( <i>int from, String pattern, bool ignoreCase=true, bool wholeWord=false</i> ) | <i>TextRange</i> |
|---------------------------------------------------------------------------------------------|------------------|

---

Find first occurrence of the text 'text' in the formatted text from a position 'from'. Returns TextRange object.

---

|                                                                            |                  |
|----------------------------------------------------------------------------|------------------|
| <b>findRegEx</b> ( <i>int from, String pattern, bool ignoreCase=true</i> ) | <i>TextRange</i> |
|----------------------------------------------------------------------------|------------------|

---

Find first occurrence of the regular expresion 'pattern' in the formatted text from a position 'from'. Returns TextRange object.

---

|                                                                                                                                        |                  |
|----------------------------------------------------------------------------------------------------------------------------------------|------------------|
| <b>findAndReplace</b> ( <i>int startPosition, String pattern, String replacement, bool ignoreCase = true, bool wholeWord = false</i> ) | <i>TextRange</i> |
|----------------------------------------------------------------------------------------------------------------------------------------|------------------|

---

|                                                                                                                     |                  |
|---------------------------------------------------------------------------------------------------------------------|------------------|
| <b>findAndReplaceRegEx</b> ( <i>int startPosition, String pattern, String replacement, bool ignoreCase = true</i> ) | <i>TextRange</i> |
|---------------------------------------------------------------------------------------------------------------------|------------------|

---

|                                                                                                                                           |             |
|-------------------------------------------------------------------------------------------------------------------------------------------|-------------|
| <b>findAndReplaceAll</b> ( <i>int startPosition, String pattern, String replacement, bool ignoreCase = true, bool wholeWord = false</i> ) | <i>bool</i> |
|-------------------------------------------------------------------------------------------------------------------------------------------|-------------|

---

|                                                                                                                        |             |
|------------------------------------------------------------------------------------------------------------------------|-------------|
| <b>findAndReplaceAllRegEx</b> ( <i>int startPosition, String pattern, String replacement, bool ignoreCase = true</i> ) | <i>bool</i> |
|------------------------------------------------------------------------------------------------------------------------|-------------|

---

Replace all occurrences of the regular expresion 'pattern' by the string 'text'.

---

|                                                                                               |             |
|-----------------------------------------------------------------------------------------------|-------------|
| <b>removeStyleSheets</b> ( <i>int begin=0, int end=length, boolean withAttributes=false</i> ) | <i>void</i> |
|-----------------------------------------------------------------------------------------------|-------------|

---

|                                                                                |             |
|--------------------------------------------------------------------------------|-------------|
| <b>removeStyleSheets</b> ( <i>TextRange range, bool withAttributes=false</i> ) | <i>void</i> |
|--------------------------------------------------------------------------------|-------------|

---

The method removes style sheet setting from the text and converts the style sheets to a normal attribute setting. If the parameter 'withAttribute' set to true, the styles are removed completely without the conversion.

---

|                                                        |            |
|--------------------------------------------------------|------------|
| <b>getAttributes</b> (AttributeGroup group, int pos=0) | Attributes |
|--------------------------------------------------------|------------|

---

Get the attributes of one attribute group at the specified position, e.g. to transfer them to another text fragment with *setAttributes()*.

*group*: AttributeGroup.Character, AttributeGroup.Paragraph or AttributeGroup.Layout.

*pos*: Position in the formatted text. The default value is 0.

---

|                                                          |      |
|----------------------------------------------------------|------|
| <b>setAttributes</b> (Attributes attrs, TextRange range) | void |
|----------------------------------------------------------|------|

---

Apply attributes obtained by *getAttributes()* to the specified text fragment.

*attrs*: Attributes obtained by *getAttributes()*.

*range*: The parameter (its properties begin and end) defines the fragment of affected text.

### Methods (character setting)

All methods that get a text attribute (e.g. *font()*, *textAlignment()*) can be called in four ways: without a parameter the whole text is inspected; with one parameter *pos* the character at this position is inspected; with *begin* and *end*, or with a *TextRange*, the specified fragment is inspected. If the attribute is not the same in the whole inspected text, the method returns *null* and *errorStatus* is set to *IsValid(2)*.

---

|                |          |
|----------------|----------|
| <b>font</b> () | FontName |
|----------------|----------|

---

|                       |          |
|-----------------------|----------|
| <b>font</b> (int pos) | FontName |
|-----------------------|----------|

---

|                                  |          |
|----------------------------------|----------|
| <b>font</b> (int begin, int end) | FontName |
|----------------------------------|----------|

---

|                             |          |
|-----------------------------|----------|
| <b>font</b> (TextRange ran) | FontName |
|-----------------------------|----------|

---

Get the name of the font which is applied to the text at the specified position.

*pos*: Position inside the text.

Returns the name of the applied font or the value *null* when the error is occurred.

---

|                                                                 |      |
|-----------------------------------------------------------------|------|
| <b>setFont</b> (FontName fontName, int begin=0, int end=length) | void |
|-----------------------------------------------------------------|------|

---

|                                                     |      |
|-----------------------------------------------------|------|
| <b>setFont</b> (FontName fontName, TextRange range) | void |
|-----------------------------------------------------|------|

---

Apply a font to specified text fragment.

*fontName*: Name of the font. E.g. "Arial-Regular", "Constantia-Bold".

*begin*: Index of the first letter of the affected text fragment. The default value is 0.

*end*: Index of the position behind the last affected character. The default value is the length of the text.

*range*: The parameter (its properties *begin* and *end*) defines the fragment of affected text.

---

|                                                                        |      |
|------------------------------------------------------------------------|------|
| <b>setFont</b> (String fontFamilyName, String styleName, TextRange tr) | void |
|------------------------------------------------------------------------|------|

---

|                                                                                    |      |
|------------------------------------------------------------------------------------|------|
| <b>setFont</b> (String fontFamilyName, String styleName, int begin=0, int end=len) | void |
|------------------------------------------------------------------------------------|------|

---

Apply a font to a specified text fragment.

*fontFamilyName*: Name of the font family. e.g. "Arial", "Constantia".

*styleName*: Name of the font style. e.g. “Regular”, “Bold”, “Italic”, etc.

*begin*: Index of the first letter of the affected text fragment. The default value is 0.

*end*: Index of the position behind the last affected character. The default value is length of the text.

*range*: The parameter (its properties *begin* and *end*) defines the fragment of affected text.

|                   |                 |
|-------------------|-----------------|
| <b>fontSize()</b> | <i>FontSize</i> |
|-------------------|-----------------|

|                          |                 |
|--------------------------|-----------------|
| <b>fontSize(int pos)</b> | <i>FontSize</i> |
|--------------------------|-----------------|

|                                     |                 |
|-------------------------------------|-----------------|
| <b>fontSize(int begin, int end)</b> | <i>FontSize</i> |
|-------------------------------------|-----------------|

|                                |                 |
|--------------------------------|-----------------|
| <b>fontSize(TextRange ran)</b> | <i>FontSize</i> |
|--------------------------------|-----------------|

Get the font size object as applied to the text at the specified position.

*pos*: Index of the character in the text. The default value is 0.

Returns the object FontSize of the font.

|                                                    |             |
|----------------------------------------------------|-------------|
| <b>setFontSize(FontSize size, TextRange range)</b> | <i>void</i> |
|----------------------------------------------------|-------------|

|                                                                |             |
|----------------------------------------------------------------|-------------|
| <b>setFontSize(FontSize size, int begin=0, int end=length)</b> | <i>void</i> |
|----------------------------------------------------------------|-------------|

Set the font size to be applied to the specified text range.

*size*: Required font object.

*begin*: Index of the first letter of the affected text fragment. The default value is 0.

*end*: Index of the position behind the last affected character. The default value is the length of the text.

*range*: The parameter (its properties *begin* and *end*) defines the fragment of affected text.

|                         |               |
|-------------------------|---------------|
| <b>languageByName()</b> | <i>String</i> |
|-------------------------|---------------|

|                                |               |
|--------------------------------|---------------|
| <b>languageByName(int pos)</b> | <i>String</i> |
|--------------------------------|---------------|

|                                           |               |
|-------------------------------------------|---------------|
| <b>languageByName(int begin, int end)</b> | <i>String</i> |
|-------------------------------------------|---------------|

|                                      |               |
|--------------------------------------|---------------|
| <b>languageByName(TextRange ran)</b> | <i>String</i> |
|--------------------------------------|---------------|

Get the name of the language that is set for the text at a specified position.

*pos*: Position in the text. The default value is 0.

Returns name of the language.

|                                                                        |             |
|------------------------------------------------------------------------|-------------|
| <b>setLanguageByName(String langName, int begin=0, int end=length)</b> | <i>void</i> |
|------------------------------------------------------------------------|-------------|

|                                                            |             |
|------------------------------------------------------------|-------------|
| <b>setLanguageByName(String langName, TextRange range)</b> | <i>void</i> |
|------------------------------------------------------------|-------------|

Set name of language for the specified fragment of the text.

*langName*: Name of selected language.

*begin*: Index of first letter of affected text fragment. The default value is 0.

*end*: Index of the position behind the last affected character. The default value is the length of the text.

*range*: The parameter (its properties *begin* and *end*) defines the fragment of affected text.

|                   |                 |
|-------------------|-----------------|
| <b>language()</b> | <i>Language</i> |
|-------------------|-----------------|

## FormattedText Objects | Class: FormattedText

|                                                                                                                     |                   |
|---------------------------------------------------------------------------------------------------------------------|-------------------|
| <b>language(int pos)</b>                                                                                            | <i>Language</i>   |
| <b>language(int begin, int end)</b>                                                                                 | <i>Language</i>   |
| <b>language(TextRange range)</b>                                                                                    | <i>Language</i>   |
| Get the id of the language that is set for the text at a specified position.                                        |                   |
| <i>pos</i> : Position in the text. The default value is 0.                                                          |                   |
| Returns name of the language.                                                                                       |                   |
| <b>setLanguage(Language lang, int begin=0, int end=length)</b>                                                      | <i>void</i>       |
| <b>setLanguage(Language lang, TextRange range)</b>                                                                  | <i>void</i>       |
| Set id of language for the specified fragment of the text.                                                          |                   |
| <i>lang</i> : ID of selected language.                                                                              |                   |
| <i>begin</i> : Index of first letter of affected text fragment. The default value is 0.                             |                   |
| <i>end</i> : Index of the position behind the last affected character. The default value is the length of the text. |                   |
| <i>range</i> : The parameter (its properties begin and end) defines the fragment of affected text.                  |                   |
| <b>brush()</b>                                                                                                      | <i>ColorBrush</i> |
| <b>brush(int pos)</b>                                                                                               | <i>ColorBrush</i> |
| <b>brush(int begin, int end)</b>                                                                                    | <i>ColorBrush</i> |
| <b>brush(TextRange range)</b>                                                                                       | <i>ColorBrush</i> |
| Get the brush (color) used for the text at the specified position.                                                  |                   |
| <i>pos</i> : Position in the formatted text. The default value is 0.                                                |                   |
| Returns the brush (color) used at the specified position.                                                           |                   |
| <b>setBrush(ColorBrush brush, int begin=0, int end=length)</b>                                                      | <i>void</i>       |
| <b>setBrush(ColorBrush brush, TextRange range)</b>                                                                  | <i>void</i>       |
| <b>setBrush(Color color, int begin=0, int end=length)</b>                                                           | <i>void</i>       |
| <b>setBrush(Color color, TextRange range)</b>                                                                       | <i>void</i>       |
| Apply a new brush (color) to a specified text fragment.                                                             |                   |
| <i>brush</i> : ColorBrush applied to the text fragment.                                                             |                   |
| <i>color</i> : One of the predefined colors of the enumeration type Color, e.g. Color.Blue.                         |                   |
| <i>begin</i> : Index of the first letter of the affected text fragment. The default value is 0.                     |                   |
| <i>end</i> : Index of the position behind the last affected character. The default value is the length of the text. |                   |
| <i>range</i> : The parameter (its properties begin and end) defines the fragment of affected text.                  |                   |
| <b>shade()</b>                                                                                                      | <i>float</i>      |
| <b>shade(int pos)</b>                                                                                               | <i>float</i>      |
| <b>shade(int begin, int end)</b>                                                                                    | <i>float</i>      |

|                                                                                                                                        |                       |
|----------------------------------------------------------------------------------------------------------------------------------------|-----------------------|
| <b>shade(<i>TextRange range</i>)</b>                                                                                                   | <i>float</i>          |
| Get value of shade level at the specified text position.                                                                               |                       |
| <i>pos</i> : Position in the formatted text. The default value is 0.                                                                   |                       |
| Returns level of shade. The value range is 0 – 100.                                                                                    |                       |
| <b>setShade(<i>float shade</i> , <i>int begin=0</i>, <i>int end=length</i>)</b>                                                        | <i>void</i>           |
| <b>setShade(<i>float shade</i>, <i>TextRange range</i>)</b>                                                                            | <i>void</i>           |
| Set level of shade to the specified text fragment.                                                                                     |                       |
| <i>shade</i> : New value of shade level.                                                                                               |                       |
| <i>begin</i> : Index of first letter of affected text fragment. The default value is 0.                                                |                       |
| <i>end</i> : Index of position behind the last affected character. The default value is length of the text.                            |                       |
| <i>range</i> : The parameter (its properties begin and end) defines the fragment of affected text.                                     |                       |
| <b>opacity()</b>                                                                                                                       | <i>float</i>          |
| <b>opacity(<i>int pos</i>)</b>                                                                                                         | <i>float</i>          |
| <b>opacity(<i>int begin</i>, <i>int end</i>)</b>                                                                                       | <i>float</i>          |
| <b>opacity(<i>TextRange range</i>)</b>                                                                                                 | <i>float</i>          |
| Get the opacity level at the specified text position.                                                                                  |                       |
| <i>pos</i> : Position in the formatted text. The default value is 0.                                                                   |                       |
| Returns level of opacity. The value range is 0 – 100.                                                                                  |                       |
| <b>setOpacity(<i>float opacity</i>, <i>int begin=0</i>, <i>int end=length</i>)</b>                                                     | <i>void</i>           |
| <b>setOpacity(<i>float opacity</i>, <i>TextRange range</i>)</b>                                                                        | <i>void</i>           |
| Set the opacity level for the specified text fragment.                                                                                 |                       |
| <i>shade</i> : New value of opacity level.                                                                                             |                       |
| <i>begin</i> : Index of first letter of affected text fragment. The default value is 0.                                                |                       |
| <i>end</i> : Index of the position behind the last affected character. The default value is length of the text.                        |                       |
| <i>range</i> : The parameter (its properties begin and end) defines the fragment of the affected text.                                 |                       |
| <b>outlineOptions()</b>                                                                                                                | <i>OutlineOptions</i> |
| <b>outlineOptions(<i>int pos</i>)</b>                                                                                                  | <i>OutlineOptions</i> |
| <b>outlineOptions(<i>int begin</i>, <i>int end</i>)</b>                                                                                | <i>OutlineOptions</i> |
| <b>outlineOptions(<i>TextRange range</i>)</b>                                                                                          | <i>OutlineOptions</i> |
| Get parameters for drawing the character outline. See the chapter “Class: OutlineOptions” to see all details about outline parameters. |                       |
| <i>pos</i> : Position in the formatted text. The default value is 0.                                                                   |                       |
| Returns an object of the class OutlineOptions with outline parameters.                                                                 |                       |

---

|                                                                                                         |             |
|---------------------------------------------------------------------------------------------------------|-------------|
| <b>setOutlineOptions</b> ( <i>OutlineOptions options</i> , <i>int begin=0</i> , <i>int end=length</i> ) | <i>void</i> |
|---------------------------------------------------------------------------------------------------------|-------------|

---

|                                                                                     |             |
|-------------------------------------------------------------------------------------|-------------|
| <b>setOutlineOptions</b> ( <i>OutlineOptions options</i> , <i>TextRange range</i> ) | <i>void</i> |
|-------------------------------------------------------------------------------------|-------------|

---

Switch the outlining of characters on or off and set the parameters.

*options*: Object containing parameters for outlining. See chapter “Class: OutlineOptions” to see all the available parameters.

*begin*: Index of first letter of affected text fragment. The default value is 0.

*end*: Index of the position behind the last affected character. The default value is length of the text.

*range*: The parameter (its properties *begin* and *end*) defines the fragment of affected text.

---

|                     |                         |
|---------------------|-------------------------|
| <b>underline</b> () | <i>UnderlineOptions</i> |
|---------------------|-------------------------|

---

|                                     |                         |
|-------------------------------------|-------------------------|
| <b>underline</b> ( <i>int pos</i> ) | <i>UnderlineOptions</i> |
|-------------------------------------|-------------------------|

---

|                                                        |                         |
|--------------------------------------------------------|-------------------------|
| <b>underline</b> ( <i>int begin</i> , <i>int end</i> ) | <i>UnderlineOptions</i> |
|--------------------------------------------------------|-------------------------|

---

|                                             |                         |
|---------------------------------------------|-------------------------|
| <b>underline</b> ( <i>TextRange range</i> ) | <i>UnderlineOptions</i> |
|---------------------------------------------|-------------------------|

---

Get parameters of the underline see is applied on the text at the specified position.

*pos*: Position in the formatted text. The default value is 0.

Returns an object of class *UnderlineOptions*. See the chapter “Class: UnderlineOptions” for detailed information about underline parameters.

---

|                                                                                                      |             |
|------------------------------------------------------------------------------------------------------|-------------|
| <b>setUnderline</b> ( <i>UnderlineOptions options</i> , <i>int begin=0</i> , <i>int end=length</i> ) | <i>void</i> |
|------------------------------------------------------------------------------------------------------|-------------|

---

|                                                                                  |             |
|----------------------------------------------------------------------------------|-------------|
| <b>setUnderline</b> ( <i>UnderlineOptions options</i> , <i>TextRange range</i> ) | <i>void</i> |
|----------------------------------------------------------------------------------|-------------|

---

Set the underline.

*options*: Detailed parameters of the underline. See the chapter “Class: UnderlineOptions” for detailed information about underline parameters.

*begin*: Index of first letter of affected text fragment. The default value is 0.

*end*: Index of the position behind the last affected character. The default is the text length.

*range*: The parameter (its properties *begin* and *end*) defines the fragment of affected text.

---

|                         |                             |
|-------------------------|-----------------------------|
| <b>strikethrough</b> () | <i>StrikethroughOptions</i> |
|-------------------------|-----------------------------|

---

|                                         |                             |
|-----------------------------------------|-----------------------------|
| <b>strikethrough</b> ( <i>int pos</i> ) | <i>StrikethroughOptions</i> |
|-----------------------------------------|-----------------------------|

---

|                                                            |                             |
|------------------------------------------------------------|-----------------------------|
| <b>strikethrough</b> ( <i>int begin</i> , <i>int end</i> ) | <i>StrikethroughOptions</i> |
|------------------------------------------------------------|-----------------------------|

---

|                                                 |                             |
|-------------------------------------------------|-----------------------------|
| <b>strikethrough</b> ( <i>TextRange range</i> ) | <i>StrikethroughOptions</i> |
|-------------------------------------------------|-----------------------------|

---

Get the strikethrough options.

*pos*: Position in the formatted text. The default value is 0.

---

|                                                                                                                 |             |
|-----------------------------------------------------------------------------------------------------------------|-------------|
| <b>setStrikethrough</b> ( <i>StrikethroughOptions options</i> ,<br><i>int begin=0</i> , <i>int end=length</i> ) | <i>void</i> |
|-----------------------------------------------------------------------------------------------------------------|-------------|

---

|                                                                                          |             |
|------------------------------------------------------------------------------------------|-------------|
| <b>setStrikethrough</b> ( <i>StrikethroughOptions options</i> , <i>TextRange range</i> ) | <i>void</i> |
|------------------------------------------------------------------------------------------|-------------|

---

Set the strikethrough data.

*options*: Detailed parameters of the strikethrough. See the chapter “Class: StrikethroughOptions” for detailed information about strikethrough parameters.

*begin*: Index of first letter of affected text fragment. The default value is 0.

*end*: Index of the position behind the last affected character. The default is the text length.

*range*: The parameter (its properties begin and end) defines the fragment of affected text.

---

|                            |                          |
|----------------------------|--------------------------|
| <b>characterPosition()</b> | <i>CharacterPosition</i> |
|----------------------------|--------------------------|

---

|                                   |                          |
|-----------------------------------|--------------------------|
| <b>characterPosition(int pos)</b> | <i>CharacterPosition</i> |
|-----------------------------------|--------------------------|

---

|                                              |                          |
|----------------------------------------------|--------------------------|
| <b>characterPosition(int begin, int end)</b> | <i>CharacterPosition</i> |
|----------------------------------------------|--------------------------|

---

|                                     |                          |
|-------------------------------------|--------------------------|
| <b>characterPosition(int pos=0)</b> | <i>CharacterPosition</i> |
|-------------------------------------|--------------------------|

---

Get the current character position, which is used for text formatting at the specified position.

*pos*: Position in the formatted text. The default value is 0.

Returns character position at the specified position. See enumeration type *CharacterPosition* in the chapter “Enumeration Objects Index” to see all available values.

---

|                                                                                             |             |
|---------------------------------------------------------------------------------------------|-------------|
| <b>setCharacterPosition(<i>CharacterPosition</i> type,<br/>int begin=0, int end=length)</b> | <i>void</i> |
|---------------------------------------------------------------------------------------------|-------------|

---

|                                                                                    |             |
|------------------------------------------------------------------------------------|-------------|
| <b>setCharacterPosition(<i>CharacterPosition</i> type, <i>TextRange</i> range)</b> | <i>void</i> |
|------------------------------------------------------------------------------------|-------------|

---

Set new character position of the specified fragment of the formatted text.

*type*: New character position. See enumeration type *CharacterPosition* in the chapter “Enumeration Objects Index” to see all available values.

*begin*: Index of first letter of affected text fragment. The default value is 0.

*end*: Index of the position behind the last affected character. The default is the text length.

*range*: The parameter (its properties begin and end) defines the fragment of affected text.

---

|                                  |                                |
|----------------------------------|--------------------------------|
| <b>characterUpperLowerCase()</b> | <i>CharacterUpperLowerCase</i> |
|----------------------------------|--------------------------------|

---

|                                         |                                |
|-----------------------------------------|--------------------------------|
| <b>characterUpperLowerCase(int pos)</b> | <i>CharacterUpperLowerCase</i> |
|-----------------------------------------|--------------------------------|

---

|                                                    |                                |
|----------------------------------------------------|--------------------------------|
| <b>characterUpperLowerCase(int begin, int end)</b> | <i>CharacterUpperLowerCase</i> |
|----------------------------------------------------|--------------------------------|

---

|                                                        |                                |
|--------------------------------------------------------|--------------------------------|
| <b>characterUpperLowerCase(<i>TextRange</i> range)</b> | <i>CharacterUpperLowerCase</i> |
|--------------------------------------------------------|--------------------------------|

---

Get the current character upper/lower case that is used for text formatting at the specified position.

*pos*: Position in the formatted text. The default value is 0.

Returns the character upper/lower case at the specified position. See the enumeration type *CharacterUpperLowerCase* in the chapter “Enumeration Objects Index” to see all the available values.

---

|                                                                                                |             |
|------------------------------------------------------------------------------------------------|-------------|
| <b>setCharacterUpperLowerCase(<i>CharacterUpperLowerCase</i> type, <i>TextRange</i> range)</b> | <i>void</i> |
|------------------------------------------------------------------------------------------------|-------------|

---

|                                                                                                         |             |
|---------------------------------------------------------------------------------------------------------|-------------|
| <b>setCharacterUpperLowerCase(<i>CharacterUpperLowerCase</i> type, int begin=0,<br/>int end=length)</b> | <i>void</i> |
|---------------------------------------------------------------------------------------------------------|-------------|

---

Set a new character upper/lower case of the specified fragment of the formatted text.

*type*: New character upper/lower case. See the enumeration type `CharacterUpperLowerCase` in the chapter “Enumeration Objects Index” to see all the available values.

*begin*: Index of first letter of affected text fragment. The default value is 0.

*end*: Index of the position behind the last affected character. The default is the text length.

*range*: The parameter (its properties *begin* and *end*) defines the fragment of affected text.

---

|                               |              |
|-------------------------------|--------------|
| <b>characterHeightScale()</b> | <i>float</i> |
|-------------------------------|--------------|

---

|                                      |              |
|--------------------------------------|--------------|
| <b>characterHeightScale(int pos)</b> | <i>float</i> |
|--------------------------------------|--------------|

---

|                                                 |              |
|-------------------------------------------------|--------------|
| <b>characterHeightScale(int begin, int end)</b> | <i>float</i> |
|-------------------------------------------------|--------------|

---

|                                              |              |
|----------------------------------------------|--------------|
| <b>characterHeightScale(TextRange range)</b> | <i>float</i> |
|----------------------------------------------|--------------|

---

Get the character height scale as a percentage of the original height at the specified position of the formatted text.

*pos*: Position in the formatted text. The default value is 0.

Returns percentage value of height scale.

---

|                                                              |             |
|--------------------------------------------------------------|-------------|
| <b>setCharacterHeightScale(float scale, TextRange range)</b> | <i>void</i> |
|--------------------------------------------------------------|-------------|

---

|                                                                          |             |
|--------------------------------------------------------------------------|-------------|
| <b>setCharacterHeightScale(float scale, int begin=0, int end=length)</b> | <i>void</i> |
|--------------------------------------------------------------------------|-------------|

---

Set the character height scale of specified text fragment as a percentage of the original height.

*scale*: New height scale in percent.

*begin*: Index of the first letter of the affected text fragment. The default value is 0.

*end*: Index of the position behind the last affected character. The default value is the length of the text.

*range*: The parameter (its properties *begin* and *end*) defines the fragment of affected text.

---

|                              |              |
|------------------------------|--------------|
| <b>characterWidthScale()</b> | <i>float</i> |
|------------------------------|--------------|

---

|                                     |              |
|-------------------------------------|--------------|
| <b>characterWidthScale(int pos)</b> | <i>float</i> |
|-------------------------------------|--------------|

---

|                                                |              |
|------------------------------------------------|--------------|
| <b>characterWidthScale(int begin, int end)</b> | <i>float</i> |
|------------------------------------------------|--------------|

---

|                                             |              |
|---------------------------------------------|--------------|
| <b>characterWidthScale(TextRange range)</b> | <i>float</i> |
|---------------------------------------------|--------------|

---

Get character width scale as a percentage of the original width at the specified position of the formatted text. The default value is 0.

---

|                                                                         |             |
|-------------------------------------------------------------------------|-------------|
| <b>setCharacterWidthScale(float scale, int begin=0, int end=length)</b> | <i>void</i> |
|-------------------------------------------------------------------------|-------------|

---

|                                                             |             |
|-------------------------------------------------------------|-------------|
| <b>setCharacterWidthScale(float scale, TextRange range)</b> | <i>void</i> |
|-------------------------------------------------------------|-------------|

---

Set character width scale of specified text fragment as a percentage of the original width.

*scale*: New height scale in percent.

*begin*: Index of the first letter of the affected text fragment. The default value is 0.

*end*: Index of the position behind the last affected character. The default value is the length of the text.

*range*: The parameter (its properties *begin* and *end*) defines the fragment of affected text.

---

|                           |              |
|---------------------------|--------------|
| <b>characterSpacing()</b> | <i>float</i> |
|---------------------------|--------------|

---

---

|                                            |              |
|--------------------------------------------|--------------|
| <b>characterSpacing</b> ( <i>int pos</i> ) | <i>float</i> |
|--------------------------------------------|--------------|

---

|                                                       |              |
|-------------------------------------------------------|--------------|
| <b>characterSpacing</b> ( <i>int begin, int end</i> ) | <i>float</i> |
|-------------------------------------------------------|--------------|

---

|                                                    |              |
|----------------------------------------------------|--------------|
| <b>characterSpacing</b> ( <i>TextRange range</i> ) | <i>float</i> |
|----------------------------------------------------|--------------|

---

Get character spacing extension as a percentage of the original spacing. The value 0 (i.e. 0%) means normal spacing. Positive values make character spacing bigger. Negative values reduce the spaces between characters.

*pos*: Position in the formatted text. The default value is 0.

Returns percentage value of current spacing.

---

|                                                                                |             |
|--------------------------------------------------------------------------------|-------------|
| <b>setCharacterSpacing</b> ( <i>float space, int begin=0, int end=length</i> ) | <i>void</i> |
|--------------------------------------------------------------------------------|-------------|

---

|                                                                    |             |
|--------------------------------------------------------------------|-------------|
| <b>setCharacterSpacing</b> ( <i>float space, TextRange range</i> ) | <i>void</i> |
|--------------------------------------------------------------------|-------------|

---

Set new character spacing that is applied to specified text fragment.

*space*: Character spacing in percent. The value 0 (i.e. 0%) means normal spacing. Positive values make character spacing bigger. Negative values reduce the spaces between characters.

*begin*: Index of the first letter of the affected text fragment. The default value is 0.

*end*: Index of the position behind the last affected character. The default value is the length of the text.

*range*: The parameter (its properties *begin* and *end*) defines the fragment of affected text.

---

|                            |                         |
|----------------------------|-------------------------|
| <b>characterKerning</b> () | <i>CharacterKerning</i> |
|----------------------------|-------------------------|

---

|                                            |                         |
|--------------------------------------------|-------------------------|
| <b>characterKerning</b> ( <i>int pos</i> ) | <i>CharacterKerning</i> |
|--------------------------------------------|-------------------------|

---

|                                                       |                         |
|-------------------------------------------------------|-------------------------|
| <b>characterKerning</b> ( <i>int begin, int end</i> ) | <i>CharacterKerning</i> |
|-------------------------------------------------------|-------------------------|

---

|                                                    |                         |
|----------------------------------------------------|-------------------------|
| <b>characterKerning</b> ( <i>TextRange range</i> ) | <i>CharacterKerning</i> |
|----------------------------------------------------|-------------------------|

---

Get character kerning object. The value 0 (i.e. 0%) means normal kerning. Positive values make character kerning bigger. Negative values reduce the spaces between characters.

*pos*: Position in the formatted text. The default value is 0.

Returns the kerning object.

---

|                                                                               |             |
|-------------------------------------------------------------------------------|-------------|
| <b>setCharacterKerning</b> ( <i>CharacterKerning value, TextRange range</i> ) | <i>void</i> |
|-------------------------------------------------------------------------------|-------------|

---

|                                                                                           |             |
|-------------------------------------------------------------------------------------------|-------------|
| <b>setCharacterKerning</b> ( <i>CharacterKerning value, int begin=0, int end=length</i> ) | <i>void</i> |
|-------------------------------------------------------------------------------------------|-------------|

---

Set object of the character kerning for the specified text fragment.

*value*: New kerning object.

*begin*: Index of the first letter of the affected text fragment. The default value is 0.

*end*: Index of the position behind the last affected character. The default value is the length of the text.

*range*: The parameter (its properties *begin* and *end*) defines the fragment of affected text.

---

|                                  |              |
|----------------------------------|--------------|
| <b>characterBaselineShift</b> () | <i>float</i> |
|----------------------------------|--------------|

---

|                                                  |              |
|--------------------------------------------------|--------------|
| <b>characterBaselineShift</b> ( <i>int pos</i> ) | <i>float</i> |
|--------------------------------------------------|--------------|

---

|                                                             |              |
|-------------------------------------------------------------|--------------|
| <b>characterBaselineShift</b> ( <i>int begin, int end</i> ) | <i>float</i> |
|-------------------------------------------------------------|--------------|

---

---

**characterBaselineShift**(*TextRange range*) *float*


---

Get current shift of character baseline at specified text position. Positive value shifts the characters upwards. Negative value means a shift downwards.

*pos*: Position in the formatted text. The default value is 0.

Returns current shift of character baseline in points (pt).

---

**setCharacterBaselineShift**(*float shift*, *int begin=0*, *int end=length*) *void*


---

**setCharacterBaselineShift** (*float shift*, *TextRange range*) *void*


---

Shift character baseline of specified text range upwards or downwards.

*shift*: Baseline shift in points (pt).

*begin*: Index of first letter of affected text fragment. The default value is 0.

*end*: Index of position behind the last affected character. The default value is length of the text. *range*: The parameter (its properties *begin* and *end*) defines the fragment of affected text.

---

**characterBaselineOrientation**() *CharacterBaselineOrientation*


---

**characterBaselineOrientation**(*int pos*) *CharacterBaselineOrientation*


---

**characterBaselineOrientation**(*int begin*, *int end*) *CharacterBaselineOrientation*


---

**characterBaselineOrientation**(*TextRange range*) *CharacterBaselineOrientation*


---

Get current orientation of the character baseline at the specified text position.

*pos*: Position in the formatted text. The default value is 0.

Returns base line orientation. See enumeration type *CharacterBaselineOrientation* in the chapter “Enumeration Objects Index” to see all available values.

---

**setCharacterBaselineOrientation**(*CharacterBaselineOrientation value*,  
*TextRange range*) *void*


---

**setCharacterBaselineOrientation**(*CharacterBaselineOrientation value*,*int begin=0*,  
*int end=length*) *void*


---

Set character baseline orientation of the specified text fragment.

*value*: Required baseline orientation. See the enumeration type *CharacterBaselineOrientation* in the chapter “Enumeration Objects Index” to see all available values.

*begin*: Index of the first letter of the affected text fragment. The default value is 0.

*end*: Index of the position behind the last affected character. The default value is the length of the text.

*range*: The parameter (its properties *begin* and *end*) defines the fragment of affected text.

---

**characterRotation**() *CharacterRotation*


---

**characterRotation**(*int pos*) *CharacterRotation*


---

**characterRotation**(*int begin*, *int end*) *CharacterRotation*


---

**characterRotation**(*TextRange range*) *CharacterRotation*


---

Get current character rotation at the specified text position.

---

*pos*: Position in the formatted text. The default value is 0.

Returns character rotation. See enumeration type *CharacterRotation* in the chapter “Enumeration Objects Index” to see all available values.

---

|                                                                                             |             |
|---------------------------------------------------------------------------------------------|-------------|
| <b>setCharacterRotation</b> ( <i>CharacterRotation value, int begin=0, int end=length</i> ) | <i>void</i> |
|---------------------------------------------------------------------------------------------|-------------|

---

|                                                                                 |             |
|---------------------------------------------------------------------------------|-------------|
| <b>setCharacterRotation</b> ( <i>CharacterRotation value, TextRange range</i> ) | <i>void</i> |
|---------------------------------------------------------------------------------|-------------|

---

Set character rotation of the specified text fragment. Notice that baseline orientation remains unchanged. Only particular characters are rotated.

*value*: New character rotation. See enumeration type *CharacterRotation* in the chapter “Enumeration Objects Index” to see all available values.

*begin*: Index of first letter of affected text fragment. The default value is 0.

*end*: Index of position behind the last affected character. The default value is length of the text.

*range*: The parameter (its properties *begin* and *end*) defines the fragment of affected text.

---

|                              |             |
|------------------------------|-------------|
| <b>characterLigatures</b> () | <i>bool</i> |
|------------------------------|-------------|

---

|                                              |             |
|----------------------------------------------|-------------|
| <b>characterLigatures</b> ( <i>int pos</i> ) | <i>bool</i> |
|----------------------------------------------|-------------|

---

|                                                         |             |
|---------------------------------------------------------|-------------|
| <b>characterLigatures</b> ( <i>int begin, int end</i> ) | <i>bool</i> |
|---------------------------------------------------------|-------------|

---

|                                                      |             |
|------------------------------------------------------|-------------|
| <b>characterLigatures</b> ( <i>TextRange range</i> ) | <i>bool</i> |
|------------------------------------------------------|-------------|

---

Get information whether the ligatures are applied to the characters at the specified position in the text.

*pos*: Position in the formatted text. The default value is 0.

Returns *true* if the ligatures are used in the text. Otherwise it returns *false*.

---

|                                                                                 |             |
|---------------------------------------------------------------------------------|-------------|
| <b>setCharacterLigatures</b> ( <i>bool value, int begin=0, int end=length</i> ) | <i>void</i> |
|---------------------------------------------------------------------------------|-------------|

---

|                                                                     |             |
|---------------------------------------------------------------------|-------------|
| <b>setCharacterLigatures</b> ( <i>bool value, TextRange range</i> ) | <i>void</i> |
|---------------------------------------------------------------------|-------------|

---

Set or reset usage of ligatures to the specified text range.

*value*: Use value *true* or *false* to set respectively reset usage of ligatures in specified text range.

*begin*: Index of first letter of affected text fragment. The default value is 0.

*end*: Index of position behind the last affected character. The default value is length of the text. *range*: The parameter (its properties *begin* and *end*) defines the fragment of affected text.

---

|                            |                    |
|----------------------------|--------------------|
| <b>tablesIndexEntry</b> () | <i>TablesIndex</i> |
|----------------------------|--------------------|

---

|                                            |                    |
|--------------------------------------------|--------------------|
| <b>tablesIndexEntry</b> ( <i>int pos</i> ) | <i>TablesIndex</i> |
|--------------------------------------------|--------------------|

---

|                                                       |                    |
|-------------------------------------------------------|--------------------|
| <b>tablesIndexEntry</b> ( <i>int begin, int end</i> ) | <i>TablesIndex</i> |
|-------------------------------------------------------|--------------------|

---

|                                                    |                    |
|----------------------------------------------------|--------------------|
| <b>tablesIndexEntry</b> ( <i>TextRange range</i> ) | <i>TablesIndex</i> |
|----------------------------------------------------|--------------------|

---

|                                                                                      |             |
|--------------------------------------------------------------------------------------|-------------|
| <b>setTablesIndexEntry</b> ( <i>TablesIndex value, int begin=0, int end=length</i> ) | <i>void</i> |
|--------------------------------------------------------------------------------------|-------------|

---

|                                                                          |             |
|--------------------------------------------------------------------------|-------------|
| <b>setTablesIndexEntry</b> ( <i>TablesIndex value, TextRange range</i> ) | <i>void</i> |
|--------------------------------------------------------------------------|-------------|

---

|                               |                       |
|-------------------------------|-----------------------|
| <b>tablesContentsEntry</b> () | <i>TablesContents</i> |
|-------------------------------|-----------------------|

---

|                                               |                       |
|-----------------------------------------------|-----------------------|
| <b>tablesContentsEntry</b> ( <i>int pos</i> ) | <i>TablesContents</i> |
|-----------------------------------------------|-----------------------|

---

|                                                                                                |                                 |
|------------------------------------------------------------------------------------------------|---------------------------------|
| <code>tablesContentsEntry(int begin, int end)</code>                                           | <code>TablesContents</code>     |
| <code>tablesContentsEntry(TextRange range)</code>                                              | <code>TablesContents</code>     |
| <code>setTablesContentsEntry(TablesContents val, int begin=0, int end=length)</code>           | <code>void</code>               |
| <code>setTablesContentsEntry(TablesContents value, TextRange range)</code>                     | <code>void</code>               |
| <code>tablesBibliographyEntry()</code>                                                         | <code>TablesBibliography</code> |
| <code>tablesBibliographyEntry(int pos)</code>                                                  | <code>TablesBibliography</code> |
| <code>tablesBibliographyEntry(int begin, int end)</code>                                       | <code>TablesBibliography</code> |
| <code>tablesBibliographyEntry(TextRange range)</code>                                          | <code>TablesBibliography</code> |
| <code>setTablesBibliographyEntry(TablesBibliography value, int begin=0, int end=length)</code> | <code>void</code>               |
| <code>setTablesBibliographyEntry(TablesBibliography value, TextRange range)</code>             | <code>void</code>               |
| <code>tablesPictureEntry()</code>                                                              | <code>TablesPicture</code>      |
| <code>tablesPictureEntry(int pos)</code>                                                       | <code>TablesPicture</code>      |
| <code>tablesPictureEntry(int begin, int end)</code>                                            | <code>TablesPicture</code>      |
| <code>tablesPictureEntry(TextRange range)</code>                                               | <code>TablesPicture</code>      |
| <code>setTablesPictureEntry(TablesPicture value, int begin=0, int end=length)</code>           | <code>void</code>               |
| <code>setTablesPictureEntry(TablesPicture value, TextRange range)</code>                       | <code>void</code>               |
| <code>tablesAbbreviationEntry()</code>                                                         | <code>TablesAbbreviation</code> |
| <code>tablesAbbreviationEntry(int pos)</code>                                                  | <code>TablesAbbreviation</code> |
| <code>tablesAbbreviationEntry(int begin, int end)</code>                                       | <code>TablesAbbreviation</code> |
| <code>tablesAbbreviationEntry(TextRange range)</code>                                          | <code>TablesAbbreviation</code> |
| <code>setTablesAbbreviationEntry(TablesAbbreviation value, int begin=0, int end=length)</code> | <code>void</code>               |
| <code>setTablesAbbreviationEntry(TablesAbbreviation value, TextRange range)</code>             | <code>void</code>               |
| <code>tablesRunningTitleEntry()</code>                                                         | <code>TablesRunningTitle</code> |
| <code>tablesRunningTitleEntry(int pos)</code>                                                  | <code>TablesRunningTitle</code> |
| <code>tablesRunningTitleEntry(int begin, int end)</code>                                       | <code>TablesRunningTitle</code> |
| <code>tablesRunningTitleEntry(TextRange range)</code>                                          | <code>TablesRunningTitle</code> |
| <code>setTablesRunningTitleEntry(TablesRunningTitle value, int begin=0, int end=length)</code> | <code>void</code>               |
| <code>setTablesRunningTitleEntry(TablesRunningTitle value, TextRange range)</code>             | <code>void</code>               |
| <code>openType()</code>                                                                        | <code>OpenType</code>           |
| <code>openType(int pos)</code>                                                                 | <code>OpenType</code>           |
| <code>openType(int begin, int end)</code>                                                      | <code>OpenType</code>           |
| <code>openType(TextRange range)</code>                                                         | <code>OpenType</code>           |
| <code>setOpenType(OpenType value, int begin=0, int end=length)</code>                          | <code>void</code>               |

|                                                                  |                            |
|------------------------------------------------------------------|----------------------------|
| <b>setOpenType</b> ( <i>OpenType value, TextRange range</i> )    | <i>void</i>                |
| <b>link</b> ()                                                   | <i>Link</i>                |
| <b>link</b> ( <i>int pos</i> )                                   | <i>Link</i>                |
| <b>link</b> ( <i>int begin, int end</i> )                        | <i>Link</i>                |
| <b>link</b> ( <i>TextRange range</i> )                           | <i>Link</i>                |
| <b>setLink</b> ( <i>Link link, int begin=0, int end=length</i> ) | <i>void</i>                |
| <b>setLink</b> ( <i>Link link, TextRange range</i> )             | <i>void</i>                |
| <b>characterBackground</b> ()                                    | <i>CharacterBackground</i> |
| <b>characterBackground</b> ( <i>int pos</i> )                    | <i>CharacterBackground</i> |
| <b>characterBackground</b> ( <i>int begin, int end</i> )         | <i>CharacterBackground</i> |
| <b>characterBackground</b> ( <i>TextRange range</i> )            | <i>CharacterBackground</i> |

Get parameters that are applied to draw the character background at the specified position.

*pos*: Position in the formatted text. The default value is 0.

Returns an object that keeps the parameters of the background. See the chapter “Class: CharacterBackground” to see all the parameters that can be used to draw the background.

|                                                                                               |             |
|-----------------------------------------------------------------------------------------------|-------------|
| <b>setCharacterBackground</b> ( <i>CharacterBackground chb, int begin=0, int end=length</i> ) | <i>void</i> |
| <b>setCharacterBackground</b> ( <i>CharacterBackground chb, TextRange range</i> )             | <i>void</i> |

Show or hide the character background of the specified text fragment and set its parameters.

*chb*: Parameters of the character background. See the chapter “Class: CharacterBackground” to see all the parameters that can be used to draw the background.

*begin*: Index of first letter of affected text fragment. The default value is 0.

*end*: Index of position behind the last affected character. The default value is length of the text.

*range*: The parameter (its properties *begin* and *end*) defines the fragment of affected text.

|                                                     |                       |
|-----------------------------------------------------|-----------------------|
| <b>characterFrame</b> ()                            | <i>CharacterFrame</i> |
| <b>characterFrame</b> ( <i>int pos</i> )            | <i>CharacterFrame</i> |
| <b>characterFrame</b> ( <i>int begin, int end</i> ) | <i>CharacterFrame</i> |
| <b>characterFrame</b> ( <i>TextRange range</i> )    | <i>CharacterFrame</i> |

Get the parameters that are applied to draw the frame around the text at the specified position.

*pos*: Position in the formatted text. The default value is 0.

Returns an object that keeps the parameters of the text frame. See the chapter “Class: CharacterFrame” to see all the parameters that can be used to draw the text frame.

|                                                                                     |             |
|-------------------------------------------------------------------------------------|-------------|
| <b>setCharacterFrame</b> ( <i>CharacterFrame chb, TextRange range</i> )             | <i>void</i> |
| <b>setCharacterFrame</b> ( <i>CharacterFrame chb, int begin=0, int end=length</i> ) | <i>void</i> |

Show or hide the text frame of the specified text fragment and set its parameters.

*chb*: Parameters of the text frame. See the chapter “Class: CharacterFrame” to see all the parameters that can be used to draw the text frame.

*begin*: Index of first letter of affected text fragment. The default value is 0.

*end*: Index of position behind the last affected character. The default value is length of the text.

*range*: The parameter (its properties *begin* and *end*) defines the fragment of affected text.

---

|                        |                      |
|------------------------|----------------------|
| <b>characterRule()</b> | <i>CharacterRule</i> |
|------------------------|----------------------|

---

|                               |                      |
|-------------------------------|----------------------|
| <b>characterRule(int pos)</b> | <i>CharacterRule</i> |
|-------------------------------|----------------------|

---

|                                          |                      |
|------------------------------------------|----------------------|
| <b>characterRule(int begin, int end)</b> | <i>CharacterRule</i> |
|------------------------------------------|----------------------|

---

|                                 |                      |
|---------------------------------|----------------------|
| <b>characterRule(TextRange)</b> | <i>CharacterRule</i> |
|---------------------------------|----------------------|

---

Get parameters that are applied to draw the character rule at the specified position.

*pos*: Position in the formatted text. The default value is 0.

Returns an object that keeps the parameters of the character rule. See the chapter “Class: CharacterRule” to see all the parameters that can be used to draw the rule.

---

|                                                             |             |
|-------------------------------------------------------------|-------------|
| <b>setCharacterRule(CharacterRule chr, TextRange range)</b> | <i>void</i> |
|-------------------------------------------------------------|-------------|

---

|                                                                         |             |
|-------------------------------------------------------------------------|-------------|
| <b>setCharacterRule(CharacterRule chr, int begin=0, int end=length)</b> | <i>void</i> |
|-------------------------------------------------------------------------|-------------|

---

Show or hide the character rule in the specified text fragment and set its parameters.

*chr*: Parameters of the character rule. See the chapter “Class: CharacterRule” to see all particular parameters that can be used to draw the rule.

*begin*: Index of first letter of affected text fragment. The default value is 0.

*end*: Index of position behind the last affected character. The default value is length of the text.

*range*: The parameter (its properties *begin* and *end*) defines the fragment of affected text.

---

|                              |                            |
|------------------------------|----------------------------|
| <b>characterStyleSheet()</b> | <i>CharacterStyleSheet</i> |
|------------------------------|----------------------------|

---

|                                     |                            |
|-------------------------------------|----------------------------|
| <b>characterStyleSheet(int pos)</b> | <i>CharacterStyleSheet</i> |
|-------------------------------------|----------------------------|

---

|                                                |                            |
|------------------------------------------------|----------------------------|
| <b>characterStyleSheet(int begin, int end)</b> | <i>CharacterStyleSheet</i> |
|------------------------------------------------|----------------------------|

---

Get an object of the character style sheet that is applied to the formatted text at the specified position. No parameter means the whole text.

Returns an object of the character style sheet applied. If there no style sheet is applied, the method returns *null*.

---

|                                                                           |             |
|---------------------------------------------------------------------------|-------------|
| <b>setCharacterStyleSheet(CharacterStyleSheet style, TextRange range)</b> | <i>void</i> |
|---------------------------------------------------------------------------|-------------|

---

|                                                                                       |             |
|---------------------------------------------------------------------------------------|-------------|
| <b>setCharacterStyleSheet(CharacterStyleSheet style, int begin=0, int end=length)</b> | <i>void</i> |
|---------------------------------------------------------------------------------------|-------------|

---

Apply a character style sheet to the specified text fragment.

*style*: Name of the style sheet to be applied.

*begin*: Index of the first letter of the affected text fragment. The default value is 0.

*end*: Index of the position behind the last affected character. The default value is the length of the text.

*range*: The parameter (its properties *begin* and *end*) defines the fragment of affected text..

|                                                                                                                                                 |               |
|-------------------------------------------------------------------------------------------------------------------------------------------------|---------------|
| <b>characterStyleSheetByName()</b>                                                                                                              | <i>String</i> |
| <b>characterStyleSheetByName(int pos)</b>                                                                                                       | <i>String</i> |
| <b>characterStyleSheetByName(int begin, int end)</b>                                                                                            | <i>String</i> |
| Get the name of the character style sheet that is applied to the formatted text at the specified position. . No parameter means the whole text. |               |
| Returns the object name of the character style sheet applied. If there is no style sheet applied, the method returns <i>null</i> .              |               |
| <b>setCharacterStyleSheetByName(String style, int begin=0, int end=length)</b>                                                                  | <i>void</i>   |
| <b>setCharacterStyleSheetByName(String style, TextRange range)</b>                                                                              | <i>void</i>   |
| Apply character style sheet to the specified text fragment.                                                                                     |               |
| <i>style</i> : Name of the style sheet to be applied.                                                                                           |               |
| <i>begin</i> : Index of the first letter of the affected text fragment. The default value is 0.                                                 |               |
| <i>end</i> : Index of the position behind the last affected character. The default value is the length of the text.                             |               |
| <i>range</i> : The parameter (its properties <i>begin</i> and <i>end</i> ) defines the fragment of affected text.                               |               |

### Methods (paragraph setting)

|                                                                                                                                                                    |                           |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------|
| <b>textAlignment()</b>                                                                                                                                             | <i>ParagraphAlignment</i> |
| <b>textAlignment(int pos)</b>                                                                                                                                      | <i>ParagraphAlignment</i> |
| <b>textAlignment(int begin, int end)</b>                                                                                                                           | <i>ParagraphAlignment</i> |
| <b>textAlignment(TextRange range)</b>                                                                                                                              | <i>ParagraphAlignment</i> |
| Get paragraph alignment at the specified position in the formatted text.                                                                                           |                           |
| <i>pos</i> : Position in the formatted text. The default value is 0.                                                                                               |                           |
| Returns the paragraph alignment code. See enumeration type <i>ParagraphAlignment</i> in the chapter “Enumeration Objects Index” to see all available alignments.   |                           |
| <b>setTextAlignment(ParagraphAlignment align, TextRange range)</b>                                                                                                 | <i>void</i>               |
| <b>setTextAlignment(ParagraphAlignment align, int begin=0, int end=length)</b>                                                                                     | <i>void</i>               |
| Set the paragraph alignment of the specified text fragment.                                                                                                        |                           |
| <i>align</i> : New paragraph alignment. See enumeration type <i>ParagraphAlignment</i> in the chapter “Enumeration Objects Index” to see all available alignments. |                           |
| <i>begin</i> : Index of the first letter of the affected text fragment. The default value is 0.                                                                    |                           |
| <i>end</i> : Index of the position behind the last affected character. The default value is length of the text.                                                    |                           |
| <i>range</i> : The parameter (its properties <i>begin</i> and <i>end</i> ) defines the fragment of affected text.                                                  |                           |
| <b>baselineGridType()</b>                                                                                                                                          | <i>BaselineGridType</i>   |
| <b>baselineGridType(int pos)</b>                                                                                                                                   | <i>BaselineGridType</i>   |

## FormattedText Objects | Class: FormattedText

|                                                                                            |                                  |
|--------------------------------------------------------------------------------------------|----------------------------------|
| <code>baselineGridType(int begin, int end)</code>                                          | <i>BaselineGridType</i>          |
| <code>baselineGridType(TextRange range)</code>                                             | <i>BaselineGridType</i>          |
| <code>setBaselineGridType(BaselineGridType type, int begin=0, int end=len)</code>          | <i>void</i>                      |
| <code>setBaselineGridType(BaselineGridType type, TextRange range)</code>                   | <i>void</i>                      |
| <code>paragraphLineSpacing()</code>                                                        | <i>String</i>                    |
| <code>paragraphLineSpacing(int pos)</code>                                                 | <i>String</i>                    |
| <code>paragraphLineSpacing(int begin, int end)</code>                                      | <i>String</i>                    |
| <code>paragraphLineSpacing(TextRange range)</code>                                         | <i>String</i>                    |
| <code>setParagraphLineSpacing(String value, int begin=0, int end=len)</code>               | <i>void</i>                      |
| <code>setParagraphLineSpacing(String value, TextRange range)</code>                        | <i>void</i>                      |
| <code>spaceToPreviousParagraph()</code>                                                    | <i>String</i>                    |
| <code>spaceToPreviousParagraph(int pos)</code>                                             | <i>String</i>                    |
| <code>spaceToPreviousParagraph(int begin, int end)</code>                                  | <i>String</i>                    |
| <code>spaceToPreviousParagraph(TextRange range)</code>                                     | <i>String</i>                    |
| <code>setSpaceToPreviousParagraph(String value, int begin=0, int end=len)</code>           | <i>void</i>                      |
| <code>setSpaceToPreviousParagraph(String value, TextRange range)</code>                    | <i>void</i>                      |
| <code>spaceToNextParagraph()</code>                                                        | <i>String</i>                    |
| <code>spaceToNextParagraph(int pos)</code>                                                 | <i>String</i>                    |
| <code>spaceToNextParagraph(int begin, int end)</code>                                      | <i>String</i>                    |
| <code>spaceToNextParagraph(TextRange range)</code>                                         | <i>String</i>                    |
| <code>setSpaceToNextParagraph(String value, int begin=0, int end=len)</code>               | <i>void</i>                      |
| <code>setSpaceToNextParagraph(String value, TextRange range)</code>                        | <i>void</i>                      |
| <code>automaticWordSpacing(int pos=0)</code>                                               | <i>AutomaticWordSpacing</i>      |
| <code>automaticWordSpacing(int pos)</code>                                                 | <i>AutomaticWordSpacing</i>      |
| <code>automaticWordSpacing(int begin, int end)</code>                                      | <i>AutomaticWordSpacing</i>      |
| <code>automaticWordSpacing(TextRange range)</code>                                         | <i>AutomaticWordSpacing</i>      |
| <code>setAutomaticWordSpacing(AutomaticWordSpacing value, int begin=0, int end=len)</code> | <i>void</i>                      |
| <code>setAutomaticWordSpacing(AutomaticWordSpacing value, TextRange range)</code>          | <i>void</i>                      |
| <code>automaticCharacterSpacing()</code>                                                   | <i>AutomaticCharacterSpacing</i> |
| <code>automaticCharacterSpacing(int pos)</code>                                            | <i>AutomaticCharacterSpacing</i> |
| <code>automaticCharacterSpacing(int begin, int end)</code>                                 | <i>AutomaticCharacterSpacing</i> |
| <code>automaticCharacterSpacing(TextRange range)</code>                                    | <i>AutomaticCharacterSpacing</i> |

## FormattedText Objects | Class: FormattedText

|                                                                                                      |                                      |
|------------------------------------------------------------------------------------------------------|--------------------------------------|
| <code>setAutomaticCharacterSpacing(AutomaticCharacterSpacing value, int begin=0, int end=len)</code> | <code>void</code>                    |
| <code>setAutomaticCharacterSpacing(AutomaticCharacterSpacing value, TextRange range)</code>          | <code>void</code>                    |
| <code>automaticCharacterWidth()</code>                                                               | <code>AutomaticCharacterWidth</code> |
| <code>automaticCharacterWidth(int pos)</code>                                                        | <code>AutomaticCharacterWidth</code> |
| <code>automaticCharacterWidth(int begin, int end)</code>                                             | <code>AutomaticCharacterWidth</code> |
| <code>automaticCharacterWidth(TextRange range)</code>                                                | <code>AutomaticCharacterWidth</code> |
| <code>setAutomaticCharacterWidth(AutomaticCharacterWidth value, int begin=0, int end=len)</code>     | <code>void</code>                    |
| <code>setAutomaticCharacterWidth(AutomaticCharacterWidth value, TextRange range)</code>              | <code>void</code>                    |
| <code>paragraphLeftIndent()</code>                                                                   | <code>String</code>                  |
| <code>paragraphLeftIndent(int pos)</code>                                                            | <code>String</code>                  |
| <code>paragraphLeftIndent(int begin, int end)</code>                                                 | <code>String</code>                  |
| <code>paragraphLeftIndent(TextRange range)</code>                                                    | <code>String</code>                  |
| <code>setParagraphLeftIndent(String value, int begin=0, int end=len)</code>                          | <code>void</code>                    |
| <code>setParagraphLeftIndent(String value, TextRange range)</code>                                   | <code>void</code>                    |
| <code>paragraphRightIndent()</code>                                                                  | <code>String</code>                  |
| <code>paragraphRightIndent(int pos)</code>                                                           | <code>String</code>                  |
| <code>paragraphRightIndent(int begin, int end)</code>                                                | <code>String</code>                  |
| <code>paragraphRightIndent(TextRange range)</code>                                                   | <code>String</code>                  |
| <code>setParagraphRightIndent(String value, int begin=0, int end=len)</code>                         | <code>void</code>                    |
| <code>setParagraphRightIndent(String value, TextRange range)</code>                                  | <code>void</code>                    |
| <code>paragraphIndent()</code>                                                                       | <code>String</code>                  |
| <code>paragraphIndent(int pos)</code>                                                                | <code>String</code>                  |
| <code>paragraphIndent(int begin, int end)</code>                                                     | <code>String</code>                  |
| <code>paragraphIndent(TextRange range)</code>                                                        | <code>String</code>                  |
| <code>setParagraphIndent (String value, int begin=0, int end=len)</code>                             | <code>void</code>                    |
| <code>setParagraphIndent (String value, TextRange range)</code>                                      | <code>void</code>                    |
| <code>paragraphTabs()</code>                                                                         | <code>Tabulators</code>              |
| <code>paragraphTabs(int pos)</code>                                                                  | <code>Tabulators</code>              |
| <code>paragraphTabs(int begin, int end)</code>                                                       | <code>Tabulators</code>              |
| <code>paragraphTabs(TextRange range)</code>                                                          | <code>Tabulators</code>              |
| <code>setParagraphTabs (Tabulators tabs, int begin=0, int end=len)</code>                            | <code>void</code>                    |

## FormattedText Objects | Class: FormattedText

|                                                                                        |                               |
|----------------------------------------------------------------------------------------|-------------------------------|
| <code>setParagraphTabs (Tabulators tabs, TextRange range)</code>                       | <code>void</code>             |
| <code>dropCaps()</code>                                                                | <code>DropCaps</code>         |
| <code>dropCaps(int pos)</code>                                                         | <code>DropCaps</code>         |
| <code>dropCaps(int begin, int end)</code>                                              | <code>DropCaps</code>         |
| <code>dropCaps(TextRange range)</code>                                                 | <code>DropCaps</code>         |
| <code>setDropCaps(DropCaps data, int begin=0, int end=len)</code>                      | <code>void</code>             |
| <code>setDropCaps(DropCaps data, TextRange range)</code>                               | <code>void</code>             |
| <code>hyphenation()</code>                                                             | <code>Hyphenation</code>      |
| <code>hyphenation(int pos)</code>                                                      | <code>Hyphenation</code>      |
| <code>hyphenation(int begin, int end)</code>                                           | <code>Hyphenation</code>      |
| <code>hyphenation(TextRange range)</code>                                              | <code>Hyphenation</code>      |
| <code>setHyphenation(Hyphenation hyp, int begin=0, int end=len)</code>                 | <code>void</code>             |
| <code>setHyphenation(Hyphenation hyp, TextRange range)</code>                          | <code>void</code>             |
| <code>enumerations()</code>                                                            | <code>Enumerations</code>     |
| <code>enumerations(int pos)</code>                                                     | <code>Enumerations</code>     |
| <code>enumerations(int begin, int end)</code>                                          | <code>Enumerations</code>     |
| <code>enumerations(TextRange range)</code>                                             | <code>Enumerations</code>     |
| <code>setEnumerations(Enumerations enum, int begin=0, int end=len)</code>              | <code>void</code>             |
| <code>setEnumerations(Enumerations enum, TextRange range)</code>                       | <code>void</code>             |
| <code>writingDirection()</code>                                                        | <code>WritingDirection</code> |
| <code>writingDirection(int pos)</code>                                                 | <code>WritingDirection</code> |
| <code>writingDirection(int begin, int end)</code>                                      | <code>WritingDirection</code> |
| <code>writingDirection(TextRange range)</code>                                         | <code>WritingDirection</code> |
| <code>setWritingDirection(WritingDirection direction, TextRange range)</code>          | <code>void</code>             |
| <code>setWritingDirection(WritingDirection direction, int begin=0, int end=len)</code> | <code>void</code>             |
| <code>opticalAlignment()</code>                                                        | <code>OpticalAlignment</code> |
| <code>opticalAlignment(int pos)</code>                                                 | <code>OpticalAlignment</code> |
| <code>opticalAlignment(int begin, int end)</code>                                      | <code>OpticalAlignment</code> |
| <code>opticalAlignment(TextRange range)</code>                                         | <code>OpticalAlignment</code> |
| <code>setOpticalAlignment(OpticalAlignment alignment, TextRange range)</code>          | <code>void</code>             |
| <code>setOpticalAlignment(OpticalAlignment alignment, int begin=0, int end=len)</code> | <code>void</code>             |

## FormattedText Objects | Class: FormattedText

|                                                                                       |                            |
|---------------------------------------------------------------------------------------|----------------------------|
| <code>supressLineNumbering()</code>                                                   | <i>bool</i>                |
| <code>supressLineNumbering(int pos)</code>                                            | <i>bool</i>                |
| <code>supressLineNumbering(int begin, int end)</code>                                 | <i>bool</i>                |
| <code>supressLineNumbering(TextRange range)</code>                                    | <i>bool</i>                |
| <code>setSupressLineNumbering(bool on, TextRange range)</code>                        | <i>void</i>                |
| <code>setSupressLineNumbering(bool on=true, int begin=0, int end=len)</code>          | <i>void</i>                |
| Get or set whether the lines of the paragraphs are excluded from the line numbering.  |                            |
| <code>paragraphFrame()</code>                                                         | <i>ParagraphFrame</i>      |
| <code>paragraphFrame(int pos)</code>                                                  | <i>ParagraphFrame</i>      |
| <code>paragraphFrame(int begin, int end)</code>                                       | <i>ParagraphFrame</i>      |
| <code>paragraphFrame(TextRange range)</code>                                          | <i>ParagraphFrame</i>      |
| <code>setParagraphFrame(ParagraphFrame pf, TextRange range)</code>                    | <i>void</i>                |
| <code>setParagraphFrame(ParagraphFrame pf, int begin=0, int end=len)</code>           | <i>void</i>                |
| <code>paragraphBackground()</code>                                                    | <i>ParagraphBackground</i> |
| <code>paragraphBackground(int pos)</code>                                             | <i>ParagraphBackground</i> |
| <code>paragraphBackground(int begin, int end)</code>                                  | <i>ParagraphBackground</i> |
| <code>paragraphBackground(TextRange range)</code>                                     | <i>ParagraphBackground</i> |
| <code>setParagraphBackground(ParagraphBackground pb, TextRange range)</code>          | <i>void</i>                |
| <code>setParagraphBackground(ParagraphBackground pb, int begin=0, int end=len)</code> | <i>void</i>                |
| <code>paragraphRuleAbove()</code>                                                     | <i>ParagraphRule</i>       |
| <code>paragraphRuleAbove(int pos)</code>                                              | <i>ParagraphRule</i>       |
| <code>paragraphRuleAbove(int begin, int end)</code>                                   | <i>ParagraphRule</i>       |
| <code>paragraphRuleAbove(TextRange range)</code>                                      | <i>ParagraphRule</i>       |
| <code>setParagraphRuleAbove(ParagraphRule pr, TextRange range)</code>                 | <i>void</i>                |
| <code>setParagraphRuleAbove(ParagraphRule pr, int begin=0, int end=len)</code>        | <i>void</i>                |
| <code>paragraphRuleBelow()</code>                                                     | <i>ParagraphRule</i>       |
| <code>paragraphRuleBelow(int pos)</code>                                              | <i>ParagraphRule</i>       |
| <code>paragraphRuleBelow(int begin, int end)</code>                                   | <i>ParagraphRule</i>       |
| <code>paragraphRuleBelow(TextRange range)</code>                                      | <i>ParagraphRule</i>       |
| <code>setParagraphRuleBelow(ParagraphRule pr, TextRange range)</code>                 | <i>void</i>                |
| <code>setParagraphRuleBelow(ParagraphRule pr, int begin=0, int end=len)</code>        | <i>void</i>                |
| <code>widowsOrphans()</code>                                                          | <i>WidowsOrphans</i>       |
| <code>widowsOrphans(int pos)</code>                                                   | <i>WidowsOrphans</i>       |

|                                                                                                                              |                            |
|------------------------------------------------------------------------------------------------------------------------------|----------------------------|
| <b>widowsOrphans</b> ( <i>int begin, int end</i> )                                                                           | <i>WidowsOrphans</i>       |
| <b>widowsOrphans</b> ( <i>TextRange range</i> )                                                                              | <i>WidowsOrphans</i>       |
| <b>setWidowsOrphans</b> ( <i>WidowsOrphans value, TextRange range</i> )                                                      | <i>void</i>                |
| <b>setWidowsOrphans</b> ( <i>WidowsOrphans value, int begin=0, int end=length</i> )                                          | <i>void</i>                |
| <b>keepParagraphTogether</b> ()                                                                                              | <i>bool</i>                |
| <b>keepParagraphTogether</b> ( <i>int pos</i> )                                                                              | <i>bool</i>                |
| <b>keepParagraphTogether</b> ( <i>int begin, int end</i> )                                                                   | <i>bool</i>                |
| <b>keepParagraphTogether</b> ( <i>TextRange range</i> )                                                                      | <i>bool</i>                |
| <b>setKeepParagraphTogether</b> ( <i>bool value, TextRange range</i> )                                                       | <i>void</i>                |
| <b>setKeepParagraphTogether</b> ( <i>bool value, int begin=0, int end=length</i> )                                           | <i>void</i>                |
| <b>paragraphStyleSheet</b> ()                                                                                                | <i>ParagraphStyleSheet</i> |
| <b>paragraphStyleSheet</b> ( <i>int pos</i> )                                                                                | <i>ParagraphStyleSheet</i> |
| <b>paragraphStyleSheet</b> ( <i>int begin, int end</i> )                                                                     | <i>ParagraphStyleSheet</i> |
| Get an object of the paragraph style sheet that is applied to the formatted text at the specified position.                  |                            |
| <i>pos</i> : Position in the formatted text. The default value is 0.                                                         |                            |
| Returns an object of the paragraph style sheet applied. If there is no style sheet applied, the method returns <i>null</i> . |                            |
| <b>setParagraphStyleSheet</b> ( <i>ParagraphStyleSheet style, int begin=0, int end=length</i> )                              | <i>void</i>                |
| <b>setParagraphStyleSheet</b> ( <i>ParagraphStyleSheet style, TextRange range</i> )                                          | <i>void</i>                |
| Apply a paragraph style sheet to all paragraphs that belong (at least partially) to the specified text fragment.             |                            |
| <i>style</i> : Name of the style sheet to be applied.                                                                        |                            |
| <i>begin</i> : Index of the first letter of the affected text fragment. The default value is 0.                              |                            |
| <i>end</i> : Index of the position behind the last affected character. The default value is the length of the text.          |                            |
| <i>range</i> : The parameter (its properties <i>begin</i> and <i>end</i> ) defines the fragment of affected text.            |                            |
| <b>paragraphStyleSheetByName</b> ( <i>int pos=0</i> )                                                                        | <i>String</i>              |
| <b>paragraphStyleSheetByName</b> ( <i>int pos</i> )                                                                          | <i>String</i>              |
| <b>paragraphStyleSheetByName</b> ( <i>int begin, int end</i> )                                                               | <i>String</i>              |
| Get the name of the paragraph style sheet that is applied to the formatted text at the specified position.                   |                            |
| <i>pos</i> : Position in the formatted text. The default value is 0.                                                         |                            |
| Returns the name of applied paragraph style sheet. If no style sheet is applied, the method returns <i>null</i> .            |                            |
| <b>setParagraphStyleSheetByName</b> ( <i>String style, int begin=0, int end=length</i> )                                     | <i>void</i>                |
| <b>setParagraphStyleSheetByName</b> ( <i>String style, TextRange range</i> )                                                 | <i>void</i>                |

Apply a paragraph style sheet to all paragraphs that belong (at least partially) to the specified text fragment.

*style*: Name of the style sheet to be applied.

*begin*: Index of the first letter of the affected text fragment. The default value is 0.

*end*: Index of the position behind the last affected character. The default value is the length of the text.

*range*: The parameter (its properties *begin* and *end*) defines the fragment of affected text.

## Methods (layout settings)

|                                                                                                   |                                |
|---------------------------------------------------------------------------------------------------|--------------------------------|
| <code>layoutLineOrientation()</code>                                                              | <i>LayoutLineDirection</i>     |
| <code>layoutLineOrientation(int pos)</code>                                                       | <i>LayoutLineDirection</i>     |
| <code>layoutLineOrientation(int begin, int end)</code>                                            | <i>LayoutLineDirection</i>     |
| <code>layoutLineOrientation(TextRange range)</code>                                               | <i>LayoutLineDirection</i>     |
| <code>setLayoutLineOrientation(LayoutLineDirection direction, TextRange range)</code>             | <i>void</i>                    |
| <code>setLayoutLineOrientation(LayoutLineDirection direction, int begin=0, int end=len)</code>    | <i>void</i>                    |
| <code>layoutLineJustification()</code>                                                            | <i>LayoutLineJustification</i> |
| <code>layoutLineJustification(int pos)</code>                                                     | <i>LayoutLineJustification</i> |
| <code>layoutLineJustification(int begin, int end)</code>                                          | <i>LayoutLineJustification</i> |
| <code>layoutLineJustification(TextRange range)</code>                                             | <i>LayoutLineJustification</i> |
| <code>setLayoutLineJustification(LayoutLineJustification justif, TextRange range)</code>          | <i>void</i>                    |
| <code>setLayoutLineJustification(LayoutLineJustification justif, int begin=0, int end=len)</code> | <i>void</i>                    |
| <code>justifiedSpaceBetweenLinesMaximum()</code>                                                  | <i>String</i>                  |
| <code>justifiedSpaceBetweenLinesMaximum(int pos)</code>                                           | <i>String</i>                  |
| <code>justifiedSpaceBetweenLinesMaximum(int begin, int end)</code>                                | <i>String</i>                  |
| <code>justifiedSpaceBetweenLinesMaximum(TextRange range)</code>                                   | <i>String</i>                  |
| <code>setJustifiedSpaceBetweenLinesMaximum(String value, TextRange range)</code>                  | <i>void</i>                    |
| <code>setJustifiedSpaceBetweenLinesMaximum(String value, int begin=0, int end=len)</code>         | <i>void</i>                    |
| <code>paragraphSpaceMaximum()</code>                                                              | <i>String</i>                  |
| <code>paragraphSpaceMaximum(int pos=0)</code>                                                     | <i>String</i>                  |
| <code>paragraphSpaceMaximum(int begin, int end)</code>                                            | <i>String</i>                  |
| <code>paragraphSpaceMaximum(TextRange range)</code>                                               | <i>String</i>                  |
| <code>setParagraphSpaceMaximum(String value, TextRange range)</code>                              | <i>void</i>                    |
| <code>setParagraphSpaceMaximum(String value, int begin=0, int end=len)</code>                     | <i>void</i>                    |
| <code>spaceToPreviousLayout(int pos = 0)</code>                                                   | <i>String</i>                  |

## FormattedText Objects | Class: FormattedText

|                                                                                    |                          |
|------------------------------------------------------------------------------------|--------------------------|
| <code>spaceToPreviousLayout(int pos)</code>                                        | <i>String</i>            |
| <code>spaceToPreviousLayout(int begin, int end)</code>                             | <i>String</i>            |
| <code>spaceToPreviousLayout(TextRange range)</code>                                | <i>String</i>            |
| <code>setSpaceToPreviousLayout(String value, TextRange range)</code>               | <i>void</i>              |
| <code>setSpaceToPreviousLayout(String value, int begin=0, int end=len)</code>      | <i>void</i>              |
| <code>spaceToNextLayout()</code>                                                   | <i>String</i>            |
| <code>spaceToNextLayout(int pos)</code>                                            | <i>String</i>            |
| <code>spaceToNextLayout(int begin, int end)</code>                                 | <i>String</i>            |
| <code>spaceToNextLayout(TextRange range)</code>                                    | <i>String</i>            |
| <code>setSpaceToNextLayout(String value, TextRange range)</code>                   | <i>void</i>              |
| <code>setSpaceToNextLayout(String value, int begin=0, int end=len)</code>          | <i>void</i>              |
| <code>layoutColumns()</code>                                                       | <i>LayoutColumnsInfo</i> |
| <code>layoutColumns(int pos)</code>                                                | <i>LayoutColumnsInfo</i> |
| <code>layoutColumns(int begin, int end)</code>                                     | <i>LayoutColumnsInfo</i> |
| <code>layoutColumns(TextRange range)</code>                                        | <i>LayoutColumnsInfo</i> |
| <code>setLayoutColumns(LayoutColumnsInfo columns, TextRange range)</code>          | <i>void</i>              |
| <code>setLayoutColumns(LayoutColumnsInfo columns, int begin=0, int end=len)</code> | <i>void</i>              |
| <code>layoutFootnotes()</code>                                                     | <i>LayoutFootnotes</i>   |
| <code>layoutFootnotes(int pos)</code>                                              | <i>LayoutFootnotes</i>   |
| <code>layoutFootnotes(int begin, int end)</code>                                   | <i>LayoutFootnotes</i>   |
| <code>layoutFootnotes(TextRange range)</code>                                      | <i>LayoutFootnotes</i>   |
| <code>setLayoutFootnotes(LayoutFootnotes fn, TextRange range)</code>               | <i>void</i>              |
| <code>setLayoutFootnotes(LayoutFootnotes fn, int begin=0, int end=len)</code>      | <i>void</i>              |
| <code>layoutStyleSheet()</code>                                                    | <i>LayoutStyleSheet</i>  |
| <code>layoutStyleSheet(int pos)</code>                                             | <i>LayoutStyleSheet</i>  |

`layoutStyleSheet(int pos, int end)` *LayoutStyleSheet*

Get an object of a layout style sheet that is applied to the formatted text at the specified position. No parameter means the whole text.

Returns an object of the layout style sheet applied. If no style sheet is applied, the method returns *null*.

`setLayoutStyleSheet(LayoutStyleSheet style, int begin=0, int end=length)` *void*

`setLayoutStyleSheet(LayoutStyleSheet style, TextRange range)` *void*

Apply a layout style sheet to all layouts that belong (at least partially) to the specified text fragment.

*style*: Name of the style sheet to be applied.

*begin*: Index of the first letter of the affected text fragment. The default value is 0.

*end*: Index of the position behind the last affected character. The default value is the length of the text.

*range*: The parameter (its properties *begin* and *end*) defines the fragment of affected text.

---

**layoutStyleSheetByName**(*int pos=0*)      *String*

---

**layoutStyleSheetByName**(*int pos*)      *String*

---

**layoutStyleSheetByName**(*int begin, int end*) *String*

---

Get the name of the layout style sheet that is applied to the formatted text at the specified position. No parameter means the whole text.

Returns the name of the layout style sheet applied. If no style sheet is applied, the method returns *null*.

---

**setLayoutStyleSheetByName**(*String style, TextRange range*)      *void*

---

**setLayoutStyleSheetByName**(*String style, int begin=0, int end=end*)      *void*

---

Apply layout style sheet to all layouts that belong (at least partially) to the specified text fragment.

*style*: Name of the style sheet to be applied.

*begin*: Index of the first letter of the affected text fragment. Default value is 0.

*end*: Index of the position behind the last affected character. The default value is length of the text.

*range*: The parameter (its properties *begin* and *end*) defines the fragment of affected text.

## Methods (chapter setting)

---

**chapterName**()      *String*

---

**chapterName**(*int pos*)      *String*

---

**chapterName**(*int begin=0, int end=length*)      *String*

---

**chapterName**(*TextRange range*)      *String*

---

**setChapterName**(*String name, TextRange range*)      *void*

---

**setChapterName**(*String name, int begin=0, int end=length*)      *void*

---

## Methods (Story setting)

---

**storyEndnotes**()      *TextEndnotes*

---

**storyEndnotes**(*int pos*)      *TextEndnotes*

---

**storyEndnotes**(*int begin=0, int end=length*) *TextEndnotes*

---

**storyEndnotes**(*TextRange range*)      *TextEndnotes*

---

**setStoryEndnotes**(*TextEndnotes value, TextRange range*)      *void*

---

**setStoryEndnotes**(*TextEndnotes value, int begin=0, int end=length*)      *void*

---

**storyFootnotes**()      *TextFootnotes*

---

**storyFootnotes**(*int pos*)      *TextFootnotes*

---

**storyFootnotes**(*int begin=0, int end=length*)      *TextFootnotes*

---

|                                                                                                                                                                                                                            |                      |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------|
| <b>storyFootnotes</b> ( <i>TextRange range</i> )                                                                                                                                                                           | <i>TextFootnotes</i> |
| <b>setStoryFootnotes</b> ( <i>TextFootnotes value, TextRange range</i> )                                                                                                                                                   | <i>void</i>          |
| <b>setStoryFootnotes</b> ( <i>TextFootnotes value, int begin=0, int end=length</i> )                                                                                                                                       | <i>void</i>          |
| <b>storyLineCounterData</b> ()                                                                                                                                                                                             | <i>LineNumbers</i>   |
| <b>storyLineCounterData</b> ( <i>int pos</i> )                                                                                                                                                                             | <i>LineNumbers</i>   |
| <b>storyLineCounterData</b> ( <i>int begin=0, int end=length</i> )                                                                                                                                                         | <i>LineNumbers</i>   |
| <b>storyLineCounterData</b> ( <i>TextRange range</i> )                                                                                                                                                                     | <i>LineNumbers</i>   |
| <b>setStoryLineCounterData</b> ( <i>LineNumbers data, TextRange range</i> )                                                                                                                                                | <i>void</i>          |
| <b>setStoryLineCounterData</b> ( <i>LineNumbers data, int begin=0, int end=length</i> )                                                                                                                                    | <i>void</i>          |
| <b>Methods (lists)</b>                                                                                                                                                                                                     |                      |
| <b>paragraphAtCharacterPosition</b> ( <i>int pos=0</i> )                                                                                                                                                                   | <i>TextRange</i>     |
| Get a paragraph, part of the formatted text, which contains a specified text position.                                                                                                                                     |                      |
| <i>pos</i> : Position in the formatted text. The default value is 0.                                                                                                                                                       |                      |
| Returns a text range which represents the appropriate paragraph.                                                                                                                                                           |                      |
| <b>chapters</b> ( <i>TextRange range</i> )                                                                                                                                                                                 | <i>TextRanges</i>    |
| <b>chapters</b> ( <i>int begin=0, int end=length</i> )                                                                                                                                                                     | <i>TextRanges</i>    |
| Returns a collection of the object TextRanges. Each TextRange object defines one chapter that is contained in the text in the range <i>range</i> . If no range input parameter is entered all text is used.                |                      |
| <b>layouts</b> ( <i>TextRange range</i> )                                                                                                                                                                                  | <i>TextRanges</i>    |
| <b>layouts</b> ( <i>int begin=0, int end=length</i> )                                                                                                                                                                      | <i>TextRanges</i>    |
| Returns a collection of the object TextRanges. Each TextRange object defines one layout that is contained in the text in the range <i>range</i> . If no range input parameter is entered all text is used.                 |                      |
| <b>paragraphs</b> ( <i>TextRange range</i> )                                                                                                                                                                               | <i>TextRanges</i>    |
| <b>paragraphs</b> ( <i>int begin=0, int end=length</i> )                                                                                                                                                                   | <i>TextRanges</i>    |
| Returns a collection of the object TextRanges. Each TextRange object defines one paragraph that is contained in the text in the range <i>range</i> . If no <i>range</i> input parameter is entered all text is used.       |                      |
| <b>spans</b> ( <i>TextRange range</i> )                                                                                                                                                                                    | <i>TextRanges</i>    |
| <b>spans</b> ( <i>int begin=0, int end=length</i> )                                                                                                                                                                        | <i>TextRanges</i>    |
| Returns a collection of the object TextRanges. Each TextRange object defines one span that is contained in the text in the range <i>range</i> . If no <i>range</i> input parameter is entered all text is used.            |                      |
| <b>endnotesRanges</b> ( <i>TextRange range</i> )                                                                                                                                                                           | <i>TextRanges</i>    |
| <b>endnotesRanges</b> ( <i>int begin=0, int end=length</i> )                                                                                                                                                               | <i>TextRanges</i>    |
| Returns a collection of the object TextRanges. Each TextRange object defines one endnotes ranges that is contained in the text in the range <i>range</i> . If no <i>range</i> input parameter is entered all text is used. |                      |
| <b>story</b> ()                                                                                                                                                                                                            | <i>TextRange</i>     |

**Examples:**

```
var doc = application.currentDocument;
if (doc) {
    // Create new text object
    var objT = doc.objects().create(ObjectType.Text);
    if (objT) {
        with (objT) {
            // Configure its shape to be a rectangle
            form = FrameForm.Rectangle;
            pagex = "15mm";
            pagey = "35.5pt";
            width = "165mm";
            height = "714.5pt";
        }
        // Insert the text object in the current page
        doc.currentPage.add(objT);
    }
    // Access the content of the text object
    with (objT.content.formattedText) {
        // Add some text
        var trLine1 = appendPlainText("This is an example text, which is
        appended to a formatted text of TextObject.\n");
        var trLine2 = appendPlainText("After a new line here is another
        line of this text.\n");
        appendPlainText("\n");
        // Apply another font, its size and color
        setFont("Courier New-Bold", trLine1);
        setFontSize(new FontSize(20), trLine1.begin, trLine1.end - 20);
        setBrush(Color.Blue, trLine1.begin + 20, trLine1.end - 20);
        // Automatic underline: no style parameters are necessary
        var optUl = new UnderlineOptions(FontStyleLineType.Automatic);
        optUl.rangeType = FontStyleLineRangeType.Words;
        setUnderline(optUl, trLine2.begin, trLine2.begin + 11);
        // Manual strikethrough: prepare parameters and apply them to a
        text
        var optSt = new StrikethroughOptions(FontStyleLineType.Manual);
        with (optSt) {
            brush = new ColorBrush(Color.Red);
            shade = 100;
            opacity = 60;
            lineWidth = "1.5pt";
            lineStyle = "Pattern 2";
            offset = "-3pt";
            rangeType = FontStyleLineRangeType.Words;
        }
        setStrikethrough(optSt, trLine2.begin + 12, trLine2.begin + 32);
        // Glyphs
        var trG1 = appendPlainText("1 x 1012 ... No Glyph Superscript
        Subscript\n");
        appendPlainText("\n");
        setFontSize(new FontSize(20), trG1);
```

```
        setCharacterPosition(CharacterPosition.Superscript, trGl.begin + 6,
            trGl.begin + 8);
        setCharacterPosition(CharacterPosition.Superscript, trGl.begin +
22,            trGl.begin + 33);
        setCharacterPosition(CharacterPosition.Subscript, trGl.begin + 34,
            trGl.begin + 43);
        setCharacterUpperLowerCase(CharacterUpperLowerCase.Caps, trGl.begin
+ 34,            trGl.begin + 43);
        // Outlines
        appendPlainText("Outlined Text:\n");
        var trOl = appendPlainText("STYLE 1 / STYLE 2 / STYLE 3\n");
        appendPlainText("\n");
        setFontSize(new FontSize(35), trOl);
        var optOl = new OutlineOptions(Color.Green, 100, 100, "2pt",
"Solid", CharacterOutlineJoinStyle.MiterJoin);
        setOutlineOptions(optOl, new TextRange(trOl.begin, trOl.begin +
7));

        optOl.brush = new ColorBrush(Color.Blue);
        optOl.joinStyle = CharacterOutlineJoinStyle.BevelJoin;
        optOl.active = true;
        setOutlineOptions(optOl, new TextRange(trOl.begin + 10, trOl.begin
+ 17));

        optOl.brush = new ColorBrush(Color.Magenta);
        optOl.joinStyle = CharacterOutlineJoinStyle.RoundJoin;
        setOutlineOptions(optOl, new TextRange(trOl.begin + 20, trOl.begin
+ 27));

        // Character scaling
        appendPlainText("\n\n\n");
        var trScl = appendPlainText("Character scaling: 0% 200% 400%\n");
        appendPlainText("\n");
        setCharacterHeightScale(200, trScl.end - 10, trScl.end - 6);
        setCharacterHeightScale(400, trScl.end - 5, trScl.end - 1);
        // Character spacing
        var trSpc = appendPlainText("Character spacing: POSITIVE-
NEGATIVE\n");
        appendPlainText("\n");
        setFontSize(new FontSize(20), trSpc);
        setCharacterSpacing(0.5, trSpc.end - 20, trSpc.end - 11);
        setCharacterSpacing(-0.2, trSpc.end - 10, trSpc.end - 1);
        // Character rotation
        var trRot = appendPlainText("Character rotation: 90 degrees / 180
degrees\n");
        appendPlainText("\n");
        setFontSize(new FontSize(15), trRot);
        setCharacterRotation(CharacterRotation.Angle90, trRot.end - 25,
            trRot.end - 15);
        setCharacterRotation(CharacterRotation.Angle180, trRot.end - 12,
```

```
        trRot.end - 1);
    // Character frame
    appendPlainText("\n");
    var trFrm = appendPlainText("Character frame: negative indent /
positive indent \n");
    appendPlainText("\n");
    setFontSize(new FontSize(15), trFrm);

    var charFrame1 = new CharacterFrame(true, Color.Green, 100, 100,
"-2mm", "-2mm", "-7pt", "-7pt", "2pt", "Pattern 8");
    setCharacterFrame(charFrame1, trFrm.end - 37, trFrm.end - 22)

    var charFrame2 = new CharacterFrame();
    charFrame2.active = true;
    charFrame2.brush = new ColorBrush(Color.Red);
    charFrame2.shade = 100;
    charFrame2.opacity = 100;
    charFrame2.leftIndent = "2mm";
    charFrame2.rightIndent = "2mm";
    charFrame2.topIndent = "2pt";
    charFrame2.bottomIndent = "2pt";
    charFrame2.frameWidth = "2pt";
    charFrame2.lineStyle = "Pattern 5";
    setCharacterFrame(charFrame2, trFrm.end - 17, trFrm.end - 2);

    // Text alignment
    var trAlig;
    for (i = 0; i < 5; i++)
        trAlig = appendPlainText("Right text alignment is demonstrated
in this paragraph.");
    appendPlainText("\n\n");
    setTextAlignment(ParagraphAlignment.Right, trAlig.begin,
trAlig.end);
    for (i = 0; i < 5; i++)
        trAlig = appendPlainText("Center text alignment is demonstrated
in this paragraph.");
    appendPlainText("\n\n");
    setTextAlignment(ParagraphAlignment.Center, trAlig.begin,
trAlig.end);
    for (i = 0; i < 4; i++)
        trAlig = appendPlainText("Forced justified alignment is
demonstrated in this paragraph.");
    appendPlainText("See the he last line of the paragraph. It is
stretched to whole width of the text object.");
    appendPlainText("\n\n");
    setTextAlignment(ParagraphAlignment.ForcedJustified, trAlig.begin,
trAlig.end);
    // Object inserted in the formatted text
    var objGr = doc.objects().create(ObjectType.Graphic);
    if (objGr) {
        with (objGr) {
            form = FrameForm.Rectangle;
```

```
        width = "30mm";
        height = "20mm";
        frameColor = new ColorBrush(Color.Red);
        frameWidth = "2pt";
        fillColor = new ColorBrush(Color.Yellow);
        fillShade = 100;
    }
    var trObj = appendPlainText("Some object is inserted here
inside the formatted text.\n");
    appendPlainText("\n");
    setObject(objGr, trObj.begin + 29);
}
// Character position correction.
var trChc = appendPlainText("Position of the letter S is corrected
by 20 points down and 40 points right.\n");
appendPlainText("\n");
setPositionCorrection(new CharacterPositionCorrection(20, 40),
    trChc.begin + 23);
}
}
```

## Class: TextRanges

The class *TextRanges* holds the collection of *TextRange* objects.

### Properties:

---

|               |                 |
|---------------|-----------------|
| <b>length</b> | <i>int Read</i> |
|---------------|-----------------|

---

Returns number of items in the collection.

### Methods:

---

|                      |                  |
|----------------------|------------------|
| <b>at(int index)</b> | <i>TextRange</i> |
|----------------------|------------------|

---

Returns an item of *TextRange* object from the *index* position of the collection.

## Class: TextRange

The class *TextRange* implements an object that describes a fragment of the text, i.e. its beginning and end.

### Constructor:

---

*new TextRange(int begin=0, int end=1)*

---

### Properties:

---

|              |                       |
|--------------|-----------------------|
| <b>begin</b> | <i>int Read/Write</i> |
|--------------|-----------------------|

---

Position of the beginning of the text fragment inside the original text.

---

|            |                       |
|------------|-----------------------|
| <b>end</b> | <i>int Read/Write</i> |
|------------|-----------------------|

---

Position of the end of the text fragment inside the original text. The value is the index of the character behind the last character that is included into the range.

|               |            |             |
|---------------|------------|-------------|
| <b>length</b> | <i>int</i> | <i>Read</i> |
|---------------|------------|-------------|

Length of the text fragment, i.e. the number of characters and all objects that are inserted into the text fragment.

### Methods:

|                                               |             |
|-----------------------------------------------|-------------|
| <b>setRange</b> ( <i>int begin, int end</i> ) | <i>void</i> |
|-----------------------------------------------|-------------|

Set the beginning and end of the text range.

*begin*: Index of the first character of the text range inside the whole text.

*end*: Index of the position behind the last character of the text range.

## Class: TextCursor

The object “TextCursor” holds an object with currently edited text with the cursor active.

### Properties:

|             |                      |             |
|-------------|----------------------|-------------|
| <b>text</b> | <i>FormattedText</i> | <i>Read</i> |
|-------------|----------------------|-------------|

Returns objects of the type FormattedText that hold all text.

|                  |                  |                   |
|------------------|------------------|-------------------|
| <b>selection</b> | <i>TextRange</i> | <i>Read/Write</i> |
|------------------|------------------|-------------------|

Set/get the range of text that is selected.

|                  |                  |             |
|------------------|------------------|-------------|
| <b>textChain</b> | <i>TextChain</i> | <i>Read</i> |
|------------------|------------------|-------------|

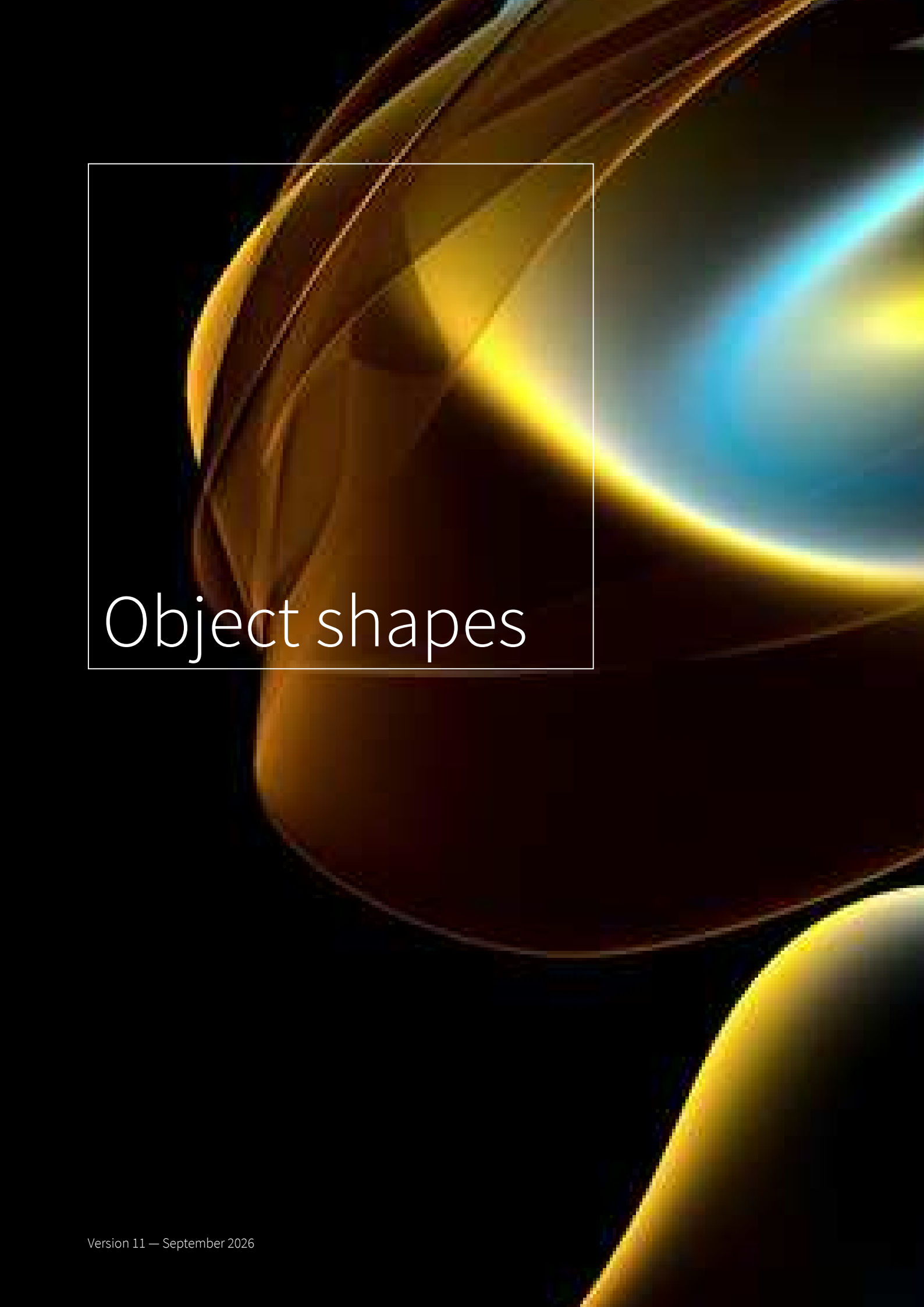
Returns the story of the text object that holds current text.

|                   |                   |             |
|-------------------|-------------------|-------------|
| <b>selections</b> | <i>TextRanges</i> | <i>Read</i> |
|-------------------|-------------------|-------------|

Returns all ranges of text that are selected.







# Object shapes

# Object shapes

## Class: Point

The class **Point** implements an object that describes one point of the polygon point list with a flag description. The point structure is an array with two or three items: [*x coordinate*, *y coordinate*, *flags*].

There are two types of points in the polygon points list: sizing point and control point. The sizing point defines the position of the point in the polygon and the control point defines a curvature in the neighborhood of a point. The sizing point can be defined without flag information (i.e. [*x,y*]) or with flag information (i.e. [*x,y,flags*]). The control point must always define the flags information. A sizing point may have one, two or no control points. The first control point is defined before the sizing point and the second one is defined after sizing point. Further polygons can be joined to one object with flags that are defined with the sizing point.

### List of usable flags for the control point:

“**A**” - after the control point, the point must be defined after the sizing point in the polygon point list

“**AC**” - after the control point, the point is bounded in the before control point

“**B**” - before the control point, the point must be defined before the sizing point in the polygon point list

“**BC**” - before the control point, the point is bounded in the after control point

### List of usable flags for the sizing point:

“**F**” - first point of the polygon point list; it must only be defined when further polygons are defined in the polygon point list

“**L**” - last point of the polygon point list; it must only be defined when more polygons is defined in the polygon point list.

### Constructor:

**new** Point(*x,y*)

**new** Point(*x,y,flags*)

**new** Point(Array [*x, y, flags*])

In the constructors of Point, **flags** is a number: a combination of PolygonPointFlag values, e.g. **PolygonPointFlag.Control** | **PolygonPointFlag.Before**. The letter codes “A”, “AC”, “B”, “BC”, “F” and “L” are only evaluated in the point array passed to the constructor of PointList.

### Properties:

|                                                                                                                       |                        |                   |
|-----------------------------------------------------------------------------------------------------------------------|------------------------|-------------------|
| <b>x</b>                                                                                                              | <i>UnitValueString</i> | <i>Read/Write</i> |
| Set/get x coordinate of the point.                                                                                    |                        |                   |
| <b>y</b>                                                                                                              | <i>UnitValueString</i> | <i>Read/Write</i> |
| Set/get y coordinate of the point.                                                                                    |                        |                   |
| <b>flags</b>                                                                                                          | <i>short</i>           | <i>Read/Write</i> |
| Set/get flag information. Each flag is represented by one bit in the word. See the enumeration type PolygonPointFlag. |                        |                   |

|                                                                       |             |                   |
|-----------------------------------------------------------------------|-------------|-------------------|
| <b>flagAfter</b>                                                      | <i>bool</i> | <i>Read/Write</i> |
| Set/get/clear flag combination for “after” control point.             |             |                   |
| <b>flagAfterConstraint</b>                                            | <i>bool</i> | <i>Read/Write</i> |
| Set/get/clear flag combination for “after constraint” control point.  |             |                   |
| <b>flagBefore</b>                                                     | <i>bool</i> | <i>Read/Write</i> |
| Set/get/clear flag combination for “before” control point.            |             |                   |
| <b>flagBeforeConstraint</b>                                           | <i>bool</i> | <i>Read/Write</i> |
| Set/get/clear flag combination for “before constraint” control point. |             |                   |
| <b>flagFirst</b>                                                      | <i>bool</i> | <i>Read/Write</i> |
| Set/get/clear flag combination for “first” sizing point.              |             |                   |
| <b>flagLast</b>                                                       | <i>bool</i> | <i>Read/Write</i> |
| Set/get/clear flag combination for “last” sizing point.               |             |                   |

**Examples:**

```
var point = [335.56, 370.928, "F"];
var point = [304.554, 643.618];
var point = new Point(["15,3cm", "462pt", PolygonPointFlag.Control |
PolygonPointFlag.Before | PolygonPointFlag.Constraint]);
```

## Class: PointList

The class *PointList* implements an object that describes a polygon points list. The polygon point list is composed with the points. (See the chapter “Class: Point” for the point description.). All points are shifted to one level array consecutively. The control points must be set before or after the sizing point that is affected by these control points. When the polygon point list contains more polygon point lists, the flags “first” and “last” sizing point must be used.

The point coordinates are related to the page.

**Constructor:**

```
new PointList()
new PointList(point list constant)
```

**Properties:**

|                                                                                     |             |          |
|-------------------------------------------------------------------------------------|-------------|----------|
| <b>length</b>                                                                       | <i>int</i>  | <i>R</i> |
| Returns the number of points in the polygon point list.                             |             |          |
| <b>isGroup</b>                                                                      | <i>bool</i> | <i>R</i> |
| Returns true when point list contains more subpolygons, otherwise it returns false. |             |          |
| <b>numberOfSizingPoints</b>                                                         | <i>int</i>  | <i>R</i> |
| Returns the number of sizing points.                                                |             |          |
| <b>numberOfSubPolygons</b>                                                          | <i>int</i>  | <i>R</i> |
| Returns the number of subpolygons in the polygon points list.                       |             |          |

**Methods:****getPoint** (*int index*) *Point*Returns the Point object from the polygon point list from its *index* position.**setPoint** (*Point point, int index*) *void*Replaces the point object to the polygon point list at its *index* position.**insertPoint** (*Point point, int index*) *void*Inserts a new point object in the polygon point list at its *index* position. The current object and all following objects move up.**appendPoint** (*Point point*) *void*

Appends a new point object at the end of the polygon point list.

**deletePoint** (*int index*) *void*

Deletes the point object from its index position in the polygon point list.

**Examples 1: The polygon point list contains two subpolygons.**

```
var pointArray = [[335.56, 370.928, "F"], [244.928, 465.535], [304.554,
643.618],
    [512.848, 662.945], [562.139, 436.914], [457.992, 380.468, "B"],
    [461.172, 350.258, "L"], [464.352, 320.047, "A"], [350.665, 464.74, "F"],
    [380.08, 544.241], [420.626, 547.421, "B"], [420.626, 547.421],
    [400.626, 527.421, "A"], ["15,3cm", "462pt"], [391.689, 407.379, "B"],
    [392.638, 447.249, "L"], [393.58, 487.119, "A"]];

var pointList = new PointList(pointArray);
var doc = application.currentDocument;
if (doc) {
    var obj = doc.objects().create(Object.TextFrame);
    if (obj) {
        obj.form = FrameForm.Polygon;
        obj.pageGeometryPoints = pointList;
        doc.currentPage.add(obj);
    }
}
```

**Examples 2:**

```
var pointArray = [[335.56, 370.928, "F"], [244.928, 465.535], [304.554,
643.618],
    [512.848, 662.945], [562.139, 436.914], [457.992, 380.468, "B"],
    [461.172, 350.258, "L"], [464.352, 320.047, "A"], [350.665, 464.74, "F"],
    [380.08, 544.241], [420.626, 547.421, "B"], [420.626, 547.421],
    [400.626, 527.421, "A"], ["15,3cm", "462pt"], [391.689, 407.379, "B"],
    [392.638, 447.249, "L"], [393.58, 487.119, "A"]];

var arr = new PointList(pointArray);

for (var j = 0; j < arr.length; j++) {
    var p = arr.getPoint(j);
    messageBox.information("Msg:", "Point: [" + p.x + "," + p.y + "," +
p.flags + " " + "\nFlagAfter: " + p.flagAfter +
```

```
        "\nFlagAfterConstraint: " + p.flagAfterConstraint +
        "\nFlagBefore: " + p.flagBefore +
        "\nFlagBeforeConstraint: " + p.flagBeforeConstraint +
        "\nFlagFirst: " + p.flagFirst +
        "\nFlagLast: " + p.flagLast);
    }

    arr.setPoint(new Point("40mm", "200mm", PolygonPointFlag.First), 0);
    var p = arr.getPoint(5);
    p.flagBeforeConstraint = true;
    arr.setPoint(p, 5);
    arr.appendPoint(new Point("14cm", "250mm"));
```

## Class: BoundingBox

This simple class *BoundingBox* contains 4 numbers which represent the bounding box of the object. The properties *skew*, *rotation* and *mirrored* have an impact on the values.

### Constructor:

```
new BoundingBox(UnitValueString x, UnitValueString y,
    UnitValueString width, UnitValueString height)
new BoundingBox(Array [x, y, width, height])
```

### Properties:

|                             |                        |                   |
|-----------------------------|------------------------|-------------------|
| <b>x</b>                    | <i>UnitValueString</i> | <i>Read/Write</i> |
| X-coordinate of the object. |                        |                   |
| <b>y</b>                    | <i>UnitValueString</i> | <i>Read/Write</i> |
| Y-coordinate of the object. |                        |                   |
| <b>width</b>                | <i>UnitValueString</i> | <i>Read/Write</i> |
| Width of the object.        |                        |                   |
| <b>height</b>               | <i>UnitValueString</i> | <i>Read/Write</i> |
| Height of the object.       |                        |                   |

## Class: Matrix

The class *Matrix* holds 6 parameters for the object shape transformation. The transformation formula is as follows:

$$\begin{aligned}x_{\text{new}} &= m11 * x + m21 * y + dx \\ y_{\text{new}} &= m22 * y + m12 * x + dy\end{aligned}$$

The class is used in the class *Object*.

### Constructor:

```
new Matrix();
new Matrix(float m11, float m12, float m21, float m22, UnitValueString dx, UnitValueString
dy)
new Matrix(Array [m11, m12, m21, m22, dx, dy])
```

## Properties:

|            |              |                   |
|------------|--------------|-------------------|
| <b>m11</b> | <i>float</i> | <i>Read/Write</i> |
| <b>m12</b> | <i>float</i> | <i>Read/Write</i> |
| <b>m21</b> | <i>float</i> | <i>Read/Write</i> |
| <b>m22</b> | <i>float</i> | <i>Read/Write</i> |
| <b>dx</b>  | <i>float</i> | <i>Read/Write</i> |
| <b>dy</b>  | <i>float</i> | <i>Read/Write</i> |

The properties *dx* and *dy* return the translation in points.







# Brushes and Colors

# Brushes and Colors

## Introduction

There are two types of brush objects: *ColorBrush* and *BlendBrush*. The class *Brush* is a root class of both brush objects. The class *Brushes* holds the repository of all (predefined and created) brushes.

### Color Constant Enumeration Types:

**Color:** *Black, Blue, Bronze, Brown, Cyan, Gold, Green, Magenta, Orange, Pink, Red, Salmon, Silver, Turquoise, Violet, White, Yellow*

## Class: Brushes

The predefined repository of brushes can be obtained from the document using the *brushes* property of the Document object:

```
var mybrushes = doc.brushes;
```

The class works as a factory for creating a new brush.

### Properties:

|                                              |            |             |
|----------------------------------------------|------------|-------------|
| <b>length</b>                                | <i>int</i> | <i>Read</i> |
| Returns number of brushes in the repository. |            |             |

### Methods:

|                                                                                                                                        |                   |
|----------------------------------------------------------------------------------------------------------------------------------------|-------------------|
| <b>addColor(<i>name</i>)</b>                                                                                                           | <i>ColorBrush</i> |
| Adds a new brush with “name” to the list of brushes and returns a ColorBrush object. If “name” is not unique, the exception is caused. |                   |
| <b>addBlend(<i>name</i>)</b>                                                                                                           | <i>BlendBrush</i> |
| Adds a brush with “name” to the list of brushes and returns BlendBrush. If “name” is not unique, the exception is caused.              |                   |
| <b>at(<i>index</i>)</b>                                                                                                                | <i>Brush</i>      |
| Returns <i>ColorBrush</i> or <i>BlendBrush</i> from the index position in the repository.                                              |                   |
| <b>find(<i>name</i>)</b>                                                                                                               | <i>Brush</i>      |
| Returns the <i>ColorBrush</i> or <i>BlendBrush</i> found or <i>null</i> if nothing is found.                                           |                   |

## Class: Brush

The class *Brush* is a root class of brush objects.

### Properties:

|                                                                                                                                                                                             |               |                   |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|-------------------|
| <b>uniqueName</b>                                                                                                                                                                           | <i>String</i> | <i>Read/Write</i> |
| Get/Set the unique name of brush object. The names of the predefined brushes (in square brackets, e.g. “[Black]”) cannot be changed. If the new name is not unique, an exception is thrown. |               |                   |

|                                                                                                         |               |                   |
|---------------------------------------------------------------------------------------------------------|---------------|-------------------|
| <b>localizedName</b>                                                                                    | <i>String</i> | <i>Read</i>       |
| Get the localized name of brush object.                                                                 |               |                   |
| <b>hidden</b>                                                                                           | <i>bool</i>   | <i>Read/Write</i> |
| Get/Set the flag hidden. The brushes [Black], [Cyan], [Magenta], [White] and [Yellow] cannot be hidden. |               |                   |
| <b>Methods:</b>                                                                                         |               |                   |
| <b>remove( )</b>                                                                                        | <i>bool</i>   |                   |
| Removes Brush from the list of brushes.                                                                 |               |                   |

## Class: ColorBrush

The object defines a color brush.

### Constructor:

*new* **ColorBrush**(*Color color*)

Returns the predefined brush of the given color constant (see Color Constant Enumeration Types) from the current document, e.g. *new ColorBrush(Color.Red)*.

### Properties:

|                                                                                                                            |                               |                   |
|----------------------------------------------------------------------------------------------------------------------------|-------------------------------|-------------------|
| <b>rgb</b>                                                                                                                 | <i>Rgb</i>                    | <i>Read/Write</i> |
| Get/Set the object of the RGB color model (object Rgb with needed values)                                                  |                               |                   |
| <b>cmyk</b>                                                                                                                | <i>Cmyk</i>                   | <i>Read/Write</i> |
| Get/Set the object of the CMYK color model (object Cmyk with needed values)                                                |                               |                   |
| <b>hsv</b>                                                                                                                 | <i>Hsv</i>                    | <i>Read/Write</i> |
| Get/Set the object of the HSV color model (object Hsv with needed values)                                                  |                               |                   |
| <b>lab</b>                                                                                                                 | <i>Lab</i>                    | <i>Read/Write</i> |
| Get/Set the object of the LAB color model (object Lab with needed values)                                                  |                               |                   |
| <b>colorSeparation</b>                                                                                                     | <i>ColorSeparation (enum)</i> | <i>Read/Write</i> |
| Get/Set a type of color separation. See the enumeration type ColorSeparation.                                              |                               |                   |
| <b>screenAngle</b>                                                                                                         | <i>ScreenAngle (enum)</i>     | <i>Read/Write</i> |
| Get/Set a type of screen angle. See the enumeration type ScreenAngle. It is used for ColorSeparation.SpotColor option only |                               |                   |
| <b>colorModel</b>                                                                                                          | <i>ColorModel (enum)</i>      | <i>Read</i>       |
| Get the current color model. See the enumeration type ColorModel.                                                          |                               |                   |
| <b>indexModel</b>                                                                                                          | <i>IndexModel (enum)</i>      | <i>Read</i>       |
| Get the current index model. See the enumeration type IndexModel                                                           |                               |                   |
| <b>indexKey</b>                                                                                                            | <i>String</i>                 | <i>Read</i>       |
| Get the current index key.                                                                                                 |                               |                   |

## Class: BlendBrush

The object defines the brush type that is composed of a colors and the transition between them. Two Stop objects are created by default when new *BlendBrush* object is created.

### Properties:

|                                                                              |                         |                   |
|------------------------------------------------------------------------------|-------------------------|-------------------|
| <b>type</b>                                                                  | <i>BlendType (enum)</i> | <i>Read/Write</i> |
| Get/Set type of the blend brush. See the enumeration type <i>BlendType</i> . |                         |                   |
| <b>stopCount</b>                                                             | <i>short</i>            | <i>Read</i>       |
| Get the number of Stop objects.                                              |                         |                   |

### Methods:

|                                                                              |             |  |
|------------------------------------------------------------------------------|-------------|--|
| <b>stopAt</b> ( <i>int index</i> )                                           | <i>Stop</i> |  |
| Get the Stop object from the index position.                                 |             |  |
| <b>setStopAt</b> ( <i>int index, Stop object</i> )                           | <i>void</i> |  |
| Update the Stop object on the index position in the BlendBrush object.       |             |  |
| <b>removeStopAt</b> ( <i>int index</i> )                                     | <i>bool</i> |  |
| Removes the Stop object from the index position in the BlendBrush object.    |             |  |
| Returns true if object was deleted succesfully, otherwise false value.       |             |  |
| <b>appendStop</b> ( <i>Stop object</i> )                                     | <i>void</i> |  |
| Appends the Stop object after the last Stop object of the BlendBrush object. |             |  |

### Examples:

```
var doc = application.currentDocument;
if (doc) {

    var brushes = doc.brushes;
    if (brushes) {

        var stop1 = new Stop(Color.Green, 5, 6, 7, 8);
        var stop2 = new Stop(Color.Yellow);
        stop2.shade = 15;
        stop2.opacity = 16;
        stop2.position = 17;
        stop2.focusPosition = 18;
        var stop3 = new Stop(Color.Red, 25, 26, 27, 28);
        var stop4 = new Stop(Color.Blue, 35, 36, 37, 38);

        try {
            var bc = brushes.addBlend("BlendNew");
            bc.type = BlendType.Radial;
            messageBox.information("INFO", "Stop Count: " + bc.stopCount);
            bc.setStopAt(1, stop1);
            bc.appendStop(stop2);
            bc.appendStop(stop3);
            bc.appendStop(stop4);
        }
    }
}
```

```

        messageBox.information("INFO", "Stop Count: " + bc.stopCount);
        bc.removeStopAt(3);
        messageBox.information("INFO", "Stop Count: " + bc.stopCount);
    }
    catch (err) {
        // color exists
    }
}
}

```

## Class: Stop

The Stop object holds the data about one color in the *BlendBrush* object.

### Constructor:

```

new Stop(ColorBrush brush)
new Stop(ColorBrush brush, float shade, float opacity,
        float position, float focusPosition)
new Stop(Color color)
new Stop(Color color, float shade, float opacity,
        float position, float focusPosition)

```

### Properties:

| Property          | Type              | Access            |
|-------------------|-------------------|-------------------|
| <b>colorBrush</b> | <i>ColorBrush</i> | <i>Read/Write</i> |

Get/Set the color brush.

| Property     | Type         | Access            |
|--------------|--------------|-------------------|
| <b>shade</b> | <i>float</i> | <i>Read/Write</i> |

Get/Set the shade of the color. The range is 0-100. [%]

| Property       | Type         | Access            |
|----------------|--------------|-------------------|
| <b>opacity</b> | <i>float</i> | <i>Read/Write</i> |

Get/Set the opacity of the color. The range is 0-100. [%]

| Property        | Type         | Access            |
|-----------------|--------------|-------------------|
| <b>position</b> | <i>float</i> | <i>Read/Write</i> |

Get/Set the position of the color. The range is 0-100. [%]

| Property             | Type         | Access            |
|----------------------|--------------|-------------------|
| <b>focusPosition</b> | <i>float</i> | <i>Read/Write</i> |

Get/Set the focus position. The range is 0-100. [%]

## Class: Rgb

The class defines the RGB color model.

### Constructor:

```

new Rgb(int red, int green, int blue)

```

### Properties:

| Property   | Type               | Access            |
|------------|--------------------|-------------------|
| <b>red</b> | <i>int (0-255)</i> | <i>Read/Write</i> |

Get/Set a red part of the color model.

| Property     | Type               | Access            |
|--------------|--------------------|-------------------|
| <b>green</b> | <i>int (0-255)</i> | <i>Read/Write</i> |

Get/Set a green part of the color model.

---

|             |                    |                   |
|-------------|--------------------|-------------------|
| <b>blue</b> | <i>int (0-255)</i> | <i>Read/Write</i> |
|-------------|--------------------|-------------------|

Get/Set a blue part of the color model.

## Class: Cmyk

The class defines the CMYK color model.

### Constructor:

**new Cmyk** (*int cyan, int magenta, int yellow, int key*)

### Properties:

---

|             |                    |                   |
|-------------|--------------------|-------------------|
| <b>cyan</b> | <i>int (0-100)</i> | <i>Read/Write</i> |
|-------------|--------------------|-------------------|

Get/Set a cyan part of the color model.

---

|                |                    |                   |
|----------------|--------------------|-------------------|
| <b>magenta</b> | <i>int (0-100)</i> | <i>Read/Write</i> |
|----------------|--------------------|-------------------|

Get/Set a magenta part of the color model.

---

|               |                    |                   |
|---------------|--------------------|-------------------|
| <b>yellow</b> | <i>int (0-100)</i> | <i>Read/Write</i> |
|---------------|--------------------|-------------------|

Get/Set a yellow part of the color model.

---

|            |                    |                   |
|------------|--------------------|-------------------|
| <b>key</b> | <i>int (0-100)</i> | <i>Read/Write</i> |
|------------|--------------------|-------------------|

Get/Set a key for the color model.

## Class: Hsv

The class defines the HSV color model.

### Constructor:

**new Hsv** (*int hue, int saturation, int value*)

### Properties:

---

|            |                             |                   |
|------------|-----------------------------|-------------------|
| <b>hue</b> | <i>int (0-359) [degree]</i> | <i>Read/Write</i> |
|------------|-----------------------------|-------------------|

Get/Set a hue for the color model.

---

|                   |                    |                   |
|-------------------|--------------------|-------------------|
| <b>saturation</b> | <i>int (0-255)</i> | <i>Read/Write</i> |
|-------------------|--------------------|-------------------|

Get/Set a saturation for the color model.

---

|              |                    |                   |
|--------------|--------------------|-------------------|
| <b>value</b> | <i>int (0-255)</i> | <i>Read/Write</i> |
|--------------|--------------------|-------------------|

Get/Set a value for the color model.

## Class: Lab

The class defines the LAB color model.

### Constructor:

**new Lab** (*int lightness, int positionA, int positionB*)

**Properties:**

|                  |                    |                   |
|------------------|--------------------|-------------------|
| <b>lightness</b> | <i>int (0-100)</i> | <i>Read/Write</i> |
|------------------|--------------------|-------------------|

Get/Set a lightness for the color model.

|          |                       |                   |
|----------|-----------------------|-------------------|
| <b>a</b> | <i>int (-128-127)</i> | <i>Read/Write</i> |
|----------|-----------------------|-------------------|

Get/Set a position between green and red for the color model. Negative values indicate green while positive values indicate red.

|          |                       |                   |
|----------|-----------------------|-------------------|
| <b>b</b> | <i>int (-128-127)</i> | <i>Read/Write</i> |
|----------|-----------------------|-------------------|

Get/Set a position between blue and yellow for the color model. Negative values indicate blue and positive values indicate yellow.

## Examples

**Example 1:**

```
var doc = application.currentDocument;
if (doc) {
    var brushes = doc.brushes;

    var bg1 = brushes.addBlend("MyBlendBrush");
    var bc1 = brushes.addColor("MyColorBrush");

    var names = "";
    for (var index = 0; index < brushes.length; index++) {
        var brush = brushes.at(index);
        names = names + brush.uniqueName + " : " + brush.hidden + "\n";
    }
    messageBox.information("Brush Unique Names", names);
}
```

**Example 2:**

```
var doc = application.currentDocument;
if (doc) {
    var cb = doc.brushes.addColor("MyColor2");
    cb.rgb = new Rgb(200, 100, 0);
    var tp = new TextPreferences();
    tp.lineNumbersColor = cb;
    doc.textPreferences = tp;
}
```

**Example 3:**

```
var vRgb = new Rgb(12, 23, 130);
var doc = application.currentDocument;
if (doc) {

    var brushes = doc.brushes;
    if (brushes) {

        try {
            var bc1 = brushes.addColor("Color1");
        }
    }
}
```

```
catch (err) {
    // color exists
}

bc1.rgb = vRgb;
bc1.hsv = new Hsv(55, 33, 44);
bc1.cmyk = new Cmyk(11, 111, 12, 113);
bc1.colorSeparation = ColorSeparation.SpotColor;
bc1.screenAngle = ScreenAngle.Cyan;

messageBox.information(
    "CMYK", "Cyan: " + bc1.cmyk.cyan +
    "; Magenta: " + bc1.cmyk.magenta +
    "; Yellow: " + bc1.cmyk.yellow +
    "; Key: " + bc1.cmyk.key);
messageBox.information("CMYK2", "Cyan: " +
    doc.brushes.find("Color1").cmyk.cyan + "; Magenta: " +
    doc.brushes.find("Color1").cmyk.magenta + "; Yellow: " +
    doc.brushes.find("Color1").cmyk.yellow + "; Key: " +
    doc.brushes.find("Color1").cmyk.key);
if(bc1.colorModel == ColorModel.Cmyk)
    messageBox.information("Msg:", "ColorModel is 'CMYK'");
if(bc1.colorSeparation == ColorSeparation.SpotColor)
    messageBox.information("Msg:", "ColorSeparation of color " +
    bc1.uniqueName + " is 'SpotColor'");

var cba = brushes.find("Color1");
if(cba)
    cba.remove();

    var stop1 = new Stop(Color.Gold, 55, 60, 0, 33);
var bc2 = null;
    try {
        bc2 = brushes.addBlend("Color2");
    }
    catch (err) {
        // color exists
    }

    bc2.type = BlendType.Focus;
    bc2.setStopAt(0, stop1);
    bc2.setStopAt(1, new Stop(Color.Green, 88, 90, 95, 15));
}
}
```







# StyleSheets

# StyleSheets

## Class: StyleSheets

The predefined repository of style sheets can be obtained from the document using the *styleSheets* property of the Document object:

```
var mystylesheets = doc.styleSheets;
```

The class works as a factory for creating new style sheets.

### Methods:

---

**length** (*StyleSheetFamily familyType*) *int*

Returns the number of style sheets of the type 'familyType'. See the enumeration type *StyleSheetFamily*.

---

**create** (*String name, StyleSheetFamily familyType*) *StyleSheet*

Creates and returns a new style sheet with "familyType". If "name" is not unique, the *null* is returned.

---

**at** (*int index, StyleSheetFamily familyType*) *StyleSheet*

Returns one of the style sheet objects depending on the familyType from the index position in the repository. See the enumeration type *StyleSheetFamily*.

---

**find** (*String name*) *StyleSheet*

Returns the StyleSheet object found or *null* if nothing is found.

### Examples:

```
var doc = application.currentDocument;
if (doc) {
    var ss = doc.styleSheets;

    var ssChar1 = ss.find("MyCharacter1");
    if (!ssChar1)
        ssChar1 = ss.create("MyCharacter1", StyleSheetFamily.Character);
    var ssChar2 = ss.find("MyCharacter2");
    if (!ssChar2)
        ssChar2 = ss.create("MyCharacter2", StyleSheetFamily.Character);
    var ssPar = ss.find("MyParagraph");
    if (!ssPar)
        ssPar = ss.create("MyParagraph", StyleSheetFamily.Paragraph);

    var txt1 = "Number of Character Stylesheets: " +
ss.length(StyleSheetFamily.Character) + "\n";
    txt1 = txt1 + "Number of Paragraph Stylesheets: " +
ss.length(StyleSheetFamily.Paragraph) + "\n";
    messageBox.information("StyleSheets", txt1);

    var txt2 = "";
    for (var i = 0; i < ss.length(StyleSheetFamily.Character) ; i++) {
        var ssr = ss.at(i, StyleSheetFamily.Character);
        txt2 = txt2 + ssr.name + " or " + doc.styleSheets.at(i,
```

```

StyleSheetFamily.Character).name + "\n";
    }
    messageBox.information("StyleSheets", txt2);

    doc.styleSheets.at(1, StyleSheetFamily.Character).remove();
    ssPar.clear();
}

```

## Class: StyleSheet

The class *StyleSheet* is a root class of all types of style sheet objects: *CharacterStyleSheet*, *ParagraphStyleSheet*, *LayoutStyleSheet*, *PictureStyleSheet*, *GraphicStyleSheet*, *TableStyleSheet*, *TableRowStyleSheet*, *TableColumnStyleSheet*, *TableCellStyleSheet*.

### Properties:

| name                                                                       | String     | Read/Write |
|----------------------------------------------------------------------------|------------|------------|
| Get/Set a name of the style sheet.                                         |            |            |
| templateStyleSheet                                                         | StyleSheet | Read/Write |
| Get/Set a template style sheet. The same type of style sheet must be used. |            |            |

### Methods:

| remove()                                                 | boolean |
|----------------------------------------------------------|---------|
| Removes the style sheet from the style sheet repository. |         |
| clear()                                                  | void    |
| Clears a current setting of the style sheet.             |         |

### Examples:

```

var doc = application.currentDocument;
var stls = doc.styleSheets;
if (stls)
{
    messageBox.information("MSG", stls.length(StyleSheetFamily.Character));

    doc.styleSheets.at(0, StyleSheetFamily.Character).name = "Base";
    messageBox.information("MSG", doc.styleSheets.at(0,
StyleSheetFamily.Character).name);

    var ssBase = doc.styleSheets.at(0, StyleSheetFamily.Character);
    var templSS = doc.styleSheets.create("TemplateChar",
StyleSheetFamily.Character);
    ssBase.templateStyleSheet = templSS;
    messageBox.information("MSG",
doc.styleSheets.find("Base").templateStyleSheet.name);

    doc.styleSheets.find("TemplateChar").remove();
}

```

## Class: CharacterStyleSheet

The class holds a group of style parameters that are applied to the characters from the text range. The property ***errorStatus*** is set when the error is occurred or the property is not set in the style sheet.

### Properties:

|                                                                                  |                                       |                   |
|----------------------------------------------------------------------------------|---------------------------------------|-------------------|
| <b>font</b>                                                                      | <i>FontName</i>                       | <i>Read/Write</i> |
| Returns the font name and the font type or <i>null</i> when the font is not set. |                                       |                   |
| Sets the font name and the font type.                                            |                                       |                   |
| The parameter format is <i>&lt;font name&gt;-&lt;font type&gt;</i> .             |                                       |                   |
| Sample: <i>'Arial-Bold', 'Georgia-Bold Italic'</i>                               |                                       |                   |
| <b>fontSize</b>                                                                  | <i>FontSize</i>                       | <i>Read/Write</i> |
| Returns the font size object.                                                    |                                       |                   |
| Sets the font size and font size reference type.                                 |                                       |                   |
| <b>language</b>                                                                  | <i>Language</i>                       | <i>Read/Write</i> |
| Returns the language id or the value <i>NoLang</i> when the language is not set. |                                       |                   |
| Sets the language.                                                               |                                       |                   |
| <b>brush</b>                                                                     | <i>ColorBrush</i>                     | <i>Read/Write</i> |
| Returns the color of the brush or <i>null</i> when the color is not set.         |                                       |                   |
| Sets the color of the brush.                                                     |                                       |                   |
| <b>shade</b>                                                                     | <i>float</i>                          | <i>Read/Write</i> |
| Returns the shade of the color.                                                  |                                       |                   |
| Sets the shade of the color.                                                     |                                       |                   |
| <b>opacity</b>                                                                   | <i>float</i>                          | <i>Read/Write</i> |
| Returns the opacity of the color.                                                |                                       |                   |
| Sets the opacity of the color.                                                   |                                       |                   |
| <b>characterPosition</b>                                                         | <i>CharacterPosition (enum)</i>       | <i>Read/Write</i> |
| <b>characterUpperLowerCase</b>                                                   | <i>CharacterUpperLowerCase (enum)</i> | <i>Read/Write</i> |
| <b>underline</b>                                                                 | <i>UnderlineOptions</i>               | <i>Read/Write</i> |
| <b>strikethrough</b>                                                             | <i>StrikethroughOptions</i>           | <i>Read/Write</i> |
| <b>characterHeightScale</b>                                                      | <i>float</i>                          | <i>Read/Write</i> |
| <b>characterWidthScale</b>                                                       | <i>float</i>                          | <i>Read/Write</i> |
| <b>characterSpacing</b>                                                          | <i>float</i>                          | <i>Read/Write</i> |
| <b>characterKerning</b>                                                          | <i>CharacterKerning</i>               | <i>Read/Write</i> |
| <b>characterBaselineShift</b>                                                    | <i>float</i>                          | <i>Read/Write</i> |

|                                     |                                     |                   |
|-------------------------------------|-------------------------------------|-------------------|
| <b>characterBaselineOrientation</b> | <i>CharacterBaselineOrientation</i> | <i>Read/Write</i> |
| <b>characterRotation</b>            | <i>CharacterRotation (enum)</i>     | <i>Read/Write</i> |
| <b>characterLigatures</b>           | <i>bool</i>                         | <i>Read/Write</i> |
| <b>outlineOptions</b>               | <i>OutlineOptions</i>               | <i>Read/Write</i> |
| <b>characterFrame</b>               | <i>CharacterFrame</i>               | <i>Read/Write</i> |
| <b>characterBackground</b>          | <i>CharacterBackground</i>          | <i>Read/Write</i> |
| <b>characterRule</b>                | <i>CharacterRule</i>                | <i>Read/Write</i> |
| <b>tablesAbbreviationEntry</b>      | <i>TablesAbbreviation</i>           | <i>Read/Write</i> |
| <b>tablesBibliographyEntry</b>      | <i>TablesBibliography</i>           | <i>Read/Write</i> |
| <b>tablesContentsEntry</b>          | <i>TablesContents</i>               | <i>Read/Write</i> |
| <b>tablesIndexEntry</b>             | <i>TablesIndex</i>                  | <i>Read/Write</i> |
| <b>tablesPictureEntry</b>           | <i>TablesPicture</i>                | <i>Read/Write</i> |
| <b>tablesRunningTitleEntry</b>      | <i>TablesRunningTitle</i>           | <i>Read/Write</i> |
| <b>link</b>                         | <i>Link</i>                         | <i>Read/Write</i> |
| <b>openType</b>                     | <i>OpenType</i>                     | <i>Read/Write</i> |

**Methods:**

**remove**(*String propertyName*)      *void*

The method removes the property setting from the style sheet. The value *propertyName* must be set to used property name from following list: 'font', 'fontSize', 'language', 'brush', 'shade', 'opacity', 'underline', 'strikethrough', 'characterPosition', 'characterUpperLowerCase', 'outlineOptions', 'characterHeightScale', 'characterWidthScale', 'characterBaselineShift', 'characterBaselineOrientation', 'characterSpace', 'characterKerning', 'characterRotation', 'characterLigatures', 'characterFrame', 'characterRule', 'characterBackground', 'tablesIndex', 'tablesContents', 'tablesRunningTitle', 'tablesBibliography', 'tablesPicture', 'tablesAbbreviation'

## Class: ParagraphStyleSheet

The property ***errorStatus*** is set when the error is occurred or the property is not set in the style sheet.

**Properties:**

|                            |                            |                   |
|----------------------------|----------------------------|-------------------|
| <b>characterStyleSheet</b> | <i>CharacterStyleSheet</i> | <i>Read/Write</i> |
| <b>font</b>                | <i>FontName</i>            | <i>Read/Write</i> |

Returns the font name and the font type or *null* when the font is not set.

Sets the font name and the font type.

The parameter format is *<font name>-<font type>*.

Sample: *'Arial-Bold', 'Georgia-Bold Italic'*

|                 |                 |                   |
|-----------------|-----------------|-------------------|
| <b>fontSize</b> | <i>FontSize</i> | <i>Read/Write</i> |
|-----------------|-----------------|-------------------|

Returns the font size object. Sets the font size and the reference size type.

|                                                                                                       |                                       |                   |
|-------------------------------------------------------------------------------------------------------|---------------------------------------|-------------------|
| <b>language</b>                                                                                       | <i>Language</i>                       | <i>Read/Write</i> |
| Returns the language id or the value NoLang when the language is not set. Sets the language.          |                                       |                   |
| <b>brush</b>                                                                                          | <i>ColorBrush</i>                     | <i>Read/Write</i> |
| Returns the color of the brush or <i>null</i> when the color is not set. Sets the color of the brush. |                                       |                   |
| <b>shade</b>                                                                                          | <i>float</i>                          | <i>Read/Write</i> |
| Returns the shade of the color. Sets the shade of the color.                                          |                                       |                   |
| <b>opacity</b>                                                                                        | <i>float</i>                          | <i>Read/Write</i> |
| Returns the opacity of the color.                                                                     |                                       |                   |
| Sets the opacity of the color.                                                                        |                                       |                   |
| <b>underline</b>                                                                                      | <i>UnderlineOptions</i>               | <i>Read/Write</i> |
| <b>strikethrough</b>                                                                                  | <i>StrikethroughOptions</i>           | <i>Read/Write</i> |
| <b>characterPosition</b>                                                                              | <i>CharacterPosition (enum)</i>       | <i>Read/Write</i> |
| <b>characterUpperLowerCase</b>                                                                        | <i>CharacterUpperLowerCase (enum)</i> | <i>Read/Write</i> |
| <b>characterHeightScale</b>                                                                           | <i>float</i>                          | <i>Read/Write</i> |
| <b>characterWidthScale</b>                                                                            | <i>float</i>                          | <i>Read/Write</i> |
| <b>characterSpacing</b>                                                                               | <i>float</i>                          | <i>Read/Write</i> |
| <b>characterKerning</b>                                                                               | <i>CharacterKerning</i>               | <i>Read/Write</i> |
| <b>characterBaselineShift</b>                                                                         | <i>float</i>                          | <i>Read/Write</i> |
| <b>characterBaselineOrientation</b>                                                                   | <i>CharacterBaselineOrientation</i>   | <i>Read/Write</i> |
| <b>characterRotation</b>                                                                              | <i>CharacterRotation (enum)</i>       | <i>Read/Write</i> |
| <b>characterLigatures</b>                                                                             | <i>bool</i>                           | <i>Read/Write</i> |
| <b>outlineOptionsData</b>                                                                             | <i>OutlineOptions</i>                 | <i>Read/Write</i> |
| <b>characterFrame</b>                                                                                 | <i>CharacterFrame</i>                 | <i>Read/Write</i> |
| <b>characterBackground</b>                                                                            | <i>CharacterBackground</i>            | <i>Read/Write</i> |
| <b>characterRule</b>                                                                                  | <i>CharacterRule</i>                  | <i>Read/Write</i> |
| <b>tablesAbbreviationEntry</b>                                                                        | <i>TablesAbbreviation</i>             | <i>Read/Write</i> |
| <b>tablesBibliographyEntry</b>                                                                        | <i>TablesBibliography</i>             | <i>Read/Write</i> |
| <b>tablesContentsEntry</b>                                                                            | <i>TablesContents</i>                 | <i>Read/Write</i> |
| <b>tablesIndexEntry</b>                                                                               | <i>TablesIndex</i>                    | <i>Read/Write</i> |
| <b>tablesPictureEntry</b>                                                                             | <i>TablesPicture</i>                  | <i>Read/Write</i> |
| <b>tablesRunningTitleEntry</b>                                                                        | <i>TablesRunningTitle</i>             | <i>Read/Write</i> |
| <b>link</b>                                                                                           | <i>Link</i>                           | <i>Read/Write</i> |

|                                                                                                                                                                                                                                                                                             |                                  |                   |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------|-------------------|
| <b>openType</b>                                                                                                                                                                                                                                                                             | <i>OpenType</i>                  | <i>Read/Write</i> |
| <b>automaticCharacterSpacing</b>                                                                                                                                                                                                                                                            | <i>AutomaticCharacterSpacing</i> | <i>Read/Write</i> |
| <b>automaticCharacterWidth</b>                                                                                                                                                                                                                                                              | <i>AutomaticCharacterWidth</i>   | <i>Read/Write</i> |
| <b>automaticWordSpacing</b>                                                                                                                                                                                                                                                                 | <i>AutomaticWordSpacing</i>      | <i>Read/Write</i> |
| <b>writingDirection</b>                                                                                                                                                                                                                                                                     | <i>WritingDirection (enum)</i>   | <i>Read/Write</i> |
| <b>opticalAlignment</b>                                                                                                                                                                                                                                                                     | <i>OpticalAlignment (enum)</i>   | <i>Read/Write</i> |
| <b>supressLineNumbering</b>                                                                                                                                                                                                                                                                 | <i>bool</i>                      | <i>Read/Write</i> |
| <b>paragraphIndent</b>                                                                                                                                                                                                                                                                      | <i>UnitValueString</i>           | <i>Read/Write</i> |
| <b>paragraphLeftIndent</b>                                                                                                                                                                                                                                                                  | <i>UnitValueString</i>           | <i>Read/Write</i> |
| <b>paragraphRightIndent</b>                                                                                                                                                                                                                                                                 | <i>UnitValueString</i>           | <i>Read/Write</i> |
| <b>hyphenation</b>                                                                                                                                                                                                                                                                          | <i>Hyphenation</i>               | <i>Read/Write</i> |
| <b>enumerations</b>                                                                                                                                                                                                                                                                         | <i>Enumerations</i>              | <i>Read/Write</i> |
| <b>dropCaps</b>                                                                                                                                                                                                                                                                             | <i>DropCaps</i>                  | <i>Read/Write</i> |
| <b>textAlignment</b>                                                                                                                                                                                                                                                                        | <i>ParagraphAlignment (enum)</i> | <i>Read/Write</i> |
| Alignment of the paragraphs, e.g. <i>ParagraphAlignment.Justified</i> for justified text. See the enumeration type ParagraphAlignment. To justify single paragraphs of a text without a style sheet, use <i>Formatted-Text.setTextAlignment(ParagraphAlignment.Justified, begin, end)</i> . |                                  |                   |
| <b>paragraphLineSpacing</b>                                                                                                                                                                                                                                                                 | <i>UnitValueString</i>           | <i>Read/Write</i> |
| Set word “auto” to set automatically.                                                                                                                                                                                                                                                       |                                  |                   |
| <b>spaceToPreviousParagraph</b>                                                                                                                                                                                                                                                             | <i>UnitValueString</i>           | <i>Read/Write</i> |
| <b>spaceToNextParagraph</b>                                                                                                                                                                                                                                                                 | <i>UnitValueString</i>           | <i>Read/Write</i> |
| <b>widowsOrphans</b>                                                                                                                                                                                                                                                                        | <i>WidowsOrphans</i>             | <i>Read/Write</i> |
| <b>keepParagraphTogether</b>                                                                                                                                                                                                                                                                | <i>bool</i>                      | <i>Read/Write</i> |
| <b>baselineGridType</b>                                                                                                                                                                                                                                                                     | <i>BaselineGridType (enum)</i>   | <i>Read/Write</i> |
| <b>paragraphFrame</b>                                                                                                                                                                                                                                                                       | <i>ParagraphFrame</i>            | <i>Read/Write</i> |
| <b>paragraphBackground</b>                                                                                                                                                                                                                                                                  | <i>ParagraphBackground</i>       | <i>Read/Write</i> |
| <b>paragraphRuleAbove</b>                                                                                                                                                                                                                                                                   | <i>ParagraphRule</i>             | <i>Read/Write</i> |
| <b>paragraphRuleBelow</b>                                                                                                                                                                                                                                                                   | <i>ParagraphRule</i>             | <i>Read/Write</i> |
| <b>paragraphTabs</b>                                                                                                                                                                                                                                                                        | <i>Tabulators</i>                | <i>Read/Write</i> |

**Methods:**

**remove**(*String propertyName*)      *void*

The method removes the property setting from the style sheet. The value propertyName must be set to the used property name from the following list: ‘font’, ‘fontSize’, ‘language’, ‘brush’, ‘shade’, ‘opacity’, ‘underline’, ‘strikethrough’, ‘characterPosition’, ‘characterUpperLowerCase’, ‘outlineOptions’, ‘characterHeightScale’, ‘characterWidthScale’, ‘characterBaselineShift’, ‘characterBaselineOrientation’,

'characterSpace', 'characterKerning', 'characterRotation', 'characterLigatures', 'characterFrame', 'characterRule', 'characterBackground', 'tablesIndex', 'tablesContents', 'tablesRunningTitle', 'tablesBibliography', 'tablesPicture', 'tablesAbbreviation', 'writingDirection', 'opticalAlignment', 'supressLineNumbering', 'paragraphFrame', 'paragraphBackground', 'paragraphRuleAbove', 'paragraphRuleBelow', 'automaticCharacterSpacing', 'automaticCharacterWidth', 'automaticWordSpacing', 'paragraphLeftIndent', 'paragraphRightIndent', 'paragraphIndent', 'enumerations', 'dropCaps', 'hyphenation', 'textAlignment', 'widowsAndOrphans', 'keepParagraphTogether', 'baselineGrid', 'paragraphLineSpacing', 'spaceToPreviousParagraph', 'spaceToNextParagraph', 'paragraphTabs'

## Class: LayoutStyleSheet

### Properties:

|                                          |                                       |                   |
|------------------------------------------|---------------------------------------|-------------------|
| <b>lineOrientation</b>                   | <i>LayoutLineDirection (enum)</i>     | <i>Read/Write</i> |
| <b>lineJustification</b>                 | <i>LayoutLineJustification (enum)</i> | <i>Read/Write</i> |
| <b>justifiedSpaceBetweenLinesMaximum</b> | <i>UnitValueString</i>                | <i>Read/Write</i> |
| <b>paragraphSpaceMaximum</b>             | <i>UnitValueString</i>                | <i>Read/Write</i> |
| <b>spaceToPreviousLayout</b>             | <i>UnitValueString</i>                | <i>Read/Write</i> |
| <b>spaceToNextLayout</b>                 | <i>UnitValueString</i>                | <i>Read/Write</i> |
| <b>columns</b>                           | <i>LayoutColumnsInfo</i>              | <i>Read/Write</i> |
| <b>footnotes</b>                         | <i>LayoutFootnotes</i>                | <i>Read/Write</i> |

### Methods:

**remove(String propertyName)** *void*

The method removes the property setting from the stylesheet. The value propertyName must be set to used property name from following list: ',lineOrientation', ',lineJustification', ',spaceToPreviousLayout', ',spaceToNextLayout', ',maximumSpaceBetweenLines', ',maximumSpaceBetweenParagraphs', ',columns', ',footnotes'

## Class: PictureStyleSheet

### Properties:

|                             |               |                   |
|-----------------------------|---------------|-------------------|
| <b>horizontalScale</b>      | <i>float</i>  | <i>Read/Write</i> |
| <b>verticalScale</b>        | <i>float</i>  | <i>Read/Write</i> |
| <b>offsetX</b>              | <i>String</i> | <i>Read/Write</i> |
| <b>offsetY</b>              | <i>String</i> | <i>Read/Write</i> |
| <b>rotation</b>             | <i>float</i>  | <i>Read/Write</i> |
| <b>skew</b>                 | <i>float</i>  | <i>Read/Write</i> |
| <b>mirroredHorizontally</b> | <i>bool</i>   | <i>Read/Write</i> |

|                           |              |                   |
|---------------------------|--------------|-------------------|
| <b>mirroredVertically</b> | <i>bool</i>  | <i>Read/Write</i> |
| <b>halftoneAngle</b>      | <i>float</i> | <i>Read/Write</i> |
| <b>halftoneScreen</b>     | <i>float</i> | <i>Read/Write</i> |

## Methods:

**remove**(*String propertyName*)      *void*

The method removes the property setting from the style sheet. The value propertyName must be set to used property name from following list: horizontalScale, verticalScale, offsetX, offsetY, rotation, skew, mirroredHorizontally, mirroredVertically, halftoneAngle, halftoneScreen

## Class: GraphicStyleSheet

## Properties:

|                         |                   |                   |
|-------------------------|-------------------|-------------------|
| <b>x</b>                | <i>String</i>     | <i>Read/Write</i> |
| <b>y</b>                | <i>String</i>     | <i>Read/Write</i> |
| <b>height</b>           | <i>String</i>     | <i>Read/Write</i> |
| <b>width</b>            | <i>String</i>     | <i>Read/Write</i> |
| <b>rotation</b>         | <i>float</i>      | <i>Read/Write</i> |
| <b>skew</b>             | <i>float</i>      | <i>Read/Write</i> |
| <b>mirrored</b>         | <i>bool</i>       | <i>Read/Write</i> |
| <b>fillBrush</b>        | <i>Brush</i>      | <i>Read/Write</i> |
| <b>fillShade</b>        | <i>float</i>      | <i>Read/Write</i> |
| <b>fillBlendAngle</b>   | <i>float</i>      | <i>Read/Write</i> |
| <b>baseOpacity</b>      | <i>float</i>      | <i>Read/Write</i> |
| <b>fillOpacity</b>      | <i>float</i>      | <i>Read/Write</i> |
| <b>contentOpacity</b>   | <i>float</i>      | <i>Read/Write</i> |
| <b>printable</b>        | <i>bool</i>       | <i>Read/Write</i> |
| <b>guideObject</b>      | <i>bool</i>       | <i>Read/Write</i> |
| <b>lineColor</b>        | <i>ColorBrush</i> | <i>Read/Write</i> |
| <b>lineShade</b>        | <i>float</i>      | <i>Read/Write</i> |
| <b>lineOpacity</b>      | <i>float</i>      | <i>Read/Write</i> |
| <b>lineWidth</b>        | <i>String</i>     | <i>Read/Write</i> |
| <b>lineCornerRadius</b> | <i>String</i>     | <i>Read/Write</i> |
| <b>lineStyle</b>        | <i>String</i>     | <i>Read/Write</i> |
| <b>lineStart</b>        | <i>LineCap</i>    | <i>Read/Write</i> |

|                                      |                         |                   |
|--------------------------------------|-------------------------|-------------------|
| <b>lineEnd</b>                       | <i>LineCap</i>          | <i>Read/Write</i> |
| <b>runaround</b>                     | <i>Runaround</i>        | <i>Read/Write</i> |
| <b>pictureIndividualAliasContent</b> | <i>bool</i>             | <i>Read/Write</i> |
| <b>leftTextIndent</b>                | <i>String</i>           | <i>Read/Write</i> |
| <b>topTextIndent</b>                 | <i>String</i>           | <i>Read/Write</i> |
| <b>rightTextIndent</b>               | <i>String</i>           | <i>Read/Write</i> |
| <b>bottomTextIndent</b>              | <i>String</i>           | <i>Read/Write</i> |
| <b>linkableWithOriginal</b>          | <i>bool</i>             | <i>Read/Write</i> |
| <b>individualAliasContent</b>        | <i>bool</i>             | <i>Read/Write</i> |
| <b>baselinePosition</b>              | <i>BaselinePosition</i> | <i>Read/Write</i> |
| <b>baselineDescenders</b>            | <i>bool</i>             | <i>Read/Write</i> |
| <b>columnCount</b>                   | <i>int</i>              | <i>Read/Write</i> |
| <b>columnGutter</b>                  | <i>String</i>           | <i>Read/Write</i> |
| <b>columnLines</b>                   | <i>ColumnLines</i>      | <i>Read/Write</i> |
| <b>baselineGrid</b>                  | <i>BaselineGrid</i>     | <i>Read/Write</i> |
| <b>header</b>                        | <i>HeaderFooter</i>     | <i>Read/Write</i> |
| <b>footer</b>                        | <i>HeaderFooter</i>     | <i>Read/Write</i> |
| <b>footnotes</b>                     | <i>GraphicFootnotes</i> | <i>Read/Write</i> |
| <b>dropShadow</b>                    | <i>Shadow</i>           | <i>Read/Write</i> |
| <b>innerShadow</b>                   | <i>Shadow</i>           | <i>Read/Write</i> |
| <b>innerGlow</b>                     | <i>InnerGlow</i>        | <i>Read/Write</i> |
| <b>outerGlow</b>                     | <i>OuterGlow</i>        | <i>Read/Write</i> |
| <b>colorOverlay</b>                  | <i>ColorOverlay</i>     | <i>Read/Write</i> |
| <b>reflection</b>                    | <i>Reflection</i>       | <i>Read/Write</i> |
| <b>transparency</b>                  | <i>Transparency</i>     | <i>Read/Write</i> |

**Method:**

**remove**(*String propertyName*)      *void*

The method removes the property setting from the stylesheet. The value *propertyName* must be set to used property name from following list: 'x', 'y', 'height', 'width', 'rotation', 'skew', 'mirrored', 'fillBrush', 'fillShade', 'fillBlendAngle', 'fillOpacity', 'baseOpacity', 'contentOpacity', 'printable', 'guideObject', 'lineColor', 'lineShade', 'lineOpacity', 'lineWidth', 'lineCornerRadius', 'lineStyle', 'lineStart', 'lineEnd', 'runaround', 'pictureIndividualAliasContent', 'leftTextIndent', 'topTextIndent', 'rightTextIndent', 'bottomTextIndent', 'linkableWithOriginal', 'individualAliasContent', 'baselinePosition', 'baselineDescenders', 'columnCount', 'columnGutter', 'columnLines', 'baselineGrid', 'header', 'footer',

'footnotes', 'dropShadow', 'innerShadow', 'innerGlow', 'outerGlow', 'colorOverlay', 'reflection', 'transparency'

## Class: TableStyleSheet

### Properties:

|                               |                           |                   |
|-------------------------------|---------------------------|-------------------|
| <b>fillBrush</b>              | <i>Brush</i>              | <i>Read/Write</i> |
| <b>fillShade</b>              | <i>float</i>              | <i>Read/Write</i> |
| <b>fillBlendAngle</b>         | <i>float</i>              | <i>Read/Write</i> |
| <b>lineBrush</b>              | <i>ColorBrush</i>         | <i>Read/Write</i> |
| <b>lineShade</b>              | <i>float</i>              | <i>Read/Write</i> |
| <b>lineStyle</b>              | <i>LineStyleString</i>    | <i>Read/Write</i> |
| <b>lineWidth</b>              | <i>UnitValueString</i>    | <i>Read/Write</i> |
| <b>separator</b>              | <i>TableLineSeparator</i> | <i>Read/Write</i> |
| <b>headerRows</b>             | <i>bool</i>               | <i>Read</i>       |
| <b>headerRowsData</b>         | <i>RowColumnStyle</i>     | <i>Read</i>       |
| <b>footerRows</b>             | <i>bool</i>               | <i>Read</i>       |
| <b>footerRowsData</b>         | <i>RowColumnStyle</i>     | <i>Read</i>       |
| <b>alternatingRows</b>        | <i>bool</i>               | <i>Read</i>       |
| <b>alternatingRowsData</b>    | <i>RowColumnStyleAlt</i>  | <i>Read</i>       |
| <b>leftColumns</b>            | <i>bool</i>               | <i>Read</i>       |
| <b>leftColumnsData</b>        | <i>RowColumnStyle</i>     | <i>Read</i>       |
| <b>rightColumns</b>           | <i>bool</i>               | <i>Read</i>       |
| <b>rightColumnsData</b>       | <i>RowColumnStyle</i>     | <i>Read</i>       |
| <b>alternatingColumns</b>     | <i>bool</i>               | <i>Read</i>       |
| <b>alternatingColumnsData</b> | <i>RowColumnStyleAlt</i>  | <i>Read</i>       |

### Methods:

|                              |                                          |
|------------------------------|------------------------------------------|
| <b>setHeaderRows</b>         | <i>(bool on, RowColumnStyle data)</i>    |
| <b>setFooterRows</b>         | <i>(bool on, RowColumnStyle data)</i>    |
| <b>setAlternatingRows</b>    | <i>(bool on, RowColumnStyleAlt data)</i> |
| <b>setLeftColumns</b>        | <i>(bool on, RowColumnStyle data)</i>    |
| <b>setRightColumns</b>       | <i>(bool on, RowColumnStyle data)</i>    |
| <b>setAlternatingColumns</b> | <i>(bool on, RowColumnStyleAlt data)</i> |
| <b>remove</b>                | <i>(String propertyName) void</i>        |

The method removes the property setting from the stylesheet. The value propertyName must be set to used property name from following list: 'fillBrush', 'fillShade', 'fillBlendAngle', 'lineBrush', 'lineShade', 'lineStyle', 'lineWidth', 'separator', 'headerRows', 'footerRow', 'alternatingRows', 'leftColumns', 'rightColumns', 'alternatingColumns'

## Class: RowColumnStyle

The class is used in the class TableStyleSheet.

### Constructor:

```
new RowColumnStyle()  
new RowColumnStyle(int count, String styleSheetName)
```

### Properties:

|                         |                   |                   |
|-------------------------|-------------------|-------------------|
| <b>active</b>           | <i>bool</i>       | <i>Read</i>       |
| <b>count</b>            | <i>int</i>        | <i>Read/Write</i> |
| <b>styleSheet</b>       | <i>StyleSheet</i> | <i>Read/Write</i> |
| <b>styleSheetByName</b> | <i>String</i>     | <i>Read/Write</i> |

Set name of row or column stylesheet.

## Class: RowColumnStyleAlt

The class is used in the class TableStyleSheet.

### Constructor:

```
new RowColumnStyleAlt()  
new RowColumnStyleAlt(  
    int firstCount, String firstStyleSheetName,  
    int secondCount, String secondStyleSheetName,  
    int skipFirstCount, int skipLastCount)
```

### Properties:

|                               |                   |                   |
|-------------------------------|-------------------|-------------------|
| <b>active</b>                 | <i>bool</i>       | <i>Read</i>       |
| <b>firstCount</b>             | <i>int</i>        | <i>Read/Write</i> |
| <b>firstStyleSheetByName</b>  | <i>String</i>     | <i>Read/Write</i> |
| <b>firstStyleSheet</b>        | <i>StyleSheet</i> | <i>Read/Write</i> |
| <b>secondCount</b>            | <i>int</i>        | <i>Read/Write</i> |
| <b>secondStyleSheetByName</b> | <i>String</i>     | <i>Read/Write</i> |
| <b>secondStyleSheet</b>       | <i>StyleSheet</i> | <i>Read/Write</i> |
| <b>skipFirstCount</b>         | <i>int</i>        | <i>Read/Write</i> |
| <b>skipLastCount</b>          | <i>int</i>        | <i>Read/Write</i> |

## Class: TableRowStyleSheet

### Properties:

|                                |                  |                   |
|--------------------------------|------------------|-------------------|
| <b>fillBrush</b>               | <i>Brush</i>     | <i>Read/Write</i> |
| <b>fillShade</b>               | <i>float</i>     | <i>Read/Write</i> |
| <b>fillBlendAngle</b>          | <i>float</i>     | <i>Read/Write</i> |
| <b>horizontalSeparator</b>     | <i>bool</i>      | <i>Read</i>       |
| <b>horizontalSeparatorData</b> | <i>LineStyle</i> | <i>Read</i>       |
| <b>verticalSeparator</b>       | <i>bool</i>      | <i>Read</i>       |
| <b>verticalSeparatorData</b>   | <i>LineStyle</i> | <i>Read</i>       |

### Methods:

**setHorizontalSeparator** (*bool on, LineStyle data*)

**setVerticalSeparator** (*bool on, LineStyle data*)

**remove**(*String propertyName*) *void*

The method removes the property setting from the stylesheet. The value propertyName must be set to used property name from following list: 'fillBrush', 'fillShade', 'fillBlendAngle', 'horizontalSeparator', 'verticalSeparator'

## Class: TableColumnStyleSheet

### Properties:

|                                |                  |                   |
|--------------------------------|------------------|-------------------|
| <b>fillBrush</b>               | <i>Brush</i>     | <i>Read/Write</i> |
| <b>fillShade</b>               | <i>float</i>     | <i>Read/Write</i> |
| <b>fillBlendAngle</b>          | <i>float</i>     | <i>Read/Write</i> |
| <b>horizontalSeparator</b>     | <i>bool</i>      | <i>Read</i>       |
| <b>horizontalSeparatorData</b> | <i>LineStyle</i> | <i>Read</i>       |
| <b>verticalSeparator</b>       | <i>bool</i>      | <i>Read</i>       |
| <b>verticalSeparatorData</b>   | <i>LineStyle</i> | <i>Read</i>       |

### Method:

**setHorizontalSeparator** (*bool on, LineStyle data*)

**setVerticalSeparator** (*bool on, LineStyle data*)

**remove**(*String propertyName*) *void*

The method removes the property setting from the stylesheet. The value propertyName must be set to used property name from following list: 'fillBrush', 'fillShade', 'fillBlendAngle', 'horizontalSeparator', 'verticalSeparator'

## Class: TableCellStyleSheet

### Properties:

|                            |                                   |                   |
|----------------------------|-----------------------------------|-------------------|
| <b>fillBrush</b>           | <i>Brush</i>                      | <i>Read/Write</i> |
| <b>fillShade</b>           | <i>float</i>                      | <i>Read/Write</i> |
| <b>fillBlendAngle</b>      | <i>float</i>                      | <i>Read/Write</i> |
| <b>horizontalAlignment</b> | <i>HorizontalAlignment (enum)</i> | <i>Read/Write</i> |
| <b>verticalAlignment</b>   | <i>VerticalAlignment (enum)</i>   | <i>Read/Write</i> |
| <b>rotation</b>            | <i>ContentRotation (enum)</i>     | <i>Read/Write</i> |
| <b>leftIndent</b>          | <i>UnitValueString</i>            | <i>Read/Write</i> |
| <b>topIndent</b>           | <i>UnitValueString</i>            | <i>Read/Write</i> |
| <b>rightIndent</b>         | <i>UnitValueString</i>            | <i>Read/Write</i> |
| <b>bottomIndent</b>        | <i>UnitValueString</i>            | <i>Read/Write</i> |
| <b>leftSeparator</b>       | <i>bool</i>                       | <i>Read</i>       |
| <b>leftSeparatorData</b>   | <i>LineStyle</i>                  | <i>Read</i>       |
| <b>topSeparator</b>        | <i>bool</i>                       | <i>Read</i>       |
| <b>topSeparatorData</b>    | <i>LineStyle</i>                  | <i>Read</i>       |
| <b>rightSeparator</b>      | <i>bool</i>                       | <i>Read</i>       |
| <b>rightSeparatorData</b>  | <i>LineStyle</i>                  | <i>Read</i>       |
| <b>bottomSeparator</b>     | <i>bool</i>                       | <i>Read</i>       |
| <b>bottomSeparatorData</b> | <i>LineStyle</i>                  | <i>Read</i>       |

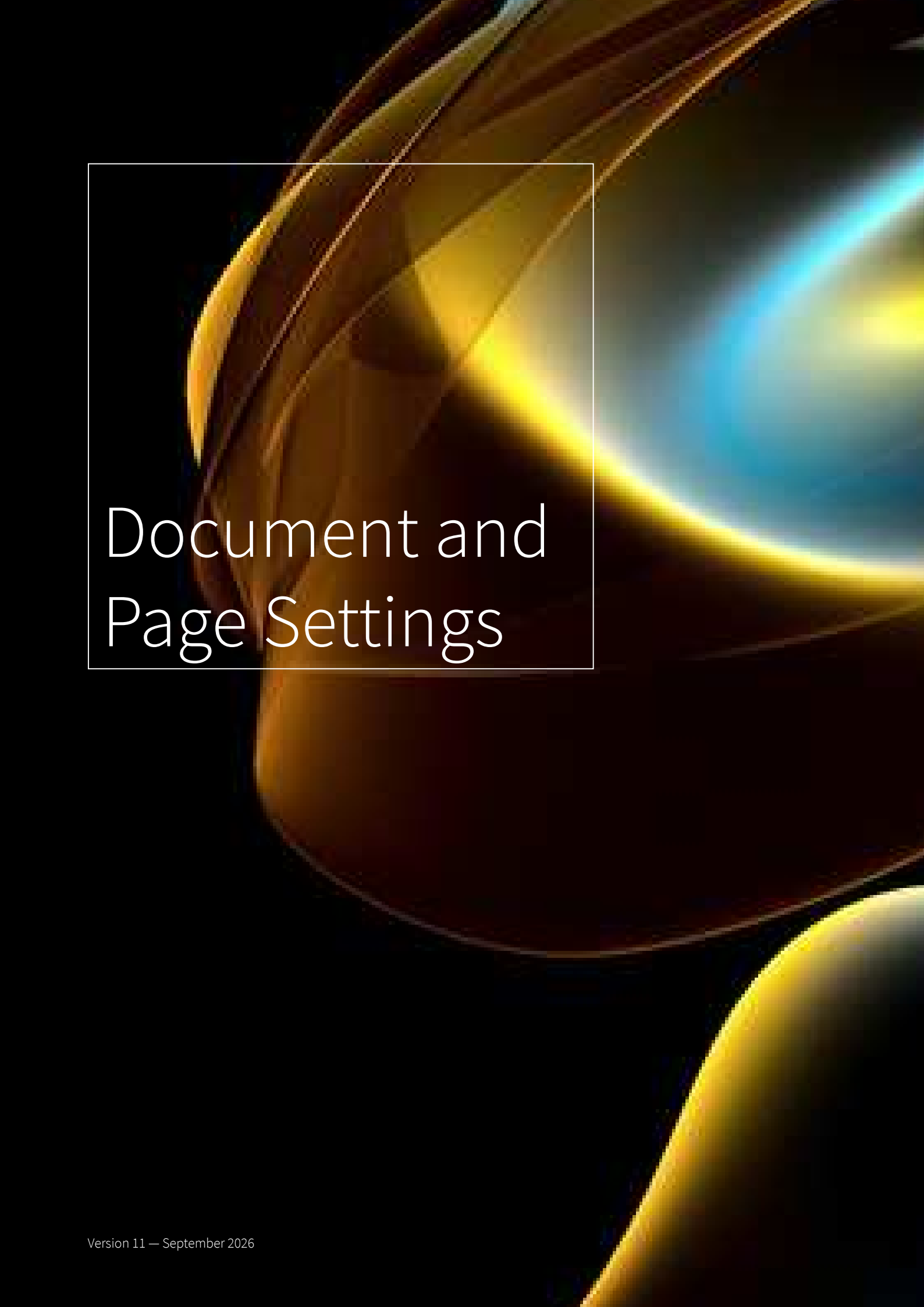
### Methods:

|                                                              |             |
|--------------------------------------------------------------|-------------|
| <b>setLeftSeparator</b> ( <i>bool on, LineStyle data</i> )   | <i>void</i> |
| <b>setTopSeparator</b> ( <i>bool on, LineStyle data</i> )    | <i>void</i> |
| <b>setRightSeparator</b> ( <i>bool on, LineStyle data</i> )  | <i>void</i> |
| <b>setBottomSeparator</b> ( <i>bool on, LineStyle data</i> ) | <i>void</i> |
| <b>remove</b> ( <i>String propertyName</i> )                 | <i>void</i> |

The method removes the property setting from the stylesheet. The value *propertyName* must be set to used property name from following list: 'fillBrush', 'fillShade', 'fillBlendAngle', 'rotation', 'horizontalAlignment', 'verticalAlignment', 'leftIndent', 'topIndent', 'rightIndent', 'bottomIndent', 'leftSeparator', 'topSeparator', 'rightSeparator', 'bottomSeparator'







# Document and Page Settings

# Document and Page Settings

## Class: DocumentSettings

An object holds the document parameters that are used when a new document is created with `application.newDocument(settings, show)`. The settings of an existing document are available through the property `documentSettings` of the class `Document`.

### Constructor:

An object can be created using the constructor:

```
new DocumentSettings()
```

and the document parameters are set using the following properties:

### Properties:

|                                                                                                                                        |                        |                   |
|----------------------------------------------------------------------------------------------------------------------------------------|------------------------|-------------------|
| <b>pageSize</b>                                                                                                                        | <i>PageSize</i>        | <i>Read/Write</i> |
| Get/Set a page format of the document created. See the enumeration type <code>PageSize</code> .                                        |                        |                   |
| <b>pageOrientation</b>                                                                                                                 | <i>PageOrientation</i> | <i>Read/Write</i> |
| Get/Set a page orientation of the document created. See the enumeration type <code>PageOrientation</code> .                            |                        |                   |
| <b>pageType</b>                                                                                                                        | <i>PageType</i>        | <i>Read/Write</i> |
| Get/Set a page type of the document created. See the enumeration type <code>PageType</code> .                                          |                        |                   |
| <b>pageOrder</b>                                                                                                                       | <i>PageOrder</i>       | <i>Read/Write</i> |
| Get/Set a page order of the document created. See the enumeration type <code>PageOrder</code> .                                        |                        |                   |
| <b>customWidth</b>                                                                                                                     | <i>UnitValueString</i> | <i>Read/Write</i> |
| Get/Set a width of the document page when for <code>PageSize</code> = 'Custom' is set.                                                 |                        |                   |
| <b>customHeight</b>                                                                                                                    | <i>UnitValueString</i> | <i>Read/Write</i> |
| Get/Set a height of the document page when for <code>PageSize</code> = 'Custom' is set.                                                |                        |                   |
| <b>automaticTextObject</b>                                                                                                             | <i>bool</i>            | <i>Read/Write</i> |
| Get/Set a flag value. If the value is set to true, the document will be created with a text object according to the document settings. |                        |                   |
| <b>layoutGrid</b>                                                                                                                      | <i>bool</i>            | <i>Read/Write</i> |
| Get/Set a flag value. If the value is set to true, the document will be created with a margin grid.                                    |                        |                   |
| <b>topMargin</b>                                                                                                                       | <i>UnitValueString</i> | <i>Read/Write</i> |
| Get/Set a top margin for the new document page.                                                                                        |                        |                   |
| <b>bottomMargin</b>                                                                                                                    | <i>UnitValueString</i> | <i>Read/Write</i> |
| Get/Set a bottom margin for the new document page.                                                                                     |                        |                   |
| <b>innerMargin</b>                                                                                                                     | <i>UnitValueString</i> | <i>Read/Write</i> |

Get/Set an inner margin for the new document page. The property is used for PageType = FacingPages only.

---

|                    |                        |                   |
|--------------------|------------------------|-------------------|
| <b>outerMargin</b> | <i>UnitValueString</i> | <i>Read/Write</i> |
|--------------------|------------------------|-------------------|

Get/Set an outer margin for the new document page. The property is used for PageType = FacingPages only.

---

|                   |                        |                   |
|-------------------|------------------------|-------------------|
| <b>leftMargin</b> | <i>UnitValueString</i> | <i>Read/Write</i> |
|-------------------|------------------------|-------------------|

Get/Set a left margin for the new document page. The property is used for PageType = SinglePage only.

---

|                    |                        |                   |
|--------------------|------------------------|-------------------|
| <b>rightMargin</b> | <i>UnitValueString</i> | <i>Read/Write</i> |
|--------------------|------------------------|-------------------|

Get/Set a right margin for the new document page. The property is used for PageType = SinglePage only.

---

|                        |            |                   |
|------------------------|------------|-------------------|
| <b>numberOfColumns</b> | <i>int</i> | <i>Read/Write</i> |
|------------------------|------------|-------------------|

Get/Set a number of columns to be used on a new document page.

---

|                     |                        |                   |
|---------------------|------------------------|-------------------|
| <b>columnGutter</b> | <i>UnitValueString</i> | <i>Read/Write</i> |
|---------------------|------------------------|-------------------|

Get/Set a gutter between the columns.

#### Example 1:

```
var ds = new DocumentSettings();
ds.pageSize = PageSize.Custom;
ds.pageOrientation = PageOrientation.Landscape;
ds.pageType = PageType.FacingPages;
ds.pageOrder = PageOrder.RightToLeft;
ds.customWidth = "150mm";
ds.customHeight = "20cm";
ds.automaticTextObject = true;
ds.layoutGrid = true;
ds.topMargin = "5mm";
ds.bottomMargin = "8mm";
messageBox.information("Msg:",
    convertUnitValue(ds.topMargin, "m", true));
```

## Class: PageSettings

An object holds the page parameters that are used when new pages are created with the method addDocumentPage(count, settings, masterPage) of the class Document.

#### Constructor:

An object can be created using the constructor:

**new PageSettings()**

and the page parameters are set using the following properties:

#### Properties:

---

|                       |                           |                   |
|-----------------------|---------------------------|-------------------|
| <b>insertPosition</b> | <i>PageInsertPosition</i> | <i>Read/Write</i> |
|-----------------------|---------------------------|-------------------|

---

Position for newly added pages. See the enumeration type `PageInsertPosition`. Default: `AfterCurrentPage`.

|                                                                                                                                                    |             |                   |
|----------------------------------------------------------------------------------------------------------------------------------------------------|-------------|-------------------|
| <b>afterPageNumber</b>                                                                                                                             | <i>int</i>  | <i>Read/Write</i> |
| Index of page after which new page is inserted (only for <code>PageInsertPosition.AfterPage</code> ). The first page has the number 1. Default: 1. |             |                   |
| <b>copyObjects</b>                                                                                                                                 | <i>bool</i> | <i>Read/Write</i> |
| Objects are taken over from the current page                                                                                                       |             |                   |
| <b>copyOnlySelectedObjects</b>                                                                                                                     | <i>bool</i> | <i>Read/Write</i> |
| Selected objects are taken over from the current page only                                                                                         |             |                   |
| <b>copyAsAlias</b>                                                                                                                                 | <i>bool</i> | <i>Read/Write</i> |
| Objects are taken over as an Alias instead of as a Copy                                                                                            |             |                   |
| <b>linkToTextChain</b>                                                                                                                             | <i>bool</i> | <i>Read/Write</i> |
| Text objects of new pages are linked to the text chain of the old page.                                                                            |             |                   |

## Class: ChapterInfo

The class is used in the class *DocumentPage*.

### Constructor:

```
new ChapterInfo()
new ChapterInfo( PageNumberingFormat pnk, short pageNumber, String prefix)
```

### Properties:

|                        |                            |                   |
|------------------------|----------------------------|-------------------|
| <b>numberingFormat</b> | <i>PageNumberingFormat</i> | <i>Read/Write</i> |
| <b>newPageNumber</b>   | <i>int</i>                 | <i>Read/Write</i> |
| <b>prefix</b>          | <i>String</i>              | <i>Read/Write</i> |

## Class: DocumentStatistics

An object holds the statistics information of the document (Document Statistics). It is used in the class *Document*.

### Constructor:

An object can be created using the constructor: `new DocumentStatistics()` and the document parameters are set using the following properties or using the constructor with parameters (all fourteen parameters must be given):

```
new DocumentStatistics (
    PrintDeviceType printDeviceType,
    PrintMaterialType printMaterialType,
    PrintImageType printImageType,
    ExposureType exposureType,
    PrintResolutionType resolutionType,
    ColorSeparationState colorSeparationState,
```

```
String scale,
String author,
String orderNumber,
String orderedBy,
String responsible,
String phone,
String fax,
String deadline)
```

**Properties:**

|                        |                             |                   |
|------------------------|-----------------------------|-------------------|
| <b>printDeviceType</b> | <i>PrintDeviceType</i>      | <i>Read/Write</i> |
| <b>printMaterial</b>   | <i>PrintMaterialType</i>    | <i>Read/Write</i> |
| <b>printImage</b>      | <i>PrintImageType</i>       | <i>Read/Write</i> |
| <b>exposure</b>        | <i>ExposureType</i>         | <i>Read/Write</i> |
| <b>printResolution</b> | <i>PrintResolutionType</i>  | <i>Read/Write</i> |
| <b>colorSeparation</b> | <i>ColorSeparationState</i> | <i>Read/Write</i> |
| <b>scale</b>           | <i>String</i>               | <i>Read/Write</i> |
| <b>author</b>          | <i>String</i>               | <i>Read/Write</i> |
| <b>orderNumber</b>     | <i>String</i>               | <i>Read/Write</i> |
| <b>orderedBy</b>       | <i>String</i>               | <i>Read/Write</i> |
| <b>responsible</b>     | <i>String</i>               | <i>Read/Write</i> |
| <b>phone</b>           | <i>String</i>               | <i>Read/Write</i> |
| <b>fax</b>             | <i>String</i>               | <i>Read/Write</i> |
| <b>deadline</b>        | <i>String</i>               | <i>Read/Write</i> |
| <b>workingTime</b>     | <i>int</i>                  | <i>Read</i>       |

## Class: SearchPaths

The object holds a list of search paths for the document. It is used in the class Document.

**Constructor:**

An object can be created using the constructor: *new SearchPaths()* and the document parameters are set using following properties:

|               |            |             |
|---------------|------------|-------------|
| <b>length</b> | <i>int</i> | <i>Read</i> |
|---------------|------------|-------------|

**Methods:**

|                                                           |             |
|-----------------------------------------------------------|-------------|
| <b>addPath</b> ( <i>String path, bool withSubdir</i> )    | <i>void</i> |
| The char ‘\’ must be doubled.                             |             |
| <b>removePath</b> ( <i>String path, bool withSubdir</i> ) | <i>void</i> |
| The char ‘\’ must be doubled.                             |             |

|                                                       |               |
|-------------------------------------------------------|---------------|
| <b>removePath</b> ( <i>String path</i> )              | <i>void</i>   |
| The char '\' must be doubled.                         |               |
| <b>removeAll</b> ()                                   | <i>void</i>   |
| <b>exists</b> ( <i>String path, bool withSubdir</i> ) | <i>bool</i>   |
| <b>exists</b> ( <i>String path</i> )                  | <i>bool</i>   |
| The char '\' must be doubled.                         |               |
| <b>pathAt</b> ( <i>int index</i> )                    | <i>String</i> |
| <b>subdirFlagAt</b> ( <i>int index</i> )              | <i>bool</i>   |

**Examples:**

```

var doc = application.currentDocument;
if (doc)
{
    var sp1 = new SearchPaths();
    sp1.addPath("c:\\Development", true);
    sp1.addPath("c:\\aaaa", true);
    sp1.addPath("c:\\aaaa", false);
    doc.searchPaths = sp1;
    application.currentDocument.searchPaths.addPath("c:\\xxxx", true);
    application.currentDocument.searchPaths.addPath("c:\\yyyy", false);
    application.currentDocument.searchPaths.addPath("c:\\zzzz", true);

    var sp = doc.searchPaths;
    MessageBox.information("Len: ",
application.currentDocument.searchPaths.length);
    MessageBox.information("PathAt: ", sp.pathAt(3));
    MessageBox.information("FlagAt: ", sp.subdirFlagAt(0));
    var txt = "";
    for (var i = 0; i < application.currentDocument.searchPaths.length; i++)
    {
        txt = txt + "\nIndex=" + i + "; Path=" +
application.currentDocument.searchPaths.pathAt(i) + "; SubdirFlag=" +
application.currentDocument.searchPaths.subdirFlagAt(i);
    }
    MessageBox.information("List of paths", txt);

    if (sp.exists("c:\\xxxx"))
        sp.removePath("c:\\xxxx");

    var txt = "";
    for (var i = 0; i < application.currentDocument.searchPaths.length; i++)
    {
        txt = txt + "\nIndex=" + i + "; Path=" +
application.currentDocument.searchPaths.pathAt(i) + "; SubdirFlag=" +
application.currentDocument.searchPaths.subdirFlagAt(i);
    }
    MessageBox.information("List of paths", txt);
}

```

```
doc.searchPaths.removeAll();  
messageBox.information("Search Paths Count: ", doc.searchPaths.length);  
}
```





# Preferences

# Preferences

## Class: EmbeddedFontsPreferences

The object is obtained from the property *embeddedFontsPreferences* of the class Document.

### Properties:

|                           |             |                   |
|---------------------------|-------------|-------------------|
| <b>disableSystemFonts</b> | <i>bool</i> | <i>Read/Write</i> |
| <b>embedAllUsedFonts</b>  | <i>bool</i> | <i>Read/Write</i> |

## Class: LanguageInfo

The class is used in the *LanguagePreferences* class.

### Properties:

|                               |                         |                   |
|-------------------------------|-------------------------|-------------------|
| <b>language</b>               | <i>Language</i>         | <i>Read</i>       |
| <b>singleQuotesType</b>       | <i>SingleQuotesType</i> | <i>Read/Write</i> |
| <b>doubleQuotesType</b>       | <i>DoubleQuotesType</i> | <i>Read/Write</i> |
| <b>hyphenationModuleIdent</b> | <i>int</i>              | <i>Read</i>       |
| <b>spellModuleIdent</b>       | <i>int</i>              | <i>Read</i>       |
| <b>thesaurusModuleIdent</b>   | <i>int</i>              | <i>Read</i>       |

## Class: LanguagePreferences

The class is used in the Document class.

### Methods:

|                                                     |                     |
|-----------------------------------------------------|---------------------|
| <b>languageInfo</b> ( <i>Language lang</i> )        | <i>LanguageInfo</i> |
| <b>setLanguageInfo</b> ( <i>LanguageInfo info</i> ) | <i>void</i>         |

### Examples:

```
var doc = application.currentDocument;
if (doc) {
    var pref = doc.languagePreferences;
    var langInfo = pref.languageInfo(Language.German);
    messageBox.information("MSG",
        "SpelModuleIdent: " + langInfo.spellModuleIdent +
        "\nDouble Quotes Type: " + langInfo.doubleQuotesType);
    langInfo.doubleQuotesType = DoubleQuotesType.DoubleQuotes4;
    pref.setLanguageInfo(langInfo);
    doc.languagePreferences = pref;}
```

## Class: LayoutPreferences

The object is obtained from the property *layoutPreferences* of the class Document.

Properties:

|                               |              |                   |
|-------------------------------|--------------|-------------------|
| <b>showGuides</b>             | <i>bool</i>  | <i>Read/Write</i> |
| <b>showSmartGuides</b>        | <i>bool</i>  | <i>Read/Write</i> |
| <b>showMarginsAndColumns</b>  | <i>bool</i>  | <i>Read/Write</i> |
| <b>showRuler</b>              | <i>bool</i>  | <i>Read/Write</i> |
| <b>showTextRuler</b>          | <i>bool</i>  | <i>Read/Write</i> |
| <b>showGuideObjects</b>       | <i>bool</i>  | <i>Read/Write</i> |
| <b>showAliasObjects</b>       | <i>bool</i>  | <i>Read/Write</i> |
| <b>aliasObjectsSelectable</b> | <i>bool</i>  | <i>Read/Write</i> |
| <b>showBaselineGrid</b>       | <i>bool</i>  | <i>Read/Write</i> |
| <b>guideObjectsMagnetic</b>   | <i>bool</i>  | <i>Read/Write</i> |
| <b>guideObjectsSelectable</b> | <i>bool</i>  | <i>Read/Write</i> |
| <b>visualizeStyleSheets</b>   | <i>bool</i>  | <i>Read/Write</i> |
| <b>quickTextView</b>          | <i>bool</i>  | <i>Read/Write</i> |
| <b>quickPictureView</b>       | <i>bool</i>  | <i>Read/Write</i> |
| <b>viewZoomFactor</b>         | <i>float</i> | <i>Read/Write</i> |

## Class: ObjectPreferences

The object is obtained from the property *objectPreferences* of the class Document.

Properties:

|                              |                          |                   |
|------------------------------|--------------------------|-------------------|
| <b>snapDistance</b>          | <i>float</i>             | <i>Read/Write</i> |
| <b>graphicStyleSheet</b>     | <i>GraphicStyleSheet</i> | <i>Read/Write</i> |
| <b>horizontalPasteOffset</b> | <i>String</i>            | <i>Read/Write</i> |
| <b>verticalPasteOffset</b>   | <i>String</i>            | <i>Read/Write</i> |
| <b>askForImages</b>          | <i>bool</i>              | <i>Read/Write</i> |
| <b>colorDepth</b>            | <i>ColorDepth</i>        | <i>Read/Write</i> |
| <b>embedAllImages</b>        | <i>bool</i>              | <i>Read/Write</i> |
| <b>pictureStyleSheet</b>     | <i>PictureStyleSheet</i> | <i>Read/Write</i> |

## Class: PagePreferences

The object is obtained from the property *pagePreferences* of the class Document.

### Properties:

|                                                                           |             |                   |
|---------------------------------------------------------------------------|-------------|-------------------|
| <b>saveSmallPreview</b>                                                   | <i>bool</i> | <i>Read/Write</i> |
| <b>saveLargePreview</b>                                                   | <i>bool</i> | <i>Read/Write</i> |
| <b>smallPreviewsMaximumSize</b><br>from 32 pixels to 640 pixel by step 16 | <i>int</i>  | <i>Read/Write</i> |
| <b>maximumPreviews</b><br>0 = all pages                                   | <i>int</i>  | <i>Read/Write</i> |
| <b>previewQuality</b><br>10 – 100 [%]                                     | <i>int</i>  | <i>Read/Write</i> |

## Class: PreflightPreferences

The object holds the preflight preferences. It is used in the class *Document*.

### Constructor:

An object can be created using the constructor:

```
new PreflightPreferences()
```

and the document parameters are set using the following properties or using the constructor with parameters (all ten parameters must be given):

```
new PreflightPreferences (  
    bool preflightDocument,  
    float colorImagesMinimumResolution,  
    float grayImagesMinimumResolution,  
    float bwImagesMinimumResolution,  
    bool warningForRgbColorMode,  
    bool warningForGrayColorMode,  
    bool warningForBwColorMode,  
    bool warningForTransparency,  
    bool warningForEffects,  
    float tolerance)
```

### Properties:

|                                     |              |                         |
|-------------------------------------|--------------|-------------------------|
| <b>preflightDocument</b>            | <i>bool</i>  | <i>Read/Write</i>       |
| <b>colorImagesMinimumResolution</b> | <i>float</i> | <i>[ppi] Read/Write</i> |
| <b>grayImagesMinimumResolution</b>  | <i>float</i> | <i>[ppi] Read/Write</i> |
| <b>bwImagesMinimumResolution</b>    | <i>float</i> | <i>[ppi] Read/Write</i> |

All three properties get/set the resolution value in ppi.

To disable the option set 0 option.

|                                |              |                                    |
|--------------------------------|--------------|------------------------------------|
| <b>warningForRgbColorMode</b>  | <i>bool</i>  | <i>Read/Write</i>                  |
| <b>warningForGrayColorMode</b> | <i>bool</i>  | <i>Read/Write</i>                  |
| <b>warningForBwColorMode</b>   | <i>bool</i>  | <i>Read/Write</i>                  |
| <b>warningForTransparency</b>  | <i>bool</i>  | <i>Read/Write</i>                  |
| <b>warningForEffects</b>       | <i>bool</i>  | <i>Read/Write</i>                  |
| <b>tolerance</b>               | <i>float</i> | <i>&lt;0-100&gt;[%] Read/Write</i> |

## Class: TextPreferences

The object holds the text preferences. The class is used in class *Document*.

### Constructor:

An object can be created using the constructor: *new TextPreferences()* and the document parameters are set using the following properties:

### Properties:

|                                                              |              |                   |
|--------------------------------------------------------------|--------------|-------------------|
| <b>underlineOffsetVerticalLayout</b>                         | <i>float</i> | <i>Read/Write</i> |
| unit: [%], valid values interval: [-100, 100], default: 60%  |              |                   |
| <b>underlineOffsetHorizontalLayout</b>                       | <i>float</i> | <i>Read/Write</i> |
| unit: [%], valid values interval: [-100, 100], default: 10%  |              |                   |
| <b>underlineLineWidth</b>                                    | <i>float</i> | <i>Read/Write</i> |
| unit: [%], valid values interval: [1, 100], default: 5%      |              |                   |
| <b>strikethroughOffsetVerticalLayout</b>                     | <i>float</i> | <i>Read/Write</i> |
| unit: [%], valid values interval: [-100, 100], default: 0%   |              |                   |
| <b>strikethroughOffsetHorizontalLayout</b>                   | <i>float</i> | <i>Read/Write</i> |
| unit: [%], valid values interval: [-100, 100], default: -30% |              |                   |
| <b>strikethroughLineWidth</b>                                | <i>float</i> | <i>Read/Write</i> |
| unit: [%], valid values interval: [5, 100], default: 5%      |              |                   |
| <b>superscriptOffset</b>                                     | <i>float</i> | <i>Read/Write</i> |
| unit: [%], valid values interval: [0, 100], default: 33%     |              |                   |
| <b>superscriptCharacterHeight</b>                            | <i>float</i> | <i>Read/Write</i> |
| unit: [%], valid values interval: [1, 100], default: 50%     |              |                   |
| <b>superscriptCharacterWidth</b>                             | <i>float</i> | <i>Read/Write</i> |
| unit: [%], valid values interval: [1, 100], default: 50%     |              |                   |
| <b>subscriptOffset</b>                                       | <i>float</i> | <i>Read/Write</i> |
| unit: [%], valid values interval: [0, 100], default: 33%     |              |                   |
| <b>subscriptCharacterHeight</b>                              | <i>float</i> | <i>Read/Write</i> |

unit: [%], valid values interval: [1, 100], default: 50%

|                                |              |                   |
|--------------------------------|--------------|-------------------|
| <b>subscriptCharacterWidth</b> | <i>float</i> | <i>Read/Write</i> |
|--------------------------------|--------------|-------------------|

unit: [%], valid values interval: [1, 100], default: 50%

|                                |              |                   |
|--------------------------------|--------------|-------------------|
| <b>superiorCharacterHeight</b> | <i>float</i> | <i>Read/Write</i> |
|--------------------------------|--------------|-------------------|

unit: [%], valid values interval: [1, 100], default: 50%

|                               |              |                   |
|-------------------------------|--------------|-------------------|
| <b>superiorCharacterWidth</b> | <i>float</i> | <i>Read/Write</i> |
|-------------------------------|--------------|-------------------|

unit: [%], valid values interval: [1, 100], default: 50%

|                                 |              |                   |
|---------------------------------|--------------|-------------------|
| <b>smallCapsCharacterHeight</b> | <i>float</i> | <i>Read/Write</i> |
|---------------------------------|--------------|-------------------|

unit: [%], valid values interval: [1, 100], default: 75%

|                                |              |                   |
|--------------------------------|--------------|-------------------|
| <b>smallCapsCharacterWidth</b> | <i>float</i> | <i>Read/Write</i> |
|--------------------------------|--------------|-------------------|

unit: [%], valid values interval: [1, 100], default: 75%

|                        |                 |                   |
|------------------------|-----------------|-------------------|
| <b>lineNumbersFont</b> | <i>FontName</i> | <i>Read/Write</i> |
|------------------------|-----------------|-------------------|

get/set name of the font

|                            |                        |                   |
|----------------------------|------------------------|-------------------|
| <b>lineNumbersFontSize</b> | <i>UnitValueString</i> | <i>Read/Write</i> |
|----------------------------|------------------------|-------------------|

get/set font size

|                         |                   |                   |
|-------------------------|-------------------|-------------------|
| <b>lineNumbersColor</b> | <i>ColorBrush</i> | <i>Read/Write</i> |
|-------------------------|-------------------|-------------------|

get/set a color of LineNumbers

|                        |                        |                   |
|------------------------|------------------------|-------------------|
| <b>lineNumbersMode</b> | <i>LineNumbersMode</i> | <i>Read/Write</i> |
|------------------------|------------------------|-------------------|

get/set mode of LineNumbers

|                        |                    |                   |
|------------------------|--------------------|-------------------|
| <b>lineNumbersData</b> | <i>LineNumbers</i> | <i>Read/Write</i> |
|------------------------|--------------------|-------------------|

get/set LineNumbers data which depends on LineNumbersMode

|                                  |              |                   |
|----------------------------------|--------------|-------------------|
| <b>textLinesAutomaticSpacing</b> | <i>float</i> | <i>Read/Write</i> |
|----------------------------------|--------------|-------------------|

unit: [%], valid values interval: [0, 100], default: 0%

|                        |                              |                   |
|------------------------|------------------------------|-------------------|
| <b>textLinesMethod</b> | <i>AutoLineSpacingMethod</i> | <i>Read/Write</i> |
|------------------------|------------------------------|-------------------|

|                 |                 |                   |
|-----------------|-----------------|-------------------|
| <b>language</b> | <i>Language</i> | <i>Read/Write</i> |
|-----------------|-----------------|-------------------|

|                   |                        |                   |
|-------------------|------------------------|-------------------|
| <b>greekBelow</b> | <i>UnitValueString</i> | <i>Read/Write</i> |
|-------------------|------------------------|-------------------|

|                              |             |                   |
|------------------------------|-------------|-------------------|
| <b>automaticGrammarCheck</b> | <i>bool</i> | <i>Read/Write</i> |
|------------------------------|-------------|-------------------|

|                            |             |                   |
|----------------------------|-------------|-------------------|
| <b>automaticSpellCheck</b> | <i>bool</i> | <i>Read/Write</i> |
|----------------------------|-------------|-------------------|

|                       |             |                   |
|-----------------------|-------------|-------------------|
| <b>showInvisibles</b> | <i>bool</i> | <i>Read/Write</i> |
|-----------------------|-------------|-------------------|

|                      |             |                   |
|----------------------|-------------|-------------------|
| <b>showTextRuler</b> | <i>bool</i> | <i>Read/Write</i> |
|----------------------|-------------|-------------------|

|                                    |                        |                   |
|------------------------------------|------------------------|-------------------|
| <b>distanceBetweenStandardTabs</b> | <i>UnitValueString</i> | <i>Read/Write</i> |
|------------------------------------|------------------------|-------------------|

|                                  |             |                   |
|----------------------------------|-------------|-------------------|
| <b>typographicQuotationMarks</b> | <i>bool</i> | <i>Read/Write</i> |
|----------------------------------|-------------|-------------------|

|                       |             |                   |
|-----------------------|-------------|-------------------|
| <b>forceLineBreak</b> | <i>bool</i> | <i>Read/Write</i> |
|-----------------------|-------------|-------------------|

|                                                                                                                            |                            |                   |
|----------------------------------------------------------------------------------------------------------------------------|----------------------------|-------------------|
| <b>visualizeStyleSheets</b>                                                                                                | <i>bool</i>                | <i>Read/Write</i> |
| <b>firstBaseline</b>                                                                                                       | <i>BaselinePosition</i>    | <i>Read/Write</i> |
| <b>baselineGrid</b>                                                                                                        | <i>BaselineGrid</i>        | <i>Read/Write</i> |
| <b>characterStyleSheet</b>                                                                                                 | <i>CharacterStyleSheet</i> | <i>Read/Write</i> |
| Set/get the default character style sheet for the text. The object CharacterStyleSheet is used to identify the stylesheet. |                            |                   |
| <b>characterStyleSheetByName</b>                                                                                           | <i>String</i>              | <i>Read/Write</i> |
| Set/get the default character style sheet. Name of style sheet is used for identify of the style sheet.                    |                            |                   |
| <b>paragraphStyleSheet</b>                                                                                                 | <i>ParagraphStyleSheet</i> | <i>Read/Write</i> |
| <b>paragraphStyleSheetByName</b>                                                                                           | <i>String</i>              | <i>Read/Write</i> |
| <b>layoutStyleSheet</b>                                                                                                    | <i>LayoutStyleSheet</i>    | <i>Read/Write</i> |
| <b>layoutStyleSheetByName</b>                                                                                              | <i>String</i>              | <i>Read/Write</i> |
| <b>footnoteReferenceStyleSheet</b>                                                                                         | <i>CharacterStyleSheet</i> | <i>Read/Write</i> |
| <b>footnoteLabelStyleSheet</b>                                                                                             | <i>CharacterStyleSheet</i> | <i>Read/Write</i> |
| <b>footnoteTextStyleSheet</b>                                                                                              | <i>ParagraphStyleSheet</i> | <i>Read/Write</i> |
| <b>footnoteReferenceStyleSheetByName</b>                                                                                   | <i>String</i>              | <i>Read/Write</i> |
| <b>footnoteLabelStyleSheetByName</b>                                                                                       | <i>String</i>              | <i>Read/Write</i> |
| <b>footnoteTextStyleSheetByName</b>                                                                                        | <i>String</i>              | <i>Read/Write</i> |
| <b>endnoteReferenceStyleSheet</b>                                                                                          | <i>CharacterStyleSheet</i> | <i>Read/Write</i> |
| <b>endnoteLabelStyleSheet</b>                                                                                              | <i>CharacterStyleSheet</i> | <i>Read/Write</i> |
| <b>endnoteTextStyleSheet</b>                                                                                               | <i>ParagraphStyleSheet</i> | <i>Read/Write</i> |
| <b>endnoteReferenceStyleSheetByName</b>                                                                                    | <i>String</i>              | <i>Read/Write</i> |
| <b>endnoteLabelStyleSheetByName</b>                                                                                        | <i>String</i>              | <i>Read/Write</i> |
| <b>endnoteTextStyleSheetByName</b>                                                                                         | <i>String</i>              | <i>Read/Write</i> |
| <b>changeTracking</b>                                                                                                      | <i>bool</i>                | <i>Read/Write</i> |
| <b>endnotes</b>                                                                                                            | <i>TextEndnotes</i>        | <i>Read/Write</i> |
| <b>footnotes</b>                                                                                                           | <i>TextFootnotes</i>       | <i>Read/Write</i> |

#### Example 1:

```

application.currentDocument.textPreferences.underlineOffsetVerticalLayout =
77;
application.currentDocument.textPreferences.baselineGrid.start = "13pt";
messageBox.information("UnderlineOffsetVerticalLayout",
    application.currentDocument.textPreferences.underlineOffsetVerticalLayout
+ "\n" +
    application.currentDocument.textPreferences.baselineGrid.start);

```

**Example 2:**

```
var doc = application.currentDocument;
if (doc) {
    var tp = new TextPreferences();
    tp.underlineOffsetVerticalLayout = 33;
    tp.lineNumbersFont = "Times New Roman-Bold";
    tp.lineNumbersFontSize = "12pt";
    var cb = doc.brushes.addColor("MyColor2");
    cb.rgb = new Rgb(200, 100, 0);
    tp.lineNumbersColor = cb;
    var blg = new BaselineGrid("22pt", "12pt",
BaselineGridNumbering.ShowAllNumbers,
BaselineGridDirection.RightToLeft);
    tp.baselineGrid = blg;

    doc.textPreferences = tp;

    var tp1 = doc.textPreferences;
    var txt = "UnderlineOffsetVertLayout = " +
tp1.underlineOffsetVerticalLayout + "\nUnderlineOffsetHorizontalLayout = " +
tp1.underlineOffsetHorizontalLayout;

    messageBox.information("TextPreferences", txt)
}
```







# Helped Data Objects

# Helped Data Objects

## Class: Annotation

The object of the class *Annotation* represents a property of the class *FormattedText*.

### Properties:

|                                                                                                                                                             |                      |             |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------|-------------|
| <b>position</b>                                                                                                                                             | <i>RubyPosition</i>  | <i>Read</i> |
| Returns the value of the enumeration type <i>RubyPosition</i> .                                                                                             |                      |             |
| <b>range</b>                                                                                                                                                | <i>TextRange</i>     | <i>Read</i> |
| Returns a <i>TextRange</i> object with the position where the base text (where the annotation text is applied) and the annotation text are placed together. |                      |             |
| <b>baseTextRange</b>                                                                                                                                        | <i>TextRange</i>     | <i>Read</i> |
| Returns a <i>TextRange</i> object with the position where only the base text (where the annotation text is applied) is placed.                              |                      |             |
| <b>annotationTextRange</b>                                                                                                                                  | <i>TextRange</i>     | <i>Read</i> |
| Returns a <i>TextRange</i> object with the position where only the annotation text is placed.                                                               |                      |             |
| <b>mainText</b>                                                                                                                                             | <i>FormattedText</i> | <i>Read</i> |
| Returns all text after the position where the annotation text was applied.                                                                                  |                      |             |

### Examples:

```
var doc = application.currentDocument;
if (doc) {
    var objP = doc.objects().create(ObjectType.Text);
    objP.pagex = "10mm";
    objP.pagey = "22mm";
    objP.width = "120mm";
    objP.height = "40mm";

    with (objP.content) {
        insertPlainText("Sample text for testing.");
        formattedText.insertAnnotation("AnnotationText",
            new TextRange(5, 10),
            RubyPosition.Below);
    }
    doc.currentPage.add(objP);
    var pc = doc.currentPage.objects().at(0).content;
    var txt = "";
    var t = pc.formattedText.annotationAt(5);

    txt = txt + "Position: " + t.position + "\n";
    txt = txt + "Range: [" + t.range.begin + "," + t.range.end + "; Text: " +
        t.mainText.formattedText(t.range.begin, t.range.end).plainText()
        + "]\n";
    txt = txt + "BaseTextRange: [" + t.baseTextRange.begin + "," +
```

```
        t.baseTextRange.end + "; Text: " +
t.mainText.formattedText(t.baseTextRange.begin,
t.baseTextRange.end).plainText() + "]\n";
        txt = txt + "AnnotationTextRange: [" + t.annotationTextRange.begin + ","
+          t.annotationTextRange.end + "; Text: " +
t.mainText.formattedText(t.annotationTextRange.begin,
t.annotationTextRange.end).plainText() + "]\n";
        txt = txt + "Text: " + t.mainText.plainText() + "\n";

        messageBox.information("TextInfo", txt);
    }
```

## Class: Attributes

The class is used in the classes *Object*, *PictureContent*, *TextContent*, *Table*, *TableCell*, *TableColumn*, *TableRow* and *TableSeparator*.

Constructor:

```
new Attributes()
```

## Class: AutomaticCharacterSpacing

The class is used in the class *FormattedText*, *ParagraphStyleSheet*.

Constructor:

```
new AutomaticCharacterSpacing()
```

Properties:

|                |              |                       |
|----------------|--------------|-----------------------|
| <b>active</b>  | <i>bool</i>  | <i>Read/Write</i>     |
| <b>minimum</b> | <i>float</i> | <i>Read/Write [%]</i> |
| <b>optimum</b> | <i>float</i> | <i>Read/Write [%]</i> |
| <b>maximum</b> | <i>float</i> | <i>Read/Write [%]</i> |

## Class: AutomaticCharacterWidth

The class is used in the class *FormattedText*, *ParagraphStyleSheet*.

Constructor:

```
new AutomaticCharacterSpacing()
```

Properties:

|                |              |                       |
|----------------|--------------|-----------------------|
| <b>minimum</b> | <i>float</i> | <i>Read/Write [%]</i> |
| <b>optimum</b> | <i>float</i> | <i>Read/Write [%]</i> |
| <b>maximum</b> | <i>float</i> | <i>Read/Write [%]</i> |

## Class: AutomaticWordSpacing

The class is used in the class *FormattedText*, *ParagraphStyleSheet*.

Constructor:

```
new AutomaticCharacterSpacing()
```

Properties:

|                |              |                       |
|----------------|--------------|-----------------------|
| <b>minimum</b> | <i>float</i> | <i>Read/Write [%]</i> |
| <b>optimum</b> | <i>float</i> | <i>Read/Write [%]</i> |
| <b>maximum</b> | <i>float</i> | <i>Read/Write [%]</i> |

## Class: BaselineGrid

The class is used in the class *TextPreferences*.

Constructor:

```
new BaselineGrid()
new BaselineGrid(String start, String spacing, BaselineGridDirection direction,
BaselineGridNumbering numbering)
```

Properties:

|                  |                              |                   |
|------------------|------------------------------|-------------------|
| <b>active</b>    | <i>bool</i>                  | <i>Read/Write</i> |
| <b>start</b>     | <i>UnitValueString</i>       | <i>Read/Write</i> |
| <b>spacing</b>   | <i>UnitValueString</i>       | <i>Read/Write</i> |
| <b>direction</b> | <i>BaselineGridDirection</i> | <i>Read/Write</i> |
| <b>numbering</b> | <i>BaselineGridNumbering</i> | <i>Read/Write</i> |

## Class: BaselinePosition

The class is used in the classes *TextPreferences* and *TextContent*.

Constructor:

```
new BaselinePosition()
new BaselinePosition(BaselineType firstBaseline, BaselineOffset applyValue,
UnitValueString value)
```

Properties:

|                                                       |                        |                   |
|-------------------------------------------------------|------------------------|-------------------|
| <b>active</b>                                         | <i>bool</i>            | <i>Read/Write</i> |
| <b>firstBaseline</b>                                  | <i>BaselineType</i>    | <i>Read/Write</i> |
| Get/Set value of the BaselineType enumeration type.   |                        |                   |
| <b>applyValue</b>                                     | <i>BaselineOffset</i>  | <i>Read/Write</i> |
| Get/Set value of the BaselineOffset enumeration type. |                        |                   |
| <b>value</b>                                          | <i>UnitValueString</i> | <i>Read/Write</i> |

Get/Set offset value.

#### Examples:

```
var objectcontent = ...; //get some object content
var bp = new BaselinePosition();
bp.firstBaseline = BaselineType.AscentHeight;
bp.applyValue = BaselineOffset.Fixed;
bp.value = "10pt";
objectcontent.baselinePosition = bp;

var bp1 = objectcontent.baselinePosition;
bp1.value = 15;
    bp1.firstBaseline = BaselineType.LineSpacing;
bp1.applyValue = BaselineOffset.Fixed;
bp1.value = "20pt";

var bp2 = new BaselinePosition(
    BaselineType.CapHeight, BaselineOffset.Fixed, "2cm");
objectcontent.baselinePosition = bp2;
```

## Class: ColorOverlay

#### Constructor:

```
new ColorOverlay()
```

#### Properties:

|                |                         |                   |                          |
|----------------|-------------------------|-------------------|--------------------------|
| <b>active</b>  | <i>bool</i>             | <i>Read/Write</i> |                          |
| <b>color</b>   | <i>ColorBrush</i>       | <i>Read/Write</i> |                          |
| <b>shade</b>   | <i>float</i>            | <i>Read/Write</i> | <i>[%] &lt;0,100&gt;</i> |
| <b>opacity</b> | <i>float</i>            | <i>Read/Write</i> | <i>[%] &lt;0,100&gt;</i> |
| <b>mode</b>    | <i>ColorMode (enum)</i> | <i>Read/Write</i> |                          |

## Class: ColumnLines

The class is used in the class GraphicStyleSheet.

#### Constructor:

```
new ColumnLines()
new ColumnLines(ColorBrush lineColor, float lineShade, float lineOpacity,
    String lineWidth, String lineStyle,
    String topIndent, String bottomIndent,
    TextColumnLineAnchor topAnchor, TextColumnLineAnchor bottomAnchor)
```

#### Properties:

|                  |                   |                   |
|------------------|-------------------|-------------------|
| <b>active</b>    | <i>bool</i>       | <i>Read/Write</i> |
| <b>lineColor</b> | <i>ColorBrush</i> | <i>Read/Write</i> |
| <b>lineShade</b> | <i>float</i>      | <i>Read/Write</i> |

|                     |                             |                   |
|---------------------|-----------------------------|-------------------|
| <b>lineOpacity</b>  | <i>float</i>                | <i>Read/Write</i> |
| <b>lineWidth</b>    | <i>String</i>               | <i>Read/Write</i> |
| <b>lineStyle</b>    | <i>String</i>               | <i>Read/Write</i> |
| <b>topIndent</b>    | <i>String</i>               | <i>Read/Write</i> |
| <b>bottomIndent</b> | <i>String</i>               | <i>Read/Write</i> |
| <b>topAnchor</b>    | <i>TextColumnLineAnchor</i> | <i>Read/Write</i> |
| <b>bottomAnchor</b> | <i>TextColumnLineAnchor</i> | <i>Read/Write</i> |

## Class: Comment

The class is used in the class FormattedText.

### Constructor:

```
new Comment()
```

### Properties:

|                                                                                    |               |                   |
|------------------------------------------------------------------------------------|---------------|-------------------|
| <b>text</b>                                                                        | <i>String</i> | <i>Read/Write</i> |
| <b>author</b>                                                                      | <i>String</i> | <i>Read/Write</i> |
| <b>creationDate</b>                                                                | <i>String</i> | <i>Read/Write</i> |
| Format: "dd.MM.yyyy hh:mm:ss" is expected.                                         |               |                   |
| <b>modificationDate</b>                                                            | <i>String</i> | <i>Read/Write</i> |
| Format: "dd.MM.yyyy hh:mm:ss" is expected.                                         |               |                   |
| <b>valid</b>                                                                       | <i>bool</i>   | <i>Read</i>       |
| Returns <i>true</i> if a comment exists at the position where the object was read. |               |                   |

## Class: DataTable

The class reads a sheet of an Excel file (.xlsx) as a table of strings. It requires the XLSX plugin of VivaDesigner.

### Constructor:

```
new DataTable()
```

### Properties:

|                                                                                                         |            |             |
|---------------------------------------------------------------------------------------------------------|------------|-------------|
| <b>columnCount</b>                                                                                      | <i>int</i> | <i>Read</i> |
| <b>rowCount</b>                                                                                         | <i>int</i> | <i>Read</i> |
| Returns the number of rows including the header row. The row 0 is the header row with the column names. |            |             |

### Methods:

```
loadExcelSheet(String filePath, String sheetName, String colorColumnName) bool
```

---

**loadExcelSheet**(*String filePath, String sheetName, String[] colorColumnNames*) *bool*

Loads the sheet *sheetName* of the Excel file *filePath*. Returns *false* if the file does not exist or contains no data.

---

**columnIndex**(*String columnName*) *int*

Returns the index of the column with the given name in the header row (upper and lower case are not distinguished) or -1.

---

**rowAt**(*int index*) *String[]*

Returns the row at the given index. *rowAt(0)* returns the header row.

---

**rowByFieldContent**(*String fieldContent, int columnIndex*) *String[]*

Returns the first row whose field in the column *columnIndex* equals *fieldContent*, or an empty list.

---

## Class: DropCaps

The feature DropCaps sets the drop caps at the beginning of each paragraph that is placed in the text range. The class is used in the classes *FormattedText* and *ParagraphStyleSheet*.

### Constructor:

*new DropCaps()*

*new DropCaps(DropCapsType mode)*

### Properties:

|                                                                                                                            |                              |                   |
|----------------------------------------------------------------------------------------------------------------------------|------------------------------|-------------------|
| <b>active</b>                                                                                                              | <i>bool</i>                  | <i>Read</i>       |
| <b>type</b>                                                                                                                | <i>DropCapsType</i>          | <i>Read/Write</i> |
| The feature DropCaps sets the drop caps at the beginning of each paragraph that is placed in the text range.               |                              |                   |
| <b>linesType</b>                                                                                                           | <i>DropCapsLinesType</i>     | <i>Read</i>       |
| Get value of the DropCapsLinesType enumeration type.                                                                       |                              |                   |
| <b>minimumLines</b>                                                                                                        | <i>int</i>                   | <i>Read</i>       |
| Defines the minimum number of lines for the DropCaps feature. It works for <i>DropCapsLinesType.Fixed-LineHeight</i> only. |                              |                   |
| <b>maximumLines</b>                                                                                                        | <i>int</i>                   | <i>Read</i>       |
| Defines the maximum number of lines for the DropCaps feature.                                                              |                              |                   |
| <b>characterCount</b>                                                                                                      | <i>int</i>                   | <i>Read/Write</i> |
| Defines the number of characters used for DropCaps. It works for <i>DropCapsType.Sequence</i> only.                        |                              |                   |
| <b>runaroundType</b>                                                                                                       | <i>DropCapsRunaroundType</i> | <i>Read/Write</i> |
| Defines the type of runaround.                                                                                             |                              |                   |
| <b>distanceToText</b>                                                                                                      | <i>UnitValueString</i>       | <i>Read/Write</i> |
| Defines distance from the DropCaps text to the following text.                                                             |                              |                   |
| <b>scale</b>                                                                                                               | <i>float</i>                 | <i>Read/Write</i> |

---

Defines the scale of the DropCaps text.

### Methods:

---

**setFixLines** (*int minimum, int maximum*) *void*

Set drop caps features. The `DropCapsLinesType = FixedLineHeight` and set the minimum and the maximum.

---

**setDynamicLines** (*int maximum*) *void*

Set drop caps features. The `DropCapsLinesType = DynamicLineHeight` and set the maximum.

### Examples:

```
var ft = ... //set FormattedText object
var dc1 = new DropCaps();
dc1.type = DropCapsType.Sequence;
dc1.setFixLines(4,8);
dc1.characterCount = 2;
dc1.runaroundType = DropCapsRunaroundType.RunaroundBox;
dc1.distanceToText = "13mm";
dc1.scale = 60;
messageBox.information("MSG", ft.dropCaps(30).type);
ft.setDropCaps(dc1, new TextRange(3, 50));
messageBox.information("MSG", ft.dropCaps(30).type);
//clear drop caps
ft.setDropCaps(new DropCaps(DropCapsType.None), 3, 50);
messageBox.information("MSG", ft.dropCaps(30).type)
```

## Class: Enumerations

The feature is named "Bullets & Numbering" in the VivaDesigner interface. Defines features of numbering paragraphs in the selected text. The text related to Enumerations object parameters is added before each paragraph.

The class is used in class `FormattedText`, `ParagraphStyleSheet`.

### Constructor:

**new Enumerations()**

The constructor for `EnumerationsType = Numbering` is the following:

```
new Enumerations(EnumerationsNumberingMode mode,
    int startWith, int level,
    EnumerationsAlignment justification,
    EnumerationsNumberFormat format,
    bool tabulator,
    String prefix, String postfix,
    String characterStyleSheet)
```

The constructor for `EnumerationsType = Bullets`

```
new Enumerations(BulletCharacters bulletType,
    Char bullet, (insignificant for bulletType != Custom)
```

```

    int level,
    EnumerationsAlignment justification,
    bool tabulator,
    String prefix, String postfix,
    String characterStyleSheet)

```

**Properties:**

|                                                                   |                                  |                               |
|-------------------------------------------------------------------|----------------------------------|-------------------------------|
| <b>on</b>                                                         | <i>bool</i>                      | <i>Read</i>                   |
| Returns <i>true</i> if bullets or numbering are switched on.      |                                  |                               |
| <b>type</b>                                                       | <i>EnumerationsType</i>          | <i>Read/Write</i>             |
| <b>mode</b>                                                       | <i>EnumerationsNumberingMode</i> | <i>Read/Write</i>             |
| <b>startWith</b>                                                  | <i>int</i>                       | <i>Read/Write (1- )</i>       |
| <b>level</b>                                                      | <i>int</i>                       | <i>Read/Write (1-10)</i>      |
| <b>alignment</b>                                                  | <i>EnumerationsAlignment</i>     | <i>Read/Write</i>             |
| <b>bullet</b>                                                     | <i>Char</i>                      | <i>Read</i>                   |
| Use the method <i>setBullet()</i> to change the bullet character. |                                  |                               |
| <b>prefix</b>                                                     | <i>String</i>                    | <i>Read/Write (maxlen=10)</i> |
| <b>postfix</b>                                                    | <i>String</i>                    | <i>Read/Write (maxlen=10)</i> |
| <b>tabulator</b>                                                  | <i>bool</i>                      | <i>Read/Write</i>             |
| <b>characterStyleSheetByName</b>                                  | <i>String</i>                    | <i>Read/Write</i>             |
| <b>characterStyleSheet</b>                                        | <i>CharacterStyleSheet</i>       | <i>Read/Write</i>             |
| <b>numberFormat</b>                                               | <i>EnumerationsNumberFormat</i>  | <i>Read/Write</i>             |

**Methods:**

```

setBullet(BulletCharacters type, Char customChar)    void

```

**Examples:**

```

    var ft = ... // set formatted text

    //set empty constructor
    var en = new Enumerations();
    en.mode = EnumerationsNumberingMode.BeginNew;
    en.startWith = 3;
    en.level = 2;
    en.alignment = EnumerationsAlignment.Center;
    en.setBullet(BulletCharacters.MiddleDot);
    en.prefix = "AAAA";
    en.postfix = "BBBB";
    en.tabulator = true;
    en.styleSheet = "MyCharacter";
    en.numberFormat = EnumerationsNumberFormat.Format_i_ii_iii;

    //constructor for EnumerationsType.Numbering
    var en2 = new Enumerations(EnumerationsNumberingMode.BeginNew, 3, 2,
        EnumerationsAlignment.Center,

```

```

EnumerationsNumberFormat.Format_i_ii_iii, true,
    "AAA", "BBB", "MyCharacter");

//constructor for EnumerationsType.Bullets
var en3 = new Enumerations(BulletCharacters.MiddleDot, 'a', 2,
    EnumerationsAlignment.Center, true,
    "AAA", "BBB", "MyCharacter");

ft.setEnumerations(EnumerationsType.Numbering, en2, 4, 30);
ft.setEnumerations(EnumerationsType.Bullets, en3, 60);

messageBox.information("MSG", "Type: " + ft.getEnumerationsType(25));
var en4 = ft.getEnumerationsData(25);
messageBox.information("MSG", "Data: StartWith, Tabulator: " +
    en4.startWith + "," + en4.tabulator);
messageBox.information("MSG", "Data: StartWith, Tabulator: " +
    ft.getEnumerationsData(25).startWith + "," +
    ft.getEnumerationsData(25).tabulator);

```

## Class: FontSize

The class is used in the classes *FormattedText*, *CharacterStyleSheet* and *ParagraphStyleSheet*.

### Constructor:

```

new FontSize(float size)
new FontSize(float size, FontReferenceSizeType type)

```

### Properties:

| type | FontReferenceSizeType | Read/Write |
|------|-----------------------|------------|
| size | float                 | Read/Write |

## Class: GraphicFootnotes

The class is used in the classes *GraphicStyleSheet* and *TextContent*.

### Constructor:

```

new GraphicFootnotes()

```

### Properties:

|                |                   |            |
|----------------|-------------------|------------|
| position       | FootnotesPosition | Read/Write |
| columnCount    | int               | Read/Write |
| columnDistance | String            | Read/Write |
| startColumn    | int               | Read/Write |
| width          | int               | Read/Write |

## Class: Guide

The object of the class represents a guide of a page or of a spread. It is returned by the method `guide(int id)` of the classes `Page` and `Spread`.

### Properties:

|               |             |             |
|---------------|-------------|-------------|
| <b>isNull</b> | <i>bool</i> | <i>Read</i> |
|---------------|-------------|-------------|

Returns *true* if no guide with the given id exists.

## Class: HeaderFooter

The class is used in the classes `GraphicStyleSheet` and `TextContent`.

### Constructor:

```
new HeaderFooter()
```

### Properties:

|               |                        |                   |
|---------------|------------------------|-------------------|
| <b>active</b> | <i>bool</i>            | <i>Read/Write</i> |
| <b>height</b> | <i>UnitValueString</i> | <i>Read/Write</i> |

## Class: Hyphenation

The class is used in the classes `ParagraphStyleSheet` and `FormattedText`.

### Constructor:

```
new Hyphenation()
new Hyphenation(short smallestWord,
                short maxHyphensInARow,
                short minPrefix,
                short minSuffix,
                HyphenationQuality quality)
```

### Properties:

|                         |                           |                   |
|-------------------------|---------------------------|-------------------|
| <b>active</b>           | <i>bool</i>               | <i>Read/Write</i> |
| <b>smallestWord</b>     | <i>int</i>                | <i>Read/Write</i> |
| <b>maxHyphensInARow</b> | <i>int</i>                | <i>Read/Write</i> |
| <b>minPrefix</b>        | <i>int</i>                | <i>Read/Write</i> |
| <b>minSuffix</b>        | <i>int</i>                | <i>Read/Write</i> |
| <b>quality</b>          | <i>HyphenationQuality</i> | <i>Read/Write</i> |

### Examples:

```
var ft = ... // get formatted text
var hyp = new Hyphenation();
hyp.on = true;
hyp.smallestWord = 10;
hyp.maxHyphensInARow = 11;
```

```
hyp.minPrefix = 12;
hyp.minSuffix = 13;
hyp.quality = HyphenationQuality.GoodQuality;

var hyp2 = new Hyphenation(20, 21, 22, 23,
HyphenationQuality.StandardQuality);
ft.setHyphenation(true, hyp);
ft.setHyphenation(true, hyp2, 10, 20);

messageBox.information("Msg:", "Hyphenation IsOn : " + ft.hyphenation());
var hyp1 = ft.hyphenationData(15);
messageBox.information("Msg:", "Hyphenation: On: " + hyp1.on);
messageBox.information("Msg:", "Hyphenation: smallestWord: " +
    hyp1.smallestWord);

ft.setHyphenation(false);
```

## Class: ChangePictureColorSpaceSettings

The class holds settings for the picture color space changing.

### Constructor:

```
new ChangePictureColorSpaceSettings();
```

### Properties:

| <b>format</b>                              | <i>PictureFormat (enum)</i> | <i>Read/Write</i> |
|--------------------------------------------|-----------------------------|-------------------|
| Valid only if method outFilePath is empty. |                             |                   |
| <b>colorSpace</b>                          | <i>ColorSpace(enum)</i>     | <i>Read/Write</i> |
| <b>colorProfile</b>                        | <i>String</i>               | <i>Read/Write</i> |
| <b>prefix</b>                              | <i>String</i>               | <i>Read/Write</i> |
| <b>postfix</b>                             | <i>String</i>               | <i>Read/Write</i> |
| <b>outFilePath</b>                         | <i>String</i>               | <i>Read/Write</i> |
| Embed the image if empty.                  |                             |                   |

## Class: CharacterBackground

The object of the class holds the parameters of the character background.

The class is used in classes FormattedText, CharacterStyleSheet and ParagraphStyleSheet.

### Constructor:

```
new CharacterBackground ()
new CharacterBackground (bool active,
    ColorBrush brush, float shade, float opacity,
    String leftIndent, String rightIndent,
    String topIndent, String bottomIndent)
```

## Properties:

|                     |                        |                   |
|---------------------|------------------------|-------------------|
| <b>active</b>       | <i>bool</i>            | <i>Read/Write</i> |
| <b>brush</b>        | <i>ColorBrush</i>      | <i>Read/Write</i> |
| <b>shade</b>        | <i>float</i>           | <i>Read/Write</i> |
| <b>opacity</b>      | <i>float</i>           | <i>Read/Write</i> |
| <b>leftIndent</b>   | <i>UnitValueString</i> | <i>Read/Write</i> |
| <b>rightIndent</b>  | <i>UnitValueString</i> | <i>Read/Write</i> |
| <b>topIndent</b>    | <i>UnitValueString</i> | <i>Read/Write</i> |
| <b>bottomIndent</b> | <i>UnitValueString</i> | <i>Read/Write</i> |

## Examples:

```
var ft = ... // get formatted text

var chb = new CharacterBackground();
chb.brush = Color.Red;
chb.shade = 34;
chb.opacity = 56;
chb.topIndent = "1pt";
chb.leftIndent = "0pt";
chb.rightIndent = "0pt";
chb.bottomIndent = "2pt";

var chb2 = new CharacterBackground( true, Color.Green,
    100, 99, "0pt", "0pt", "-3mm", "-8mm");
ft.setCharacterBackground(true, chb2, new TextRange(10, 34));
ft.setCharacterBackground(true, chb, new TextRange(15, 25));
var chb1 = ft.characterBackgroundData(16);
var txt = "Brush: " + chb1.brush.uniqueName + "\nShade: " +
    chb1.shade + "\nOpacity: " + chb1.opacity;
messageBox.information("CharacterBackground", txt);
ft.setCharacterBackground(false, chb, 28, 30);
```

## Class: CharacterFrame

The object of the class holds the parameters of the character frame.

The class is used in classes FormattedText, CharacterStyleSheet and ParagraphStyleSheet.

## Constructor:

```
new CharacterFrame()
new CharacterFrame(bool active,
    ColorBrush brush, float shade, float opacity,
    UnitValueString leftIndent, UnitValueString rightIndent,
    UnitValueString topIndent, UnitValueString bottomIndent,
    UnitValueString frameWidth, String lineStyle)
```

## Properties:

|                     |                        |                   |
|---------------------|------------------------|-------------------|
| <b>active</b>       | <i>bool</i>            | <i>Read/Write</i> |
| <b>brush</b>        | <i>ColorBrush</i>      | <i>Read/Write</i> |
| <b>shade</b>        | <i>float</i>           | <i>Read/Write</i> |
| <b>opacity</b>      | <i>float</i>           | <i>Read/Write</i> |
| <b>leftIndent</b>   | <i>UnitValueString</i> | <i>Read/Write</i> |
| <b>rightIndent</b>  | <i>UnitValueString</i> | <i>Read/Write</i> |
| <b>topIndent</b>    | <i>UnitValueString</i> | <i>Read/Write</i> |
| <b>bottomIndent</b> | <i>UnitValueString</i> | <i>Read/Write</i> |
| <b>frameWidth</b>   | <i>UnitValueString</i> | <i>Read/Write</i> |
| <b>lineStyle</b>    | <i>LineStyleString</i> | <i>Read/Write</i> |

## Examples:

```

var ft = ... // get formatted text

var chf = new CharacterFrame();
chf.active = true;
chf.brush = Color.Red;
chf.shade = 34;
chf.opacity = 56;
chf.frameWidth = "1pt";
chf.lineStyle = "SolidLine";
chf.topIndent = "0mm";
chf.leftIndent = "0mm";
chf.rightIndent = "0mm";
chf.bottomIndent = "0mm";
ft.setCharacterFrame(true, chf, 10, 11);

var chf1 = new CharacterFrame(true, Color.Green,
    55, 77, "2pt", "3pt", "1pt", "-8pt", "2pt", "DashDotLine");
ft.setCharacterFrame(chf1, new TextRange(15, 30));
ft.setCharacterFrame(chf, new TextRange(18, 20));

```

## Class: CharacterKerning

The class is used in the class *FormattedText*, *CharacterStyleSheet* and *ParagraphStyleSheet*.

## Constructor:

```
new CharacterKerning()
```

## Properties:

|              |                    |                          |
|--------------|--------------------|--------------------------|
| <b>type</b>  | <i>KerningType</i> | <i>Read/Write (enum)</i> |
| <b>value</b> | <i>float</i>       | <i>Read/Write</i>        |

## Class: CharacterPositionCorrection

The class is used in the class `FormattedText`.

Constructor:

```
new CharacterPositionCorrection()
new CharacterPositionCorrection(int horizontal, int vertical)
```

Properties:

|                             |            |                   |
|-----------------------------|------------|-------------------|
| <b>horizontalCorrection</b> | <i>int</i> | <i>Read/Write</i> |
| <b>verticalCorrection</b>   | <i>int</i> | <i>Read/Write</i> |

Examples:

```
var ft = ... // get formatted text
var cps = new CharacterPositionCorrection();
cps.horizontalCorrection = 35;
cps.verticalCorrection = -15;
ft.setPositionCorrection(cps, 10);
messageBox.information("MSG",
ft.positionCorrection(10).verticalCorrection);
```

## Class: CharacterRule

The object of the class holds parameters of the character rule function for the text.

The class is used in classes *FormattedText*, *CharacterStyleSheet* and *ParagraphStyleSheet*.

Constructor:

```
new CharacterRule()
new CharacterRule(bool active, ColorBrush brush, float shade, float opacity,
UnitValueString lineWidth, String lineStyle, UnitValueString
offset,
UnitValueString leftIndent, UnitValueString rightIndent)
```

Properties:

|                    |                        |                   |
|--------------------|------------------------|-------------------|
| <b>active</b>      | <i>boolean</i>         | <i>Read/Write</i> |
| <b>brush</b>       | <i>ColorBrush</i>      | <i>Read/Write</i> |
| <b>shade</b>       | <i>float</i>           | <i>Read/Write</i> |
| <b>opacity</b>     | <i>float</i>           | <i>Read/Write</i> |
| <b>lineWidth</b>   | <i>UnitValueString</i> | <i>Read/Write</i> |
| <b>lineStyle</b>   | <i>LineStyleString</i> | <i>Read/Write</i> |
| <b>offset</b>      | <i>UnitValueString</i> | <i>Read/Write</i> |
| <b>leftIndent</b>  | <i>UnitValueString</i> | <i>Read/Write</i> |
| <b>rightIndent</b> | <i>UnitValueString</i> | <i>Read/Write</i> |

Examples:

```
var ft = ... // get formatted text
```

```
var chr1 = new CharacterRule( true, Color.Green, 60, 50, "2pt",  
"SolidLine",  
    "0mm", "0mm", "0mm");  
var chr = new CharacterRule();  
chr.active = true;  
chr.brush = Color.Magenta;  
chr.shade = 91;  
chr.opacity = 82;  
chr.lineWidth = "3mm";  
chr.lineStyle = "DotLine";  
chr.offset = "2mm";  
chr.leftIndent = "1mm";  
chr.rightIndent = "1mm";  
ft.setCharacterRule(chr, new TextRange(10, 20));  
ft.setCharacterRule(chr, 15, 17);  
ft.setCharacterRule(chr1, new TextRange(25,35));
```

## Class: Image

The object of the class holds a bitmap image.

The class is used in the class `PictureContent` (properties `image` and `preview`).

### Constructor:

```
new Image()  
new Image(String data)
```

### Properties:

| data                                                                                                                                                                                      | String      | Read |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------|------|
| Returns the image as a JSON string with the image size, the format and the pixel data (Base64). The same string is accepted by the constructor and by the method <code>setData()</code> . |             |      |
| width                                                                                                                                                                                     | int         | Read |
| height                                                                                                                                                                                    | int         | Read |
| format                                                                                                                                                                                    | ImageFormat | Read |

### Methods:

|                                     |                   |
|-------------------------------------|-------------------|
| <code>setData( String data )</code> | <code>void</code> |
|-------------------------------------|-------------------|

## Class: InnerGlow

The class is used in the class `GraphicObject` for all objects and in the class `GraphicStyleSheet`. The class holds the parameters used for the “Inner Glow” effect of the object.

### Constructor:

```
new InnerGlow()
```

## Properties:

|                |                         |                        |            |                      |
|----------------|-------------------------|------------------------|------------|----------------------|
| <b>active</b>  | <i>bool</i>             | <i>Read/Write</i>      |            |                      |
| <b>color</b>   | <i>ColorBrush</i>       | <i>Read/Write</i>      |            |                      |
| <b>shade</b>   | <i>float</i>            | <i>Read/Write</i>      | <i>[%]</i> | <i>&lt;0-100&gt;</i> |
| <b>opacity</b> | <i>float</i>            | <i>Read/Write</i>      | <i>[%]</i> | <i>&lt;0-100&gt;</i> |
| <b>mode</b>    | <i>ColorMode (enum)</i> | <i>Read/Write</i>      |            |                      |
| <b>blur</b>    | <i>UnitValueString</i>  | <i>Read/Write [mm]</i> |            | <i>&lt;0,20&gt;</i>  |

## Class: LayoutColumnsInfo

The class is used in the classes *LayoutStyleSheet* and *FormattedText*.

## Constructor:

```
new LayoutColumnsInfo()
```

## Properties:

|              |                          |                   |
|--------------|--------------------------|-------------------|
| <b>count</b> | <i>int</i>               | <i>Read</i>       |
| <b>type</b>  | <i>LayoutColumnsType</i> | <i>Read/Write</i> |

## Methods:

|                                                                           |                        |
|---------------------------------------------------------------------------|------------------------|
| <b>setCount</b> ( <i>int n, UnitValueString margin</i> )                  | <i>void</i>            |
| <b>setWidthAt</b> ( <i>int index, UnitValueString width</i> )             | <i>void</i>            |
| <b>widthAt</b> ( <i>int index</i> )                                       | <i>UnitValueString</i> |
| <b>setMarginAt</b> ( <i>int index, UnitValueString margin</i> )           | <i>void</i>            |
| <b>marginAt</b> ( <i>int index</i> )                                      | <i>UnitValueString</i> |
| <b>addColumn</b> ( <i>UnitValueString width, UnitValueString margin</i> ) | <i>void</i>            |

Appends a column with the given width and margin.

## Examples:

```
var ft = ... // get formatted text
var lci = new LayoutColumnsInfo();

// set columns for type "Automatic"
lci.setCount(4, "5mm");

// set columns for type "Manual"
lci.setWidthAt(0, "33pt");
lci.setMarginAt(0, "2pt");
lci.setWidthAt(1, "50pt");
lci.setMarginAt(1, "5pt");
ft.setColumns(lci, new TextRange(10, 15));

var lci1 = ft.columnsData(12);
var txt = "Width(1)=" + lci1.widthAt(1) + "; Margin(1)=" +
```

```
lcil.marginAt(1) +      "; Count=" + lcil.count;
    messageBox.information("MSG", txt);
```

## Class: LayoutFootnotes

The class is used in the classes *LayoutStyleSheet* and *FormattedText*.

### Constructor:

```
new LayoutFootnotes()
```

### Properties:

|                                |                            |                   |
|--------------------------------|----------------------------|-------------------|
| <b>mode</b>                    | <i>LayoutFootnotesMode</i> | <i>Read/Write</i> |
| <b>columnCount</b>             | <i>int</i>                 | <i>Read/Write</i> |
| <b>margin</b>                  | <i>UnitValueString</i>     | <i>Read/Write</i> |
| <b>placeStartColumn</b>        | <i>int</i>                 | <i>Read/Write</i> |
| <b>placeMaximumColumnCount</b> | <i>int</i>                 | <i>Read/Write</i> |

### Examples:

```
var ft = ... // get formatted text

var fn = new LayoutFootnotes();
fn.columnCount = 4;
fn.margin = "1cm";
fn.placeStartColumn = 2;
fn.placeMaximumColumnCount = 3;
ft.setLayoutFootnotes(LayoutFootnotesMode.LayoutDependent, fn,
    new TextRange(10, 20));

var fn1 = ft.layoutFootnotesData(15);
var info = "LayoutFootnotes: \n" +
    "  Mode: " + fn1.mode + "\n" +
    "  Column Count: " + fn1.columnCount + "\n" +
    "  Margin: " + fn1.margin + "\n" +
    "  PlaceStartColumn: " + fn1.placeStartColumn + "\n" +
    "  PlaceMaximumColumnCount: " + fn1.placeMaximumColumnCount + "\n";
messageBox.information("MSG", info);
```

## Class: LineStyle

The class is used in the classes *TableCellStyleSheet*, *TableColumnStyleSheet*, *TableRowStyleSheet*.

### Constructor:

```
new LineStyle()
new LineStyle(ColorBrush color, float shade,
    String lineStyle, String lineWidth)
```

### Properties:

|                  |                   |                   |
|------------------|-------------------|-------------------|
| <b>active</b>    | <i>bool</i>       | <i>Read</i>       |
| <b>lineBrush</b> | <i>ColorBrush</i> | <i>Read/Write</i> |

|                             |                        |                   |
|-----------------------------|------------------------|-------------------|
| <b>lineShade</b>            | <i>float</i>           | <i>Read/Write</i> |
| <b>lineStyle</b>            | <i>LineStyleString</i> | <i>Read/Write</i> |
| Returns LineStyle name.     |                        |                   |
| <b>lineWidth</b>            | <i>UnitValueString</i> | <i>Read/Write</i> |
| Returns line width in [pt]. |                        |                   |

## Class: LineNumbers

The class represents Line Counter data. It is used in classes *TextPreferences* and *FormattedText*.

### Constructor:

```
new LineNumbers()

new LineNumbers (LineNumbersMode mode,
    int startsWith, int step,
    UnitValueString offset,
    LineNumbersRestartType restartType,
    bool supressEmptyParagraphs)
```

### Properties:

|                               |                               |                   |
|-------------------------------|-------------------------------|-------------------|
| <b>on</b>                     | <i>bool</i>                   | <i>Read</i>       |
| <b>mode</b>                   | <i>LineNumbersMode</i>        | <i>Read/Write</i> |
| <b>startsWith</b>             | <i>int</i>                    | <i>Read/Write</i> |
| <b>step</b>                   | <i>int</i>                    | <i>Read/Write</i> |
| <b>offset</b>                 | <i>UnitValueString</i>        | <i>Read/Write</i> |
| <b>restartType</b>            | <i>LineNumbersRestartType</i> | <i>Read/Write</i> |
| <b>supressEmptyParagraphs</b> | <i>bool</i>                   | <i>Read/Write</i> |

## Class: Link

The class is used in the class *FormattedText*, *CharacterStyleSheet* and *ParagraphStyleSheet*.

### Constructor:

```
new Link()

new Link (bool active,
    String documentName,
    LinkAction action,
    int pageNumber)
```

### Properties:

|                                   |             |                   |
|-----------------------------------|-------------|-------------------|
| <b>active</b> ( <i>obsolete</i> ) | <i>bool</i> | <i>Read/Write</i> |
|-----------------------------------|-------------|-------------------|

|                                                          |                        |                   |
|----------------------------------------------------------|------------------------|-------------------|
| <b>documentName</b> <i>(only if destination != None)</i> | <i>string</i>          | <i>Read/Write</i> |
| <b>linkAction</b> <i>(only if destination = Page)</i>    | <i>linkAction</i>      | <i>Read</i>       |
| <b>pageNumber</b> <i>(only if destination = Page)</i>    | <i>int</i>             | <i>Read</i>       |
| <b>destination</b>                                       | <i>LinkDestination</i> | <i>Read/Write</i> |

Use the numeric values 0 (None), 1 (Page) and 2 (Url).

**Methods:**

|                                                                                                  |             |
|--------------------------------------------------------------------------------------------------|-------------|
| <b>setAction</b> <i>(LinkAction action, int pageNumber)</i><br><i>only if destination = Page</i> | <i>void</i> |
|--------------------------------------------------------------------------------------------------|-------------|

## Class: Note

The class is used in a class `FormattedText`.

**Properties:**

|                                                                    |                        |                   |
|--------------------------------------------------------------------|------------------------|-------------------|
| <b>type</b>                                                        | <i>NoteType</i>        | <i>Read/Write</i> |
| <b>text</b>                                                        | <i>String</i>          | <i>Read/Write</i> |
| <b>(Err: If text is set, the anchor is not shown in the note.)</b> |                        |                   |
| <b>anchorText</b>                                                  | <i>String</i>          | <i>Read/Write</i> |
| <b>position</b>                                                    | <i>int</i> <i>Read</i> |                   |
| <b>mainText</b>                                                    | <i>FormattedText</i>   | <i>Read</i>       |

## Class: OpenType

**Constructor:**

*new* **OpenType**( )

**Properties:**

|                             |                    |                   |
|-----------------------------|--------------------|-------------------|
| <i>contextualAlternates</i> | <i>ThreeStates</i> | <i>Read/Write</i> |
| <i>capitalSpacing</i>       | <i>ThreeStates</i> | <i>Read/Write</i> |
| <i>fractions</i>            | <i>ThreeStates</i> | <i>Read/Write</i> |
| <i>ordinals</i>             | <i>ThreeStates</i> | <i>Read/Write</i> |
| <i>swash</i>                | <i>ThreeStates</i> | <i>Read/Write</i> |
| <i>slashedZero</i>          | <i>ThreeStates</i> | <i>Read/Write</i> |
| <i>caseSensitiveForms</i>   | <i>ThreeStates</i> | <i>Read/Write</i> |

Helped Data Objects | Class: OpenType

|                                   |                                |                   |
|-----------------------------------|--------------------------------|-------------------|
| <i>stylisticAlternates</i>        | <i>int</i>                     | <i>Read/Write</i> |
| <i>stylistic1</i>                 | <i>int</i>                     | <i>Read/Write</i> |
| <i>stylistic2</i>                 | <i>int</i>                     | <i>Read/Write</i> |
| <i>stylistic3</i>                 | <i>int</i>                     | <i>Read/Write</i> |
| <i>stylistic4</i>                 | <i>int</i>                     | <i>Read/Write</i> |
| <i>stylistic5</i>                 | <i>int</i>                     | <i>Read/Write</i> |
| <i>stylistic6</i>                 | <i>int</i>                     | <i>Read/Write</i> |
| <i>stylistic7</i>                 | <i>int</i>                     | <i>Read/Write</i> |
| <i>stylistic8</i>                 | <i>int</i>                     | <i>Read/Write</i> |
| <i>stylistic9</i>                 | <i>int</i>                     | <i>Read/Write</i> |
| <i>stylistic10</i>                | <i>int</i>                     | <i>Read/Write</i> |
| <i>stylistic11</i>                | <i>int</i>                     | <i>Read/Write</i> |
| <i>stylistic12</i>                | <i>int</i>                     | <i>Read/Write</i> |
| <i>stylistic13</i>                | <i>int</i>                     | <i>Read/Write</i> |
| <i>stylistic14</i>                | <i>int</i>                     | <i>Read/Write</i> |
| <i>stylistic15</i>                | <i>int</i>                     | <i>Read/Write</i> |
| <i>stylistic16</i>                | <i>int</i>                     | <i>Read/Write</i> |
| <i>stylistic17</i>                | <i>int</i>                     | <i>Read/Write</i> |
| <i>stylistic18</i>                | <i>int</i>                     | <i>Read/Write</i> |
| <i>stylistic19</i>                | <i>int</i>                     | <i>Read/Write</i> |
| <i>stylistic20</i>                | <i>int</i>                     | <i>Read/Write</i> |
| <i>placement</i>                  | <i>OpenTypePlacement</i>       | <i>Read/Write</i> |
| <i>numerals</i>                   | <i>OpenTypeNumerals</i>        | <i>Read/Write</i> |
| <i>positionalForms</i>            | <i>OpenTypePositionalForms</i> | <i>Read/Write</i> |
| <i>smallCapitalsFromCapitals</i>  | <i>ThreeStates</i>             | <i>Read/Write</i> |
| <i>smallCapitals</i>              | <i>ThreeStates</i>             | <i>Read/Write</i> |
| <i>petiteCapitalsFromCapitals</i> | <i>ThreeStates</i>             | <i>Read/Write</i> |
| <i>petiteCapitals</i>             | <i>ThreeStates</i>             | <i>Read/Write</i> |
| <i>titling</i>                    | <i>ThreeStates</i>             | <i>Read/Write</i> |
| <i>unicase</i>                    | <i>ThreeStates</i>             | <i>Read/Write</i> |

|                                                                                                           |                    |                   |
|-----------------------------------------------------------------------------------------------------------|--------------------|-------------------|
| <i>discretionaryLigatures</i>                                                                             | <i>ThreeStates</i> | <i>Read/Write</i> |
| <i>historicalLigatures</i>                                                                                | <i>ThreeStates</i> | <i>Read/Write</i> |
| <b>featuresString</b>                                                                                     | <i>String</i>      | <i>Read/Write</i> |
| Gets or sets all OpenType features as one string of feature tags and values, for example "calt=1,frac=0". |                    |                   |

## Class: OptimizePictureSettings

The class holds settings for the picture optimizing.

Constructor:

```
new OptimizePictureSettings();
```

Properties:

|                                                |                             |                         |
|------------------------------------------------|-----------------------------|-------------------------|
| <b>crop</b>                                    | <i>bool</i>                 | <i>Read/Write</i>       |
| <b>prefix</b>                                  | <i>String</i>               | <i>Read/Write</i>       |
| <b>postfix</b>                                 | <i>String</i>               | <i>Read/Write</i>       |
| <b>format</b>                                  | <i>PictureFormat (enum)</i> | <i>Read/Write</i>       |
| Valid only if the method outFilePath is empty. |                             |                         |
| <b>bleed</b>                                   | <i>float</i>                | <i>Read/Write [pt]</i>  |
| <b>colorImageResolution</b>                    | <i>float</i>                | <i>Read/Write [ppi]</i> |
| <b>grayImageResolution</b>                     | <i>float</i>                | <i>Read/Write [ppi]</i> |
| <b>bwImageResolution</b>                       | <i>float</i>                | <i>Read/Write [ppi]</i> |
| <b>outFilePath</b>                             | <i>String</i>               | <i>Read/Write</i>       |

Embed the image if empty.

## Class: OuterGlow

The class is used in the class *GraphicObject* for all objects and the class *GraphicStyleSheet*. The class holds the parameters used for the “Outer Glow” effect of the object.

Constructor:

```
new OuterGlow()
```

Properties:

|                |                   |                   |            |         |
|----------------|-------------------|-------------------|------------|---------|
| <b>active</b>  | <i>bool</i>       | <i>Read/Write</i> |            |         |
| <b>color</b>   | <i>ColorBrush</i> | <i>Read/Write</i> |            |         |
| <b>shade</b>   | <i>float</i>      | <i>Read/Write</i> | <i>[%]</i> | <0-100> |
| <b>opacity</b> | <i>float</i>      | <i>Read/Write</i> | <i>[%]</i> | <0-100> |

|             |                         |                   |                           |
|-------------|-------------------------|-------------------|---------------------------|
| <b>mode</b> | <i>ColorMode (enum)</i> | <i>Read/Write</i> |                           |
| <b>blur</b> | <i>UnitValueString</i>  | <i>Read/Write</i> | <i>[mm] &lt;0, 20&gt;</i> |

## Class: OutlineOptions

The object of the class holds the parameters of the text outline.

The class is used in classes *FormattedText*, *CharacterStyleSheet* and *ParagraphStyleSheet*.

### Constructor:

```
new OutlineOptions (ColorBrush brush, float shade, float opacity,
                    UnitValueString lineWidth, String lineStyle,
                    CharacterOutlineJoinStyle joinStyle)
```

### Properties:

|                  |                                  |                   |
|------------------|----------------------------------|-------------------|
| <b>active</b>    | <i>bool</i>                      | <i>Read/Write</i> |
| <b>brush</b>     | <i>ColorBrush</i>                | <i>Read/Write</i> |
| <b>shade</b>     | <i>float</i>                     | <i>Read/Write</i> |
| <b>opacity</b>   | <i>float</i>                     | <i>Read/Write</i> |
| <b>lineWidth</b> | <i>UnitValueString</i>           | <i>Read/Write</i> |
| <b>lineStyle</b> | <i>LineStyleString</i>           | <i>Read/Write</i> |
| <b>joinStyle</b> | <i>CharacterOutlineJoinStyle</i> | <i>Read/Write</i> |

### Examples:

```
var ol = new OutlineOptions(Color.Red, 44, 56, "1mm", "SolidLine",
                           CharacterOutlineJoinStyle.RoundJoin);
ft.setOutlineOptions(true, ol, new TextRange(10, 15, ft));
//switch off the feature
ft.setOutlineOptions(false, ol);
```

## Class: PackageSettings

The class is used in classes *Document*.

### Constructor:

```
new PackageSettings()
```

### Properties:

|                        |             |                   |
|------------------------|-------------|-------------------|
| <b>includeProtocol</b> | <i>bool</i> | <i>Read/Write</i> |
| <b>includeFonts</b>    | <i>bool</i> | <i>Read/Write</i> |
| <b>includeImages</b>   | <i>bool</i> | <i>Read/Write</i> |
| <b>includeDocument</b> | <i>bool</i> | <i>Read/Write</i> |
| <b>cleanupFolder</b>   | <i>bool</i> | <i>Read/Write</i> |

|                               |               |                   |
|-------------------------------|---------------|-------------------|
| <b>overwriteExistingFiles</b> | <i>bool</i>   | <i>Read/Write</i> |
| <b>useSubfolders</b>          | <i>bool</i>   | <i>Read/Write</i> |
| <b>fontsSubfolder</b>         | <i>String</i> | <i>Read/Write</i> |
| <b>imagesSubfolder</b>        | <i>String</i> | <i>Read/Write</i> |
| <b>zip</b>                    | <i>bool</i>   | <i>Read/Write</i> |

## Class: ParagraphBackground

The class is used in the classes *ParagraphStyleSheet* and *FormattedText*.

### Constructor:

```
new ParagraphBackground (bool active,
    ColorBrush brush, float shade, float opacity,
    UnitValueString leftIndent, UnitValueString rightIndent,
    UnitValueString topIndent, UnitValueString bottomIndent,
    ExtendToNextParagraphType extNext, bool extToColumn)
```

### Properties:

|                              |                                  |                   |
|------------------------------|----------------------------------|-------------------|
| <b>active</b>                | <i>bool</i>                      | <i>Read/Write</i> |
| <b>brush</b>                 | <i>ColorBrush</i>                | <i>Read/Write</i> |
| <b>shade</b>                 | <i>float</i>                     | <i>Read/Write</i> |
| <b>opacity</b>               | <i>float</i>                     | <i>Read/Write</i> |
| <b>leftIndent</b>            | <i>UnitValueString</i>           | <i>Read/Write</i> |
| <b>rightIndent</b>           | <i>UnitValueString</i>           | <i>Read/Write</i> |
| <b>topIndent</b>             | <i>UnitValueString</i>           | <i>Read/Write</i> |
| <b>bottomIndent</b>          | <i>UnitValueString</i>           | <i>Read/Write</i> |
| <b>extendToNextParagraph</b> | <i>ExtendToNextParagraphType</i> | <i>Read/Write</i> |
| <b>extendToColumnBorder</b>  | <i>bool</i>                      | <i>Read/Write</i> |

### Examples:

```
var pb = new ParagraphBackground(true, Color.Yellow, 82, 73,
    "0mm", "0mm", "0mm", "0mm", ExtendToNextParagraphType.No, false);
pb.brush = Color.Yellow;
pb.shade = 82;
pb.opacity = 73;
pb.topIndent = "-2mm";
pb.leftIndent = "-2mm";
pb.rightIndent = "-1mm";
pb.bottomIndent = "-1mm";
pb.extendToNextParagraph = ExtendToNextParagraphType.Auto;
pb.extendToColumnBorder = true;
```

```

var pb2 = new ParagraphBackground( true, Color.Green, 82, 73,
    "-2mm", "-2mm", "-1mm", "-1mm",
ExtendToNextParagraphType.Auto, true);
ft.setParagraphBackground(pb, new TextRange(10, 11));

//remove the feature
ft.setParagraphBackground(pb, new TextRange(10, 11));

```

## Class: ParagraphFrame

The class is used in the classes *ParagraphStyleSheet* and *FormattedText*.

### Constructor:

```

new ParagraphFrame (bool active,
    ColorBrush brush, float shade, float opacity,
    UnitValueString leftIndent, UnitValueString rightIndent,
    UnitValueString topIndent, UnitValueString bottomIndent,
    UnitValueString lineWidth, String lineStyle,
    bool leftBorder, bool topBorder, bool rightBorder,
    bool bottomBorder, ExtendToNextParagraphType extendToNextType,
    bool extendToColumn)

```

### Properties:

|                              |                                  |                   |
|------------------------------|----------------------------------|-------------------|
| <b>active</b>                | <i>bool</i>                      | <i>Read/Write</i> |
| <b>brush</b>                 | <i>ColorBrush</i>                | <i>Read/Write</i> |
| <b>shade</b>                 | <i>float</i>                     | <i>Read/Write</i> |
| <b>opacity</b>               | <i>float</i>                     | <i>Read/Write</i> |
| <b>leftIndent</b>            | <i>UnitValueString</i>           | <i>Read/Write</i> |
| <b>rightIndent</b>           | <i>UnitValueString</i>           | <i>Read/Write</i> |
| <b>topIndent</b>             | <i>UnitValueString</i>           | <i>Read/Write</i> |
| <b>bottomIndent</b>          | <i>UnitValueString</i>           | <i>Read/Write</i> |
| <b>frameWidth</b>            | <i>UnitValueString</i>           | <i>Read/Write</i> |
| <b>lineStyle</b>             | <i>LineStyleString</i>           | <i>Read/Write</i> |
| <b>leftBorder</b>            | <i>bool</i>                      | <i>Read/Write</i> |
| <b>topBorder</b>             | <i>bool</i>                      | <i>Read/Write</i> |
| <b>rightBorder</b>           | <i>bool</i>                      | <i>Read/Write</i> |
| <b>bottomBorder</b>          | <i>bool</i>                      | <i>Read/Write</i> |
| <b>extendToNextParagraph</b> | <i>ExtendToNextParagraphType</i> | <i>Read/Write</i> |
| <b>extendToColumnBorder</b>  | <i>bool</i>                      | <i>Read/Write</i> |

### Examples:

```

var pf = new ParagraphFrame(true, Color.Blue, 34, 56,
    "0mm", "0mm", "0mm", "0mm", "1mm", "SolidLine",

```

```

    false, false, false, false, ExtendToNextParagraphType.No, false);
pf.brush = Color.Blue;
pf.shade = 34;
pf.opacity = 56;
pf.frameWidth = "1mm";
pf.lineStyle = "DotLine";
pf.topIndent = "1mm";
pf.leftIndent = "2mm";
pf.rightIndent = "3mm";
pf.bottomIndent = "2mm";
pf.leftBorder = true;
pf.topBorder = true;
pf.rightBorder = true;
pf.bottomBorder = true;
pf.extendToNextParagraph = ExtendToNextParagraphType.Auto;
pf.extendToColumnBorder = true;
ft.setParagraphFrame(pf, new TextRange(10, 20));

```

## Class: ParagraphRule

The class is used in the classes *ParagraphStyleSheet* and *FormattedText*.

### Constructor:

```

new ParagraphRule()
new ParagraphRule(bool active,
    ColorBrush brush, float shade, float opacity,
    UnitValueString lineWidth, String lineStyle,
    UnitValueString offset,
    UnitValueString leftIndent, UnitValueString rightIndent,
    ParagraphRuleLengthType lengthType)

```

### Properties:

|                    |                                |                   |
|--------------------|--------------------------------|-------------------|
| <b>active</b>      | <i>bool</i>                    | <i>Read/Write</i> |
| <b>brush</b>       | <i>ColorBrush</i>              | <i>Read/Write</i> |
| <b>shade</b>       | <i>float</i>                   | <i>Read/Write</i> |
| <b>opacity</b>     | <i>float</i>                   | <i>Read/Write</i> |
| <b>lineWidth</b>   | <i>UnitValueString</i>         | <i>Read/Write</i> |
| <b>lineStyle</b>   | <i>String</i>                  | <i>Read/Write</i> |
| <b>offset</b>      | <i>UnitValueString</i>         | <i>Read/Write</i> |
| <b>leftIndent</b>  | <i>UnitValueString</i>         | <i>Read/Write</i> |
| <b>rightIndent</b> | <i>UnitValueString</i>         | <i>Read/Write</i> |
| <b>lengthType</b>  | <i>ParagraphRuleLengthType</i> | <i>Read/Write</i> |

### Examples:

```

var pre = new ParagraphRule();
pre.active = true;
pre.brush = Color.Green;

```

```

pre.shade = 67;
pre.opacity = 78;
pre.lineWidth = "1mm";
pre.lineStyle = "DashLine";
pre.offset = "-12mm";
pre.leftIndent = "1mm";
pre.rightIndent = "7mm";
pre.lengthType = ParagraphRuleLengthType.Space;
ft.setParagraphRuleBelow(pre, new TextRange( 33, 35));

```

## Class: PDFExportSettings

Constructor:

```
new PDFExportSettings()
```

Properties:

|                             |                              |                   |
|-----------------------------|------------------------------|-------------------|
| <b>pagesRangeType</b>       | <i>PrintPagesRangeType</i>   | <i>Read/Write</i> |
| <b>fromPage</b>             | <i>int</i>                   | <i>Read/Write</i> |
| <b>toPage</b>               | <i>int</i>                   | <i>Read/Write</i> |
| <b>pageSelection</b>        | <i>String</i>                | <i>Read/Write</i> |
| <b>printSpreads</b>         | <i>bool</i>                  | <i>Read/Write</i> |
| <b>printEmptyPages</b>      | <i>bool</i>                  | <i>Read/Write</i> |
| <b>printSpellCorrection</b> | <i>bool</i>                  | <i>Read/Write</i> |
| <b>layersRange</b>          | <i>PrintLayersRange</i>      | <i>Read/Write</i> |
| <b>printSeparately</b>      | <i>bool</i>                  | <i>Read/Write</i> |
| <b>pageNumbering</b>        | <i>PrintPageNumbering</i>    | <i>Read/Write</i> |
| <b>numberOfDigits</b>       | <i>int</i>                   | <i>Read/Write</i> |
| <b>nameChangePolicy</b>     | <i>PrintNameChangePolicy</i> | <i>Read/Write</i> |
| <b>pagesPerJob</b>          | <i>int</i>                   | <i>Read/Write</i> |
| <b>registrationMarks</b>    | <i>bool</i>                  | <i>Read/Write</i> |
| <b>cropmarks</b>            | <i>bool</i>                  | <i>Read/Write</i> |
| <b>cropMarkLength</b>       | <i>UnitValueString</i>       | <i>Read/Write</i> |
| <b>cropMarkDistance</b>     | <i>UnitValueString</i>       | <i>Read/Write</i> |
| <b>colorSeparation</b>      | <i>PrintColorSeparation</i>  | <i>Read/Write</i> |
| <b>useBleed</b>             | <i>bool</i>                  | <i>Read/Write</i> |
| <b>bleedLeft</b>            | <i>UnitValueString</i>       | <i>Read/Write</i> |
| <b>bleedTop</b>             | <i>UnitValueString</i>       | <i>Read/Write</i> |
| <b>bleedRight</b>           | <i>UnitValueString</i>       | <i>Read/Write</i> |

|                                      |                                 |                   |
|--------------------------------------|---------------------------------|-------------------|
| <b>bleedBottom</b>                   | <i>UnitValueString</i>          | <i>Read/Write</i> |
| <b>horizontalAlignment</b>           | <i>PrintHorizontalAlignment</i> | <i>Read/Write</i> |
| <b>verticalAlignment</b>             | <i>PrintVerticalAlignment</i>   | <i>Read/Write</i> |
| <b>outputPath</b>                    | <i>String</i>                   | <b>Read/Write</b> |
| <b>fontEmbedding</b>                 | <i>PrintFontEmbedding</i>       | <i>Read/Write</i> |
| <b>ignoreTrapping</b>                | <i>bool</i>                     | <i>Read/Write</i> |
| <b>imageQuality</b>                  | <i>PrintImageQuality</i>        | <i>Read/Write</i> |
| <b>opi</b>                           | <i>PrintOPI</i>                 | <i>Read/Write</i> |
| <b>pdfCompatibility</b>              | <i>PDFCompatibility</i>         | <i>Read/Write</i> |
| <b>pdfOptimize</b>                   | <i>bool</i>                     | <i>Read/Write</i> |
| <b>pdfCreateLayers</b>               | <i>bool</i>                     | <i>Read/Write</i> |
| <b>pdfCreateBookmarks</b>            | <i>bool</i>                     | <i>Read/Write</i> |
| <b>pdfCompressText</b>               | <i>bool</i>                     | <i>Read/Write</i> |
| <b>pdfWarningScale</b>               | <i>bool</i>                     | <i>Read/Write</i> |
| <b>pdfWarningScaleBelow</b>          | <i>int</i>                      | <i>Read/Write</i> |
| <b>pdfWarningScaleAbove</b>          | <i>int</i>                      | <i>Read/Write</i> |
| <b>pdfWarningTextOverflow</b>        | <i>bool</i>                     | <i>Read/Write</i> |
| <b>pdfCompressionMethod</b>          | <i>PDFCompressionMethod</i>     | <i>Read/Write</i> |
| <b>pdfCompressionQuality</b>         | <i>int</i>                      | <i>Read/Write</i> |
| <b>pdfColorConversion</b>            | <i>PDFColorConversion</i>       | <i>Read/Write</i> |
| <b>pdfTargetProfile</b>              | <i>String</i>                   | <i>Read/Write</i> |
| <b>pdfxOutputIntent</b>              | <i>String</i>                   | <i>Read/Write</i> |
| <b>pdfxGrayProfile</b>               | <i>String</i>                   | <i>Read/Write</i> |
| <b>pdfxAllowEmbeddedICCPProfiles</b> | <i>bool</i>                     | <i>Read/Write</i> |

## Class: PreviewInstructions

The class holds the parameters for a preview image. It is used in the method `preview()` of the classes `Object` and `Page`.

### Constructor:

```
new PreviewInstructions()
new PreviewInstructions(float scale, HorizontalAlignment
horizontalAlignment,
    VerticalAlignment verticalAlignment, UnitValueString cropsX,
UnitValueString cropsY,
    UnitValueString cropsWidth, UnitValueString cropsHeight, bool
```

---

withFrame, ColorBrush brush)

Properties:

|                            |                            |                   |
|----------------------------|----------------------------|-------------------|
| <b>scale</b>               | <i>float</i>               | <i>Read/Write</i> |
| <b>horizontalAlignment</b> | <i>HorizontalAlignment</i> | <i>Read/Write</i> |
| <b>verticalAlignment</b>   | <i>VerticalAlignment</i>   | <i>Read/Write</i> |
| <b>cropsX</b>              | <i>UnitValueString</i>     | <i>Read/Write</i> |
| <b>cropsY</b>              | <i>UnitValueString</i>     | <i>Read/Write</i> |
| <b>cropsWidth</b>          | <i>UnitValueString</i>     | <i>Read/Write</i> |
| <b>cropsHeight</b>         | <i>UnitValueString</i>     | <i>Read/Write</i> |
| <b>withFrame</b>           | <i>bool</i>                | <i>Read/Write</i> |
| <b>brush</b>               | <i>ColorBrush</i>          | <i>Write</i>      |

## Class: Reflection

Constructor:

```
new Reflection()
```

Properties:

|                |                        |                   |                                  |
|----------------|------------------------|-------------------|----------------------------------|
| <b>active</b>  | <i>bool</i>            | <i>Read/Write</i> |                                  |
| <b>size</b>    | <i>float</i>           | <i>Read/Write</i> | <i>[%] &lt;0,100&gt;</i>         |
| <b>offset</b>  | <i>UnitValueString</i> | <i>Read/Write</i> | <i>[pt] &lt;-60000,60000&gt;</i> |
| <b>opacity</b> | <i>float</i>           | <i>Read/Write</i> | <i>[%] &lt;0,100&gt;</i>         |

## Class: Runaround

The class is used in the classes GraphicStyleSheet and Object. It determines the way text wraps around the object.

Constructor:

```
new Runaround()
```

```
new Runaround(RunaroundMode mode, RunaroundShapeType shapeType, UnitValueString distance)
```

```
new Runaround(RunaroundMode mode, RunaroundShapeType shapeType, UnitValueString left, UnitValueString top, UnitValueString right, UnitValueString bottom)
```

Properties:

|                  |                           |                   |                              |
|------------------|---------------------------|-------------------|------------------------------|
| <b>mode</b>      | <i>RunaroundMode</i>      | <i>Read/Write</i> |                              |
| <b>shapeType</b> | <i>RunaroundShapeType</i> | <i>Read/Write</i> |                              |
| <b>distance</b>  | <i>UnitValueString</i>    | <i>Read/Write</i> | <i>&lt;0, 20000&gt; [mm]</i> |

---

|               |                        |             |                              |
|---------------|------------------------|-------------|------------------------------|
| <b>left</b>   | <i>UnitValueString</i> | <i>Read</i> | <i>&lt;0, 20000&gt; [mm]</i> |
| <b>top</b>    | <i>UnitValueString</i> | <i>Read</i> | <i>&lt;0, 20000&gt; [mm]</i> |
| <b>right</b>  | <i>UnitValueString</i> | <i>Read</i> | <i>&lt;0, 20000&gt; [mm]</i> |
| <b>bottom</b> | <i>UnitValueString</i> | <i>Read</i> | <i>&lt;0, 20000&gt; [mm]</i> |

**Method:**

```
setBlock(UnitValueString left, UnitValueString top,
          UnitValueString right, UnitValueString bottom) void
```

**Examples:**

```
var ra1 = new Runaround( RunaroundMode.Right,
RunaroundShapeType.ObjectContours, "1mm");

var ra2 = new Runaround(RunaroundMode.Left, RunaroundShapeType.Block,
    "2mm", "12mm", "13mm", "14mm");
textObject1.runaround = ra1;
textObject2.runaround = ra2;
```

## Class: Shadow

The class is used in the class *GraphicObject* for all objects and the class *GraphicStyleSheet*. The class holds the parameters used for the “Drop Shadow” effect of the object.

**Constructor:**

```
new Shadow()
```

**Properties:**

|                            |                         |                   |                                  |
|----------------------------|-------------------------|-------------------|----------------------------------|
| <b>active</b>              | <i>bool</i>             | <i>Read/Write</i> |                                  |
| <b>color</b>               | <i>ColorBrush</i>       | <i>Read/Write</i> |                                  |
| <b>shade</b>               | <i>float</i>            | <i>Read/Write</i> | <i>[%] &lt;0-100&gt;</i>         |
| <b>opacity</b>             | <i>float</i>            | <i>Read/Write</i> | <i>[%] &lt;0-100&gt;</i>         |
| <b>mode</b>                | <i>ColorMode (enum)</i> | <i>Read/Write</i> |                                  |
| <b>angle</b>               | <i>float</i>            | <i>Read/Write</i> | <i>[degree] &lt;-360,360&gt;</i> |
| <b>offset</b>              | <i>UnitValueString</i>  | <i>Read/Write</i> | <i>[pt] &lt;-60000,60000&gt;</i> |
| <b>blur</b>                | <i>UnitValueString</i>  | <i>Read/Write</i> | <i>[mm] &lt;0,20&gt;</i>         |
| <b>scale</b>               | <i>float</i>            | <i>Read/Write</i> | <i>[%] &lt;0,20000&gt;</i>       |
| <b>coverShadow</b>         | <i>bool</i>             | <i>Read/Write</i> |                                  |
| <b>applyElementOpacity</b> |                         | <i>bool</i>       | <i>Read/Write</i>                |

**Examples:**

```
var sh = new Shadow();
sh.active = true;
sh.color = Color.Red;
sh.shade = 90;
sh.opacity = 88;
```

```
sh.mode = ColorMode.HardLight;
sh.angle = 166;
sh.offset = "55mm";
sh.blur = "1pt";
sh.scale = 100;
sh.coverShadow = false;
sh.applyElementOpacity = false;

if (textObject) {
    textObject.shadow = sh;
}
```

## Class: SpellLang

### Constructor:

*new SpellLang (Language type)*

## Class: StrikethroughOptions

The object of this class holds the parameters of a strikethrough line.

The class is used in classes FormattedText, CharacterStyleSheet and ParagraphStyleSheet.

### Constructor:

An object can be created using one of the following constructors:

```
new StrikethroughOptions (FontStyleLineType type,
    ColorBrush brush, float shade, float opacity,
    UnitValueString lineWidth, String lineStyle,
    UnitValueString offset, FontStyleLineRangeType type)
```

or

```
new StrikethroughOptions (FontStyleLineType type)
```

### Properties:

|                                                                                 |                          |                   |
|---------------------------------------------------------------------------------|--------------------------|-------------------|
| <b>type</b>                                                                     | <i>FontStyleLineType</i> | <i>Read/Write</i> |
| <b>brush</b>                                                                    | <i>ColorBrush</i>        | <i>Read/Write</i> |
| Get/Set the text color.                                                         |                          |                   |
| <b>shade</b>                                                                    | <i>float</i>             | <i>Read/Write</i> |
| Get/Set the shade of the text color.                                            |                          |                   |
| <b>opacity</b>                                                                  | <i>float</i>             | <i>Read/Write</i> |
| Get/Set the opacity of the text color.                                          |                          |                   |
| <b>lineWidth</b>                                                                | <i>UnitValueString</i>   | <i>Read/Write</i> |
| Get/Set the line width of the line.                                             |                          |                   |
| <b>lineStyle</b>                                                                | <i>String</i>            | <i>Read/Write</i> |
| Get/Set the line style of the line. See the enumeration type <i>LineStyle</i> . |                          |                   |
| <b>offset</b>                                                                   | <i>UnitValueString</i>   | <i>Read/Write</i> |

Get/Set the distance under the text where the line is applied.

|                  |                               |                   |
|------------------|-------------------------------|-------------------|
| <b>rangeType</b> | <i>FontStyleLineRangeType</i> | <i>Read/Write</i> |
|------------------|-------------------------------|-------------------|

Get/Set the line range. See the enumeration type *FontStyleLineRangeType*.

## Class: TablesAbbreviation

The class is used in classes *FormattedText*, *CharacterStyleSheet* and *ParagraphStyleSheet*.

### Constructor:

An object can be created using the following constructor:

```
new TablesAbbreviation(bool active, String alternativeText)
```

### Properties:

|                        |               |                   |
|------------------------|---------------|-------------------|
| <b>active</b>          | <i>bool</i>   | <i>Read/Write</i> |
| <b>alternativeText</b> | <i>String</i> | <i>Read/Write</i> |

## Class: TablesBibliography

The class is used in classes *FormattedText*, *CharacterStyleSheet* and *ParagraphStyleSheet*.

### Constructor:

An object can be created using the constructor: *new TablesBibliography()* and the document parameters are set using properties. Alternatively you can use the following constructor with parameters:

```
new TablesBibliography(bool active, String alternativeText)
```

### Properties:

|                        |               |                   |
|------------------------|---------------|-------------------|
| <b>active</b>          | <i>bool</i>   | <i>Read/Write</i> |
| <b>alternativeText</b> | <i>String</i> | <i>Read/Write</i> |

## Class: TablesContents

The class is used in classes *FormattedText*, *CharacterStyleSheet* and *ParagraphStyleSheet*.

### Constructor:

An object can be created using the constructor: *new TablesContents()* and the document parameters are set using properties. Alternatively you can use the following constructor with parameters:

```
new TablesContents(bool active, String alternativeText, int level)
```

### Properties:

|                        |               |                   |
|------------------------|---------------|-------------------|
| <b>active</b>          | <i>bool</i>   | <i>Read/Write</i> |
| <b>alternativeText</b> | <i>String</i> | <i>Read/Write</i> |

|              |            |                   |
|--------------|------------|-------------------|
| <b>level</b> | <i>int</i> | <i>Read/Write</i> |
|--------------|------------|-------------------|

## Class: TablesIndex

The class is used in classes *FormattedText*, *CharacterStyleSheet* and *ParagraphStyleSheet*.

### Constructor:

An object can be created using the constructor: *new TablesIndex()* and the document parameters are set using properties. Alternatively you can use the following constructor with parameters:

```
new TablesIndex( bool active, String alternativeText, bool appendPageNumber)
```

### Properties:

|                         |               |                   |
|-------------------------|---------------|-------------------|
| <b>active</b>           | <i>bool</i>   | <i>Read/Write</i> |
| <b>alternativeText</b>  | <i>String</i> | <i>Read/Write</i> |
| <b>appendPageNumber</b> | <i>bool</i>   | <i>Read/Write</i> |

## Class: TablesPicture

The class is used in classes *FormattedText*, *CharacterStyleSheet* and *ParagraphStyleSheet*.

### Constructor:

An object can be created using the constructor: *new TablesPicture()* and the document parameters are set using properties. Alternatively you can use the following constructor with parameters:

```
new TablesPicture(bool active, String alternativeText)
```

### Properties:

|                        |               |                   |
|------------------------|---------------|-------------------|
| <b>active</b>          | <i>bool</i>   | <i>Read/Write</i> |
| <b>alternativeText</b> | <i>String</i> | <i>Read/Write</i> |

## Class: TablesRunningTitle

The class is used in classes *FormattedText*, *CharacterStyleSheet* and *ParagraphStyleSheet*.

### Constructor:

An object can be created using the constructor: *new TablesRunningTitle()* and the document parameters are set using properties. Alternatively you can use the following constructor with parameters::

```
new TablesRunningTitle(bool active, String alternativeText, int level)
```

### Properties:

|               |             |                   |
|---------------|-------------|-------------------|
| <b>active</b> | <i>bool</i> | <i>Read/Write</i> |
|---------------|-------------|-------------------|

|                        |               |                   |
|------------------------|---------------|-------------------|
| <b>alternativeText</b> | <i>String</i> | <i>Read/Write</i> |
| <b>level</b>           | <i>int</i>    | <i>Read/Write</i> |

## Class: Tabulator

The class is used in the class *Tabulators*.

### Constructor:

```
new Tabulator ()
new Tabulator (String position, TabulatorAlignment alignment, String
fillCharacter)
```

### Properties:

|                       |                           |                   |
|-----------------------|---------------------------|-------------------|
| <b>position</b>       | <i>UnitValueString</i>    | <i>Read/Write</i> |
| <b>alignment</b>      | <i>TabulatorAlignment</i> | <i>Read/Write</i> |
| <b>alignCharacter</b> | <i>Char</i>               | <i>Read/Write</i> |
| <b>fillCharacter</b>  | <i>String</i>             | <i>Read/Write</i> |

## Class: Tabulators

The class is used in the classes *ParagraphStyleSheet* and *FormattedText*

### Constructor:

```
new Tabulators (TabsType type)
new Tabulators (TabsType type, bool standardTabs, bool indent)
```

### Properties:

|                          |                 |                   |
|--------------------------|-----------------|-------------------|
| <b>type</b>              | <i>TabsType</i> | <i>Read/Write</i> |
| <b>active</b>            | <i>bool</i>     | <i>Read</i>       |
| <b>count</b>             | <i>int</i>      | <i>Read</i>       |
| <b>applyStandardTabs</b> | <i>bool</i>     | <i>Read/Write</i> |
| <b>applyAsIndent</b>     | <i>bool</i>     | <i>Read/Write</i> |

### Methods:

|                                                               |                  |
|---------------------------------------------------------------|------------------|
| <b>getTabulator</b> ( <i>int index</i> )                      | <i>Tabulator</i> |
| <b>addTabulator</b> ( <i>Tabulator tabulator</i> )            | <i>void</i>      |
| <b>removeTabulator</b> ( <i>int index</i> )                   | <i>void</i>      |
| <b>removeAllTabulators</b> ()                                 | <i>void</i>      |
| <b>setTabulator</b> ( <i>int index, Tabulator tabulator</i> ) | <i>void</i>      |

## Class: TaggedPdfOptions

The class is used in the class Object. It defines how the object is tagged when a tagged PDF is exported.

### Constructor:

```
new TaggedPdfOptions ()
new TaggedPdfOptions (TaggedPdfType type)
new TaggedPdfOptions (TaggedPdfType type, String alternativeText)
```

### Properties:

|                        |                      |                   |
|------------------------|----------------------|-------------------|
| <b>tagType</b>         | <i>TaggedPdfType</i> | <i>Read/Write</i> |
| <b>alternativeText</b> | <i>String</i>        | <i>Read/Write</i> |

## Class: TextEndnotes

The class is used in class TextPreferences and FormattedText.

### Constructor:

An object can be created using a constructor:

```
new TextEndnotes ()
```

and the document parameters are set using the following Properties:

### Properties:

|                                  |                                     |                   |
|----------------------------------|-------------------------------------|-------------------|
| <b>displayPosition</b>           | <i>EndnotesBreakType</i>            | <i>Read/Write</i> |
| <b>saveTo</b>                    | <i>EndnotesPositionType</i>         | <i>Read/Write</i> |
| <b>textOffset</b>                | <i>UnitValueString</i>              | <i>Read/Write</i> |
| <b>prefix</b>                    | <i>String</i>                       | <i>Read/Write</i> |
| <b>postfix</b>                   | <i>String</i>                       | <i>Read/Write</i> |
| <b>numberFormat</b>              | <i>EnumerationsNumberFormat</i>     | <i>Read/Write</i> |
| <b>numberingType</b>             | <i>ReferenceAnchorNumberingType</i> | <i>Read/Write</i> |
| <b>startNumber</b>               | <i>int</i>                          | <i>Read/Write</i> |
| <b>separatorVisible</b>          | <i>bool</i>                         | <i>Read/Write</i> |
| <b>separatorVerticalOffset</b>   | <i>UnitValueString</i>              | <i>Read/Write</i> |
| <b>separatorHorizontalOffset</b> | <i>UnitValueString</i>              | <i>Read/Write</i> |
| <b>separatorLineWidth</b>        | <i>UnitValueString</i>              | <i>Read/Write</i> |
| <b>separatorLineBrush</b>        | <i>ColorBrush</i>                   | <i>Read/Write</i> |
| <b>separatorLineStyle</b>        | <i>String</i>                       | <i>Read/Write</i> |
| <b>separatorLineLength</b>       | <i>UnitValueString</i>              | <i>Read/Write</i> |

## Class: TextFootnotes

The class is used in the class TextPreferences and FormattedText.

### Constructor:

An object can be created using constructor:

```
new TextFootnotes ()
```

and the document parameters are set using following Properties:

### Properties:

|                                  |                                     |                   |
|----------------------------------|-------------------------------------|-------------------|
| <b>textOffset</b>                | <i>UnitValueString</i>              | <i>Read/Write</i> |
| <b>prefix</b>                    | <i>String</i>                       | <i>Read/Write</i> |
| <b>postfix</b>                   | <i>String</i>                       | <i>Read/Write</i> |
| <b>numberFormat</b>              | <i>EnumerationsNumberFormat</i>     | <i>Read/Write</i> |
| <b>numberingType</b>             | <i>ReferenceAnchorNumberingType</i> | <i>Read/Write</i> |
| <b>startNumber</b>               | <i>int</i>                          | <i>Read/Write</i> |
| <b>separatorVisible</b>          | <i>bool</i>                         | <i>Read/Write</i> |
| <b>separatorVerticalOffset</b>   | <i>UnitValueString</i>              | <i>Read/Write</i> |
| <b>separatorHorizontalOffset</b> | <i>UnitValueString</i>              | <i>Read/Write</i> |
| <b>separatorLineWidth</b>        | <i>UnitValueString</i>              | <i>Read/Write</i> |
| <b>separatorLineBrush</b>        | <i>ColorBrush</i>                   | <i>Read/Write</i> |
| <b>separatorLineStyle</b>        | <i>String</i>                       | <i>Read/Write</i> |
| <b>separatorLineLength</b>       | <i>UnitValueString</i>              | <i>Read/Write</i> |

## Class: TransparencyStop

### Constructor:

```
new TransparencyStop()
```

```
new TransparencyStop(float opacity, float position, float focusPosition)
```

### Properties:

|                      |              |                   |            |                      |
|----------------------|--------------|-------------------|------------|----------------------|
| <b>opacity</b>       | <i>float</i> | <i>Read/Write</i> | <i>[%]</i> | <i>&lt;0,100&gt;</i> |
| <b>position</b>      | <i>float</i> | <i>Read/Write</i> | <i>[%]</i> | <i>&lt;0,100&gt;</i> |
| <b>focusPosition</b> | <i>float</i> | <i>Read/Write</i> | <i>[%]</i> | <i>&lt;0,100&gt;</i> |

## Class: Transparency

### Constructor:

```
new Transparency()
```

## Properties:

|                  |                                |                                                       |
|------------------|--------------------------------|-------------------------------------------------------|
| <b>active</b>    | <i>bool</i>                    | <i>Read/Write</i>                                     |
| <b>type</b>      | <i>TransparencyType (enum)</i> | <i>Read/Write</i>                                     |
| <b>angle</b>     | <i>float</i>                   | <i>Read/Write [degree]</i><br><i>&lt;-360,360&gt;</i> |
| <b>stopCount</b> | <i>int</i>                     | <i>Read</i>                                           |

## Methods:

|                     |                                           |                         |
|---------------------|-------------------------------------------|-------------------------|
| <b>stopAt</b>       | <i>(int index)</i>                        | <i>TransparencyStop</i> |
| <b>setStopAt</b>    | <i>(int index, TransparencyStop stop)</i> | <i>void</i>             |
| <b>removeStopAt</b> | <i>(int index)</i>                        | <i>void</i>             |
| <b>appendStop</b>   | <i>(TransparencyStop stop)</i>            | <i>void</i>             |

## Class: UnderlineOptions

The object of this class holds the parameters of the text underline.

The class is used in the classes FormattedText, CharacterStyleSheet and ParagraphStyleSheet.

## Constructor:

An object can be created using the constructor:

```
new UnderlineOptions ()
```

or

```
new UnderlineOptions (FontStyleLineType type)
```

and the parameters are set using properties.

## Properties:

|                                                                         |                          |                   |
|-------------------------------------------------------------------------|--------------------------|-------------------|
| <b>type</b>                                                             | <i>FontStyleLineType</i> | <i>Read/Write</i> |
| <b>brush</b>                                                            | <i>ColorBrush</i>        | <i>Read/Write</i> |
| Get/Set the text color.                                                 |                          |                   |
| <b>shade</b>                                                            | <i>float</i>             | <i>Read/Write</i> |
| Get/Set the shade of the text color.                                    |                          |                   |
| <b>opacity</b>                                                          | <i>float</i>             | <i>Read/Write</i> |
| Get/Set the opacity of the text color.                                  |                          |                   |
| <b>lineWidth</b>                                                        | <i>UnitValueString</i>   | <i>Read/Write</i> |
| Get/Set the line width of the line.                                     |                          |                   |
| <b>lineStyle</b>                                                        | <i>String</i>            | <i>Read/Write</i> |
| Get/Set the line style (line type). See the enumeration type LineStyle. |                          |                   |
| <b>offset</b>                                                           | <i>UnitValueString</i>   | <i>Read/Write</i> |

Get/Set the distance under the text where the line is applied.

| <b>rangeType</b>                                                                 | <i>FontStyleLineRangeType</i> | <i>Read/Write</i> |
|----------------------------------------------------------------------------------|-------------------------------|-------------------|
| Get/Set the line range. See the enumeration type <i>FontStyleLineRangeType</i> . |                               |                   |

#### Examples:

```
var opt = new UnderlineOptions();
opt.brush = Color.Red;
opt.shade = 92;
opt.opacity = 99;
opt.lineWidth = "2pt";
opt.lineStyle = "DashLine";
opt.offset = "0mm";
opt.rangeType = FontStyleLineRangeType.Words;
ft.setManualUnderline(opt);
ft.setAutomaticUnderline(FontStyleLineRangeType.Words);
```

## Class: Variable

The object of the class *Variable* represents a text variable in a *FormattedText* object, for example a page number, the number of pages or a running title. A *Variable* object is read from a text with the method *variable(pos)* of the class *FormattedText* and inserted with the method *insertVariable(variable, pos)*.

#### Properties:

| <b>type</b>                                                                                                                                                                                                                                                                                                                | <i>String</i> | <i>Read</i> |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|-------------|
| Returns the type key of the variable, for example "Viva Technology/Page Number", "Viva Technology/Number of Pages", "Viva Technology/Previous Page Number", "Viva Technology/Next Page Number", "Viva Technology/Running Title", "Viva Technology/Document Creation Date" or "Viva Technology/Document Modification Date". |               |             |

#### Methods:

| <b>text(<i>SpellLang language, TextObject textObject</i>)</b>                                | <i>String</i> |
|----------------------------------------------------------------------------------------------|---------------|
| Returns the text that the variable shows in the given text object, using the given language. |               |

#### Examples:

```
// copy the page number variable at position 0 to the end of the text
var ft = textObject.content.formattedText;
var v = ft.variable(0);
if (v.type == "Viva Technology/Page Number")
    ft.insertVariable(v, ft.length);
```

## Class: WidowsOrphans

The class is used in the class *FormattedText* and *ParagraphStyleSheet*.

#### Constructor:

```
new WidowsOrphans(WidowsOrphansMode mode)
new WidowsOrphans()
```

## Properties:

|                   |                          |                   |
|-------------------|--------------------------|-------------------|
| <i>mode</i>       | <i>WidowsOrphansMode</i> | <i>Read/Write</i> |
| <i>firstLines</i> | <i>int</i>               | <i>Read/Write</i> |
| <i>lastLines</i>  | <i>int</i>               | <i>Read/Write</i> |

---



The background features a dynamic, abstract composition of flowing, organic shapes in shades of gold, yellow, and blue against a dark, almost black, backdrop. The shapes appear to be layered and moving, creating a sense of depth and motion. A bright, glowing blue and yellow light source is visible on the right side, casting a soft glow across the scene.

# User Interface Objects

# User Interface Objects

## Class: MessageBox

### Methods:

|                                                            |             |
|------------------------------------------------------------|-------------|
| <b>information</b> ( <i>String title, String message</i> ) | <i>void</i> |
| <b>warning</b> ( <i>String title, String message</i> )     | <i>void</i> |

## Class: Directory

The class represents a directory or a folder.

### Constructor:

An object can be created using a constructor:

```
new Directory(String path)  
new Directory()
```

Without a path the object refers to the current directory.

### Properties:

|                                           |               |                   |
|-------------------------------------------|---------------|-------------------|
| <b>fullPath</b>                           | <i>String</i> | <i>Read</i>       |
| Get full path string.                     |               |                   |
| <b>name</b>                               | <i>String</i> | <i>Read</i>       |
| Get relative path name for the directory. |               |                   |
| <b>path</b>                               | <i>String</i> | <i>Read/Write</i> |
| <b>count</b>                              | <i>int</i>    | <i>Read</i>       |
| <b>isReadable</b>                         | <i>bool</i>   | <i>Read</i>       |
| <b>exists</b>                             | <i>bool</i>   | <i>Read</i>       |
| <b>isRoot</b>                             | <i>bool</i>   | <i>Read</i>       |
| <b>isRelative</b>                         | <i>bool</i>   | <i>Read</i>       |
| <b>isAbsolute</b>                         | <i>bool</i>   | <i>Read</i>       |

### Methods:

|                                                                                                                      |             |
|----------------------------------------------------------------------------------------------------------------------|-------------|
| <b>makeDir</b> ( <i>String dir</i> )                                                                                 | <i>bool</i> |
| Creates the subdirectory <i>dir</i> in the current directory. Returns TRUE if the directory is created successfully. |             |
| <b>removeDir</b> ( <i>String dir</i> )                                                                               | <i>bool</i> |
| Removes the subdirectory <i>dir</i> in the current directory. Returns TRUE if the directory is removed successfully. |             |
| <b>changeDir</b> ( <i>String dir</i> )                                                                               | <i>bool</i> |

Changes the current directory to the directory `dir`. Returns `TRUE` if the directory is changed successfully.

The methods `mkdir()`, `rmdir()` and `chdir()` are equivalent to `makeDir()`, `removeDir()` and `changeDir()`.

---

|                                                            |             |
|------------------------------------------------------------|-------------|
| <b>renameDir</b> ( <i>String oldName, String newName</i> ) | <i>bool</i> |
|------------------------------------------------------------|-------------|

Renames the subdirectory `oldName` in the current directory to the new name `newName`. Returns `TRUE` if the directory is renamed successfully.

---

|                                                                                                                                                         |                 |
|---------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| <b>getFileList</b> ( <i>String nameFilters, DirectoryFilter filters=DirectoryFilter::NoFilter, DirectorySortFlag sort = DirectorySortFlag::NoSort</i> ) | <i>String[]</i> |
|---------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|

Returns list of file names from the directory.

---

|                                                                                                                                       |  |
|---------------------------------------------------------------------------------------------------------------------------------------|--|
| <b>getFileList</b> ( <i>DirectoryFilter filters = DirectoryFilter::NoFilter, DirectorySortFlag sort = DirectorySortFlag::NoSort</i> ) |  |
|---------------------------------------------------------------------------------------------------------------------------------------|--|

Returns list of file names from the directory.

---

|                                              |             |
|----------------------------------------------|-------------|
| <b>zip</b> ( <i>String filePath = null</i> ) | <i>bool</i> |
|----------------------------------------------|-------------|

Returns `TRUE` if the zip operation on the directory was processed successfully, otherwise returns `FALSE`.

*filePath* - a full file name of a result zip file. If it is empty, the name of a source directory is used with 'zip' extension.

## Class: File

The class represents a general file.

### Constructor:

An object can be created using a constructor:

```
new File(String fullPath)
```

### Properties:

---

|                 |               |             |
|-----------------|---------------|-------------|
| <b>fileName</b> | <i>String</i> | <i>Read</i> |
|-----------------|---------------|-------------|

Returns the full file path.

---

|               |             |             |
|---------------|-------------|-------------|
| <b>exists</b> | <i>bool</i> | <i>Read</i> |
|---------------|-------------|-------------|

Returns the information if the file exists or not.

---

|             |            |             |
|-------------|------------|-------------|
| <b>size</b> | <i>int</i> | <i>Read</i> |
|-------------|------------|-------------|

Returns the size of file in bytes.

### Methods:

---

|                                       |             |
|---------------------------------------|-------------|
| <b>copy</b> ( <i>String newName</i> ) | <i>bool</i> |
|---------------------------------------|-------------|

Copies the current file to the new position `newName`.

*newName*: full path file name

---

|                                         |             |
|-----------------------------------------|-------------|
| <b>rename</b> ( <i>String newName</i> ) | <i>bool</i> |
|-----------------------------------------|-------------|

Renames the current file to the new name.

*newName*: full path file name

---

|                 |             |
|-----------------|-------------|
| <b>remove()</b> | <i>bool</i> |
|-----------------|-------------|

Removes the file.

---

|                                    |             |
|------------------------------------|-------------|
| <b>zip(String filePath = null)</b> | <i>bool</i> |
|------------------------------------|-------------|

Returns *TRUE* if the zip operation on the file was processed succesfully, otherwise returns *FALSE*.

*filePath* - a full file name of a result zip file. If it is empty, the name of a source file is used with 'zip' extension.

---

|                                           |             |
|-------------------------------------------|-------------|
| <b>unzip(String directoryPath = null)</b> | <i>bool</i> |
|-------------------------------------------|-------------|

Returns *TRUE* if the zip operation on the file was processed succesfully, otherwise returns *FALSE*.

*directoryPath* - a full result directory name. If it is empty, the same directory as a source file has is used.

---

## Class: TextFile

The class represents a file as a text file. The class *TextFile* extends the *File* class. The data is read line by line.

### Constructor:

An object can be created using a constructor:

```
new TextFile(String fullFilePath)
```

### Properties:

---

|            |             |             |
|------------|-------------|-------------|
| <b>eof</b> | <i>bool</i> | <b>Read</b> |
|------------|-------------|-------------|

Checks the reading file is on the end of the file.

---

### Methods:

---

|                                    |             |
|------------------------------------|-------------|
| <b>open(OpenFileMode fileMode)</b> | <i>bool</i> |
|------------------------------------|-------------|

Opens the file in a defined mode. If the file doesn't exist, the new file is created. Returns true if the file is opened successfully.

---

|                |             |
|----------------|-------------|
| <b>close()</b> | <i>void</i> |
|----------------|-------------|

Close the open file.

---

|                   |               |
|-------------------|---------------|
| <b>readLine()</b> | <i>String</i> |
|-------------------|---------------|

Read one line of the text from the open text file at a current position. Returns *null* if no following line exists in the file.

---

|                              |             |
|------------------------------|-------------|
| <b>writeLine(String str)</b> | <i>void</i> |
|------------------------------|-------------|

Write one line of the text to the open text file at a current position. The new line character is added at the end of the line.

---

|                |             |
|----------------|-------------|
| <b>flush()</b> | <i>bool</i> |
|----------------|-------------|

Writes buffered data to the file. Returns true if successful.

---

### Examples:

```
// correct the directory
```

---

```
var d = new Directory(appScriptsPath);
d.mkdir("mydir");
d.chdir("mydir");

// open or create text file
var file = new TextFile(d.fullPath + "/TestFileName.txt");
messageBox.information("MSG", "The file name is " + file.fileName + ".
\nDoes the file exist? " + file.exists);
if (file.open(OpenFileMode.ReadWrite))
{
    // write new texts to the file
    file.writeLine("Test row in the text file 1!");
    file.writeLine("Test row in the text file 2!");
    file.writeLine("Test row in the text file 3!");
    file.writeLine("Test row in the text file 4!");
    // close file
    file.close();
}
// open or create the file
var rText = "";
if (file.open(OpenFileMode.ReadWrite))
{
    // read all text from the file to one variable
    while (!file.eof)
    {
        rText = rText + file.readLine();
    }
}
file.close();

// create new file and write the one row text in this file
var nfile = new TextFile(d.fullPath + "/TestNewFileName.txt");
if (nfile.open(OpenFileMode.ReadWrite))
{
    nfile.writeLine(rText);
}
nfile.close();

// create new directory
var oldFullPath = d.fullPath;
d.chdir("..");
d.mkdir("yourdir");
d.chdir("yourdir");

// copy new file name and remove source file
nfile.copy(d.fullPath + "/" + nfile.fileName.split('/').pop());
nfile.remove();

// rename old file
file.rename(file.fileName + ".old");
```



## Class: FileDialog

The class works with *FileDialog* for selecting the directory or the files.

### Constructor:

```
new FileDialog()
```

### Properties:

|                             |                             |                   |
|-----------------------------|-----------------------------|-------------------|
| <b>caption</b>              | <i>String</i>               | <i>Read/Write</i> |
| Title of the dialog window. |                             |                   |
| <b>acceptMode</b>           | <i>FileDialogAcceptMode</i> | <i>Read/Write</i> |
| <b>fileMode</b>             | <i>FileDialogFileMode</i>   | <i>Read/Write</i> |
| <b>viewMode</b>             | <i>FileDialogViewMode</i>   | <i>Read/Write</i> |
| <b>lookInLabel</b>          | <i>String</i>               | <i>Read/Write</i> |
| <b>fileNameLabel</b>        | <i>String</i>               | <i>Read/Write</i> |
| <b>fileTypeLabel</b>        | <i>String</i>               | <i>Read/Write</i> |
| <b>acceptLabel</b>          | <i>String</i>               | <i>Read/Write</i> |
| <b>rejectLabel</b>          | <i>String</i>               | <i>Read/Write</i> |
| <b>directory</b>            | <i>String</i>               | <i>Read/Write</i> |
| <b>directoryFilter</b>      | <i>int</i>                  | <i>Read/Write</i> |
| <b>nameFilter</b>           | <i>String</i>               | <i>Read/Write</i> |
| <b>mimeTypeFilter</b>       | <i>String</i>               | <i>Read/Write</i> |
| <b>selectedFiles</b>        | <i>StringList</i>           | <i>Read</i>       |

### Methods:

|                                                                                                                                                                                                           |               |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|
| <b>open()</b>                                                                                                                                                                                             | <i>bool</i>   |
| Opens the file dialog depending on the paramters that are set to the objects before opening. Returns the value true if the slection is accepted. Or it returns the value false if the dialog is canceled. |               |
| <b>addOption</b> ( <i>FileDialogOption option</i> )                                                                                                                                                       | <i>void</i>   |
| <b>removeOption</b> ( <i>FileDialogOption option</i> )                                                                                                                                                    | <i>void</i>   |
| <b>testOption</b> ( <i>FileDialogOption option</i> )                                                                                                                                                      | <i>bool</i>   |
| <b>testDirectoryFilter</b> ( <i>DirectoryFilter filter</i> )                                                                                                                                              | <i>bool</i>   |
| <b>getExistingDirectory</b> ( <i>String caption, String dir, FileDialogOption options</i> )                                                                                                               | <i>String</i> |
| <b>getExistingDirectory</b> ()                                                                                                                                                                            | <i>String</i> |
| <b>getOpenFileName</b> ( <i>String caption, String dir, String filter, FileDialogOption opts</i> )                                                                                                        | <i>String</i> |
| <b>getOpenFileName</b> ()                                                                                                                                                                                 | <i>String</i> |
| <i>filter</i> : The filters must be divided by double semicolon char (*.jpg;*.png)                                                                                                                        |               |

---

```
getSaveFileName(String caption, String dir, String filter,
FileDialogOption opts)      String
```

---

```
getSaveFileName()      String
```

*filter*: The filters must be divided by double semicolon char (\*.jpg;\*.png)

## Class: ProgressReporter

The class represents the window with the progress bar and with the information about the progress some operation.

### Constructor:

```
new ProgressReporter(String windowTitle, ProgressReporterType type,
    int startPosition, int endPosition)
```

### Properties:

|                            |               |                   |
|----------------------------|---------------|-------------------|
| <b>progressBarText</b>     | <i>String</i> | <i>Write</i>      |
| <b>progressBarSubText</b>  | <i>String</i> | <i>Write</i>      |
| <b>progressBarPosition</b> | <i>int</i>    | <i>Read/Write</i> |
| <b>progressBarLength</b>   | <i>int</i>    | <i>Read</i>       |

### Methods:

|                                             |             |
|---------------------------------------------|-------------|
| <b>close()</b>                              | <i>void</i> |
| <b>progressBarReset(int begin, int end)</b> | <i>void</i> |
| <b>progressBarFinish()</b>                  | <i>void</i> |

The method finishes the progress bar. It set the property `progressBarPosition` to value less then the begin value of the progress bar.

|                      |             |
|----------------------|-------------|
| <b>forceRedraw()</b> | <i>void</i> |
|----------------------|-------------|

Processes pending drawing events so that the progress window is updated during a long operation.

## Class: Dialog

The class represents a user dialog.

### Constructor:

```
new Dialog()
```

### Properties:

|                     |               |                   |
|---------------------|---------------|-------------------|
| <b>caption</b>      | <i>String</i> | <i>Read/Write</i> |
| <b>minimumWidth</b> | <i>int</i>    | <i>Read/Write</i> |
| <b>maximumWidth</b> | <i>int</i>    | <i>Read/Write</i> |
| <b>resizable</b>    | <i>bool</i>   | <i>Read/Write</i> |

Defines if the user can change the size of the dialog.

## Methods:

|                                                                                                                                              |                 |
|----------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| <b>addLabel</b> ( <i>String label</i> )                                                                                                      | <i>Label</i>    |
| <b>addLabel</b> ( <i>int id, String label</i> )                                                                                              | <i>Label</i>    |
| <b>addComboBox</b> ( <i>String label</i> )                                                                                                   | <i>ComboBox</i> |
| <b>addComboBox</b> ( <i>int id, String label</i> )                                                                                           | <i>ComboBox</i> |
| <b>addComboBox</b> ( <i>int id, int parentId, String label, HorizontalAlignment alignment</i> )                                              | <i>ComboBox</i> |
| <b>addCheckBox</b> ( <i>String label, bool defaultValue</i> )                                                                                | <i>CheckBox</i> |
| <b>addCheckBox</b> ( <i>int id, String label, bool defaultValue</i> )                                                                        | <i>CheckBox</i> |
| <b>addLineEdit</b> ( <i>String label, String defaultValue, LineEditType type</i> )                                                           | <i>LineEdit</i> |
| <b>addLineEdit</b> ( <i>String label, String defaultValue, LineEditType type, int width</i> )                                                | <i>LineEdit</i> |
| <b>addLineEdit</b> ( <i>int id, String label, String defValue, LineEditType type, int width</i> )                                            | <i>LineEdit</i> |
| <b>addTextEdit</b> ( <i>String label, String defaultValue</i> )                                                                              | <i>TextEdit</i> |
| <b>addTextEdit</b> ( <i>int id, String label, String defaultValue</i> )                                                                      | <i>TextEdit</i> |
| <b>addTextEdit</b> ( <i>String label, String defaultValue, int width, int height</i> )                                                       | <i>TextEdit</i> |
| <b>addTextEdit</b> ( <i>int id, String label, String defaultValue, int width, int height</i> )                                               | <i>TextEdit</i> |
| <b>addGroupBox</b> ( <i>String label</i> )                                                                                                   | <i>GroupBox</i> |
| <b>addGroupBox</b> ( <i>int id, String label</i> )                                                                                           | <i>GroupBox</i> |
| The method adds a box for radio buttons.                                                                                                     |                 |
| <b>addButton</b> ( <i>int id, String label, String clickedMethodName</i> )                                                                   | <i>Button</i>   |
| <b>addAcceptButton</b> ( <i>int id, String label</i> )                                                                                       | <i>Button</i>   |
| <b>addRejectButton</b> ( <i>int id, String label</i> )                                                                                       | <i>Button</i>   |
| <b>addButtons</b> ()                                                                                                                         | <i>void</i>     |
| <b>addButtons</b> ( <i>String acceptLabel, String rejectLabel</i> )                                                                          | <i>void</i>     |
| <b>addButtons</b> ( <i>String acceptLabel, String rejectLabel, bool center</i> )                                                             | <i>void</i>     |
| The method adds an accept button and a cancel button in the dialog.                                                                          |                 |
| <b>exec</b> ()                                                                                                                               | <i>int</i>      |
| The method shows a prepared dialog. It returns the code 1=OK button, 0=CANCEL button or the changeEventCode from the combo box if it is set. |                 |
| <b>accept</b> ()                                                                                                                             | <i>void</i>     |
| <b>done</b> ( <i>int resultCode</i> )                                                                                                        | <i>void</i>     |

**reject()***void*

The methods close the dialog from a callback method: **accept()** with the result 1, **reject()** with the result 0, **done()** with the given result code. The result is returned by **exec()**.

**Example:**

```
var d = new Dialog();
d.caption = "New Dialog";

var line1 = d.addLineEdit("MyDouble: ", 10.3, LineEditType.Double);
var line2 = d.addLineEdit("MyString: ", "default line text",
LineEditType.String);
var line3 = d.addLineEdit("MyInteger: ", 11, LineEditType.Integer);

var chb = d.addCheckBox("MyCheckBox:", true);
var text = d.addTextEdit("MyTextBoxLabel:", "abcd efgh");

var label = d.addLabel("Some label text");

var groupbox = d.addGroupBox("MyRadioBox");
var rb1= groupbox.addRadioButton("RB1", false);
var rb2 = groupbox.addRadioButton("RB2", true);

var combo = d.addComboBox("MyCombo");
combo.addItem("ITM1", 1);
combo.addItem("ITM2", 2);
combo.addItem("ITM3", 3);
combo.addItem("ITM4", 4);
combo.addItem("ITM5", 5);

d.addButtons("MyOK", "MyCancel");

var res = d.show();

messageBox.information("MSG", d.caption + ": Result = " + res + "\n" +
    "LABEL: " + label.value + "\n" +
    "TEXT EDIT: " + text.value + "\n" +
    "LINES EDIT: " + line1.value + ", " + line2.value + ", " + line3.value
+ "\n" +
    "CHECK BOX: " + chb.value + "\n" +
    "RADIO BUTTONS: " + rb1.value + ", " + rb2.value + "\n" +
    "COMBO: " + combo.text + ", " + combo.value);
```

## Class: Label

The class represents a item of a user dialog.

**Properties:**

|              |               |                   |
|--------------|---------------|-------------------|
| <b>id</b>    | <i>int</i>    | <i>Read/Write</i> |
| <b>text</b>  | <i>String</i> | <i>Read/Write</i> |
| <b>value</b> | <i>String</i> | <i>Read</i>       |

|                  |                |             |
|------------------|----------------|-------------|
| <b>isEnabled</b> | <i>boolean</i> | <i>Read</i> |
| <b>isHidden</b>  | <i>boolean</i> | <i>Read</i> |
| Methods:         |                |             |
| <b>hide()</b>    | <i>void</i>    |             |
| <b>show()</b>    | <i>void</i>    |             |
| <b>enable()</b>  | <i>void</i>    |             |
| <b>disable()</b> | <i>void</i>    |             |

## Class: CheckBox

The class represents a item of a user dialog.

### Properties:

|                  |                |                   |
|------------------|----------------|-------------------|
| <b>id</b>        | <i>int</i>     | <i>Read/Write</i> |
| <b>text</b>      | <i>String</i>  | <i>Read/Write</i> |
| <b>isEnabled</b> | <i>boolean</i> | <i>Read</i>       |
| <b>isHidden</b>  | <i>boolean</i> | <i>Read</i>       |
| <b>value</b>     | <i>boolean</i> | <i>Read/Write</i> |

It returns *TRUE* if the checkbox is checked or *FALSE* if it is not checked.

|                           |               |                   |
|---------------------------|---------------|-------------------|
| <b>stateChangedMethod</b> | <i>String</i> | <i>Read/Write</i> |
|---------------------------|---------------|-------------------|

A callback method has two parameters: **id**, **value**.

### Methods:

|                  |             |  |
|------------------|-------------|--|
| <b>hide()</b>    | <i>void</i> |  |
| <b>show()</b>    | <i>void</i> |  |
| <b>enable()</b>  | <i>void</i> |  |
| <b>disable()</b> | <i>void</i> |  |

## Class: ComboBox

The class represents an item of a user dialog.

### Properties:

|                                                        |               |                   |
|--------------------------------------------------------|---------------|-------------------|
| <b>id</b>                                              | <i>int</i>    | <i>Read/Write</i> |
| Returns a unique id of the combo object.               |               |                   |
| <b>parentId</b>                                        | <i>int</i>    | <i>Read</i>       |
| Returns a unique id of the parent of the combo object. |               |                   |
| <b>text</b>                                            | <i>String</i> | <i>Read/Write</i> |

Returns a label of a selected item.

|              |            |                   |
|--------------|------------|-------------------|
| <b>value</b> | <i>int</i> | <i>Read/Write</i> |
|--------------|------------|-------------------|

Returns a value of a selected item.

|                     |            |                   |
|---------------------|------------|-------------------|
| <b>currentIndex</b> | <i>int</i> | <i>Read/Write</i> |
|---------------------|------------|-------------------|

Returns or sets the position of the selected item.

|              |            |             |
|--------------|------------|-------------|
| <b>count</b> | <i>int</i> | <i>Read</i> |
|--------------|------------|-------------|

Returns the number of items.

|              |            |             |
|--------------|------------|-------------|
| <b>width</b> | <i>int</i> | <i>Read</i> |
|--------------|------------|-------------|

|                   |            |              |
|-------------------|------------|--------------|
| <b>fixedWidth</b> | <i>int</i> | <i>Write</i> |
|-------------------|------------|--------------|

|                  |            |                   |
|------------------|------------|-------------------|
| <b>iconWidth</b> | <i>int</i> | <i>Read/Write</i> |
|------------------|------------|-------------------|

|                   |            |                   |
|-------------------|------------|-------------------|
| <b>iconHeight</b> | <i>int</i> | <i>Read/Write</i> |
|-------------------|------------|-------------------|

|                  |             |                   |
|------------------|-------------|-------------------|
| <b>iconFrame</b> | <i>bool</i> | <i>Read/Write</i> |
|------------------|-------------|-------------------|

The icon size and the frame are used for items added with an object preview (see *addItem()*).

|                  |                |             |
|------------------|----------------|-------------|
| <b>isEnabled</b> | <i>boolean</i> | <i>Read</i> |
|------------------|----------------|-------------|

|                 |                |             |
|-----------------|----------------|-------------|
| <b>isHidden</b> | <i>boolean</i> | <i>Read</i> |
|-----------------|----------------|-------------|

|                                  |               |                   |
|----------------------------------|---------------|-------------------|
| <b>currentIndexChangedMethod</b> | <i>String</i> | <i>Read/Write</i> |
|----------------------------------|---------------|-------------------|

Set// returns a name of the script callback method. The method process is a 'change index' event. The method has three parameters that replace '?' in the method name: combobox id, value, item position.

Sample setting: `combo.currentIndexChangeMethod = "processComboBoxEvent(?)";`

Sample method definition: `processComboBoxEvent(id, value, index) {...}`

|                        |               |                   |
|------------------------|---------------|-------------------|
| <b>activatedMethod</b> | <i>String</i> | <i>Read/Write</i> |
|------------------------|---------------|-------------------|

A callback method has three parameters: id, value, item position

|                          |               |                   |
|--------------------------|---------------|-------------------|
| <b>highlightedMethod</b> | <i>String</i> | <i>Read/Write</i> |
|--------------------------|---------------|-------------------|

A callback method has three parameters: id, value, item position.

## Methods:

|                                                |             |
|------------------------------------------------|-------------|
| <b>addItem(<i>String label, int value</i>)</b> | <i>void</i> |
|------------------------------------------------|-------------|

|                                                                  |             |
|------------------------------------------------------------------|-------------|
| <b>addItem(<i>String label, int value, boolean selected</i>)</b> | <i>void</i> |
|------------------------------------------------------------------|-------------|

The method adds an item into an end of a combo box list.

|                                                                              |             |
|------------------------------------------------------------------------------|-------------|
| <b>addItem(<i>Object object, String label, int value, bool selected</i>)</b> | <i>void</i> |
|------------------------------------------------------------------------------|-------------|

Adds an item with a preview of the layout object as icon.

|                       |             |
|-----------------------|-------------|
| <b>addSeparator()</b> | <i>void</i> |
|-----------------------|-------------|

|                                          |             |
|------------------------------------------|-------------|
| <b>insertSeparator(<i>int index</i>)</b> | <i>void</i> |
|------------------------------------------|-------------|

|                                                     |             |
|-----------------------------------------------------|-------------|
| <b>setItemKey(<i>int itemIndex, String key</i>)</b> | <i>void</i> |
|-----------------------------------------------------|-------------|

|                                                                                                |               |
|------------------------------------------------------------------------------------------------|---------------|
| <b>itemKey</b> ( <i>int itemIndex</i> )                                                        | <i>String</i> |
| Stores and returns an additional text key for the item at the position itemIndex.              |               |
| <b>hide</b> ()                                                                                 | <i>void</i>   |
| The method hides a combo box object.                                                           |               |
| <b>show</b> ()                                                                                 | <i>void</i>   |
| The method shows (makes visible) a combo box object and sets the item to the current position. |               |
| <b>enable</b> ()                                                                               | <i>void</i>   |
| <b>disable</b> ()                                                                              | <i>void</i>   |

## Class: GroupBox

The class represents a item of a user dialog. It is a box for a radiobutton group.

### Properties:

|                                                            |                |                   |
|------------------------------------------------------------|----------------|-------------------|
| <b>id</b>                                                  | <i>int</i>     | <i>Read/Write</i> |
| <b>clickedMethod</b>                                       | <i>String</i>  | <i>Read/Write</i> |
| A callback method has one parameter: ID of a radio button. |                |                   |
| <b>isEnabled</b>                                           | <i>boolean</i> | <i>Read</i>       |
| <b>isHidden</b>                                            | <i>boolean</i> | <i>Read</i>       |

### Methods:

|                                                                          |                    |
|--------------------------------------------------------------------------|--------------------|
| <b>addRadioButton</b> ( <i>int id, String label, bool defaultValue</i> ) | <i>RadioButton</i> |
| <b>addRadioButton</b> ( <i>String label, bool defaultValue</i> )         | <i>RadioButton</i> |
| The methods add a radiobutton item into a groupbox.                      |                    |
| <b>hide</b> ()                                                           | <i>void</i>        |
| <b>show</b> ()                                                           | <i>void</i>        |
| <b>enable</b> ()                                                         | <i>void</i>        |
| <b>disable</b> ()                                                        | <i>void</i>        |

## Class: RadioButton

The class represents a item of a user dialog.

### Properties:

|                  |                |                   |
|------------------|----------------|-------------------|
| <b>id</b>        | <i>int</i>     | <i>Read/Write</i> |
| <b>text</b>      | <i>String</i>  | <i>Read/Write</i> |
| <b>isEnabled</b> | <i>boolean</i> | <i>Read</i>       |
| <b>isHidden</b>  | <i>boolean</i> | <i>Read</i>       |

|                                                                                                                                                          |                |                   |
|----------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|-------------------|
| <b>value</b>                                                                                                                                             | <i>boolean</i> | <i>Read/Write</i> |
| It returns TRUE if the radiobutton is selected or FALSE if it is not selected.                                                                           |                |                   |
| <b>clickedMethod</b>                                                                                                                                     | <i>String</i>  | <i>Read/Write</i> |
| A callback method has one parameter: ID of the radio button. The 'clickedMethod' takes precedence over the "clickedMethod" defined by a GroupBox object. |                |                   |
| <b>Methods:</b>                                                                                                                                          |                |                   |
| <b>hide()</b>                                                                                                                                            |                | <i>void</i>       |
| <b>show()</b>                                                                                                                                            |                | <i>void</i>       |
| <b>enable()</b>                                                                                                                                          |                | <i>void</i>       |
| <b>disable()</b>                                                                                                                                         |                |                   |

## Class: LineEdit

The class represents a item of a user dialog.

### Properties:

|                                                                                 |                |                   |
|---------------------------------------------------------------------------------|----------------|-------------------|
| <b>id</b>                                                                       | <i>int</i>     | <i>Read/Write</i> |
| <b>value</b>                                                                    | <i>String</i>  | <i>Read/Write</i> |
| It sets and returns a text from a line.                                         |                |                   |
| <b>isEnabled</b>                                                                | <i>boolean</i> | <i>Read</i>       |
| <b>isHidden</b>                                                                 | <i>boolean</i> | <i>Read</i>       |
| <b>cursorPositionChangedMethod</b>                                              | <i>String</i>  | <i>Read/Write</i> |
| A previous callback method has three parameter: ID, old position, new position. |                |                   |
| <b>editingFinishedMethod</b>                                                    | <i>String</i>  | <i>Read/Write</i> |
| <b>returnPressedMethod</b>                                                      | <i>String</i>  | <i>Read/Write</i> |
| <b>selectionChangedMethod</b>                                                   | <i>String</i>  | <i>Read/Write</i> |
| A previous five callback methods have one parameter: <i>ID</i>                  |                |                   |
| <b>textChangedMethod</b>                                                        | <i>String</i>  | <i>Read/Write</i> |
| <b>textEditedMethod</b>                                                         | <i>String</i>  | <i>Read/Write</i> |
| The previous two callback methods have two parameters: <i>ID, TEXT</i> .        |                |                   |
| <b>inputRejectedMethod</b>                                                      | <i>String</i>  | <i>Read/Write</i> |
| The callback method has one parameter: <i>ID</i> .                              |                |                   |
| <b>width</b>                                                                    | <i>int</i>     | <i>Read</i>       |
| <b>fixedWidth</b>                                                               | <i>int</i>     | <i>Write</i>      |

## Methods:

|                  |             |
|------------------|-------------|
| <i>hide()</i>    | <i>void</i> |
| <i>show()</i>    | <i>void</i> |
| <i>enable()</i>  | <i>void</i> |
| <i>disable()</i> | <i>void</i> |

## Class: TextEdit

The class represents a item of a user dialog.

## Properties:

|                                                                                |                |                   |
|--------------------------------------------------------------------------------|----------------|-------------------|
| <b>id</b>                                                                      | <i>int</i>     | <i>Read/Write</i> |
| <b>value</b>                                                                   | <i>String</i>  | <i>Read/Write</i> |
| It sets and returns a text from a textbox.                                     |                |                   |
| <b>cursorPositionChangedMethod</b>                                             | <i>String</i>  | <i>Read/Write</i> |
| A previous callback method has three parameter: ID, old position, new position |                |                   |
| <b>selectionChangedMethod</b>                                                  | <i>String</i>  | <i>Read/Write</i> |
| A previous callback method has one parameter: <i>ID</i> .                      |                |                   |
| <b>textChangedMethod</b>                                                       | <i>String</i>  | <i>Read/Write</i> |
| A previous two callback methods has two parameters: <i>ID,TEXT</i> .           |                |                   |
| <b>isEnabled</b>                                                               | <i>boolean</i> | <i>Read</i>       |
| <b>isHidden</b>                                                                | <i>boolean</i> | <i>Read</i>       |
| <b>width</b>                                                                   | <i>int</i>     | <i>Read</i>       |
| <b>height</b>                                                                  | <i>int</i>     | <i>Read</i>       |
| <b>fixedWidth</b>                                                              | <i>int</i>     | <i>Write</i>      |
| <b>fixedHeight</b>                                                             | <i>int</i>     | <i>Write</i>      |
| Methods:                                                                       |                |                   |
| <b>hide()</b>                                                                  | <i>void</i>    |                   |
| <b>show()</b>                                                                  | <i>void</i>    |                   |
| <b>enable()</b>                                                                | <i>void</i>    |                   |
| <b>disable()</b>                                                               | <i>void</i>    |                   |

## Class: Button

The class represents a item of a user dialog.





# Enumeration Objects Index

# Enumeration Objects Index

|                               |                                                                                                                                                                                      |
|-------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>AutoLineSpacingMethod:</b> | <i>AscentDescent, FontSize</i>                                                                                                                                                       |
| <b>AttributeGroup:</b>        | <i>Character, Paragraph, Layout, Story</i>                                                                                                                                           |
| <b>BaselineGridNumbering:</b> | <i>NoneNumbering, ShowAllNumbers, ShowAllEvenNumbers, ShowFirstAndLastNumbers</i>                                                                                                    |
| <b>BaselineGridDirection:</b> | <i>LeftToRight, RightToLeft, TopToBottom</i>                                                                                                                                         |
| <b>BaselineGridType:</b>      | <i>None, FirstLine, AllLines</i>                                                                                                                                                     |
| <b>BaselineType:</b>          | <i>LargestCharacter, AscentHeight, CapHeight, XHeight, Baseline, LineSpacing</i>                                                                                                     |
| <b>BaselineOffset:</b>        | <i>Off, Fixed, Minimum</i>                                                                                                                                                           |
| <b>BlendType:</b>             | <i>Linear, Rectangle, Focus, Radial</i>                                                                                                                                              |
| <b>BulletCharacters:</b>      | <i>Custom, Bullet, MiddleDot, Dash, Plus, Triangle, Circle</i>                                                                                                                       |
| <b>Color:</b>                 | <i>Black, Blue, Bronze, Brown, Cyan, Gold, Green, Magenta, Orange, Pink, Red, Salmon, Silver, Turquoise, Violet, White, Yellow, NoColor, AlwaysPrintingColor, NeverPrintingColor</i> |
| <b>ColorDepth:</b>            | <i>Depth8bit, Depth24bit</i>                                                                                                                                                         |
| <b>ColorMode:</b>             | <i>Normal, Multiply, Screen, Darken, Lighten, HardLight</i>                                                                                                                          |
| <b>ColorModel:</b>            | <i>Rgb, Cmyk, Hsv, Lab</i>                                                                                                                                                           |
| <b>ColorSeparation:</b>       | <i>CMYK, SpotColor</i>                                                                                                                                                               |
| <b>ColorSeparationState:</b>  | <i>Yes, No, Prepare</i>                                                                                                                                                              |
| <b>ColorSpace:</b>            | <i>Unknown, Mono, Gray, RGB, CMYK, LAB</i>                                                                                                                                           |
| <b>ContentRotation:</b>       | <i>Angle0, Angle90, Angle180, Angle270</i>                                                                                                                                           |
| <b>ContentType:</b>           | <i>Custom, Text, Picture, Table</i>                                                                                                                                                  |
| <b>DialogItemType:</b>        | <i>Unknown, LineEdit, TextEdit, CheckBox, Label, GroupBox, ComboBox, Button</i>                                                                                                      |

**DirectoryFilter:** *NoFilter*

*Dirs:* List directories that match the filters.

*AllDirs:* List all directories. Don't apply the filters to directory names.

*Files:* List files.

*Drives:* List disk drives(ignored under Unix).

*NoSymLinks:* Do not list symbolic links. (Ignored by operating systems that don't support symbolic links).

*NoDot:* Do not list the special entry “.”.

*NoDotDot:* Do not list the special entry “..”.

## Enumeration Objects Index

**NoDotAndDotDot:** Do not list the special entries “.” and “..”.

**AllEntries:** List directories, files, drives and symlinks. (This does not list broken symlinks unless you specify System).

**Readable:** List files for which the application has read access. The Readable value needs to be combined with Dirs or Files.

**Writable:** List files for which the application has write access. The Writable value needs to be combined with Dirs or Files.

**Executable:** List files for which the application has execute access. The Executable value needs to be combined with Dirs or Files.

**Modified:** Only list files that have been modified. (ignored on Unix).

**Hidden:** List hidden files (on Unix, files starting with a “.”).

**System:** List system files (on Unix, FIFOs, sockets and device files are included; on Windows, .lnk files are included)

**CaseSensitive:** The filter should be case sensitive.

|                                   |                                                                                                                                     |
|-----------------------------------|-------------------------------------------------------------------------------------------------------------------------------------|
| <b>DirectorySortFlag:</b>         | <i>NoSort, Name, Time, Size, Unsorted, DirsFirst, Reversed, IgnoreCase, DirsLast, LocaleAware, Type</i>                             |
| <b>DoubleQuotesType:</b>          | <i>InvalidDoubleQuotes, DoubleQuotes1, DoubleQuotes2, DoubleQuotes3, DoubleQuotes4, DoubleQuotes5, DoubleQuotes6, DoubleQuotes7</i> |
| <b>DropCapsType:</b>              | <i>None, WholeWord, Sequence</i>                                                                                                    |
| <b>DropCapsLinesType:</b>         | <i>FixedLineHeight, DynamicLineHeight</i>                                                                                           |
| <b>DropCapsRunaroundType:</b>     | <i>RunaroundBox, RunaroundShape</i>                                                                                                 |
| <b>EndnotesBreakType:</b>         | <i>PlaceInText, PlaceInNewTextColumn, PlaceInNewTextObject</i>                                                                      |
| <b>EndnotesPositionType:</b>      | <i>ChapterEnd, TextEnd</i>                                                                                                          |
| <b>EnumerationsType:</b>          | <i>Off, Numbering, Bullets</i>                                                                                                      |
| <b>EnumerationsAlignment:</b>     | <i>Left, Center, Right</i>                                                                                                          |
| <b>EnumerationsLimits:</b>        | <i>MaxLevels</i>                                                                                                                    |
| <b>EnumerationsNumberingMode:</b> | <i>Continue, BeginNew</i>                                                                                                           |
| <b>EnumerationsNumberFormat:</b>  | <i>Format_1_2_3, Format_a_b_c, Format_A_B_C, Format_i_ii_iii, Format_I_II_III</i>                                                   |
| <b>ErrorStatus:</b>               | <i>Ok, IsNotProcessed, IsNotValid, IsNotSet, IsNotUsable, IsNotAccessToFileSystem</i>                                               |
| <b>ExposureType:</b>              | <i>Positive, Negative</i>                                                                                                           |
| <b>ExtendToNextParagraphType:</b> | <i>No, Auto</i>                                                                                                                     |
| <b>FileDialogAcceptMode:</b>      | <i>Open, Save</i>                                                                                                                   |
| <b>FileDialogFileMode:</b>        | <i>AnyFile, ExistingFile, Directory, ExistingFiles</i>                                                                              |

## Enumeration Objects Index

|                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|--------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>FileDialogOption:</b>             | <i>ShowDirsOnly, DontResolveSymlinks, DontConfirmOverwrite, DontUseNativeDialog, ReadOnly, HideNameFilterDetails, DontUseSheet, DontUseCustomDirectoryIcons</i>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <b>FileDialogViewMode:</b>           | <i>Detail, List</i>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <b>FontReferenceSizeType:</b>        | <i>AscentHeight, CapHeight</i>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <b>FontStyleLineRangeType:</b>       | <i>All, Words</i>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <b>FontStyleLineType:</b>            | <i>None, Automatic, Manual</i>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <b>FootnotesPosition:</b>            | <i>None, Layout, TextFrame, EveryTextColumn</i>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <b>FrameForm:</b>                    | <i>Rectangle, Ellipse, Polygon</i>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| <b>HeightType:</b>                   | <i>Exactly, Minimum</i>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <b>HorizontalAlignment:</b>          | <i>Left, Center, Right</i>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <b>HyphenationQuality:</b>           | <i>StandardQuality, GoodQuality, BestQuality, BestAvailableQuality</i>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <b>ChangeOrderType:</b>              | <i>ToFront, Forward, Backward, ToBack</i>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <b>CharacterBaselineOrientation:</b> | <i>Auto, Horizontal, Vertical</i>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <b>CharacterFunction:</b>            | <i>NormalCharacter, Tab, BorderTab, Indent, EmSpace, EnSpacer, ThreePerEmSpace, FourPerEmSpace, SixPerEmSpace, ThinSpace, HairSpace, PunctuationSpace, NoBreakingEmSpace, EnSpaceNonBreaking, ThreePerEmSpaceNonBreaking, FourPerEmSpaceNonBreaking, SixPerEmSpaceNonBreaking, ThinSpaceNonBreaking, HairSpaceNonBreaking, PunctuationSpaceNonBreaking, SpaceNonBreaking, FixedSpaceNonBreaking, FigureSpace, EmDash, Endash, Hyphen, DiscretionaryHyphen, EmDashNonBreaking, EndashNonBreaking, HyphenNonBreaking, LineBreak, ParagraphBreak, LayoutColumnBreak, TextColumnBreak, TextObjectBreak, LayoutBreak, ChapterBreak, ZeroWidthSpace, ZeroWidthNonJoiner, ZeroWidthJoiner</i> |
| <b>CharacterOutlineJoinStyle:</b>    | <i>MiterJoin, BevelJoin, RoundJoin</i>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <b>CharacterPosition:</b>            | <i>Normal, Subscript, Superscript, Superior</i>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <b>CharacterRotation:</b>            | <i>Auto, Angle0, Angle90, Angle180, Angle270</i>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| <b>CharacterUpperLowerCase:</b>      | <i>Normal, SmallCaps, Caps, LowerCase, TitleCase</i>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |

## Enumeration Objects Index

|                                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|---------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>ImageFormat:</b>             | <i>Format_Invalid, Format_Mono, Format_MonoLSB, Format_Indexed8, Format_RGB32, Format_ARGB32, Format_ARGB32_Premultiplied, Format_RGB16, Format_ARGB8565_Premultiplied, Format_RGB666, Format_ARGB6666_Premultiplied, Format_RGB555, Format_ARGB8555_Premultiplied, Format_RGB888, Format_RGB444, Format_ARGB4444_Premultiplied, Format_RGBX8888, Format_RGBA8888, Format_RGBA8888_Premultiplied, Format_BGR30, Format_A2BGR30_Premultiplied, Format_RGB30, Format_A2RGB30_Premultiplied, Format_Alpha8, Format_Grayscale8, Format_RGBX64, Format_RGBA64, Format_RGBA64_Premultiplied, Format_Grayscale16, Format_BGR888</i>                                                                                                                                                                                                                                                      |
| <b>IndexModel:</b>              | <i>InvalidIndex, Pantone, Focoltone, Trumatch, HKS_K, HKS_E, HKS_Z, HKS_N, Toyo, Dic, Web, FirstCustomIndexModel</i>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <b>KerningType:</b>             | <i>Automatic, Manual</i>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| <b>Language:</b>                | <i>NoLang, Afrikaans_South Africa, Arabic, Bulgarian, Catalan, Chichewa, Chinese, Croatian, Czech, Danish, Dutch, Dutch_Secondary, English, English_AU, English_CA, English_GB, English_NZ, English_US, English_UK_ise, English_UK_ize, Faroese, Finnish, French, French_BE, French_UppercaseUnaccented, Galician, German_AT, German_CH, German_CH_Reformed, German_DE, German_DE_Reformed, German_DE_Combined, Greek, Hebrew, Hungarian, Icelandic, Indonesian, Irish, Italian, Japanese, Kinyarwanda, Kiswahili, Korean, Kurdish, Lithuanian, Latvian, Malagasy, Malay, Maori, Norwegian, Norwegian_Nynorsk, Polish, Portuguese, Portuguese_Brazilian, Romanian, Russian, Russian_Je, Russian_Jo, Setswana, ScottishGaelic, Slovak, Slovenian, Spanish, Spanish_MX, Spanish_UppercaseAccented, Swedish, Tagalog, Tetum, Thai, Ukrainian, Vietnamese, Welsh, Zulu, Universal</i> |
| <b>LayoutColumnsType:</b>       | <i>None, Automatic, Manual</i>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <b>LayoutFootnotesMode:</b>     | <i>None, ColumnDependent, LayoutDependent</i>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| <b>LayoutLineDirection:</b>     | <i>None, TopToBottom, LeftToRight, RightToLeft</i>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <b>LayoutLineJustification:</b> | <i>None, Beginning, Middle, End</i>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| <b>LineCap:</b>                 | <i>Normal, ArrowHead, Bullet, ArrowEnd</i>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <b>LineEditType:</b>            | <i>Double, Integer, String</i>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <b>LineForm:</b>                | <i>FreeLine</i> (simple line),<br><i>ConstrainedLine</i> (line with angles 0°, 45°, 90°, 135°, 180°),<br><i>PolyLine</i> (line composed of multiple lines (path))                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <b>LineNumbersMode:</b>         | <i>None, Left, Right</i>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| <b>LineNumbersRestartType:</b>  | <i>Continue, RestartEveryLayoutColumn, RestartEveryLayout, RestartEveryTextColumn, RestartEveryTextObject, RestartEveryChapter</i>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |

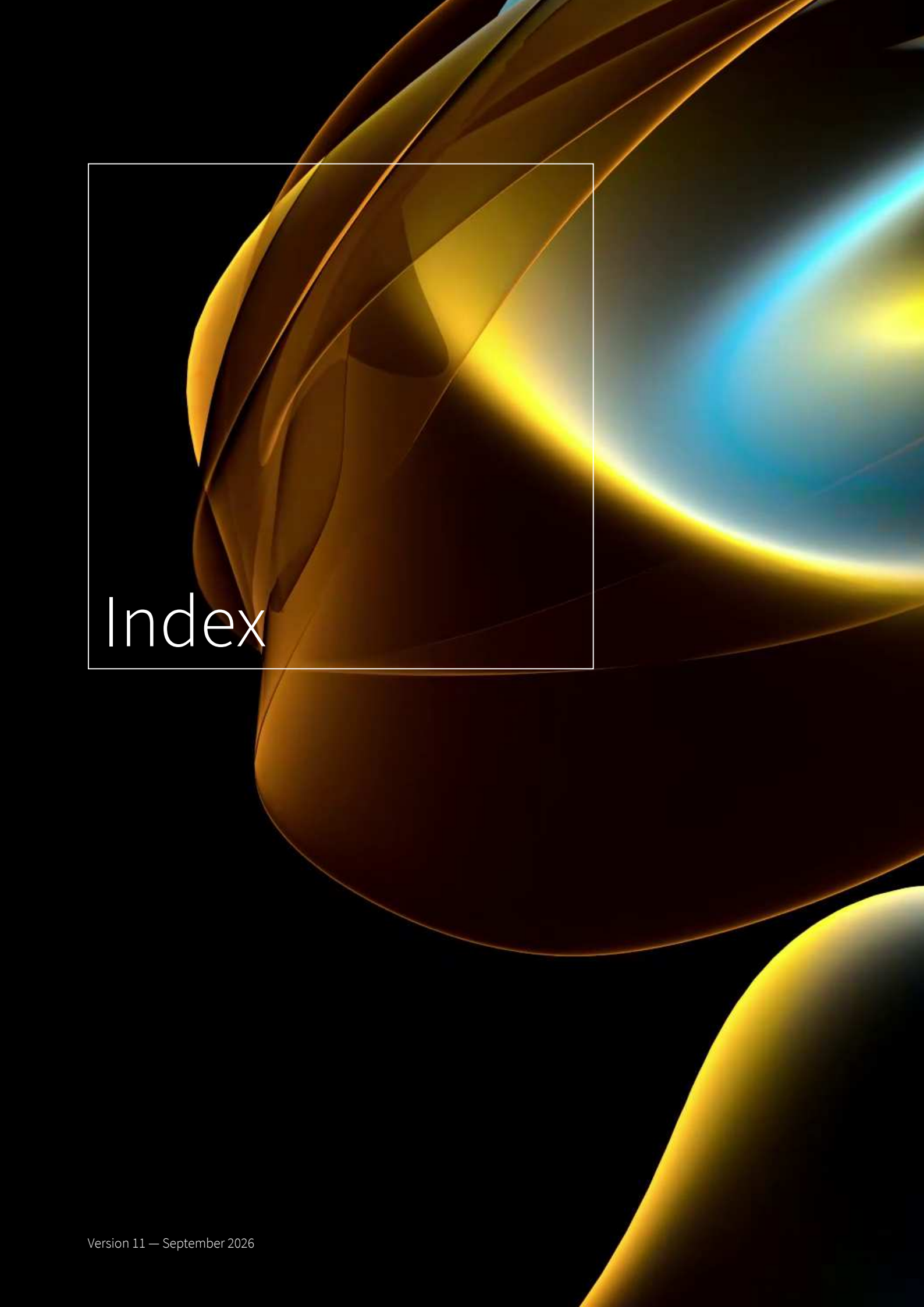
## Enumeration Objects Index

|                                   |                                                                                                                                                                                         |
|-----------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>LinkAction:</b>                | <i>GotoPageNumber, GotoFirstPage, GotoPreviousPage, GotoNextPage, GotoLastPage</i>                                                                                                      |
| <b>LinkDestination:</b>           | <i>None, Page, Url</i>                                                                                                                                                                  |
| <b>MoveReferencePoint:</b>        | <i>OriginTopLeft, BBoxTopLeft</i>                                                                                                                                                       |
| <b>NoteType:</b>                  | <i>Footnote, Endnote</i>                                                                                                                                                                |
| <b>ObjectType:</b>                | <i>All (for getobject list only), Text, Picture, Table, Graphic, Line, Group, Alias, TextAlias, PictureAlias</i>                                                                        |
| <b>OpenFileMode:</b>              | <i>ReadOnly, WriteOnly, ReadWrite, Append</i>                                                                                                                                           |
| <b>OpenTypePlacement:</b>         | <i>Undefined, None, Denominators, Numerators, Superscript, Subscript</i>                                                                                                                |
| <b>OpenTypeNumerals:</b>          | <i>Undefined, None, TabularFigures, OldstyleFigures, LiningFigures, ProportionalFigures</i>                                                                                             |
| <b>OpenTypePositionalForms:</b>   | <i>Undefined, None, TerminalForms, InitialForms, MedialForms, IsolatedForms</i>                                                                                                         |
| <b>OpticalAlignment:</b>          | <i>None, Font, Auto</i>                                                                                                                                                                 |
| <b>PageInsertPosition:</b>        | <i>BeforeCurrentPage, AfterCurrentPage, AtStartOfDocument, AtEndOfDocument, AfterPage</i>                                                                                               |
| <b>PageMode:</b>                  | <i>AliasPages, SinglePages, DoublePages, Spreads</i>                                                                                                                                    |
| <b>PageNumberingFormat:</b>       | <i>None, Arabic, UpperLetter, LowerLetter, UpperRoman, LowerRoman</i><br>Example: 1 2 3 4, A B C D, a b c d, I II III IV, i ii iii iv                                                   |
| <b>PageOrder:</b>                 | <i>LeftRight, RightLeft</i>                                                                                                                                                             |
| <b>PageOrientation:</b>           | <i>Portrait, Landscape</i>                                                                                                                                                              |
| <b>PageType:</b>                  | <i>SinglePage, FacingPages</i>                                                                                                                                                          |
| <b>PageSize:</b>                  | <i>A0, A1, A2, A3, A4, A5, A6, A7, B5, USLetter, USLegal, PF_11_17, CompactDisk, LetterHalf, Custom</i>                                                                                 |
| <b>ParagraphAlignment:</b>        | <i>Left, Right, Center, Justified, ForcedJustified</i>                                                                                                                                  |
| <b>ParagraphRuleLengthType:</b>   | <i>Text, Column, Space</i>                                                                                                                                                              |
| <b>PDFCompatibility:</b>          | <i>PDF_13, PDF_14, PDF_15, PDF_16, PDF_17, PDF_17ext3, PDF_17ext8, PDF_20, PDFX1_2001, PDFX1a_2001, PDFX1a_2003, PDFX2_2003, PDFX3_2002, PDFX3_2003, PDFX4, PDFX4p, PDFX5g, PDFX5pg</i> |
| <b>PDFCompressionMethod:</b>      | <i>None, ZIP, JPEG</i>                                                                                                                                                                  |
| <b>PDFColorConversion:</b>        | <i>None, Always, PreserverNumbers</i>                                                                                                                                                   |
| <b>PictureFormat:</b>             | <i>TIFF, JPEG, PNG</i>                                                                                                                                                                  |
| <b>PictureHorizontalPosition:</b> | <i>Ignore, Left, Center, Right</i>                                                                                                                                                      |
| <b>PictureVerticalPosition:</b>   | <i>Ignore, Top, Center, Bottom</i>                                                                                                                                                      |
| <b>PolygonPointFlag:</b>          | <i>After, Default, Control, Before, Constraint, First, Last, Align</i>                                                                                                                  |

## Enumeration Objects Index

|                                      |                                                                                                                                                                                                                                                                                                                                                                         |
|--------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>PrintDeviceType:</b>              | <i>LaserPrinter, ImageSetter, FilmRecorder, ColorPrinter, ColorProof, Plotter, ComputerToPlate</i>                                                                                                                                                                                                                                                                      |
| <b>PrintImageType:</b>               | <i>NoMirroring, Mirroring</i>                                                                                                                                                                                                                                                                                                                                           |
| <b>PrintMaterialType:</b>            | <i>Paper, Film, Foil</i>                                                                                                                                                                                                                                                                                                                                                |
| <b>PrintResolutionType:</b>          | <i>Low, High, R300dpi, R600dpi, R1280dpi, R2400dpi</i>                                                                                                                                                                                                                                                                                                                  |
| <b>PrintPagesRangeType:</b>          | <i>CurrentPage, PagesRange, PageSelection</i>                                                                                                                                                                                                                                                                                                                           |
| <b>PrintLayersRange:</b>             | <i>AllLayers, VisibleLayers, PrintableLayers, VisibleAndPrintableLayers</i>                                                                                                                                                                                                                                                                                             |
| <b>PrintPageNumbering:</b>           | <i>Physical, Logical, PhysicalAndLogical</i>                                                                                                                                                                                                                                                                                                                            |
| <b>PrintColorSeparation:</b>         | <i>Off, App, InRIP</i>                                                                                                                                                                                                                                                                                                                                                  |
| <b>PrintHorizontalAlignment:</b>     | <i>Left, Center, Right</i>                                                                                                                                                                                                                                                                                                                                              |
| <b>PrintVerticalAlignment:</b>       | <i>Top, Center, Bottom</i>                                                                                                                                                                                                                                                                                                                                              |
| <b>PrintFontEmbedding:</b>           | <i>None, AllButBase14, OnlyType1, All</i>                                                                                                                                                                                                                                                                                                                               |
| <b>PrintImageQuality:</b>            | <i>None, LoRes, HiRes</i>                                                                                                                                                                                                                                                                                                                                               |
| <b>PrintOPI:</b>                     | <i>None, TIFF, EPS, Both, Generic</i>                                                                                                                                                                                                                                                                                                                                   |
| <b>PrintNameChangePolicy:</b>        | <i>AppendPageNumber, ReplaceWithPageNumber, ReplaceWithPageName</i>                                                                                                                                                                                                                                                                                                     |
| <b>ProgressReporterType:</b>         | <i>NoItems, TextItem, TwoTextItems, CancelButton</i>                                                                                                                                                                                                                                                                                                                    |
| <b>ReferenceAnchorNumberingType:</b> | <i>ContinuousNumbering, ChapterWiseNumbering</i>                                                                                                                                                                                                                                                                                                                        |
| <b>RubyPosition:</b>                 | <i>Above, Before, Below, After, Hidden</i>                                                                                                                                                                                                                                                                                                                              |
| <b>RunaroundMode:</b>                | <i>None, Left, Right, All</i>                                                                                                                                                                                                                                                                                                                                           |
| <b>RunaroundShapeType:</b>           | <i>ObjectContours, Block, ImageBoundaries, ClippingPath</i>                                                                                                                                                                                                                                                                                                             |
| <b>ScalePictureType:</b>             | <i>Proportional, FitToFrame, FillFrameProportionally, FitToPicture, OriginalSize</i>                                                                                                                                                                                                                                                                                    |
| <b>ScreenAngle:</b>                  | <i>Cyan, Magenta, Yellow, Black</i>                                                                                                                                                                                                                                                                                                                                     |
| <b>SingleQuotesType:</b>             | <i>InvalidSingleQuotes, SingleQuotes1, SingleQuotes2, SingleQuotes3, SingleQuotes4, SingleQuotes5, SingleQuotes6, SingleQuotes7</i>                                                                                                                                                                                                                                     |
| <b>SpecialCharacters:</b>            | <i>SingleQuote1, SingleQuote2, SingleQuote3, SingleQuote4, SingleQuote5, SingleQuote6, DoubleQuote1, DoubleQuote2, DoubleQuote3, DoubleQuote4, DoubleQuote5, DoubleQuote6, Bullet, ATSign, SectionSign, BaskslashSign, VerticalLine, PerMilleSign, HorizontalEllipsis, EuroSign, DollarSign, PoundSign, YenSign, CopyrightSymbol, RegisteredSymbol, TrademarkSymbol</i> |
| <b>StyleSheetFamily:</b>             | <i>Character, Paragraph, Layout, Graphic, Picture, Table, TableRow, TableColumn, TableCell</i>                                                                                                                                                                                                                                                                          |
| <b>TableLineSeparator:</b>           | <i>None, Vertical, Horizontal</i>                                                                                                                                                                                                                                                                                                                                       |



The background of the page is an abstract composition of flowing, organic shapes in shades of gold, yellow, and blue against a dark background. A white rectangular box is positioned on the left side, containing the word 'Index'.

# Index

## Index

## A

absolutePath . . . . . 12  
accept . . . . . 205  
acceptLabel . . . . . 203  
acceptMode . . . . . 203  
activatedMethod . . . . . 208  
active134, 159, 160, 161, 163, 167, 169, 170, 171, 173, 174, 178, 179, 180, 181, 182, 185, 186, 188, 189, 190, 193  
active ( . . . . . 175  
add . . . . . 19, 24, 63  
addAcceptButton . . . . . 205  
addAliasPage . . . . . 16  
addBlend . . . . . 114  
addButton . . . . . 205  
addButtons . . . . . 205  
addCheckBox . . . . . 205  
addColor . . . . . 114  
addColumn . . . . . 173  
addComboBox . . . . . 205  
addDocumentPage . . . . . 16  
addGroupBox . . . . . 205  
addHorizontalGuide . . . . . 25  
addItem . . . . . 208  
addLabel . . . . . 205  
addLayer(). . . . . 17  
addLineEdit . . . . . 205  
addOption . . . . . 203  
addPath . . . . . 143  
addRadioButton . . . . . 209  
addRejectButton . . . . . 205  
addSeparator . . . . . 208  
addTabulator . . . . . 190  
addTextEdit . . . . . 205  
addVerticalGuide . . . . . 25  
afterPageNumber . . . . . 142  
aliasObjectsSelectable . . . . . 149  
aliasPage . . . . . 21  
aliasPageAt . . . . . 16  
aliasPageByName . . . . . 16  
aliasPageCount . . . . . 13  
aliasSp . . . . . 14  
aliasSpreadAt . . . . . 16  
alignCharacter . . . . . 190  
alignment . . . . . 165, 190  
alternatingColumns . . . . . 133  
alternatingColumnsData . . . . . 133  
alternatingRows . . . . . 133  
alternatingRowsData . . . . . 133  
alternativeText . . . . . 188, 189, 190, 191  
anchorText . . . . . 176  
angle . . . . . 186, 193  
annotation . . . . . 72  
annotationTextRange . . . . . 158  
appendPageNumber . . . . . 189  
appendPlainText . . . . . 70  
appendPoint . . . . . 108  
appendStop . . . . . 116, 193  
application . . . . . 8  
applyAsIndent . . . . . 190  
applyElementOpacity . . . . . 186  
applyStandardTabs . . . . . 190  
applyValue . . . . . 160  
appScriptsPath . . . . . 8  
areaAt . . . . . 51  
areaCount . . . . . 51  
areaIndex . . . . . 51  
askForImages . . . . . 149  
at . . . . . 31, 51, 101, 114, 124  
AttributeGroup . . . . . 214

Attributes(). . . . . 159  
author . . . . . 143, 162  
AutoLineSpacingMethod . . . . . 214  
automaticCharacterSpacing . . . . . 89, 129  
AutomaticCharacterSpacing(). . . . . 159, 160  
automaticCharacterWidth . . . . . 90, 129  
automaticGrammarCheck . . . . . 152  
automaticSpellCheck . . . . . 152  
automaticTextObject . . . . . 140  
automaticWordSpacing . . . . . 89, 129

## B

baselineDescenders . . . . . 45, 132  
baselineGrid . . . . . 45, 132, 153  
BaselineGrid . . . . . 160  
BaselineGrid(). . . . . 160  
BaselineGridDirection . . . . . 214  
BaselineGridNumbering . . . . . 214  
baselineGridType . . . . . 88, 89, 129  
BaselineGridType . . . . . 214  
BaselineOffset . . . . . 214  
baselinePosition . . . . . 45, 132  
BaselinePosition . . . . . 160  
BaselineType . . . . . 214  
baseOpacity . . . . . 131  
baseTextRange . . . . . 158  
begin . . . . . 101  
bleed . . . . . 178  
bleedBottom . . . . . 184  
bleedLeft . . . . . 183  
bleedRight . . . . . 183  
bleedTop . . . . . 183  
blendAngle . . . . . 54, 56, 57  
BlendType . . . . . 214  
blue . . . . . 118  
blur . . . . . 173, 179, 186  
bottom . . . . . 186  
bottomAnchor . . . . . 162  
bottomBorder . . . . . 181  
bottomIndent . . . . . 44, 57, 136, 162, 169, 170, 180, 181  
bottomMargin . . . . . 140  
bottomSeparator . . . . . 136  
bottomSeparatorData . . . . . 136  
bottomTextIndent . . . . . 132  
brush . . . . . 27, 55, 56, 57, 61, 77, 126, 128, 169, 170, 171, 179, 180, 181, 182, 185, 187, 193  
brushes . . . . . 14  
bullet . . . . . 165  
BulletCharacters . . . . . 214  
bwImageResolution . . . . . 178  
bwImagesMinimumResolution . . . . . 150

## C

caption . . . . . 203, 204  
cell . . . . . 53  
changeColorSpace . . . . . 50  
changed . . . . . 18  
changeDir . . . . . 198  
changeOrder . . . . . 35  
ChangeOrderType . . . . . 216  
ChangePictureColorSpaceSettings(). . . . . 168  
changeTracking . . . . . 153  
chapterInfo . . . . . 21  
ChapterInfo . . . . . 142  
ChapterInfo(). . . . . 142  
chapterName . . . . . 96  
chapters . . . . . 97  
characterBackground . . . . . 86, 127, 128  
CharacterBackground . . . . . 168

characterBaselineOrientation . . . . . 83, 127, 128  
CharacterBaselineOrientation . . . . . 216  
characterBaselineShift . . . . . 82, 83, 126, 128  
characterCount . . . . . 163  
characterFrame . . . . . 86, 127, 128  
CharacterFrame . . . . . 169  
CharacterFrame(). . . . . 169  
characterFunction . . . . . 73  
CharacterFunction . . . . . 216  
characterHeightScale . . . . . 81, 126, 128  
characterKerning . . . . . 82, 126, 128  
CharacterKerning(). . . . . 170  
characterLigatures . . . . . 84, 127, 128  
CharacterOutlineJoinStyle . . . . . 216  
characterPosition . . . . . 80, 126, 128  
CharacterPosition . . . . . 216  
characterPosition(int begin, int end) . . . . . 80  
characterPosition(int pos) . . . . . 80  
characterPosition(int pos=0) . . . . . 80  
CharacterPositionCorrection . . . . . 171  
CharacterPositionCorrection(). . . . . 171  
characterRotation . . . . . 83, 127, 128  
CharacterRotation . . . . . 216  
characterRule . . . . . 87, 127, 128  
CharacterRule . . . . . 171  
CharacterRule(). . . . . 171  
characterSpacing . . . . . 81, 82, 126, 128  
characterStyleSheet . . . . . 87, 127, 153, 165  
characterStyleSheetByName . . . . . 88, 153, 165  
characterUpperLowerCase . . . . . 80, 126, 128  
CharacterUpperLowerCase . . . . . 216  
characterWidthScale . . . . . 81, 126, 128  
charAsCode . . . . . 71  
charAsString . . . . . 71  
cleanupFolder . . . . . 179  
clear(). . . . . 74, 125  
clearGuides . . . . . 25  
clickedMethod . . . . . 209, 210  
clippingEnabled . . . . . 49  
clippingTight . . . . . 49  
clippingTolerance . . . . . 49  
close . . . . . 17, 200  
close(). . . . . 204  
closeAndSave(). . . . . 17  
cmyk . . . . . 115  
color . . . . . 161, 173, 178, 186  
Color . . . . . 214  
colorBrush . . . . . 117  
ColorBrush . . . . . 115  
colorDepth . . . . . 149  
ColorDepth . . . . . 214  
colorImageResolution . . . . . 178  
colorImagesMinimumResolution . . . . . 150  
colorMode . . . . . 49  
ColorMode . . . . . 214  
colorModel . . . . . 115  
ColorModel . . . . . 214  
colorOverlay . . . . . 37, 132  
ColorOverlay(). . . . . 161  
colorProfile . . . . . 168  
colorRgb . . . . . 27  
colorSeparation . . . . . 115, 143, 183  
ColorSeparation . . . . . 214  
ColorSeparationState . . . . . 214  
colorSpace . . . . . 49, 168  
ColorSpace . . . . .  
column . . . . . 52, 58  
columnCount . . . . . 44, 52, 132, 162, 166, 174  
columnDistance . . . . . 166

## Index

columnGutter . . . . . 132, 141  
columnIndex . . . . . 163  
columnLines . . . . . 44, 132  
ColumnLines . . . . . 161  
ColumnLines() . . . . . 161  
columnOffset . . . . . 44  
columns . . . . . 130  
columnSpan . . . . . 57  
comment . . . . . 72  
Comment() . . . . . 162  
connectTo . . . . . 43  
containsUserProperty . . . . . 17, 20, 25, 35  
content . . . . . 42, 47, 64  
contentExpandableObject . . . . . 58  
contentObject . . . . . 58  
contentOpacity . . . . . 42, 47, 131  
contentRotation . . . . . 57  
ContentRotation . . . . . 214  
contentType . . . . . 58  
ContentType . . . . . 214  
convertUnitValue . . . . . 8  
copy . . . . . 199  
copy() . . . . . 35, 74  
copyAsAlias . . . . . 142  
copyAttributes . . . . . 36, 46, 50, 55, 56, 59, 61  
copyContentFrom . . . . . 50  
copyObjectPreferences . . . . . 18  
copyObjects . . . . . 142  
copyOnlySelectedObjects . . . . . 142  
copyUserProperties . . . . . 34  
count . . . . . 134, 173, 190, 198, 208  
Count . . . . . 13, 14  
Courier Bold . . . . . 2  
coverShadow . . . . . 186  
create . . . . . 31, 124  
createAlias() . . . . . 35  
creationDate . . . . . 162  
crop . . . . . 178  
cropMarkDistance . . . . . 183  
cropMarkLength . . . . . 183  
cropmarks . . . . . 183  
cropsHeight . . . . . 185  
cropsWidth . . . . . 185  
cropsX . . . . . 185  
cropsY . . . . . 185  
currentDocument . . . . . 12  
currentIndex . . . . . 208  
currentIndexChangedMethod . . . . . 208  
currentLayer . . . . . 15  
currentPage . . . . . 14  
currentPictureObject . . . . . 15  
currentSpread . . . . . 15  
currentText . . . . . 15  
cursorPositionChangedMethod . . . . . 210, 211  
customHeight . . . . . 140  
customProperty . . . . . 35  
customPropertyBoolean . . . . . 35  
customWidth . . . . . 140  
cyan . . . . . 118

## D

deadline . . . . . 143  
deleteObjects() . . . . . 20  
deletePoint . . . . . 108  
deleteSelectedObjects() . . . . . 20  
destination . . . . . 176  
Dialog . . . . . 204  
DialogItemType . . . . . 214  
direction . . . . . 160  
directory . . . . . 203  
Directory . . . . . 198  
directoryFilter . . . . . 203  
DirectoryFilter . . . . . 214  
DirectorySortFlag  
disable . . . . . 207, 209, 211

disableSystemFonts . . . . . 148  
displayPosition . . . . . 191  
distance . . . . . 185  
distanceBetweenStandardTabs . . . . . 152  
distanceToText . . . . . 163  
documentAt . . . . . 12  
documentCount . . . . . 12  
documentExportFormats . . . . . 12  
documentImportFormats . . . . . 12  
documentName . . . . . 176  
documentOpenFormats . . . . . 12  
documentPageAt . . . . . 15  
documentPageCount . . . . . 13  
documentSaveFormats . . . . . 12  
documentSettings . . . . . 15  
DocumentSettings . . . . . 140  
documentSp . . . . . 13  
documentSpreadAt . . . . . 16  
documentStatistics . . . . . 14  
DocumentStatistics . . . . . 142  
done . . . . . 205  
doubleQuotesType . . . . . 148  
DoubleQuotesType . . . . . 215  
dropCaps . . . . . 91, 129  
DropCaps . . . . . 163  
DropCaps() . . . . . 163  
DropCapsLinesType . . . . . 215  
DropCapsRunaroundType . . . . . 215  
DropCapsType . . . . . 215  
dropShadow . . . . . 36, 132  
dump . . . . . 8  
dx . . . . . 110  
dy . . . . . 110

## E

editingFinishedMethod . . . . . 210  
embedAllImages . . . . . 149  
embedAllUsedFonts . . . . . 148  
embeddedFontsPreferences . . . . . 14  
enable . . . . . 207, 209, 210, 211  
end . . . . . 101  
endnoteLabelStyleSheet . . . . . 153  
endnoteLabelStyleSheetByName . . . . . 153  
endnoteReferenceStyleSheet . . . . . 153  
endnoteReferenceStyleSheetByName . . . . . 153  
endnotes . . . . . 153  
EndnotesBreakType . . . . . 215  
EndnotesPositionType . . . . . 215  
endnotesRanges . . . . . 97  
endnoteTextStyleSheet . . . . . 153  
endnoteTextStyleSheetByName . . . . . 153  
enumerations . . . . . 91, 129  
Enumerations . . . . . 164  
Enumerations() . . . . . 164  
EnumerationsAlignment . . . . . 215  
EnumerationsLimits  
EnumerationsNumberFormat . . . . . 215  
EnumerationsNumberingMode . . . . . 215  
EnumerationsType . . . . . 215  
eof . . . . . 200  
equals . . . . . 51  
ErrorStatus  
exec . . . . . 205  
execute . . . . . 8  
exists . . . . . 144, 198, 199  
exportToPDF . . . . . 17  
exportToXML . . . . . 17  
exposure . . . . . 143  
ExposureType . . . . . 215  
extendToColumnBorder . . . . . 180, 181  
extendToNextParagraph . . . . . 180, 181  
ExtendToNextParagraphType . . . . . 215

## F

fax . . . . . 143

featuresString . . . . . 178  
File . . . . . 199  
FileDialogAcceptMode . . . . . 215  
FileDialogFileMode . . . . . 215  
FileDialogOption . . . . . 216  
FileDialogViewMode . . . . . 216  
fileMode . . . . . 203  
fileName . . . . . 199  
fileNameLabel . . . . . 203  
filePath . . . . . 12, 14, 48  
fileTypeLabel . . . . . 203  
fillBlendAngle . . . . . 41, 131, 133, 135, 136  
fillBrush . . . . . 131, 133, 135, 136  
fillCharacter . . . . . 190  
fillColor . . . . . 41  
fillOpacity . . . . . 41, 131  
fillShade . . . . . 41, 131, 133, 135, 136  
find . . . . . 74, 114, 124  
findAndReplace . . . . . 74  
findAndReplaceAll . . . . . 74  
findAndReplaceAllRegEx . . . . . 74  
findAndReplaceRegEx . . . . . 74  
findRegEx . . . . . 74  
firstBaseline . . . . . 153, 160  
firstCount . . . . . 134  
firstStyleSheet . . . . . 134  
firstStyleSheetByName . . . . . 134  
fixedHeight . . . . . 211  
fixedWidth . . . . . 208, 210, 211  
flagAfter . . . . . 107  
flagAfterConstraint . . . . . 107  
flagBefore . . . . . 107  
flagBeforeConstraint . . . . . 107  
flagFirst . . . . . 107  
flagLast . . . . . 107  
flags . . . . . 106  
flush . . . . . 200  
focusPosition . . . . . 117, 192  
font . . . . . 75, 126, 127  
FontReferenceSizeType . . . . . 216  
fontSize . . . . . 76, 126, 127  
FontSize . . . . . 166  
fontsSubfolder . . . . . 180  
FontStyleLineRangeType . . . . . 216  
FontStyleLineType . . . . . 216  
footer . . . . . 45, 132  
footerFormattedText . . . . . 45  
footerRows . . . . . 133  
footerRowsData . . . . . 133  
footnoteLabelStyleSheet . . . . . 153  
footnoteLabelStyleSheetByName . . . . . 153  
footnoteReferenceStyleSheet . . . . . 153  
footnoteReferenceStyleSheetByName . . . . . 153  
footnotes . . . . . 45, 130, 132, 153  
FootnotesPosition . . . . . 216  
footnoteTextStyleSheet . . . . . 153  
footnoteTextStyleSheetByName . . . . . 153  
forceLineBreak . . . . . 152  
forceRedraw . . . . . 204  
form . . . . . 37, 40  
format . . . . . 168, 178  
formattedText . . . . . 45, 71  
frameColor . . . . . 40  
FrameForm . . . . . 216  
frameOpacity . . . . . 40  
frameRadius . . . . . 41  
frameShade . . . . . 40  
frameStyle . . . . . 40  
frameWidth . . . . . 40, 170, 181  
fromPage . . . . . 183  
fullCopy() . . . . . 35  
fullPath . . . . . 198

## G

getAttributes . . . . . 36, 46, 50, 55, 56, 59, 61, 75

## Index

getExistingDirectory . . . . . 203  
getFileList . . . . . 199  
getObject . . . . . 15, 19, 24, 27, 62  
getOpenFileName . . . . . 203  
getPoint . . . . . 108  
getSaveFileName . . . . . 204  
getSaveFileName() . . . . . 204  
getTabulator . . . . . 190  
gotoPage . . . . . 16  
GraphicFootnotes() . . . . . 166  
graphicStyleSheet . . . . . 149  
grayImageResolution . . . . . 178  
grayImagesMinimumResolution . . . . . 150  
greekBelow . . . . . 152  
green . . . . . 117  
groupObject . . . . . 34  
guide . . . . . 20, 25, 33  
guideCount . . . . . 19, 24  
guideObject . . . . . 131  
guideObjectsMagnetic . . . . . 149  
guideObjectsSelectable . . . . . 149

## H

halftoneAngle . . . . . 49, 131  
halftoneScreen . . . . . 49, 131  
header . . . . . 45, 132  
HeaderFooter() . . . . . 167  
headerFormattedText . . . . . 45  
headerRows . . . . . 133  
headerRowsData . . . . . 133  
height . . . . . 40, 48, 54, 109, 131, 167, 211  
heightType . . . . . 54  
HeightType . . . . . 216  
hidden . . . . . 27, 115  
hide . . . . . 207, 209, 210, 211  
highlightedMethod . . . . . 208  
horizontalAlignment . . . . . 57, 136, 184, 185  
HorizontalAlignment . . . . . 216  
horizontalCorrection . . . . . 171  
horizontalMirrored . . . . . 49  
horizontalOffset . . . . . 48  
horizontalPasteOffset . . . . . 149  
horizontalScale . . . . . 49, 130  
horizontalSeparator . . . . . 135  
horizontalSeparatorAbove . . . . . 52  
horizontalSeparatorData . . . . . 135  
hsv . . . . . 115  
Hsv . . . . . 118  
hue . . . . . 118  
hyphenation . . . . . 91, 129  
Hyphenation . . . . . 167  
Hyphenation() . . . . . 167  
hyphenationModuleIdent . . . . . 148  
HyphenationQuality . . . . . 216

## I

iconFrame . . . . . 208  
iconHeight . . . . . 208  
iconWidth . . . . . 208  
id . . . . . 32, 206, 207, 209, 210, 211  
identifier . . . . . 12  
ignoreTrapping . . . . . 184  
image . . . . . 50  
Image() . . . . . 172  
Image(String data) . . . . . 172  
imageData . . . . . 50  
ImageFormat . . . . . 217  
imageQuality . . . . . 184  
imagesSubfolder . . . . . 180  
importPicture . . . . . 50  
include . . . . . 7  
includeDocument . . . . . 179  
includeFonts . . . . . 179  
includeImages . . . . . 179  
includeProtocol . . . . . 179

indexKey . . . . . 115  
indexModel . . . . . 115  
IndexModel . . . . . 217  
individualAliasContent . . . . . 42, 47, 132  
information ( . . . . . 198  
innerGlow . . . . . 37, 132  
InnerGlow() . . . . . 172  
innerMargin . . . . . 140  
innerShadow . . . . . 36, 132  
inputRejectedMethod . . . . . 210  
insertAfter . . . . . 62  
insertAnnotation . . . . . 72  
insertAt . . . . . 63  
insertCharacter . . . . . 71  
insertComment . . . . . 72  
insertFormattedText . . . . . 71  
insertHtml . . . . . 71  
insertNote . . . . . 73  
insertPlainText . . . . . 45, 70  
insertPoint . . . . . 108  
insertPosition . . . . . 141  
insertSeparator . . . . . 208  
insertVariable . . . . . 74  
isAbsolute . . . . . 198  
isAliasPage . . . . . 19  
isEmbedded . . . . . 49  
isEmpty . . . . . 49  
isEnabled . . . . . 207, 208, 209, 210, 211  
isGroup . . . . . 107  
isHidden . . . . . 207, 208, 209, 210, 211  
isNull . . . . . 167  
isOnline . . . . . 30  
isReadable . . . . . 198  
isRelative . . . . . 198  
isRoot . . . . . 198  
itemKey . . . . . 209

## J

joinStyle . . . . . 179  
justifiedSpaceBetweenLinesMaximum . . . . . 94, 130

## K

keepParagraphTogether . . . . . 93, 129  
keepRunaround . . . . . 27  
KerningType . . . . . 217  
key . . . . . 118

## L

lab . . . . . 115  
Lab . . . . . 118  
label . . . . . 21  
language . . . . . 76, 77, 126, 128, 148, 152  
Language . . . . . 217  
languageByName . . . . . 76  
languageInfo . . . . . 148  
languagePreferences . . . . . 14  
layer . . . . . 34  
layerAt . . . . . 17  
layerCount . . . . . 14  
layersRange . . . . . 183  
layoutColumns . . . . . 95  
LayoutColumnsInfo() . . . . . 173  
LayoutColumnsType . . . . . 217  
layoutFootnotes . . . . . 95  
LayoutFootnotes() . . . . . 174  
LayoutFootnotesMode . . . . . 217  
layoutGrid . . . . . 140  
LayoutLineDirection . . . . . 217  
layoutLineJustification . . . . . 94  
LayoutLineJustification . . . . . 217  
layoutLineOrientation . . . . . 94  
layoutPositionColumn . . . . . 19  
layoutPositionRow . . . . . 19  
layoutPreferences . . . . . 14  
layouts . . . . . 97

layoutStyleSheet . . . . . 95, 153  
layoutStyleSheetByName . . . . . 96, 153  
left . . . . . 186  
leftBorder . . . . . 181  
leftColumns . . . . . 133  
leftColumnsData . . . . . 133  
leftIndent . . . . . 44, 57, 136, 169, 170, 171, 180, 181, 182  
leftMargin . . . . . 141  
leftSeparator . . . . . 136  
leftSeparatorData . . . . . 136  
leftTextIndent . . . . . 132  
length . . . . . 30, 51, 62, 70, 101, 102, 107, 114, 124, 143  
lengthType . . . . . 182  
level . . . . . 165, 189, 190  
lightness . . . . . 119  
lineAngle . . . . . 37  
lineBrush . . . . . 133, 174  
LineCap . . . . . 217  
lineColor . . . . . 37, 131, 161  
lineCornerRadius . . . . . 131  
LineEditType . . . . . 217  
lineEnd . . . . . 38, 132  
LineForm . . . . . 217  
lineJustification . . . . . 130  
lineLength . . . . . 37  
LineNumbers . . . . . 175  
LineNumbers() . . . . . 175  
lineNumbersColor . . . . . 152  
lineNumbersData . . . . . 152  
lineNumbersFont . . . . . 152  
lineNumbersFontSize . . . . . 152  
lineNumbersMode . . . . . 152  
LineNumbersMode . . . . . 217  
LineNumbersRestartType . . . . . 217  
lineOpacity . . . . . 38, 131, 162  
lineOrientation . . . . . 130  
lineShade . . . . . 37, 131, 133, 161, 175  
lineStart . . . . . 38, 131  
lineStyle . . . . . 38, 131, 133, 162, 170, 171, 175, 179, 181, 182, 187, 193  
LineStyle . . . . . 174  
LineStyle() . . . . . 174  
linesType . . . . . 163  
lineWidth . . . . . 38, 131, 133, 162, 171, 175, 179, 182, 187, 193  
link . . . . . 86, 127, 128  
Link . . . . . 175  
Link() . . . . . 175  
linkableWithOriginal . . . . . 45, 132  
linkAction . . . . . 176  
LinkAction . . . . . 218  
LinkDestination . . . . . 218  
linkToTextChain . . . . . 142  
loadExcelSheet . . . . . 162, 163  
localizedName . . . . . 115  
locked . . . . . 27, 33  
logicalNumber . . . . . 21  
lookInLabel . . . . . 203

## M

m11 . . . . . 110  
m12 . . . . . 110  
m21 . . . . . 110  
m22 . . . . . 110  
magenta . . . . . 118  
mainText . . . . . 158, 176  
makeDir . . . . . 198  
margin . . . . . 174  
marginAt . . . . . 173  
matrix . . . . . 33  
Matrix . . . . . 109  
Matrix() . . . . . 109  
maxHyphensInARow . . . . . 167  
maximum . . . . . 159, 160  
maximumLines . . . . . 163  
maximumPreviews . . . . . 150

## Index

maximumWidth . . . . . 204  
message) . . . . . 198  
messageBox . . . . . 8  
mimeTypeFilter . . . . . 203  
minimum . . . . . 159, 160  
minimumLines . . . . . 163  
minimumWidth . . . . . 204  
minPrefix . . . . . 167  
minSuffix . . . . . 167  
mirrored . . . . . 33, 131  
mirroredHorizontally . . . . . 130  
mirroredVertically . . . . . 131  
mode . . . . . 161, 165, 173, 174, 175, 179, 185, 186  
modificationDate . . . . . 162  
move . . . . . 36  
moveLayer . . . . . 17  
moveOnPageTo . . . . . 36  
moveOnSpreadTo . . . . . 36  
movePicture . . . . . 50  
MoveReferencePoint . . . . . 218

## N

name . . . . . 22, 26, 27, 32, 125, 198  
nameChangePolicy . . . . . 183  
nameFilter . . . . . 203  
new Cmyk . . . . . 118  
new PDFExportSettings() . . . . . 183  
new Rgb . . . . . 117  
new TextEndnotes . . . . . 191  
new TextFootnotes . . . . . 192  
new TransparencyStop() . . . . . 192  
newDocument . . . . . 13  
newDocument() . . . . . 13  
newName() . . . . . 199  
newPageNumber . . . . . 142  
note . . . . . 34, 72  
notesVisible . . . . . 70  
NoteType . . . . . 218  
numberFormat . . . . . 165, 191, 192  
numbering . . . . . 160  
numberingFormat . . . . . 142  
numberingType . . . . . 191, 192  
numberOfColumns . . . . . 141  
numberOfDigits . . . . . 183  
numberOfSizingPoints . . . . . 107  
numberOfSubPolygons . . . . . 107

## O

object . . . . . 71  
objectCount . . . . . 15, 19, 24, 27, 62  
objectOpacity . . . . . 41  
objectPage . . . . . 24  
objectPreferences . . . . . 14  
objects . . . . . 15, 19, 24, 62  
objects(ObjectType objectType = All) . . . . . 27  
objectType . . . . . 62  
ObjectType . . . . . 218  
offset . . . . . 171, 175, 182, 185, 186, 187, 193  
offsetOnSpreadX . . . . . 19  
offsetOnSpreadY . . . . . 19  
offsetX . . . . . 130  
offsetY . . . . . 130  
oldName . . . . . 199  
on . . . . . 165, 175  
opacity . . . . . 78, 117, 126, 128, 161, 169, 170, 171, 173, 178, 179, 180, 181, 182, 185, 186, 187, 192, 193  
open . . . . . 200, 203  
openDocument . . . . . 13  
OpenFileMode . . . . . 218  
openType . . . . . 85, 127, 129  
OpenType() . . . . . 176  
OpenTypeNumerals . . . . . 218  
OpenTypePlacement . . . . . 218  
OpenTypePositionalForms . . . . . 218  
opi . . . . . 184

opticalAlignment . . . . . 91, 129  
OpticalAlignment . . . . . 218  
optimizePicture . . . . . 50  
OptimizePictureSettings() . . . . . 178  
optimum . . . . . 159, 160  
orderBy . . . . . 143  
orderedObjects . . . . . 27  
orderNumber . . . . . 143  
outerGlow . . . . . 37, 132  
OuterGlow() . . . . . 178  
outerMargin . . . . . 141  
outFilePath . . . . . 168, 178  
outlineOptions . . . . . 78, 127  
OutlineOptions . . . . . 179  
outlineOptionsData . . . . . 128  
outputPath . . . . . 184  
overwriteExistingFiles . . . . . 180  
owner . . . . . 64

## P

package . . . . . 17  
PackageSettings() . . . . . 179  
page . . . . . 34  
pageAt . . . . . 24  
pageBoundingBox . . . . . 34  
pageCount . . . . . 24  
pageGeometryPoints . . . . . 38, 40  
pageHeight . . . . . 15  
PageInsertPosition . . . . . 218  
pageMode . . . . . 13  
PageMode . . . . . 218  
pageNumber . . . . . 176  
pageNumbering . . . . . 183  
PageNumberingFormat . . . . . 218  
pageOrder . . . . . 140  
PageOrder . . . . . 218  
pageOrientation . . . . . 140  
PageOrientation . . . . . 218  
pagePreferences . . . . . 14  
pageSelection . . . . . 183  
PageSettings() . . . . . 141  
pageSize . . . . . 140  
PageSize . . . . . 218  
pagesPerJob . . . . . 183  
pagesRangeType . . . . . 183  
pageType . . . . . 22, 140  
PageType . . . . . 218  
pageWidth . . . . . 15  
pagex . . . . . 32  
pagey . . . . . 32  
ParagraphAlignment . . . . . 218  
paragraphAtCharacterPosition . . . . . 97  
paragraphBackground . . . . . 92, 129  
ParagraphBackground . . . . . 180  
paragraphFrame . . . . . 92, 129  
ParagraphFrame . . . . . 181  
paragraphIndent . . . . . 90, 129  
paragraphLeftIndent . . . . . 90, 129  
paragraphLineSpacing . . . . . 89, 129  
paragraphRightIndent . . . . . 90, 129  
ParagraphRule . . . . . 182  
ParagraphRule() . . . . . 182  
paragraphRuleAbove . . . . . 92, 129  
paragraphRuleBelow . . . . . 92, 129  
ParagraphRuleLengthType . . . . . 218  
paragraphs . . . . . 97  
paragraphSpaceMaximum . . . . . 94, 130  
paragraphStyleSheet . . . . . 93, 153  
paragraphStyleSheetByName . . . . . 93, 153  
paragraphTabs . . . . . 90, 129  
parentId . . . . . 207  
path . . . . . 198  
pathAt . . . . . 144  
pdfColorConversion . . . . . 184  
PDFColorConversion

pdfCompatibility . . . . . 184  
PDFCompatibility . . . . . 218  
pdfCompressionMethod . . . . . 184  
PDFCompressionMethod . . . . . 218  
pdfCompressionQuality . . . . . 184  
pdfCompressText . . . . . 184  
pdfCreateBookmarks . . . . . 184  
pdfCreateLayers . . . . . 184  
pdfOptimize . . . . . 184  
pdfTargetProfile . . . . . 184  
pdfWarningScale . . . . . 184  
pdfWarningScaleAbove . . . . . 184  
pdfWarningScaleBelow . . . . . 184  
pdfWarningTextOverflow . . . . . 184  
pdfxAllowEmbeddedICCProfiles . . . . . 184  
pdfxGrayProfile . . . . . 184  
pdfxOutputIntent . . . . . 184  
phone . . . . . 143  
physicalNumber . . . . . 19  
PictureFormat . . . . . 218  
PictureHorizontalPosition . . . . . 218  
pictureIndividualAliasContent . . . . . 132  
pictureStyleSheet . . . . . 149  
PictureVerticalPosition . . . . . 218  
placeMaximumColumnCount . . . . . 174  
placeStartColumn . . . . . 174  
plainText . . . . . 45, 70  
Point . . . . . 106  
PointList . . . . . 107  
PointList() . . . . . 107  
PolygonPointFlag . . . . . 218  
position . . . . . 117, 158, 166, 176, 190, 192  
positionCorrection . . . . . 73  
postfix . . . . . 165, 168, 178, 191, 192  
prefix . . . . . 142, 165, 168, 178, 191, 192  
preflightDocument . . . . . 150  
preflightPreferences . . . . . 14  
PreflightPreferences . . . . . 150  
PreflightPreferences() . . . . . 150  
preview . . . . . 20, 35, 50  
previewQuality . . . . . 150  
printable . . . . . 27, 33, 131  
PrintColorSeparation . . . . . 219  
printDeviceType . . . . . 143  
PrintDeviceType . . . . . 219  
printEmptyPages . . . . . 183  
PrintFontEmbedding . . . . . 219  
PrintHorizontalAlignment  
printImage . . . . . 143  
PrintImageQuality . . . . . 219  
PrintImageType . . . . . 219  
PrintLayersRange . . . . . 219  
printMaterial . . . . . 143  
PrintMaterialType . . . . . 219  
PrintNameChangePolicy . . . . . 219  
PrintOPI . . . . . 219  
PrintPageNumbering . . . . . 219  
PrintPagesRangeType . . . . . 219  
printResolution . . . . . 143  
PrintResolutionType . . . . . 219  
printSeparately . . . . . 183  
printSpellCorrection . . . . . 183  
printSpreads . . . . . 183  
PrintVerticalAlignment . . . . . 219  
progressBarFinish . . . . . 204  
progressBarLength . . . . . 204  
progressBarPosition . . . . . 204  
progressBarReset . . . . . 204  
progressBarSubText . . . . . 204  
progressBarText . . . . . 204  
ProgressReporter . . . . . 204  
ProgressReporterType  
protected . . . . . 33

## Index

### Q

grid . . . . . 49  
quality . . . . . 167  
quickPictureView . . . . . 149  
quickTextView . . . . . 149

### R

range . . . . . 158  
rangeType . . . . . 188, 194  
Read/Write . . . . . 184  
readLine . . . . . 200  
red . . . . . 117  
ReferenceAnchorNumberingType . . . . . 219  
reflection . . . . . 37, 132  
Reflection() . . . . . 185  
refresh . . . . . 31  
registrationMarks . . . . . 183  
reject . . . . . 206  
rejectLabel . . . . . 203  
remove . . . 43, 47, 64, 65, 127, 129, 130, 131, 132, 133, 135, 136, 200  
remove() . . . . . 21, 23, 26, 27, 35, 53, 63, 115, 125  
removeAll() . . . . . 144  
removeAllTabulators . . . . . 190  
removeContent() . . . . . 58  
removeDir . . . . . 198  
removeDocumentPages . . . . . 16  
removeGuide . . . . . 25  
removeOption . . . . . 203  
removePath . . . . . 143, 144  
removeRange . . . . . 74  
removeStopAt . . . . . 116, 193  
removeStyleSheets . . . . . 74  
removeTabulator . . . . . 190  
rename . . . . . 199  
renameDir . . . . . 199  
resizable . . . . . 204  
resolutionX . . . . . 48  
resolutionY . . . . . 48  
responsible . . . . . 143  
restartType . . . . . 175  
returnPressedMethod . . . . . 210  
rgb . . . . . 115  
right . . . . . 186  
rightBorder . . . . . 181  
rightColumns . . . . . 133  
rightColumnsData . . . . . 133  
rightIndent . . . . . 44, 57, 136, 169, 170, 171, 180, 181, 182  
rightMargin . . . . . 141  
rightSeparator . . . . . 136  
rightSeparatorData . . . . . 136  
rightTextIndent . . . . . 132  
rotation . . . . . 33, 49, 130, 131, 136  
row . . . . . 52, 58  
rowAt . . . . . 163  
rowByFieldContent . . . . . 163  
RowColumnStyle . . . . . 134  
RowColumnStyle() . . . . . 134  
RowColumnStyleAlt . . . . . 134  
rowCount . . . . . 52, 162  
rowSpan . . . . . 57  
RubyPosition . . . . . 219  
runaround . . . . . 34, 132  
Runaround . . . . . 185  
Runaround() . . . . . 185  
RunaroundMode . . . . . 219  
RunaroundShapeType . . . . . 219  
runaroundType . . . . . 163

### S

saturation . . . . . 118  
save . . . . . 17  
saveLargePreview . . . . . 150  
saveSmallPreview . . . . . 150

saveTo . . . . . 191  
saveToFile . . . . . 50  
scale . . . . . 143, 163, 185, 186  
scalePicture . . . . . 50  
ScalePictureType . . . . . 219  
screenAngle . . . . . 115  
ScreenAngle . . . . . 219  
searchPaths . . . . . 14  
secondCount . . . . . 134  
secondStyleSheet . . . . . 134  
secondStyleSheetByName . . . . . 134  
selected . . . . . 33  
selectedFiles . . . . . 203  
selection . . . . . 102  
selectionChangedMethod . . . . . 210, 211  
selections . . . . . 102  
separator . . . . . 53, 133  
separatorHorizontalOffset . . . . . 191, 192  
separatorLineBrush . . . . . 191, 192  
separatorLineLength . . . . . 191, 192  
separatorLineStyle . . . . . 191, 192  
separatorLineWidth . . . . . 191, 192  
separatorVerticalOffset . . . . . 191, 192  
separatorVisible . . . . . 191, 192  
setAction . . . . . 176  
setAliasPage . . . . . 21  
setAlternatingColumns . . . . . 133  
setAlternatingRows . . . . . 133  
setAttributes . . . . . 36, 46, 50, 55, 56, 59, 61, 75  
setAutomaticCharacterSpacing . . . . . 90  
setAutomaticCharacterWidth . . . . . 90  
setAutomaticWordSpacing . . . . . 89  
setBaselineGridType . . . . . 89  
setBlock . . . . . 186  
setBottomSeparator . . . . . 136  
setBrush . . . . . 77  
setBullet . . . . . 165  
setChapterName . . . . . 96  
setCharacterBackground . . . . . 86  
setCharacterBaselineOrientation . . . . . 83  
setCharacterBaselineShift . . . . . 83  
setCharacterFrame . . . . . 86  
setCharacterFunction . . . . . 73  
setCharacterHeightScale . . . . . 81  
setCharacterKerning . . . . . 82  
setCharacterLigatures . . . . . 84  
setCharacterPosition . . . . . 80  
setCharacterRotation . . . . . 84  
setCharacterRule . . . . . 87  
setCharacterSpacing . . . . . 82  
setCharacterStyleSheet . . . . . 87  
setCharacterStyleSheetByName . . . . . 88  
setCharacterUpperLowerCase . . . . . 80  
setCharacterWidthScale . . . . . 81  
setClipping . . . . . 50  
setCount . . . . . 173  
setCustomProperty . . . . . 35  
setData . . . . . 172  
setDropCaps . . . . . 91  
setDynamicLines . . . . . 164  
setEnumerations . . . . . 91  
setFillStyle . . . . . 53, 55, 56, 58  
setFixLines . . . . . 164  
setFont . . . . . 75  
setFontSize . . . . . 76  
setFooterRows . . . . . 133  
setFrameStyle . . . . . 61  
setHeaderRows . . . . . 133  
setHorizontalSeparator . . . . . 135  
setHyphenation . . . . . 91  
setImageData . . . . . 50  
setIndents . . . . . 59  
setItemKey . . . . . 208  
setJustifiedSpaceBetweenLinesMaximum . . . . . 94  
setKeepParagraphTogether . . . . . 93  
setLanguage . . . . . 77

setLanguageByName . . . . . 76  
setLanguageInfo . . . . . 148  
setLayoutColumns . . . . . 95  
setLayoutFootnotes . . . . . 95  
setLayoutLineJustification . . . . . 94  
setLayoutLineOrientation . . . . . 94  
setLayoutStyleSheet . . . . . 95  
setLayoutStyleSheetByName . . . . . 96  
setLeftColumns . . . . . 133  
setLeftSeparator . . . . . 136  
setLink . . . . . 86  
setMarginAt . . . . . 173  
setObject . . . . . 71  
setOpacity . . . . . 78  
setOpenType . . . . . 85, 86  
setOpticalAlignment . . . . . 91  
setOutlineOptions . . . . . 79  
setParagraphBackground . . . . . 92  
setParagraphFrame . . . . . 92  
setParagraphIndent . . . . . 90  
setParagraphLeftIndent . . . . . 90  
setParagraphLineSpacing . . . . . 89  
setParagraphRightIndent . . . . . 90  
setParagraphRuleAbove . . . . . 92  
setParagraphRuleBelow . . . . . 92  
setParagraphSpaceMaximum . . . . . 94  
setParagraphStyleSheet . . . . . 93  
setParagraphStyleSheetByName . . . . . 93  
setParagraphTabs . . . . . 90, 91  
setPoint . . . . . 108  
setPositionCorrection . . . . . 73  
setQrCodeToImage . . . . . 13  
setRange . . . . . 102  
setRightColumns . . . . . 133  
setRightSeparator . . . . . 136  
setShade . . . . . 78  
setSpaceToNextLayout . . . . . 95  
setSpaceToNextParagraph . . . . . 89  
setSpaceToPreviousLayout . . . . . 95  
setSpaceToPreviousParagraph . . . . . 89  
setStopAt . . . . . 116, 193  
setStoryEndnotes . . . . . 96  
setStoryFootnotes . . . . . 97  
setStoryLineCounterData . . . . . 97  
setStrikethrough . . . . . 79  
setSuppressLineNumbering . . . . . 92  
setTablesAbbreviationEntry . . . . . 85  
setTablesBibliographyEntry . . . . . 85  
setTablesContentsEntry . . . . . 85  
setTablesIndexEntry . . . . . 84  
setTablesPictureEntry . . . . . 85  
setTablesRunningTitleEntry . . . . . 85  
setTabulator . . . . . 190  
setTextAlignment . . . . . 88  
setTopSeparator . . . . . 136  
setUnderline . . . . . 79  
setUserProperty . . . . . 17, 20, 25, 34  
setUserXMLProperty . . . . . 18, 20, 25, 35  
setVerticalSeparator . . . . . 135  
setWidowsOrphans . . . . . 93  
setWidthAt . . . . . 173  
setWritingDirection . . . . . 91  
shade . . . 54, 56, 57, 61, 77, 78, 117, 126, 128, 161, 169, 170, 171, 173, 178, 179, 180, 181, 182, 186, 187, 193  
Shadow() . . . . . 186  
shapeType . . . . . 185  
show . . . . . 207, 209, 210, 211  
showAliasObjects . . . . . 149  
showBaselineGrid . . . . . 149  
showGuideObjects . . . . . 149  
showGuides . . . . . 149  
showInvisibles . . . . . 152  
showMarginsAndColumns . . . . . 149  
showRuler . . . . . 149  
showSmartGuides . . . . . 149  
showTextRuler . . . . . 149, 152

## Index

singleQuotesType . . . . . 148  
SingleQuotesType . . . . . 219  
size . . . . . 166, 185, 199  
skew . . . . . 33, 49, 130, 131  
skipFirstCount . . . . . 134  
skipLastCount . . . . . 134  
smallCapsCharacterHeight . . . . . 152  
smallCapsCharacterWidth . . . . . 152  
smallestWord . . . . . 167  
smallPreviewsMaximumSize . . . . . 150  
snapDistance . . . . . 149  
spaceToNextLayout . . . . . 95, 130  
spaceToNextParagraph . . . . . 89, 129  
spaceToPreviousLayout . . . . . 94, 95, 130  
spaceToPreviousParagraph . . . . . 89, 129  
spacing . . . . . 160  
spans . . . . . 97  
SpecialCharacters . . . . . 219  
SpellLang . . . . . 187  
spellModuleIdent . . . . . 148  
spread . . . . . 19, 34  
spreadBoundingBox . . . . . 34  
spreadGeometryPoints . . . . . 38, 40  
spreadx . . . . . 33  
spready . . . . . 33  
start . . . . . 160  
startColumn . . . . . 166  
startNumber . . . . . 191, 192  
startsWith . . . . . 175  
startWith . . . . . 165  
stateChangedMethod . . . . . 207  
step . . . . . 175  
Stop . . . . . 117  
stopAt . . . . . 116, 193  
stopCount . . . . . 116, 193  
story . . . . . 97  
storyEndnotes . . . . . 96  
storyFootnotes . . . . . 96, 97  
storyLineCounterData . . . . . 97  
strikethrough . . . . . 79, 126, 128  
strikethroughLineWidth . . . . . 151  
strikethroughOffsetHorizontalLayout . . . . . 151  
strikethroughOffsetVerticalLayout . . . . . 151  
StrikethroughOptions . . . . . 187  
String fillCharacter . . . . . 190  
styleSheet . . . . . 37, 49, 52, 55, 56, 58, 134  
styleSheetByName . . . . . 37, 49, 52, 55, 56, 58, 134  
StyleSheetFamily . . . . . 219  
styleSheets . . . . . 14  
subdirFlagAt . . . . . 144  
subscriptCharacterHeight . . . . . 151  
subscriptCharacterWidth . . . . . 152  
subscriptOffset . . . . . 151  
superiorCharacterHeight . . . . . 152  
superiorCharacterWidth . . . . . 152  
superscriptCharacterHeight . . . . . 151  
superscriptCharacterWidth . . . . . 151  
superscriptOffset . . . . . 151  
supressEmptyParagraphs . . . . . 175  
supressLineNumbering . . . . . 92, 129

## T

tableHeight . . . . . 52  
TableLineSeparator . . . . . 219  
TablesAbbreviation . . . . . 188  
tablesAbbreviationEntry . . . . . 85, 127, 128  
TablesBibliography . . . . . 188  
tablesBibliographyEntry . . . . . 85, 127, 128  
TablesContents . . . . . 188  
tablesContentsEntry . . . . . 84, 85, 127, 128  
TableSeparatorPosition . . . . . 219  
TablesIndex . . . . . 189  
tablesIndexEntry . . . . . 84, 127, 128  
TablesPicture(bool active, String alternativeText) . . . 189  
tablesPictureEntry . . . . . 85, 127, 128

TablesRunningTitle(bool active, String alternativeText, int level) . . . . . 189  
tablesRunningTitleEntry . . . . . 85, 127, 128  
tableWidth . . . . . 52  
tableWidthType . . . . . 52  
tabulator . . . . . 165  
Tabulator . . . . . 190  
Tabulator (String position, TabulatorAlignment alignment, . . . . . 190  
Tabulators . . . . . 190  
taggedPdfOptions . . . . . 34  
tagType . . . . . 191  
templateStyleSheet . . . . . 125  
testDirectoryFilter . . . . . 203  
testOption . . . . . 203  
text . . . . . 51, 102, 162, 176, 194, 206, 207, 209  
textAlignment . . . . . 88, 129  
textChain . . . . . 42, 64, 102  
textChains . . . . . 15  
textChangedMethod . . . . . 210, 211  
textCursor . . . . . 14  
textEditedMethod . . . . . 210  
TextFile . . . . . 200  
textLinesAutomaticSpacing . . . . . 152  
textLinesMethod . . . . . 152  
textOffset . . . . . 191, 192  
textPreferences . . . . . 14  
textRange . . . . . 45  
TextRange . . . . . 101  
thesaurusModuleIdent . . . . . 148  
title . . . . . 198  
toggleColumn . . . . . 15  
tolerance . . . . . 151  
top . . . . . 186  
toPage . . . . . 183  
topAnchor . . . . . 162  
topBorder . . . . . 181  
topIndent . . . . . 44, 57, 136, 162, 169, 170, 180, 181  
topLevel . . . . . 33  
topMargin . . . . . 140  
topSeparator . . . . . 136  
topSeparatorData . . . . . 136  
topTextIndent . . . . . 132  
transparency . . . . . 37, 132  
type . . . . . 32, 116, 163, 165, 166, 170, 173, 176, 187, 190, 193, 194  
typeString . . . . . 32  
typographicQuotationMarks . . . . . 152

## U

underline . . . . . 79, 126, 128  
underlineLineWidth . . . . . 151  
underlineOffsetHorizontalLayout . . . . . 151  
underlineOffsetVerticalLayout . . . . . 151  
UnderlineOptions . . . . . 193  
ungroup() . . . . . 62  
uniqueName . . . . . 114  
unzip . . . . . 200  
updateCustomPropertis . . . . . 18  
updateView() . . . . . 18  
useBleed . . . . . 183  
userProperty . . . . . 17, 20, 25, 34  
userScriptsPath . . . . . 9  
userXMLProperty . . . . . 17, 20, 25, 35  
useSubfolders . . . . . 180

## V

valid . . . . . 58, 162  
value . . . . . 118, 160, 170, 206, 207, 208, 210, 211  
var myObjGroup . . . . . 4  
var myRgb . . . . . 4  
var variableName1 . . . . . 4  
var variableName2 . . . . . 4  
var variableName3 . . . . . 4  
var variableName4 . . . . . 4

variable . . . . . 73  
verticalAlignment . . . . . 57, 136, 184, 185  
verticalCorrection . . . . . 171  
verticalMirrored . . . . . 49  
verticalOffset . . . . . 48  
verticalPasteOffset . . . . . 149  
verticalScale . . . . . 49, 130  
verticalSeparator . . . . . 135  
verticalSeparatorData . . . . . 135  
viewModel . . . . . 203  
viewZoomFactor . . . . . 149  
visible . . . . . 33, 61  
visualizeStyleSheets . . . . . 149, 153

## W

warning ( . . . . . 198  
warningForBwColorMode . . . . . 151  
warningForEffects . . . . . 151  
warningForGrayColorMode . . . . . 151  
warningForRgbColorMode . . . . . 151  
warningForTransparency . . . . . 151  
widowsOrphans . . . . . 92, 93, 129  
WidowsOrphans . . . . . 194  
width . . . . . 40, 48, 55, 109, 131, 166, 208, 210, 211  
widthAt . . . . . 173  
widthType . . . . . 55  
withFrame . . . . . 185  
workingTime . . . . . 143  
writeLine . . . . . 200  
writingDirection . . . . . 91, 129

## Y

yellow . . . . . 118

## Z

zip . . . . . 180, 199, 200



