Illustrator Scripting Guide

Release 0.0.1

Sep 08, 2020

Introduction

CHAPTER 1

What is Scripting?

A script is a series of commands that tells Illustrator to perform one or more tasks. These tasks can be simple, affecting only one object in the current document, or complex, affecting objects in all your Illustrator documents.

The tasks might even involve other applications, like word processors, spreadsheets, and database management programs.

For the most part, the building blocks of scripting correspond to the Illustrator tools, menus, panels, and dialog boxes with which you are already an expert. If you know what you want Illustrator to do, you can write a script to do it.

1.1 Why use scripting?

Graphic design is a field characterized by creativity, but aspects of the work are anything but creative. In fact, you probably notice that the time you spend placing and replacing images, correcting errors in text, and preparing files for printing at an image-setting service provider often reduces the time you have available for doing creative work.

With a small investment of time and effort, you can learn to write short, simple scripts that perform repetitive tasks for you. As your scripting skills grow, you can move on to more complex scripts.

Scripting also can enhance your creativity, by quickly performing tasks you might not have time to try. For example, you could write a script to systematically create a series of objects, modifying the new objects' position, stroke, and fill properties along the way. You also could write a script that accesses built-in transformation matrix functions to stretch, scale, and distort a series of objects. Without scripting, you would likely miss out on the creative potential of such labor-intensive techniques.

1.2 What about actions?

Both actions and scripts are ways of automating repetitive tasks, but they work very differently:

- Actions use a program's user interface to do their work. As an action runs, menu choices are executed, objects are selected, and recorded paths are created.

  Scripts do not use a program's user interface to perform tasks, and scripts can execute faster than actions.

- Actions have very limited facilities for getting and responding to information.

  You cannot add conditional logic to an action; therefore, actions cannot make decisions based on the current situation, like changing the stroke type of rectangles but not ellipses.

  Scripts can get information and make decisions and calculations based on the information they receive from Illustrator.

- A script can execute an action, but actions cannot execute scripts.

CHAPTER 2

Changelog

What's new and changed for scripting?

2.1 Illustrator 24.0 (CC 2020) <>

- Added: *Document.getPageItemFromUuid()*
- Added: *PageItem.uuid*

2.2 Illustrator XX.X (CC 2017) <>

- Added: *Application.getIsFileOpen()*

2.3 Illustrator XX.X (CC) <>

- ?

CHAPTER 3

Scripting language support in Adobe Illustrator CC

Illustrator scripting supports VBScript and JavaScript scripts for Windows, and AppleScript and JavaScript scripts for Mac OS.

3.1 Script file extensions

For a file to be recognized by Adobe Illustrator CC 2017 as a valid script file, the file must have the correct file name extension:

Script Type	File type (extension)	Platforms
AppleScript	compiled script (.scpt) OSAS file (no extension)	Mac OS
JavaScript or ExtendScript	text (.js or .jsx)	Windows Mac OS
VBScript	text (.vbs)	Windows

3.2 JavaScript development options

You can use the ExtendScript Toolkit to create JavaScript scripts explicitly for Illustrator, or you can use Adobe Extension Builder and the Creative Cloud SDK to develop extensions in ActionScript.

Extensions are Flash-based (SWF) and can potentially work in a variety of Creative Cloud applications.

3.2.1 Developing a CC extension using ActionScript

Creative Cloud applications have an extensibility infrastructure that allows developers to extend the capabilities of the applications; the infrastructure is based on Flash/Flex technology, and each extension is delivered as compiled Flash (SWF) file.

Creative Cloud includes the Extension Manager to enable installation of extensions.

An example of an extension that ships with the point products is Adobe Kuler. Kuler has a consistent user interface across the different suite applications, but has different logic in each, adapted to the host application.

The user interface for an extension is written in ActionScript, using the Flex framework. An extension is typically accessed through its own menu item in the application's Extensions menu.

Adobe Extension Builder allows you to design the user interface interactively using the Design view of Flash Builder. The Creative Cloud SDK also allows you to develop all of the application logic for your extension in ActionScript; you can develop and debug your extension in the familiar Flash Builder environment.

To develop your application logic, we recommend using the ActionScript Wrapper Library (CSAWLib), which exposes the scripting DOM of each host application as an ActionScript library. This is tightly integrated with the Adobe Extension Builder environment, which includes wizards to help you build your extension's basic structure, and run and debug your code against suite applications such as Adobe InDesign, Photoshop and Illustrator.

The methods, properties, and behavior of the scripting DOM is as described in the JavaScript Scripting Reference for the host application.

For details of how to use Adobe Extension Builder and the wrapper libraries, see the Creative Cloud SDK documentation, which is accessible from within Adobe Extension Builder.

3.2.2 Scripting plug-ins

The CC JavaScript scripting interface allows for limited scripting for plug-ins. A plug-in can define a command, with an event and notifier, and a handler that performs some action. A JavaScript script can then use the `app.sendScriptMessage()` method to send parameters to that plug-in-defined command, and receive a plug-in-defined response.

For example, the Adobe Custom Workspace plug-in defines a command "Switch Workspace". A script can invoke this command with the following code

```
result = app.sendScriptMessage (
  "Adobe Custom Workspace",
  "Switch Workspace",
  '<workspace="Essentials" >'
);
```

In this case, the value that the plug-in returns is the string

```
"<error= errNo>".
```

3.2.3 ExtendScript features

If you write Illustrator-specific scripts that use the Illustrator JavaScript DOM directly, you will create ExtendScript files, which are distinguished by the .jsx extension.

Giving your JavaScript files a .jsx extension (rather than the standard .js extension for a JavaScript file) allows you to take advantage of ExtendScript features and tools.

ExtendScript offers all standard JavaScript features, plus a development and debugging environment, the ExtendScript Toolkit (ESTK).

The ESTK is installed with all scriptable Adobe applications, and is the default editor for JSX files. The ESTK includes an Object Model Viewer that contains complete documentation of the methods and properties of JavaScript objects. For information on accessing the ESTK and the Object Model Viewer see *Viewing the object model*.

ExtendScript also provides various tools and utilities, including the following:

- A localization utility
- Tools that allow you to combine scripts and direct them to particular applications
- Platform-independent file and folder representation
- Tools for building user interfaces to your scripts
- A messaging framework that allows you to send and receive scripts and data among scripting-enabled Adobe applications

All of these features are available whether you use the DOM directly with a JSX file, or indirectly through the ActionScript wrapper library and Adobe Extension Builder. For details of these and other features, see JavaScript Tools Guide.

CHAPTER 4

Viewing Sample Scripts

Adobe provides sample scripts for many objects, properties, and methods in the lllustrator CC DOM. You can view script samples in two locations:

- In the `/Scripting/Sample Scripts/` folder in your lllustrator CC installation directory
- In this document :)

CHAPTER 5

Viewing the object model

Each of the supported scripting languages provides a facility for viewing the scripting objects defined by Illustrator, with reference details.

5.1 Viewing the JavaScript object model

To view the JavaScript object model for Illustrator, follow these steps:

In a default Adobe installation, the ESTK is in the following location:

Windows	`\system drive\Program Files\Adobe\Adobe Utilities CC\ExtendScript Toolkit CC`
Mac OS	`\system drive\Applications\Utilities\Adobe Utilities CC\ExtendScript Toolkit CC`

1. Start the ESTK.
2. In the ESTK, choose Help > Object Model Viewer.
3. In the Object Model Viewer window, select Adobe lllustrator CC Type Library from the Browser drop-down list.

Several extended sample scripts are in the `/Scripting/Sample Scripts/` folder in your Illustrator CC installation directory.

You also can view script samples and information about individual classes, objects, properties, methods, and parameters in *Illustrator Scripting Reference: Javascript*.

5.2 Viewing the AppleScript object model

Apple provides a Script Editor with all Mac OS systems. You can use Script Editor to view the AppleScript dictionary that describes Illustrator objects and commands.

For details of how to use Script Editor, see Script Editor Help.

Note: In a default Mac OS installation, Script Editor is in `Applications/AppleScript/Script Editor`

If you cannot find the Script Editor application, you must reinstall it from your Mac OS system CD.

---

1. Start Script Editor.
2. Choose File > Open Dictionary. Script Editor displays an Open Dictionary dialog.
3. In the Open Dictionary dialog, find and select Adobe lllustrator CC, and click Open.

Script Editor displays a list of the Illustrator objects and commands, which include the properties and elements associated with each object and the parameters for each command.

Several extended sample scripts are in the `/Scripting/Sample Scripts/` folder in your Illustrator CC installation directory.

You also can view script samples and information about individual classes, objects, properties, methods, and parameters in *Illustrator Scripting Reference: Applescript*.

5.3 Viewing the VBScript object model

VBScript provides a type library you can use to view Illustrator object properties and methods. This procedure explains how to view the type library through any Microsoft Office program. Your VBScript editor probably provides access to the library. For information see your editor's Help.

1. In any Microsoft Office application, choose Tools > Macro > Visual Basic Editor.
2. In the Visual Basic Editor, choose Tools > References.
3. In the dialog that appears, select the check box for Adobe lllustrator CC Type Library, and click OK.
4. Choose View > Object Browser, to display the Object Browser window.
5. Choose "Illustrator" from the list of open libraries in the top-left pull-down menu of the Object Browser window.

Several extended sample scripts are in the `/Scripting/Sample Scripts/` folder in your lllustrator CC installation directory.

You also can view script samples and information about individual classes, objects, properties, methods, and parameters in *Illustrator Scripting Reference: VBScript*.

CHAPTER 6

Executing Scripts

The Illustrator interface includes a Scripts menu (File > Scripts) that provides quick and easy access to your scripts.

Scripts can be listed directly as menu items that run when you select them. See *Installing scripts in the Scripts menu*.

You can navigate from the menu to any script in your file system and then run the script. See *Executing scripts from the Other Scripts menu item*.

You also can have JavaScript scripts with a .jsx extension start automatically when you launch the application. For information, see *Startup scripts (.jsx scripts only)*.

6.1 Installing scripts in the Scripts menu

To include a script in the Scripts menu (File > Scripts), save the script in the Scripts folder, located in the `/lllustrator CC/Presets` folder in your lllustrator CC installation directory.

The script's filename, minus the file extension, appears in the Scripts menu.

Scripts that you add to the Scripts folder while Illustrator is running do not appear in the Scripts menu until the next time you launch Illustrator.

Any number of scripts can be installed in the Scripts menu. If you have many scripts, use subfolders in the Scripts folder to help organize the scripts in the Scripts menu.

Each subfolder is displayed as a separate submenu containing the scripts in that subfolder.

6.2 Executing scripts from the Other Scripts menu item

The Other Scripts item at the end of the Scripts menu `(File > Scripts > Other Scripts)` allows you to execute scripts that are not installed in the Scripts folder.

Selecting Other Scripts displays a Browse dialog, which you use to navigate to a script file. When you select the file, the script is executed.

Only files that are of one of the supported file types are displayed in the browse dialog. For details, see *Scripting language support in Adobe Illustrator CC*.

6.3 Startup scripts (.jsx scripts only)

JavaScript scripts with a .jsx file extension can be installed in one of two folders, so the scripts run automatically when you launch Illustrator and each time you run a script.

The folders are:

- An application-specific startup scripts folder, which contains scripts for IllustratorCC
- A general startup scripts folder, which contains scripts that run automatically when you start any Creative Cloud application

6.3.1 Application-specific startup scripts folder

You must place application-specific startup scripts in a folder named **Startup Scripts**, which you create in the Illustrator installation directory.

For example, when IllustratorCC is installed to its default location, you would create the Startup Scripts folder at the following location:

Windows	`C:\Program Files\Adobe\Adobe lllustratorCC\Startup Scripts\`
Mac OS	`/Applications/Adobe lllustrator CC/Startup Scripts/`

JavaScript scripts with a .jsx extension placed in the Startup Scripts folder run automatically when:

- The application is launched.
- Any JavaScript file is selected from the Scripts menu (File > Scripts).

6.3.2 General startup scripts folder

The general startup scripts folder contains scripts that run automatically when you start any Creative Cloud application.

You create the folder in the following location:

Windows	`/Program Files/Common Files/Adobe/Startup Scripts CC/Illustrator`
Mac OS	`/Library/Application Support/Adobe/Startup Scripts CC/Illustrator`

If a script in the general startup folder is meant to be executed only by Illustrator, the script must include the ExtendScript #target directive (`#target illustrator`) or code like the following

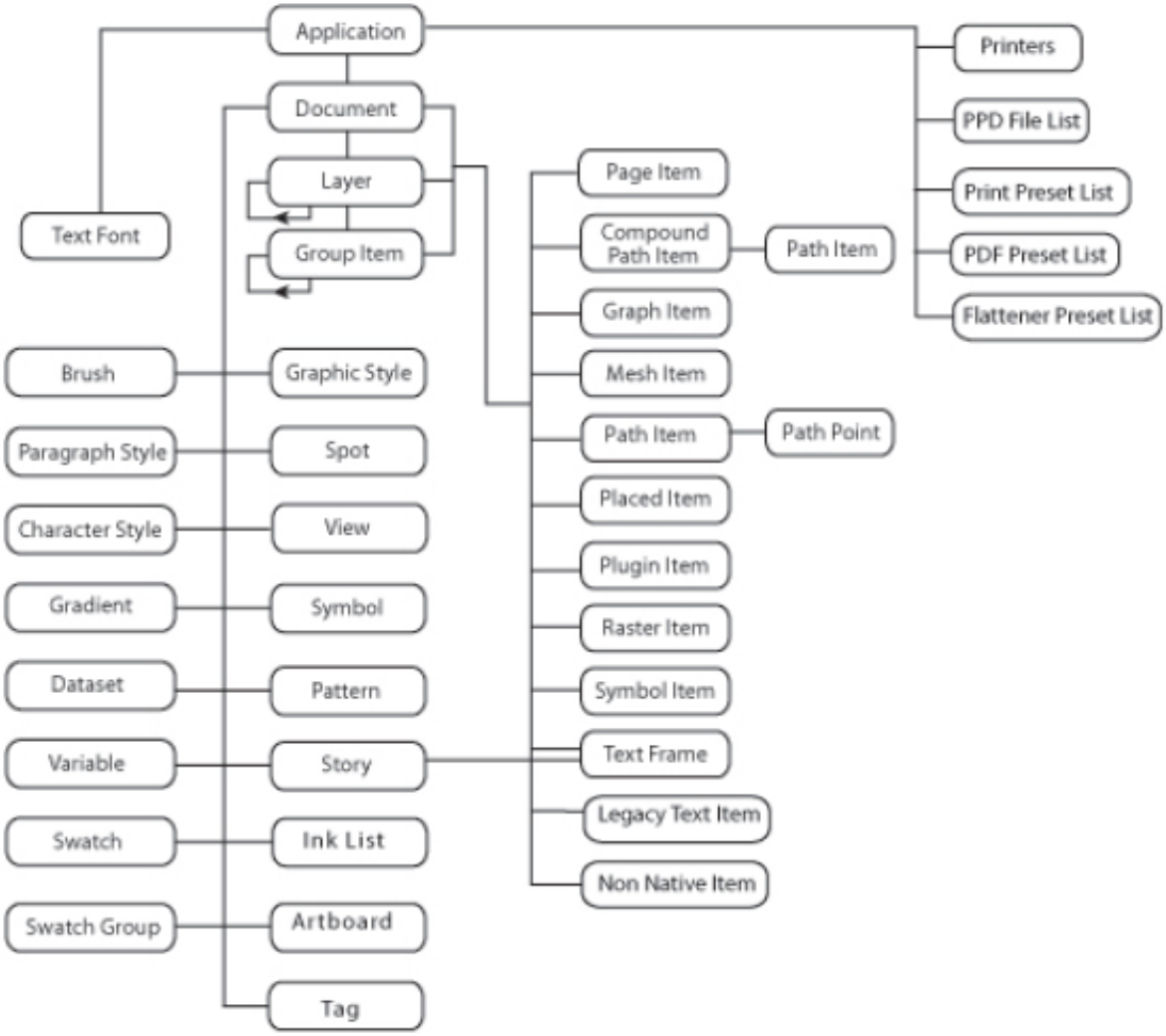
```
if (BridgeTalk.appName == "illustrator") {
  // continue executing script
}
```

CHAPTER 7

The Illustrator Scripting Object Model

A good understanding of the Illustrator object model will improve your scripting abilities. The following figure shows the containment hierarchy of the object model, starting with the application object.

Note that the layer and group item classes can contain nested objects of the same class which can, in turn, contain additional nested objects.



In addition to this application-specific object model, JavaScript provides certain utility objects, such as the File and Folder objects, which give you operating-system-independent access to the file system.

For details, see JavaScript Tools Guide.

CHAPTER 8

Object Naming Conventions

There is one object model for the Illustrator scripting interface, but actual object names vary slightly in the different scripting languages:

- AppleScript names are lower case, and individual words are separated by a space; for example: `graphic style`
- VBScript names are capitalized, and additional words in the name are indicated by uppercase initial letters; for example: `GraphicStyle`
- JavaScript names begin with lowercase letters, and additional words in the name are indicated by uppercase initial letters; for example: `graphicStyle`

This chapter uses generic object and property names, but you can easily apply these conventions to determine the corresponding language-specific names.

Throughout this document, names of properties, methods, and object are in a monospaced font.

CHAPTER 9

Top-Level (containing) Objects

Use these objects to access global information about the Illustrator application or an individual document.

9.1 Application

The properties of the `application` object give your script access to global values, such as:

- User `preferences`, which a user sets interactively in the Illustrator application by using the Preferences dialog (Edit > Preferences).
- System information like installed fonts (the `text fonts` property) and printers (the `printer list` property).

Also, there are properties that provide application-specific information and higher-level information about any open documents:

- Application information like the installation `path`, `version`, and whether Illustrator is `visible`.
- The `current active` document; that is, the art canvas that is displayed and accepting user input.
- All open `documents`.

The `application` object's methods or commands allow your script to perform application-wide actions; for example:

- `Open` files
- `Undo` and `redo` transactions
- `Quit` Illustrator

9.2 Document

The `document` object, which your scripts can create or access through the `application` object, represents an art canvas or loaded Illustrator file.

The `document` object's properties give you access to the document's content; for example:

- The current `selection`, or art objects that the user selected in the document
- All contained art objects, called `page items`, that make up the artwork tree
- Art objects of particular types, like `symbols` and `text frames`
- All `layers` and the currently `active layer`

Document properties also tell you about the state of the document itself; for example:

- User settings for the document, such as `ruler units`
- Whether the document was `saved` since the last alteration of content
- The `path` of the associated file

The document object's methods allow your scripts to act on the document; for example:

- `Save` to an Illustrator file or `save as` the various supported file formats
- `Activate` or `close` a document
- `Print` the document. Your scripts can select a printer by referencing a `print options` object, or they can reference available printers through the application object's `printer list` property.

9.3 Layer

The `layer` object provides access to the contents, or artwork tree, of a specific layer.

You access the `layer` object through the `document` object.

The `layer` object properties provide access to, or information about, the layer, such as:

- Whether the layer is `visible` or `locked`.
- The layer's `opacity` (overall transparency) and `z order position` (position in the stacking order).
- Art-creation preferences for the layer, like `artwork knockout` and `blending mode`.

CHAPTER 10

The Artwork Tree

The content of an Illustrator document is called the artwork tree. Artwork is represented by the following objects:

- `compound path item`
- `graph item`
- `group item`
- `legacy text item`
- `mesh item`
- `non native item`
- `path item`
- `placed item`
- `plugin item`
- `raster item`
- `symbol item` (see *Dynamic Objects*)
- `text frame`

Your scripts can access and manipulate art objects through collections in the document and layer objects.

There are two types of art-object collections:

- Collection objects that correspond to each individual artwork object type, such as the `graph items` object or the `mesh items` object.
- The `page items` object, which includes art objects of all types.

Also, you can use the group `item` object to reference a grouped set of art items.

You can create new art objects using the `make` command (AppleScript) or add method of an artwork item collection object. For example, to create a new `path item` object:

AppleScript	`set myPathItem to make new path item in current document`
JavaScript	`var myPathItem = activeDocument.pathItems.add();`
VBScript	`Set myPathItem = appRef.ActiveDocument.PathItems.Add()`

The following artwork collections do not allow the creation of new objects using the `make` command or add method:

- `graph items` object
- `mesh items` object
- `plugin items` object
- `legacy text items` object

For details on creating objects of these types, see the Adobe lllustrator CC Scripting Reference for your language.

10.1 Art styles

Your script can apply a graphic style to artwork using the `graphic style` object. To apply a graphic style, use the `graphic styles` property of the `document` object to access the `apply to` method of the `graphic style` object.

Similarly, the `brush` object allows you to specify the brush to apply to artwork. You access any brush through the `brushes` collection object, which is a property of the `document` object.

10.2 Color objects

Your script can apply a color, pattern or gradient to a `path item` object, using the `fill color` or `stroke color` properties:

- Scripts can define new color swatches using the `make` command or add method of the `swatches` object. Your script also can create a new spot color, using the `make` command or `add` property of the `spots` object.
- You can define the attributes of an ink object using the `ink info` object, which is an `ink` object property. You access `ink` objects through the `ink list` property of the `document` object.

The following objects allow you to create colors within defined color spaces:

- The `RGB color` object, using the range 0.0 to 255.0 for the each of the three individual color values.
- The `CMYK color` object, using the percentage values 0.0 through 100.0 for each of the four individual color values.
- The `grayscale color` or `LAB color` objects, using the same range and number of values that you use in the Illustrator application.

CHAPTER 11

Text Objects

When you type content in an Illustrator document, the type automatically becomes a `text frame` object and, at the same time, a story object.

To observe this, open a new document in Illustrator and use the horizontal text tool to type some text, then use the vertical text tool to type more text.

Finally, create a rectangle and type some text inside it.

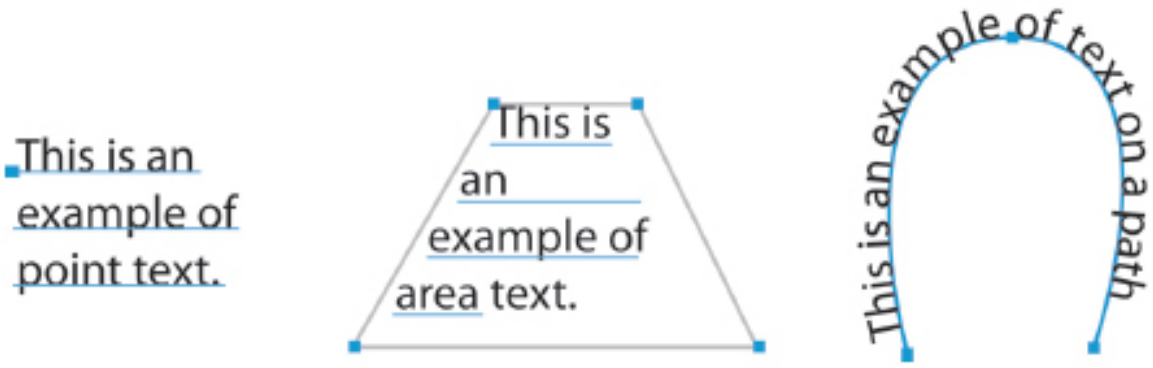
Now run the following JavaScript script

```
var myDoc = app.activeDocument;
alert("There are " + myDoc.textFrames.length + " text frames.");
alert("There are " + myDoc.stories.length + " stories.");
```

11.1 Text Frames

There are three types of text frames:

- point
- area
- path



To create a specific kind of text frame, use the `kind` property of the `text frames` object in AppleScript.

The JavaScript and VBScript `text frames` objects contain specific methods for creating area text frames and path text frames.

As in the Illustrator application, you can thread area or path text frames.

To thread existing text frames, use the `next frame` or `previous frame` property of the `text frame` object.

Threaded frames make a single `story` object.

For information on creating or threading text frames, see the chapter in this manual for your scripting language.

11.1.1 Text Geometry

While the three kinds of text frames have common characteristics, like `orientation`, each has type-specific qualities, as reflected in the `text frame` object's properties. For example:

- An area text frame can have rows and columns, which you access through the `row count` and `column count` properties.
- Path text has `start T` value and `end T` value properties that indicate where on the path the text begins and ends.
- Area and path text frames are associated with a text path object, which is specified using the `text frame` object's `text path` property. The text path defines the text frame's position and orientation (horizontal or vertical) on the artboard (while the `text frame` object's orientation property defines the `orientation` of text within the text frame). The `text path` property is not valid for point text, because point-text position and orientation are defined completely by the properties of the text frame itself.

Note: A text path is not the same as a path art item. Text paths are associated with path art items that can be accessed and manipulated to modify the appearance of the associated text frame.

11.2 Objects that represent text content

Within a text frame or story, the actual text content can be accessed as any of the following objects:

- `characters`

- words
- paragraphs
- lines

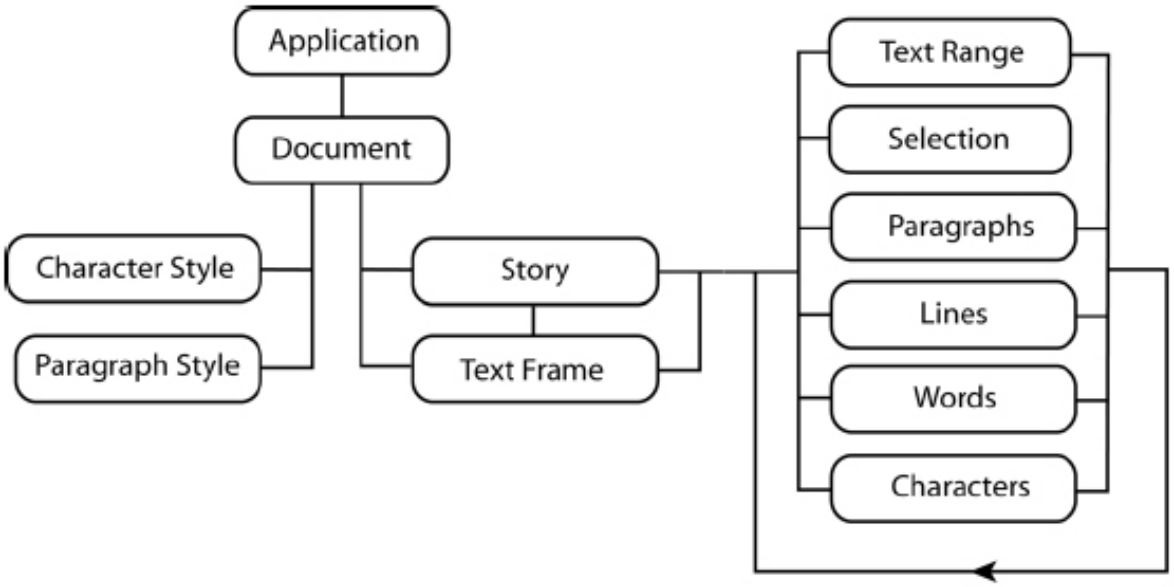
A line object is all the characters that fit on one line in a text frame or story object.

All text-art items have at least one line of text, defined as a line object.

Text art can have multiple text lines, if the text contains hard line breaks or its characters flow to a new line because they do not fit in the width of the text art.

Text objects are accessed and identified by collections within the text frame and story objects; for example

```
textFrame("My Text Frame").paragraphs
// or
story("My Story").paragraphs
```



Both text frame and story objects have insertion point and text selection properties.

The text frame object's properties also include the defining features of the text frame, such as:

- The frame width, height, and position
- Whether the frame is hidden or locked
- Whether the text is editable

Note: A line object cannot be created in a script. Your script can create character, paragraph, and word objects.

11.2.1 Text ranges

The various text objects within a text frame or story also are represented collectively by the text range object.

For example, a character is a text range with a length of 1, and a word is a text range that has a space before it.

You can set the content of a text range object by passing a string using the `contents` property.

11.3 Text styles

Text-style elements, like `font`, `capitalization`, and `justification`, are represented by `paragraph attribute` and `character attribute` objects.

These attribute objects are properties of the `paragraph style` and `character style` objects.

The `paragraph style` and `character style` objects have `apply to` and `remove` methods that allow your script to assign or remove attributes in a specific paragraph, character, or text range.

You can change the display properties of a text range by applying an appropriate style or providing local overrides of attributes at the text or paragraph levels:

- character style objects apply to sets of one or more characters. They control character features like `font`, `alignment`, `leading`, `language`, and `capitalization`, which are properties of the `character attribute` object.
- `paragraph style` objects apply to paragraphs. They control paragraph features like `first line indent`, `left indent`, and `right indent`, which are properties of the `paragraph attribute` object.

CHAPTER 12

Dynamic Objects

By creating dynamic objects, you can create data-driven graphics.

In the Illustrator application, you use the Variables panel to create or edit variables like graph data, linked file, text string, and visibility, or variables whose type is not specified.

In scripting, you use the `variable` object to represent this type of variable.

The `variable` object's `kind` property indicates the type of dynamic data that a `variable` object holds. `Variable` objects are document-level objects; you create them in a `document` object.

Note: Do not confuse variable objects with scripting variables. For details on Illustrator variables, dynamic objects, and data-driven graphics, see Illustrator Help.

Datasets, which collect variables and their associated dynamic data into one object, are represented in scripting by the `dataset` object.

The `dataset` object provides methods to update and delete `dataset` objects in your scripts.

CHAPTER 13

Symbols

In Illustrator, symbols are art items that are stored in the Symbols panel.

Your scripts can create, delete, and duplicate `symbol` objects.

When you create `symbol` objects in your script, Illustrator adds them to the Symbols panel for the target document.

A `symbol item` is an instance of a `symbol` object in a document. Each `symbol item` is linked to its symbol definition, so changing the definition of a symbol updates all instances of the symbol.

Your script can create, delete, and duplicate symbol items. Symbol items are Illustrator art items; therefore, they can be treated in the same way as other art items or page items.

You can rotate, resize, select, lock, hide, and perform other operations on symbol items.

CHAPTER 14

Transformations

The `matrix` object provides access to the power of geometric-transformation matrices.

Transformation matrices in Illustrator store the settings of an operation that scales, rotates, or moves (translates) an object on a page.

There are advantages to using matrices:

- By storing transformation values in a `matrix` object, you can use the values repeatedly on different objects in your script.
- By concatenating rotation, translation, and/or scaling matrices and applying the resulting matrix, you can perform many geometric transformations with only one script statement.
- You can invert matrix values.
- You can compare the values of two matrices.

The `application` object. has commands or methods to create, get, invert, compare, or concatenate matrices.

The command or method used to apply a matrix is the `transform` command, which belongs to any type of object on which transformations can be performed.

CHAPTER 15

Launching and quitting Illustrator from a script

Your scripts can control the activation and termination of Illustrator.

15.1 Launching and activating Illustrator

15.1.1 AppleScript

In AppleScript, you use a tell statement to target Illustrator.

The activate command activates Illustrator if it is not already active

```
tell application "Adobe Illustrator"
activate
end tell
```

15.1.2 JavaScript

Typically, you run JavaScript scripts from the application's Scripts menu (File > Scripts) or start-up folder, so there is no need to launch Illustrator from your script.

Information on launching Illustrator in JavaScript is beyond the scope of this guide.

For details, search for Interapplication Communication or Javascript Messaging Framework in JavaScript Tools Guide.

15.1.3 VBScript

In VBScript, there are several ways to create an instance of Illustrator:

- `CreateObject` launches Illustrator as an invisible application if it is not already running. If Illustrator is launched as an invisible application you must manually activate the application to make it visible:

```
Set appRef = CreateObject("Illustrator.Application")
```

If you have multiple versions of Illustrator installed on the same machine and use the CreateObject method to obtain an application reference, using "Illustrator.Application" creates a reference to the latest Illustrator version. To specifically target an earlier version, use a version identifier at the end of the string:

Illustrator 10	"Illustrator.Application.1"
Illustrator CS	"Illustrator.Application.2"
Illustrator CS2	"Illustrator.Application.3"
Illustrator CS3	"Illustrator.Application.4"
Illustrator CS4	"Illustrator.Application.CS4"
Illustrator CS5	"Illustrator.Application.CS5"
Illustrator CS6	"Illustrator.Application.CS6"
Illustrator CC	"Illustrator.Application.CC"
Illustrator CC 2014	"Illustrator.Application.CC2014"
Illustrator CC 2015	"Illustrator.Application.CC2015"
Illustrator CC 2017	"Illustrator.Application.CC2017"

- Use the `New` operator if you added a reference to the Illustrator type library to the project. For example, the following line creates a new reference to the Application object:

```
Set appRef = New Illustrator.Application
```

15.2 Quitting Illustrator

15.2.1 AppleScript

Use the quit command:

```
tell application "Adobe Illustrator"
quit
end tell
```

15.2.2 JavaScript

Use the app.quit() method:

```
app.quit();
```

15.2.3 VBScript

Use the Application object's Quit method:

```
Set appRef = CreateObject("Illustrator.Application")
appRef.Quit
```

CHAPTER 16

Working with objects

16.1 Getting the frontmost document or layer

To refer to the selected document, use the `application` object's `current document` property in AppleScript or the `active document` property in JavaScript or VBScript. Similarly, you can use the `document` object's `current layer` or `active layer` property to refer to the selected layer.

There are other types of "active" or "current" object properties, like `active dataset` or `active view`. For details, see the Adobe lllustrator CC 2017 Scripting Reference for your language.

16.2 Creating new objects

Several objects (besides the `application` object itself) cannot be obtained from containers or parent objects. Your script must create these objects directly.

The following objects must be created explicitly:

- `CMYK color`
- `document preset`
- `EPS save options`
- `export options AutoCAD`
- `export options Flash`
- `export options GIF`
- `export options JPEG`
- `export options Photoshop`
- `export options PNG8`

- export options PNG24
- export options SVG
- export options TIFF
- file
- folder
- gradient color
- gray color
- Illustrator save options
- ink
- ink info
- lab color
- matrix
- MXG save options
- no color
- open options
- open options AutoCAD
- open options FreeHand
- open options PDF
- open options Photoshop
- paper info
- Pattern color
- PDF save options
- PPD file
- PPD file info
- print color management options
- print color separation options
- print coordinate options
- printer
- printer info
- print flattener options
- print font options
- print job options
- print options
- print page marks options
- print paper options
- print postscript options

- raster effect options
- rasterize options
- screen
- screen spot function
- RGB color
- spot color
- tracing options

The file and folder objects are Adobe ExtendScript devices designed to provide platform-independent access to the underlying file system. For information on using these objects, see JavaScript Tools Guide.

For information on creating an object explicitly, see the chapter for your scripting language.

16.3 Collection objects

Most collection objects must be obtained from a container. For example, a path items collection object can be contained by a document object or a layer object; to obtain an object in a path items collection, refer to either containing of these objects. For example, see the language-specific sections below.

16.3.1 AppleScript

To refer to a path items object in a document

```
path item 1 in document 1
```

To refer to a path items object in a layer

```
path item 1 in layer 1 in document 1
```

16.3.2 JavaScript

To refer to a path items object in a document

```
documents[0].pathItems[1]
```

To refer to a path items object in a layer

```
documents[0].layers[0].pathItems[0]
```

16.3.3 VBScript

To refer to a path items object in a document

```
Documents(1).PathItems(1)
```

To refer to a path items object in a layer

```
Documents(1).Layers(1).PathItems(1)
```

For more examples of collection-item containers, see the document object Elements table in Adobe lllustrator CC 2017 Scripting Reference: AppleScript or the Properties table in Adobe lllustrator CC 2017 Scripting Reference: JavaScript or Adobe lllustrator CC 2017 Scripting Reference: VBScript. A diagram of the lllustrator CC 2017 object model is in *The Illustrator Scripting Object Model*.

16.4 Selected objects

Sometimes, you want to write scripts that act on the currently selected object or objects. For example, you might want to apply formatting to selected text or change a selected path's shape.

16.4.1 Selecting Text

To select text, use the `select` command or method of the `text range` object.

16.4.2 Selecting art items

You can select an art object (like graph items, mesh items, raster items, and symbol items) by setting its selected property to true. (In AppleScript, `selected` is a property of the `page items` object.)

16.4.3 Referring to selected art items

To refer to all currently selected objects in a document, use the `document` object's `selection` property. To work with the objects in the selection array, you must determine their type, so you will know which properties and methods or commands you can use with them. In JavaScript and VBScript, each artwork object type has a read-only `typename` property that you can use to determine the object's type. In AppleScript, use the `class` property.

16.5 Notes on renaming objects stored in the application's panels

Several objects can be renamed; that is, their `name` property is writeable. The following types of objects can be sorted alphabetically in the corresponding Illustrator panel. If a script modifies the name of such an object, references to that object by index can become invalid.

- `Brush`
- `Gradient`
- `Graphic Style`
- `Pattern`
- `Swatch`
- `Symbol`
- `Variable`

CHAPTER 17

Measurement Units

Illustrator uses points as the unit of measurement for almost all distances. One inch equals 72 points. The exception is values for properties like `kerning`, `tracking`, and the `aki` properties (used for Japanese text composition), which use em units. (See *Em space units*)

Illustrator uses points when communicating with your scripts regardless of the current ruler units. If your script depends on adding, subtracting, multiplying, or dividing specific measurement values for units other than points, it must perform any unit conversions needed to represent your measurements as points. For example, to use inches for coordinates or measurement units, you must multiply all inch values by 72 when entering the values in your script.

The following table shows conversion formulas for various units of measurement:

Unit	Conversion formula
centimeters	28.346 points = 1 centimeter
inches	72 points = 1 inch
millimeters	2.834645 points = 1 millimeter
picas	12 points = 1 pica
Qs	0.709 point = 1 Q (1 Q equals 0.23 millimeter)

JavaScript provides the UnitValue object type, which offers unit-conversion utilities. For details, see JavaScript Tools Guide

17.1 Em space units

Values that use em units instead of points are measured in thousandths of an em.

An em is proportional to the current font size.

For example, in a 6-point font, 1 em equals 6 points; in a 10-point font, 1 em equals 10 points.

In a 10-point font, a kerning value of 20 em units is equivalent to

```
(20 units x 10 points) / 1000 units/em = 0.2 points
```

CHAPTER 18

Page-item positioning and dimensions

Illustrator uses simple, two-dimensional geometry in the form of points to record the position of page item objects in a document. Every page item object in a document has a position property that defines a fixed point as a pair of page coordinates in the format [x, y]. The fixed point is the top-left corner of the object's bounding box.

For information on the types of objects that comprise the page items collection, see "The artwork tree" on

A point is designated by a pair of coordinates:

- The horizontal position, x
- The vertical position, y

You can see these coordinates in the Info panel when you select or create an object in Illustrator.

For the artboard, the default coordinate origin, (0,0), is the top-left corner, reflected in the ruler origin property of the artboard object. X coordinate values increase from left to right, and Y values increase from top to bottom. This changed in the CS5 release; to maintain script compatability, a document created by a script still uses the older system, with the origin at the bottom left of the artboard, and the Y value increasing from bottom to top. The page origin property of a document object defines the bottom-left corner of the printable region of the document as a fixed point.

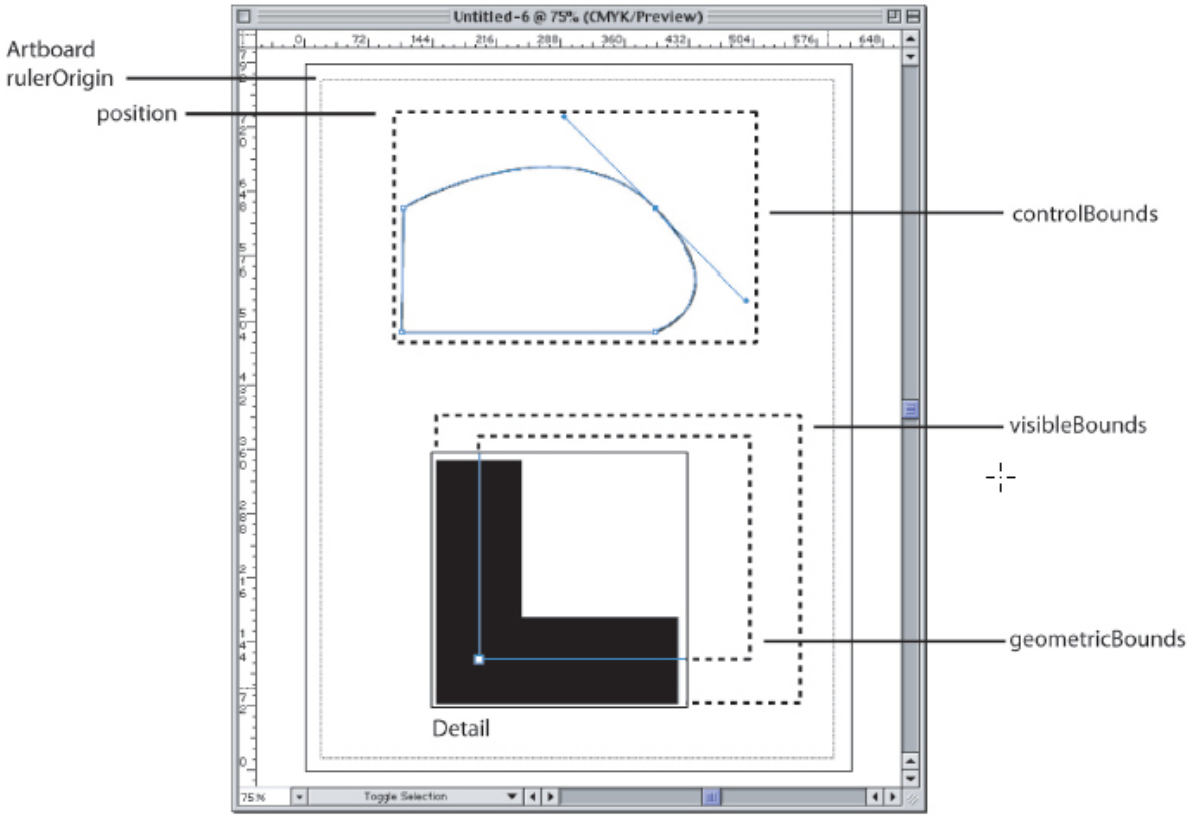
Each page item object has width and height properties. The maximum value allowed for the width or height of a page item is 16348 points.

18.1 Art item bounds

Every page item object has three properties that use fixed rectangles to describe the object's overall extent:

- The geometric bounds of a page item are the rectangular dimensions of the object's bounding box, excluding stroke width.
- The visible bounds of a page item are the dimensions of the object, including any stroke widths.
- The control bounds define the rectangular dimensions of the object, including in and out control points.

The following figure illustrates these properties, using JavaScript naming conventions.



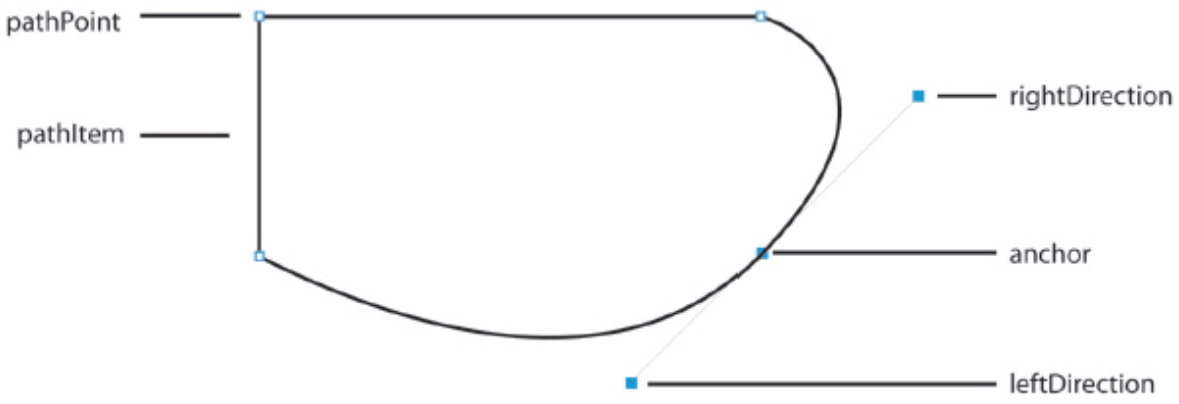
CHAPTER 19

Paths and shapes

Paths are represented in the Illustrator DOM by the `path item` object. Path items include all artwork that contains paths, such as rectangles, ellipses, and polygons, as well as freeform paths.

A freeform path consists of a series of path points. A path point can be specified in two ways:

- As an array of x and y page coordinates.
- As a `path point` object, which defines an anchor point and two direction points or handles that define the path segment's curve:



For details, samples, and information on creating shapes, see the chapter for your scripting language.

CHAPTER 20

User-interaction levels

When user feedback is required, an application typically presents a dialog. This is called user interaction. It is useful and expected when you are directly interacting with the application; however, when a script is interacting with an application, a dialog brings the execution of the script to a halt until the dialog is dismissed. This can be a serious problem in an automation environment, where there is no one present to deal with dialogs.

The `application` object contains a `user interaction level` property that allows you to control the level of interaction allowed during script execution. You can suppress interaction in an automation environment or allow some interaction where scripts are being used in a more interactive fashion.

20.1 AppleScript

Using AppleScript, it is possible to send commands from one machine to another, so additional types of interaction are possible. In AppleScript:, there are four possible values for the `user interaction` level property:

Property Value	Result
`never interact`	No interaction is allowed.
`interact with self`	Interact only with scripts executed from the Scripts menu (File > Scripts).
`interact with local`	Interact with scripts executed on the local machine (including self).
`interact with all`	Interact with all scripts.

The four values allow you to control interaction based on the source of the script commands. For example, if the application is acting as a server for remote users, it would be difficult for a remote user to dismiss a dialog, but it would be no problem for someone sitting in front of the machine. In this case, an interaction level of interact with local would prevent dialogs from halting remote scripts but would allow dialogs to be presented for local scripts.

20.2 JavaScript

In JavaScript, there are two possible values for the `app.userInteractionLevel` property:

Property Value	Result
`DISPLAYALERTS`	Interaction is allowed.
`DONTDISPLAYALERTS`	No interaction is allowed.

20.3 VBScript

In VBScript, there are two possible values for the `UserInteractionLevel` property of the `Application` object:

Property Value	Result
`aiDisplayAlerts`	Interaction is allowed.
`aiDontDisplayAlerts`	No interaction is allowed.

CHAPTER 21

Printing Illustrator documents

Using the `print options` scripting feature, you can capture and automate parts of your print workflow. Scripting exposes the full capabilities of Illustrator printing, some of which may not be accessible through the application's user interface.

Illustrator supports at most one print session at a time, because of limits in the current printing architecture.

The `document` object's print command or method takes one optional parameter, which allows you to specify a `print options` object.

The print options object allows you to define print settings like PPD, PostScript options, paper options, and color-management options. The `print options` object also has a `print preset` property, which allows you to specify a preset to define your print job.

When defining the properties of a `print options` object, you can find out which printers, PPDs, print presets, and other items are available by using the `application` object's read-only "list" properties, such as the `printer list`, `PPD file list`, and `print presets list` properties.

CHAPTER 22

For more information

Several extended sample scripts are in the `/Scripting/Sample Scripts/` folder in your Illustrator CC 2017 installation directory.

For information about individual classes, objects, properties, commands, and parameters, as well as script samples that demonstrate how to use many of these items, see asobjref/applescript-object-reference.

You also can view the lllustrator CC 2017 dictionary from the Script Editor application; see *Viewing the AppleScript object model*

If you do not understand the concepts and terms used in this chapter, read Adobe Introduction to Scripting.

CHAPTER 23

Your first Illustrator script

The traditional first project in any programming language is displaying the message "Hello World!" In this example, you create a new Illustrator document, then add a text frame containing this message. Follow these steps:

In a default Mac OS installation, Script Editor is in `/Applications/AppleScript/Script Editor/`.

If you cannot find the Script Editor application, you must reinstall it from your Mac OS system CD.

1. Open Script Editor.
2. Enter the following script:

```
--Send the following commands to Illustrator
tell application "Adobe Illustrator"
--Create a new document
set docRef to make new document
--Create a new text frame with the string "Hello World"
set textRef to make new text frame in docRef
with properties {contents: "Hello World!", position:{200, 200}}
end tell
```

3. In the Script Editor toolbar, click Run.

Tip: To add the script to the Illustrator Scripts menu (File > Scripts), save the script in the Scripts folder. The script will appear on the menu the next time you start Illustrator. For details, see *Installing scripts in the Scripts menu*.

23.1 Adding features to "Hello World"

Next, we create a new script that makes changes to the Illustrator document you created with your first script. Our second script demonstrates how to:

- Get the active document.

- Get the width of the active document.
- Resize the text frame to match the document's width.

If you already closed the Illustrator document, run your first script again to create a new document.

Follow these steps:

1. In Script Editor, choose File > New to create a new script.
2. Enter the following code:

```
tell application "Adobe Illustrator"
-- current document is always the active document
set docRef to the current document
set docWidth to the width of docRef
-- resize the text frame to match the page width
set width of text frame 1 of docRef to docWidth
-- alternatively, one can reference the item directly, as follows:
set width of text frame 1 of current document to docWidth
end tell
```

3. Run the script.

CHAPTER 24

Object references

In AppleScript, Illustrator returns object references by index position or name. For example, this is a reference to the first path in layer 2

```
path item 1 of layer 2 of document 1
```

An object's index position may change when other objects are created or deleted. For example, when a new path item is created on `layer 2`, the new path item becomes `path item 1 of layer 2 of document 1`.

This new object displaces the original path item, forcing the original to index position 2; therefore, any references made to `path item 1 of layer 2 of document 1` refer to the new object. This method of applying index numbers assures that lowest index number refers to the object that was worked on most recently.

Consider the following sample script:

```
-- Make 2 new objects and try to select both
tell application "Adobe Illustrator"
  set newDocument to make new document
  set rectPath to make new rectangle in newDocument
  set starPath to make new star in newDocument
  set selection of newDocument to {rectPath, starPath}
end tell
```

This script does not select both the rectangle and the star, as intended; instead, it selects only the star. Try running the script with the Event Log window open, to observe the references returned from Illustrator for each consecutive `make` command. (Choose Event Log at the bottom of the Script Editor window.) Notice that both commands return the same object reference: `path item 1 of layer 1 of document 1`; therefore, the last line resolves to

```
set selection of document 1 to {path item 1 of layer 1 of document 1,
  path item 1 of layer 1 of document 1}
```

A better approach is to reference the objects by name:

```
tell application "Adobe Illustrator"
  set newDocument to make new document
```

(continues on next page)

(continued from previous page)

```
  make new rectangle in newDocument with properties {name:"rectangle"}
  make new star in newDocument with properties {name:"star"}
  set selection of newDocument to
    {path item "rectangle" of newDocument,
    path item "star" of newDocument}
end tell
```

This example illustrates the need to uniquely identify objects in AppleScript scripts. We recommend that you assign names or variables to objects you need to access at a later time, as there is no guarantee you are accessing the objects you expect when accessing them by index.

24.1 Obtaining objects from documents and layers

This script references an object as part of a document:

```
-- Get reference for first page item of document 1
tell application "Adobe Illustrator"
  set pageItemRef to page item 1 of document 1
end tell
```

In the following script, the pageItemRef variable does not necessarily refer to the same object as in the previous script, because this script includes a reference to a layer:

```
-- Get reference for first page item of layer 1 of document 1
tell application "Adobe Illustrator"
  set pageItemRef to page item 1 of layer 1 of document 1
end tell
```

24.2 Creating new objects

To create a new object in AppleScript, use the `make` command.

24.3 Working with selections

When the user makes a selection in a document, the selected objects are stored in the document's selection property. To access all selected objects in the active document:

```
tell application "Adobe Illustrator"
  set myDoc to current document
  set selectedObjects to selection of myDoc
end tell
```

Depending on what is selected, the `selection` property value can be an array of any type of art objects. To get or manipulate the properties of the selected art items, you must retrieve the individual items in the array. To find out an object's type, use the `class` property.

The following sample gets the first object in the array, then displays the object's type:

```
tell application "Adobe Illustrator"
  set myDoc to current document
  set selectedObjects to selection of myDoc
  set topObject to item 1 of selectedObjects
  display dialog (class of topObject)
end tell
```

The first object in a selection array is the selected object that was last added to the page, not the last object selected.

24.3.1 Working with selections

To select an art object, the object's `selected` property.

CHAPTER 25

Working with text frames

To create a text frame of a specific type in AppleScript, use the `kind` property of the `text` frame object

```
set myRect to make new rectangle in current document with properties
{position:{100, 700}, height:100, width:100}
set myAreaText to make new text frame in current document with properties
{kind:point text,contents:"Text Frame 1"}
```

25.1 Threaded frames

As in the Illustrator application, you can thread area text frames or path text frames.

To thread existing text frames, use the `next frame` or `previous frame` property of the `text frame` object.

When copying the following script to your script editor, place the value of the contents property on one line. The long-line character (¬) is not valid within a string value.

```
tell application "Adobe Illustrator"
  make new document
  make new rectangle in current document with properties
    {position:{100, 500}, height:100, width:100}
  make new text frame in current document with properties
    {kind:area text, text path:the result, name:"tf1", contents:"This is two text␣
↪frames linked together as one story, with text flowing from the first to the last.␣
↪First frame content. "}
  make new rectangle in current document with properties
    {position:{300, 700}, height:100, width:100}
  make new text frame in current document with properties
    {kind:area text, text path:the result, name:"tf2", contents:"Second frame content.
↪" }
  --use the next frame property to thread the frames
    set next frame of text frame "tf1" of current document to
```

(continues on next page)

(continued from previous page)

```
    text frame "tf2" of current document
  redraw
end tell
```

25.1.1 Threaded frames make one story object

Threaded frames make a single story object. To observe this, run the following AppleScript after running the script above.

```
display dialog ("There are " & (count(text frames of current document)) & " text␣
↪frames.")
display dialog("There are " & (count(stories of current document)) & " stories.")
```

CHAPTER 26

Creating paths and shapes

This section explains how to create items that contain paths.

26.1 Paths

To create line or a freeform path, specify a series of path points, as a series of x-y coordinates or `path point` objects.

Using x-y coordinates limits the path to straight segments. To created a curved path, you must create `path point` objects. A path can comprise a combination of page coordinates and `path point` objects.

26.1.1 Specifying a series of x-y coordinates

To specify a path using page-coordinate pairs, use the `entire path` property of the `path items` object. The following script specifies three pairs of x-y coordinates, to create a path with three points:

```
tell application "Adobe Illustrator"
set docRef to make new document
-- set stroked to true so we can see the path
set lineRef to make new path item in docRef with properties {stroked:true}
set entire path of lineRef to {{220, 475},{200, 300},{375, 300}}
end tell
```

26.1.2 Using path point objects

To create a `path point` object, you must define three values for the point.

- A fixed `anchor` point, which is the point on the path.
- A pair of direction points— `left direction` and `right direction` —which allow you to control the path segment's curve.

You define each property as an array of page coordinates in the format [x, y]:

- If all three properties of a `path point` object have the same coordinates, and the properties of the next `path point` in the line are equal to each other, you create a straight-line segment.
- If two or more properties in a `path point` object have different values, the segment connected to the point is curved.

To create a path or add points to an existing path using `path point` objects, create a `path item` object, then add the path points as child objects in the `path item`:

```
tell application "Adobe Illustrator"
set docRef to make new document
-- set stroked to true so we can see the path
set lineRef to make new path item in docRef with properties {stroked:true}
  --giving the direction points the same value as the
  --anchor point creates a straight line segment
set newPoint to make new path point of lineRef with properties
  {anchor:{220, 475},left direction:{220, 475},right direction:{220, 475},
  point type:corner}

set newPoint2 to make new path point of lineRef with properties
  {anchor:{375, 300},left direction:{375, 300},right direction:{375, 300},
  point type:corner}

  --giving the direction points the different values
  --creates a curve
set newPoint3 to make new path point of lineRef with properties
  {anchor:{220, 300},left direction:{180, 260},right direction:{240, 320},
  point type:corner}

end tell
```

26.1.3 Combining path point types

The following script sample creates a path with three points, by combining the entire path property with a `path point` object

```
tell application "Adobe Illustrator"
set docRef to make new document
-- set stroked to true so we can see the path
set lineRef to make new path item in docRef with properties {stroked:true}
set entire path of lineRef to {{220, 475},{375, 300}}
set newPoint to make new path point of lineRef with properties
  {anchor:{220, 300},left direction:{180, 260},right direction:{240, 320},
  point type:corner}
end tell
```

26.2 Shapes

To create a shape, you use the object that corresponds to the shape's name (like `ellipse`, `rectangle`, or `polygon`), and use the object's properties to specify the shape's position, size, and other information like the number of sides in a polygon.

Remember:

- The scripting engine processes all measurements and page coordinates as points. For details, see *Measurement Units*.
- x and y coordinates are measured from the bottom-left corner of the document, as indicated in the Info panel in the Illustrator application. For details, see *Page-item positioning and dimensions*.

26.2.1 Write-once access

Properties for path-item shapes use the "write-once" access status, which indicates that the property is writeable only when the object is created. For existing path-item objects, the properties are read-only properties whose values cannot be changed.

26.2.2 Creating a rectangle

Consider the following sample:

```
tell application "Adobe Illustrator"
set docRef to make new document
set rectRef to make new rectangle in docRef with properties
  {bounds:{288, 360, 72, 144}}
end tell
```

The sample creates a rectangle with these properties:

- The top-right corner of the of the rectangle is inset 4 inches (288 points) from the bottom of the page and 5 inches (360 points) from the left edge of the page.
- The lower-left corner of the rectangle is inset 1 inch (72 points) from the left edge of the page and 2 inches (144 points) from the bottom of the page.

26.2.3 Creating a polygon

Consider the following sample:

```
tell application "Adobe Illustrator"
set docRef to make new document
set pathRef to make new polygon in docRef with properties
{center point:{144, 288},sides:7,radius:72.0}
end tell
```

The sample creates a polygon with these properties:

- The center point of the object is inset is 2 inches (144 points) on the horizontal axis and 4 inches (288 points) on the vertical axis.
- The polygon has 7 sides.
- The length of the radius from the center point to each corner is 1 inch (72 points).

CHAPTER 27

Working with the perspective grid

The Perspective Grid is a new feature in lllustrator CC 2017 that enables you to create and manipulate art in a spatial environment using established laws of perspective. Enable Perspective Grid using the View > Perspective Grid menu or the perspective tools in the toolbar.

The SDK provides an API for working with the perspective grid programmatically, and your scripts have some access to this API. A script can:

- Set a the default grid parameters using preset values.
- Show or hide the grid.
- Set the active plane.
- Draw an object in perspective on the active plane.
- Bring an object into perspective.

27.1 Use perspective presets

Illustrator provides default grid-parameter presets for one-point, two-point, and three-point perspectives. The presets are named `"[1P-NormalView]"`, `"[2P-NormalView]"`, and `"[3P-NormalView]"`.

The script shows how to select the two-point perspective preset programmatically:

```
tell application "Adobe Illustrator"
  --Create a new document
  set docRef to make new document
  tell docRef
    --Select the default two-point perspective preset
    select perspective preset perspective preset "[2P-Normal View]"
  end tell
end tell
```

You can create new perspective presets, export presets to files, and import presets from files. These scripts shows how to export and import presets:

```
tell application "Adobe Illustrator"
  set docRef to make new document
  set filePath to "Macintosh HD:scripting:PGPresetsExported"
  export perspective grid preset of docRef to file filePath
end tell

tell application "Adobe Illustrator"
  set docRef to make new document
  set filePath to "Macintosh HD:scripting:PGPresets"
  import perspective grid preset of docRef from file filePath
end tell
```

27.2 Show or hide the grid

This script shows or hides the Perspective Grid programmatically:

```
tell application "Adobe Illustrator"
  --Create a new document
  set docRef to make new document
  tell docRef
    --Display the perspective grid defined in the document
    show perspective grid
    --Hide the perspective grid defined in the document
    hide perspective grid
  end tell
end tell
```

27.3 Set the active plane

The perspective grid plane types are:

Left plane	`perspective grid plane leftplane`
Right plane	`perspective grid plane rightplane`
Floor plane	`perspective grid plane floorplane`
Invalid plane	`perspective grid plane noplane`

For a one-point perspective grid, only the left and floor plane are valid.

This script sets the active perspective plane to the left plane:

```
tell application "Adobe Illustrator"
  --Create a new document
  set docRef to make new document
  tell docRef
    --Set the active plane to the left plane
```

(continues on next page)

(continued from previous page)

```
    set perspective active plane perspective grid plane leftplane
  end tell
end tell
```

27.4 Draw on a perspective grid

When the Perspective Grid is on, drawing methods allow you to draw or operate on objects in perspective. This script creates a new document, shows a two-point perspective grid, and draws art objects on the left plane

```
tell application "Adobe Illustrator"
  --Create a new document
  set docRef to make new document
  tell docRef
    --Select the default two-point perspective preset
    select perspective preset perspective preset "[2P-Normal View]"

    --Display the perspective grid defined in the document
    show perspective grid

    --Check if active plane is set to left, otherwise set it to left
    if (get perspective active plane) is not leftplane then
      set perspective active plane perspective grid plane leftplane
    end if

    --Draw rectangle in perspective, then resize to 200% and move
    set rectRef to make new rectangle with properties {bounds:{0, 0, 30, 30},␣
↪reversed:false}
    scale rectRef horizontal scale 200 vertical scale 200 about top left with␣
↪transforming objects
    translate rectRef delta x -420 delta y 480

    --Draw ellipse in perspective
    set ellipseRef to make new ellipse with properties {bounds:{60, -60, 90, -30},␣
↪reversed:false, inscribed:true}

    --Draw rounded rectangle in perspective
    set rrectRef to make new rounded rectangle with properties {bounds:{90, -90, 30,␣
↪30}, horizontal radius:10, vertical radius:10, reversed:false}

    --Draw polygon in perspective
    set polyRef to make new polygon with properties {center point:{105, 105},␣
↪radius:15, sides:7, reversed:false}

    --Draw star in perspective
    set starRef to make new star with properties {center point:{135, 135}, radius:15,␣
↪inner radius:10, point count:6, reversed:false}

    --Draw path in perspective
    set newPath to make new path item with properties {entire path:{{anchor:{0, 0}},
↪{anchor:{60, 0}}, {anchor:{30, 45}}, {anchor:{90, 110}}}}
  end tell
end tell
```

27.5 Bring objects into perspective

If an art object is not in perspective, use the `bringInPerspective()` method to bring it into perspective and place it on a plane.

This script creates a new document, draws an art object, and brings it into perspective on a three-point perspective grid:

```
tell application "Adobe Illustrator"
  --Create a new document
  set docRef to make new document
  tell docRef
    --Draw star
    set starRef to make new star with properties {center point:{135, 135}, radius:15,␣
↪inner radius:10, point count:6, reversed:false}

    --Select the default three-point perspective preset
    select perspective preset perspective preset "[3P-Normal View]"

    --Display the perspective grid defined in the document
    show perspective grid

    --Check if active plane is set to left, otherwise set it to left
    if (get perspective active plane) is not leftplane then
      set perspective active plane perspective grid plane leftplane
    end if

    --Bring star to floor plane
    bring in perspective starRef position x 100 position y 100 perspective grid plane␣
↪floorplane
  end tell
end tell
```

CHAPTER 28

For more information

Several extended sample scripts are in the `/Scripting/Sample Scripts/` folder in your lllustrator CC 2017 installation directory.

For information about individual classes, objects, properties, methods, and parameters, as well as script samples that demonstrate how to use many of these items, see Adobe lllustrator CC 2017 Scripting Reference: JavaScript, in the `/Scripting/Documentation/` folder in your lllustrator CC 2017 installation directory. You also can use the Illustrator dictionary, which you access from the Object Model Viewer in the ESTK. For information on using the ExtendScript Toolkit and the Object Model Viewer, see *Viewing the JavaScript object model*.

If you do not understand the concepts and terms used in this chapter, read Adobe Introduction to Scripting.

CHAPTER 29

Your first Illustrator script

The traditional first project in any programming language is displaying the message "Hello World!" In this example, you create a new Illustrator document, then add a text frame containing this message. Follow these steps:

For information on locating the ExtendScript Toolkit, see *Viewing the JavaScript object model*.

1. Using any text editor (including Adobe^ InDesign® or the ESTK), enter the following text:

```
//Hello World!
var myDocument = app.documents.add();
//Create a new text frame and assign it to the variable "myTextFrame"
var myTextFrame = myDocument.textFrames.add();
// Set the contents and position of the text frame
myTextFrame.position = [200,200];
myTextFrame.contents = "Hello World!"
```

2. To test the script, do either of the following:

- If you are using the ESTK, select Adobe lllustrator CC 2017 from the drop-down list in the upper-left corner, select Yes to start Illustrator, then choose Debug > Run in the ESTK to run the script.
- If you are using a different text editor than the ESTK, save the file as text-only in a folder of your choice, using the file extension .jsx , then start Illustrator. In Illustrator, choose File > Scripts > Other Scripts, and navigate to and run your script file.

Tip: To add the script to the Illustrator Scripts menu (File > Scripts), save the script in the Scripts folder. The script will appear on the menu the next time you start Illustrator. For details, see *Installing scripts in the Scripts menu*.

29.1 Adding features to "Hello World"

Next, we create a new script that makes changes to the Illustrator document you created with your first script. Our second script demonstrates how to:

- Get the active document.
- Get the width of the active document.
- Resize the text frame to match the document's width.

If you already closed the Illustrator document, run your first script again to create a new document.

Follow these steps:

1. In Script Editor, choose File > New to create a new script.
2. Enter the following code:

```
var docRef = app.activeDocument;
var docWidth = docRef.width
var frameRef = docRef.textFrames[0]
frameRef.width = docWidth
```

3. Run the script.

CHAPTER 30

Working with methods in JavaScript

When you work with methods that have multiple parameters, you may omit optional parameters at the end of the parameter list, but you may not omit parameters in the middle of the list. If you do not want to specify a particular parameter in the middle of the list, you must insert the value `undefined` to use the parameter's default value. For example, the following definition describes the `rotate()` method for an art object.

```
rotate
  (angle
    [,changePositions]
    [,changeFillPatterns]
    [,changeFillGradients]
    [,changeStrokePattern]
    [,rotateAbout])
```

In the definition, taken from Adobe lllustrator CC 2017 Scripting Reference: JavaScript, optional parameters are enclosed in square brackets ([]).

To rotate the object 30 degrees and change the `fillGradients`, you would use the following script statement

```
myObject.rotate(30, undefined, undefined, true);
```

You need to specify `undefined` for the `changePositions` and `changeFillPatterns` parameters. You do not have to specify anything for the two optional parameters following `changeFillGradients`, since they are at the end of the parameter list.

CHAPTER 31

Accessing and referencing objects

When you write a script, you must first decide which file, or `document`, the script should act on. Through the `application` object, the script can create a new document, open an existing document, or act on a document that is already open.

The script can create new objects in the document, operate on objects that the user selected, or operate on objects in one of the object collections. The following sections illustrate various techniques for accessing, referencing, and manipulating Illustrator objects.

31.1 Referencing the application object

To obtain a reference to a specific object, you need to navigate the containment hierarchy. Because all JavaScript scripts are executed from within the Illustrator application, however, a specific reference to the application object is not required. For example, to assign the active document in Illustrator to the variable `frontMostDocument`, you could reference the `activeDocument` property of the `application` object, as follows

```
var frontMostDocument = activeDocument;
```

It is permissible to use the `application` object in a reference. To reference the `application` object, use the app global variable. The following two statements appear identical to the JavaScript engine:

```
var frontMostDocument = activeDocument;

var frontMostDocument = app.activeDocument;
```

31.2 Accessing objects in collections

All open documents, as well as the objects in a document, are collected into collection objects for the object type. A collection object contains an array of the objects that you can access by index or name. The collection object takes the plural form of the object name. For example, the collection object for the `document` object is `documents`.

The following script sample gets all `graphic style` objects in the `graphic styles` collection; that is, it gets all graphic styles available to the active document

```
var myStyles = app.activeDocument.graphicStyles;
```

All numeric collection references in JavaScript are zero-based: the first object in the collection has the index [0].

As a rule, JavaScript index numbers do not shift when you add an object to a collection. There is one exception: `documents[0` is always the active or frontmost document.

To access the first style in a `graphic styles` collection, you can use the variable declared in the previous script sample, or you can use the containment hierarchy to refer to the collection:

- Using the myStyles variable

```
var firstStyle = myStyles[0];
```

- Using the containment hierarchy:

```
var firstStyle = app.activeDocument.graphicStyles[0];
```

The following statements assign the name of the first graphic style in the collection to a variable. You can use these statements interchangeably.

```
var styleName = myStyles[0].name

var styleName = firstStyle.name

var styleName = app.activeDocument.graphicStyles[0].name
```

To get the total number of objects in a collection, use the `length` property:

```
alert ( myStyles.length );
```

The index of the last graphic style in the collection is `myStyles.length - 1` (-1 because the collection starts the index count at 0 and the length property counts from 1):

```
var lastStyle = myStyles[ myStyles.length - 1 ];
```

Note that an expression representing the index value is enclosed in square brackets ([]) as well as quotes.

If you know the name of an object, you can access the object in the collections using the name surrounded by square brackets; for example:

```
var getStyle = myStyles["Ice Type"];
```

Each element in the collection is an object of the desired type, and you can access its properties through the collection. For example, to get an object's name, use the `name` property:

```
var styleName = app.activeDocument.graphicStyles[0].name;
```

To apply `lastStyle` to the first `pageItem` in the document, use its `applyTo()` method:

```
lastStyle.applyTo( app.activeDocument.pageItems[0] );
```

31.3 Creating new objects

You can use a script to create new objects. To create objects that are available from collection objects, or containers, use the container object's `add()` method

```
var myDoc = app.documents.add()
var myLayer = myDoc.layers.add()
```

Some object types are not available from containers. To create an object of this type, define a variable, then use the `new` operator with an object constructor to assign an object as the value. For example, to create a new `CMYKColor` object using the variable name `myColor`:

```
var myColor = new CMYKColor()
```

31.4 Working with selections

When the user makes a selection in a document, the selected objects are stored in the document's `selection` property. To access all selected objects in the active document:

```
var selectedObjects = app.activeDocument.selection;
```

The `selection` property value can be an array of any type of art objects, depending on what types of objects are selected. To get or manipulate the properties of the selected art items, you must retrieve the individual items in the array. To find out an object's type, use the `typename` property.

The following sample gets the first object in the array, then displays the object's type

```
var topObject = app.activeDocument.selection[0];
alert(topObject.typename)
```

The first object in a selection array is the selected object that was last added to the page, not the last object selected.

31.4.1 Selecting artwork objects

To select an art object, the object's `selected` property.

CHAPTER 32

Working with text frames

To create a text frame of a specific type in JavaScript, use the `kind` property of the `text` frame object:

```
var rectRef = docRef.pathItems.rectangle(700, 50, 100, 100);
//use the areaText method to create the text frame
var areaTextRef = docRef.textFrames.areaText(rectRef);
```

32.1 Threaded frames

As in the Illustrator application, you can thread area text frames or path text frames.

To thread existing text frames, use the `nextFrame` or `previousFrame` property of the `text frame` object.

When copying the following script to the ESTK, place the value of the contents property on one line:

```
var myDoc = documents.add();
var myPathItem1 = myDoc.pathItems.rectangle(244, 64, 82, 76);
var myTextFrame1 = myDoc.textFrames.areaText(myPathItem1);
var myPathItem2 = myDoc.pathItems.rectangle(144, 144, 42, 116);
var myTextFrame2 = myDoc.textFrames.areaText(myPathItem2);

// use the nextFrame property to thread the text frames
myTextFrame1.nextFrame = myTextFrame2;
var sText = "This is two text frames linked together as one story, with text
flowing from the first to the last. This is two text frames linked together as one
story, with text flowing from the first to the last. This is two text frames linked
together as one story. ";
myTextFrame1.contents = sText;
redraw();
```

32.1.1 Threaded frames make one story object

Threaded frames make a single story object. To observe this, run the following JavaScript after running the script above.

```
var myDoc = app.activeDocument
alert("There are " + myDoc.textFrames.length + " text frames.")
alert("There are " + myDoc.stories.length + " stories.")
```

CHAPTER 33

Creating paths and shapes

This section explains how to create items that contain paths.

33.1 Paths

To create line or a freeform path, specify a series of path points, as a series of x-y coordinates or `path` point objects.

Using x-y coordinates limits the path to straight segments. To created a curved path, you must create `pathPoint` objects. Your path can comprise a combination of page coordinates and `pathPoint` objects.

33.1.1 Specifying a series of x-y coordinates

To specify a path using page-coordinate pairs, use the `setEntirePath()` property of the `pathItems` object. The following script specifies three pairs of x-y coordinates, to create a path with three points

```
var myDoc = app.activeDocument;
var myLine = myDoc.pathItems.add();
//set stroked to true so we can see the path
myLine.stroked = true;
myLine.setEntirePath([[220, 475], [375, 300], [200, 300]]);
```

33.1.2 Using path point objects

To create a `pathPoint` object, you must define three values for the point.

- A fixed `anchor` point, which is the point on the path.
- A pair of direction points— `left direction` and `right direction` —which allow you to control the path segment's curve.

You define each property as an array of page coordinates in the format [x, y]:

- If all three properties of a `pathPoint` object have the same coordinates, and the properties of the next `pathPoint` in the line are equal to each other, you create a straight-line segment.
- If two or more properties in a `pathPoint` object have different values, the segment connected to the point is curved.

To create a path or add points to an existing path using `pathPoint` objects, create a `pathItem` object, then add the path points as child objects in the `pathItem`:

```
var myDoc = app.activeDocument;
var myLine = myDoc.pathItems.add();

//set stroked to true so we can see the path
myLine.stroked = true;

var newPoint = myLine.pathPoints.add();
newPoint.anchor = [220, 475];
//giving the direction points the same value as the
//anchor point creates a straight line segment
newPoint.leftDirection = newPoint.anchor;
newPoint.rightDirection = newPoint.anchor;
newPoint.pointType = PointType.CORNER;

var newPoint1 = myLine.pathPoints.add();
newPoint1.anchor = [375, 300];
newPoint1.leftDirection = newPoint1.anchor;
newPoint1.rightDirection = newPoint1.anchor;
newPoint1.pointType = PointType.CORNER;

var newPoint2 = myLine.pathPoints.add();
newPoint2.anchor = [220, 300];
//giving the direction points different values
//than the anchor point creates a curve
newPoint2.leftDirection =[180, 260];
newPoint2.rightDirection = [240, 320];
newPoint2.pointType = PointType.CORNER;
```

33.1.3 Combining path point types

The following script sample creates a path with three points:

```
var myDoc = app.activeDocument;
var myLine = myDoc.pathItems.add();
myLine.stroked = true;
myLine.setEntirePath( [[220, 475], [375, 300]]);

// Append another point to the line
var newPoint = myDoc.myLine.pathPoints.add();
newPoint.anchor = [220, 300];
newPoint.leftDirection = newPoint.anchor;
newPoint.rightDirection = newPoint.anchor;
newPoint.pointType = PointType.CORNER;
```

33.2 Shapes

To create a shape, you use the object that corresponds to the shape's name (like `ellipse`, `rectangle`, or `polygon`), and use the object's properties to specify the shape's position, size, and other information like the number of sides in a polygon.

Remember:

- All measurements and page coordinates are processed as points by the scripting engine. For details, see *Measurement Units*.
- x and y coordinates are measured from the bottom-left corner of the document, as indicated in the Info panel in the Illustrator application. For details, see *Page-item positioning and dimensions*.

33.2.1 Creating a rectangle

Consider the following sample:

```
var myDocument = app.documents.add()
var artLayer = myDocument.layers.add()
var rect = artLayer.pathItems.rectangle( 144, 144, 72, 216 );
```

The sample uses the `pathItems` object's `rectangle()` method to create a rectangle with these properties:

- The top of the rectangle is 2 inches (144 points) from the bottom edge of the page.
- The left edge is 2 inches (144 points) from the left edge of the page.
- The rectangle is 1 inch (72 points) wide and 3 inches (216 points) long.

33.2.2 Creating a polygon

Consider the following sample:

```
var myDocument = app.documents.add()
var artLayer = myDocument.layers.add()
var poly = artLayer.pathItems.polygon( 144, 288, 72.0, 7 );
```

The sample uses the `polygon()` method to create a polygon with these properties:

- The center point of the object is inset is 2 inches (144 points) on the horizontal axis and 4 inches (288 points) on the vertical axis.
- The length of the radius from the center point to each corner is 1 inch (72 points).
- The polygon has 7 sides.

CHAPTER 34

Working with the perspective grid

The Perspective Grid is a new feature in lllustrator CC 2017 that enables you to create and manipulate art in a spatial environment using established laws of perspective. Enable Perspective Grid using the View > Perspective Grid menu or the perspective tools in the toolbar.

The SDK provides an API for working with the perspective grid programmatically, and your scripts have some access to this API. A script can:

- Set a the default grid parameters using preset values.
- Show or hide the grid.
- Set the active plane.
- Draw an object in perspective on the active plane.
- Bring an object into perspective.

34.1 Use perspective presets

Illustrator provides default grid-parameter presets for one-point, two-point, and three-point perspectives. The presets are named `"[1P-NormalView]"`, `"[2P-NormalView]"`, and `"[3P-NormalView]"`.

The script shows how to select the two-point perspective preset programmatically:

```
//Set the default one-point perspective preset
app.activeDocument.selectPerspectivePreset("[1P-Normal View]");

//Set the default two-point perspective preset
app.activeDocument.selectPerspectivePreset("[2P-Normal View]");

//Set the default three-point perspective preset
app.activeDocument.selectPerspectivePreset("[3P-Normal View]");
```

You can create new perspective presets, export presets to files, and import presets from files. These scripts shows how to export and import presets:

```
//Create a new document
var mydoc = app.documents.add();
//Export perspective presets to a file
var exportPresetFile = new File("C:/scripting/PGPresetsExported")
mydoc.exportPerspectiveGridPreset(exportPresetFile);

//Create a new document
var mydoc = app.documents.add();
//Import perspective presets from a file
var importPresetFile = new File("C:/scripting/PGPresets")
mydoc.importPerspectiveGridPreset(importPresetFile);
```

34.2 Show or hide the grid

This script shows or hides the Perspective Grid programmatically:

```
//Show the Perspective Grid defined in the document
app.activeDocument.showPerspectiveGrid();

//Hide the Perspective Grid defined in the document
mydoc.hidePerspectiveGrid();
```

34.3 Set the active plane

The perspective grid plane types are:

Left plane	`PerspectiveGridPlaneType.LEFTPLANE`
Right plane	`PerspectiveGridPlaneType.RIGHTPLANE`
Floor plane	`PerspectiveGridPlaneType.FLOORPLANE`
Invalid plane	`PerspectiveGridPlaneType.NOPLANE`

For a one-point perspective grid, only the left and floor plane are valid.

This script sets the active perspective plane:

```
//Set left plane as the active plane
app.activeDocument.setPerspectiveActivePlane(PerspectiveGridPlaneType.LEFTPLANE);

//Set right plane as the active plane
app.activeDocument.setPerspectiveActivePlane(PerspectiveGridPlaneType.RIGHTPLANE);

//Set floor plane as the active plane
app.activeDocument.setPerspectiveActivePlane(PerspectiveGridPlaneType.FLOORPLANE);
```

34.4 Draw on a perspective grid

When the Perspective Grid is on, drawing methods allow you to draw or operate on objects in perspective. This script creates a new document, shows a two-point perspective grid, and draws art objects on the left plane

```
//Create a new document
var mydoc = app.documents.add();

//Select the default two-point perspective preset
mydoc.selectPerspectivePreset("[2P-Normal View]");

//Display the perspective grid defined in the document
mydoc.showPerspectiveGrid();

//Check if active plane is set to left; if not, set it to left
if (mydoc.getPerspectiveActivePlane() != PerspectiveGridPlaneType.LEFTPLANE) {
  mydoc.setPerspectiveActivePlane(PerspectiveGridPlaneType.LEFTPLANE);
}

//Draw rectangle in perspective, then resize to 200% and move
var myrect = mydoc.pathItems.rectangle(30, -30, 30, 30, false);
myrect.resize(200, 200, true, false, false, false, 100, Transformation.TOPLEFT);
myrect.translate (-420, 480);

//Draw ellipse in perspective
var myellipse = mydoc.pathItems.ellipse(60, -60, 30, 30, false, true);

//Draw rounded rectangle in perspective
var myrrect = mydoc.pathItems.roundedRectangle(90, -90, 30, 30, 10, 10, false);

//Draw polygon in perspective
var mypoly = mydoc.pathItems.polygon(-105, 105, 15, 7, false);

//Draw star in perspective
var mystar = mydoc.pathItems.star(-135, 135, 15, 10, 6, false);

//Draw path in perspective
var newPath = mydoc.pathItems.add();
var lineList = new Array(4);
lineList[0] = new Array(0,0);
lineList[1] = new Array(60,0);
lineList[2] = new Array(30,45);
lineList[3] = new Array(90,110);
newPath.setEntirePath(lineList);
```

34.5 Bring objects into perspective

If an art object is not in perspective, use the `bringInPerspective()` method to bring it into perspective and place it on a plane.

This script creates a new document, draws an art object, and brings it into perspective on a three-point perspective grid:

```
//Create a new document
var mydoc = app.documents.add();

//Draw ellipse
var myellipse = mydoc.pathItems.ellipse(60, -60, 30, 30, false, true);

//Draw polygon
var mypoly = mydoc.pathItems.polygon(-105, 105, 15, 7, false);

//Draw star
var mystar = mydoc.pathItems.star(-135, 135, 15, 10, 6, false);

//Select the default three-point perspective preset
mydoc.selectPerspectivePreset("[3P-Normal View]");

//Display the perspective grid defined in the document
mydoc.showPerspectiveGrid();

//Check if active plane is set to left; if not, set it to left
if (mydoc.getPerspectiveActivePlane() != PerspectiveGridPlaneType.LEFTPLANE) {
  mydoc.setPerspectiveActivePlane(PerspectiveGridPlaneType.LEFTPLANE);
}

//Bring the ellipse to the active plane (left plane)
myellipse.bringInPerspective(-100,-100, PerspectiveGridPlaneType.LEFTPLANE);

//Bring the polygon to the right plane
mypoly.bringInPerspective(100,-100,PerspectiveGridPlaneType.RIGHTPLANE);

//Bring the star to the floor plane
mystar.bringInPerspective(100,100,PerspectiveGridPlaneType.FLOORPLANE);
```

CHAPTER 35

For more information

Several extended sample scripts are in the :Scripting:Sample Scripts folder in your Illustrator CC 2017 installation directory.

For information about individual classes, objects, properties, commands, and parameters, as well as script samples that demonstrate how to use many of these items, see Adobe lllustrator CC 2017 Scripting Reference: VBScript, in the `/Scripting/Documentation/` folder in your lllustrator CC 2017 installation directory. You also can view the lllustrator CC 2017 dictionary from the Script Editor application; see *Viewing the VBScript object model*.

If you do not understand the concepts and terms used in this chapter, read Adobe Introduction to Scripting.

CHAPTER 36

Your first Illustrator script

The traditional first project in any programming language is displaying the message "Hello World!" In this example, you create a new Illustrator document, then add a text frame containing this message. Follow these steps:

1. Start any text editor (for example, Notepad).
2. Type the following code:

```
Rem Hello World
Set appRef = CreateObject("Illustrator.Application")
Rem Create a new document and assign it to a variable
Set documentRef = appRef.Documents.Add
Rem Create a new text frame item and assign it to a variable
Set sampleText = documentRef.TextFrames.Add
Rem Set the contents and position of the TextFrame
sampleText.Position = Array(200, 200)
sampleText.Contents = "Hello World!"
```

3. Save the file as text-only in a folder of your choice, using the file extension `.vbs`.
4. To test the script, do one of the following:

- Double-click the file.
- Start Illustrator, choose File > Scripts > Other Scripts, and navigate to and run your script file.

Tip: To add the script to the Illustrator Scripts menu (File > Scripts), save the script in the Scripts folder. The script will appear on the menu the next time you start Illustrator. For details, see *Installing scripts in the Scripts menu*.

In general, when you launch a VBScript script from the Scripts menu, any `msgBox` dialogs will not display correctly.

36.1 Adding features to “Hello World”

Next, we create a new script that makes changes to the Illustrator document you created with your first script. Our second script demonstrates how to:

- Get the active document.
- Get the width of the active document.
- Resize the text frame to match the document’s width.

If you already closed the Illustrator document, run your first script again to create a new document.

Follow these steps:

1. Copy the following script into your text editor, and save the file:

```
Set appRef = CreateObject("Illustrator.Application")
'Get the active document
Set documentRef = appRef.ActiveDocument
Set sampleText = documentRef.TextFrames(1)
' Resize the TextFrame item to match the document width
sampleText.Width = documentRef.Width
sampleText.Left = 0
```

2. Run the script.

CHAPTER 37

Accessing and referencing objects

When you write a script, you must first decide which file, or `Document`, the script should act on. Through the `Application` object, the script can create a new document, open an existing document, or act on a document that is already open.

The script can create new objects in the document, operate on objects that the user selected, or operate on objects in one of the object collections. The following sections illustrate techniques for accessing, referencing, and manipulating Illustrator objects

37.1 Obtaining objects from collections

Generally, to obtain a reference to a specific object, you can navigate the containment hierarchy. For example, to use the `myPath` variable to store a reference to the first `PathItem` in the second layer of the active document

```
Set myPath = appRef.ActiveDocument.Layers(2).PathItems(1)
```

The following scripts demonstrate how to reference an object as part of a document:

```
Set documentRef = appRef.ActiveDocument

Set pageItemRef = documentRef.PageItems(1)
```

In the script below, the variable `pageItemRef` will not necessarily refer to the same object as the above script, since this script includes a reference to a layer:

```
Set documentRef = appRef.ActiveDocument
Set pageItemRef = documentRef.Layers(1).PageItems(1)
```

VBScript indexes start at 1 for object collections; however, VBScript allows you to specify whether array indexes start at 1 or 0. For information on specifying the index start number for arrays, see any VBScript textbook or tutorial.

37.2 Creating new objects

You can use a script to create new objects. To create objects that are available from collection objects, use the collection object's `Add` method:

```
Set myDoc = appRef.Documents.Add()

Set myLayer = myDoc.Layers.Add()
```

Some collection objects do not have an `Add` method. To create an object of this type, define a variable and use the `CreateObject` method. For example, the following code creates a new `CMYKColor` object using the variable name `newColor`

```
Set newColor = CreateObject ("Illustrator.CMYKColor")
```

37.3 Working with selections

When the user makes a selection in a document, the selected objects are stored in the document's `selection` property. To access all selected objects in the active document

```
Set appRef = CreateObject ("Illustrator.Application")
Set documentRef = appRef.ActiveDocument
selectedObjects = documentRef.Selection
```

Depending on what is selected, the selection property value can be an array of any type of art objects. To get or manipulate the properties of the selected art items, you must retrieve the individual items in the array. To find out an object's type, use the `typename` property.

The following sample gets the first object in the array, then displays the object's type

```
Set appRef = CreateObject ("Illustrator.Application")
Set documentRef = appRef.ActiveDocument
selectedObjects = documentRef.Selection
Set topObject = selectedObjects(0)
MsgBox(topObject.Typename)
```

The `MsgBox` method does not display a dialog when the script is run from the Illustrator Scripts menu (File > Scripts).

The first object in a selection array is the selected object that was last added to the page, not the last object selected.

37.3.1 Working with selections

To select an art object, the object's `Selected` property.

CHAPTER 38

Working with text frames

To create a text frame of a specific type in VBScript, use the `TextFrames` method that corresponds to the type of frame you want to create::

```
Set rectRef = docRef.PathItems.Rectangle(700, 50, 100, 100)

' Use the AreaText method to create the text frame
Set areaTextRef = docRef.TextFrames.AreaText(rectRef)
```

38.1 Threaded frames

As in the Illustrator application, you can thread area text frames or path text frames.

To thread existing text frames, use the `NextFrame` or `PreviousFrame` property of the `TextFrames` object.

When copying the following script to your script editor, place the value of the `Contents` property on one line. The long-line character (_) is not valid within a string value.

```
Set appRef = CreateObject("Illustrator.Application")
Set myDoc = appRef.Documents.Add
Set myPathItem1 = myDoc.PathItems.Rectangle(244, 64, 82, 76)
Set myTextFrame1 = myDoc.TextFrames.AreaText(myPathItem1)
  myTextFrame1.Contents = "This is two text frames linked together as one story, with␣
↪text flowing from the first to the last."
Set myPathItem2 = myDoc.PathItems.Rectangle(144, 144, 42, 116)
Set myTextFrame2 = myDoc.TextFrames.AreaText(myPathItem2)

'Use the NextFrame property to thread the frames
myTextFrame1.NextFrame = myTextFrame2

appRef.Redraw()
```

38.1.1 Threaded frames make one story object

Threaded frames make a single story object. To observe this, run the following VBScript after running the script above.

```
Set myDoc = appRef.ActiveDocument
myMsg = "alert(""There are " & CStr(myDoc.TextFrames.Count) & " text frames. "")"
appRef.DoJavaScript myMsg
myMsg = "alert(""There are " & CStr(myDoc.Stories.Count) & " storiess. "")"
appRef.DoJavaScript myMsg
```

CHAPTER 39

Creating paths and shapes

This section explains how to create items that contain paths.

39.1 Paths

To create line or a freeform path, specify a series of path points, as a series of x-y coordinates or `PathPoint` objects.

Using x-y coordinates limits the path to straight segments. To created a curved path, you must create `PathPoint` objects. A path can comprise a combination of page coordinates and `PathPoint` objects.

39.1.1 Specifying a series of x-y coordinates

To specify a path using page-coordinate pairs, use the `entire path` property of the `PathItems` object. The following script specifies three pairs of x-y coordinates, to create a path with three points

```
Set appRef = CreateObject ("Illustrator.Application")

Set firstPath = appRef.ActiveDocument.PathItems.Add
  firstPath.Stroked = True
firstPath.SetEntirePath(Array(Array(220, 475),Array(375, 300),Array(200, 300)))
```

39.1.2 Using path point objects

To create a `PathPoint` object, you must define three values for the point.

- A fixed `anchor` point, which is the point on the path.
- A pair of direction points— `left direction` and `right direction` —which allow you to control the path segment's curve.

You define each property as an array of page coordinates in the format `(Array (x,y))`:

- If all three properties of a `PathPoint` object have the same coordinates, and the properties of the next `PathPoint` in the line are equal to each other, you create a straight-line segment.
- If two or more properties in a `PathPoint` object hold different values, the segment connected to the point is curved.

To create a path or add points to an existing path using `PathPoint` objects, create a `PathItem` object, then add the path points as child objects in the `PathItem`

```
Set appRef = CreateObject ("Illustrator.Application")

Set firstPath = appRef.ActiveDocument.PathItems.Add
  firstPath.Stroked = true
Set newPoint = firstPath.PathPoints.Add
'Using identical coordinates creates a straight segment
newPoint.Anchor = Array(75, 300)
newPoint.LeftDirection = Array(75, 300)
newPoint.RightDirection = Array(75, 300)

Set newPoint2 = firstPath.PathPoints.Add
newPoint2.Anchor = Array(175, 250)
newPoint2.LeftDirection = Array(175, 250)
newPoint2.RightDirection = Array(175, 250)

Set newPoint3 = firstPath.PathPoints.Add
'Using different coordinates creates a curve
newPoint3.Anchor = Array(275, 290)
newPoint3.LeftDirection = Array(135, 150)
newPoint3.RightDirection = Array(155, 150)
```

39.1.3 Combining path point types

The following script sample creates a path with three points

```
Set appRef = CreateObject("Illustrator.Application")
Set myDoc = appRef.ActiveDocument
Set myLine = myDoc.PathItems.Add
  myLine.Stroked = True
  myLine.SetEntirePath( Array( Array(320, 475), Array(375, 300)))

' Append another point to the line
Set newPoint = myLine.PathPoints.Add
  'Using identical coordinates creates a straight segment
  newPoint.Anchor = Array(220, 300)
  newPoint.LeftDirection = Array(220, 300)
  newPoint.RightDirection = Array(220, 300)
```

39.2 Shapes

To create a shape, you use the object that corresponds to the shape's name (like `ellipse`, `rectangle`, or `polygon`), and use the object's properties to specify the shape's position, size, and other information like the number of sides in a polygon.

Remember:

- The scripting engine processes all measurements and page coordinates as points. For details, see *Measurement Units*.
- x and y coordinates are measured from the bottom-left corner of the document, as indicated in the Info panel in the Illustrator application. For details, see *Page-item positioning and dimensions*.

39.2.1 Creating a rectangle

Consider the following sample

```
Set appRef = CreateObject("Illustrator.Application")
Set frontDocument = appRef.ActiveDocument
' Create a new rectangle with
' top = 144, left side = 144, width = 72, height = 144
Set newRectangle = frontDocument.PathItems.Rectangle(144,144,72,144)
```

The sample creates a rectangle with these properties:

- The top of the rectangle is 2 inches (144 points) from the bottom edge of the page.
- The left edge is 2 inches (144 points) from the left edge of the page.
- The rectangle is 1 inch (72 points) wide and 2 inches (144 points) long.

39.2.2 Creating a polygon

Consider the following sample

```
Set appRef = CreateObject("Illustrator.Application")
Set frontDocument = appRef.ActiveDocument
' Create a new polygon with
' top = 144, left side = 288, width = 72, height = 144
Set newPolygon = frontDocument.PathItems.Polygon(144,288,72,7)
```

The sample creates a polygon with these properties:

- The center point of the object is inset is 2 inches (144 points) on the horizontal axis and 4 inches (288 points) on the vertical axis.
- The polygon has 7 sides.
- The length of the radius from the center point to each corner is 1 inch (72 points).

CHAPTER 40

Working with enumeration values

Properties that use enumeration values in VBScript use a numeral rather than a text value. For example, the Orientation property of the `TextFrame` object specifies whether text content in the text frame is horizontal or vertical. The property uses the `aiTextOrientation` enumeration, which has two possible values, `aiHorizontal` and `aiVertical`.

To find the numeral values of enumerations, use either of the following:

- The object browser in your scripting editor environment. See *Viewing the VBScript object model*.
- The Adobe lllustrator CC 2017 Scripting Reference: VBScript, which lists the numeral values directly after the constant value in the "Enumerations" chapter at the end of the book. The following example is from that table:

Enumeration type	Values	What it means
`AiTextOrientation`	`aiHorizontal = 0  aiVertical = 1`	The orientation of text in a text frame

The following sample specifies vertical text orientation

```
Set appRef = CreateObject ("Illustrator.Application")
Set docRef = appRef.Documents.Add
Set textRef = docRef.TextFrames.Add
textRef.Contents = "This is some text content."
textRef.Left = 50
textRef.Top = 700
textRef.Orientation = 1
```

Generally, it is considered good scripting practice to place the text value in a comment following the numeral value, as in the following sample statement:

```
textRef.Orientation = 1 ' aiVertical
```

CHAPTER 41

Working with the perspective grid

The Perspective Grid is a new feature in lllustrator CC 2017 that enables you to create and manipulate art in a spatial environment using established laws of perspective. Enable Perspective Grid using the View > Perspective Grid menu or the perspective tools in the toolbar.

The SDK provides an API for working with the perspective grid programmatically, and your scripts have some access to this API. A script can:

- Set a the default grid parameters using preset values.
- Show or hide the grid.
- Set the active plane.
- Draw an object in perspective on the active plane.
- Bring an object into perspective.

41.1 Use perspective presets

Illustrator provides default grid-parameter presets for one-point, two-point, and three-point perspectives. The presets are named `"[1P-NormalView]"`, `"[2P-NormalView]"`, and `"[3P-NormalView]"`.

The script shows how to select the two-point perspective preset programmatically:

```
Set appRef = CreateObject ("Illustrator.Application")

Rem Create a new document
Set docRef = appRef.Documents.Add()

Rem Select the default two-point perspective preset
docRef.SelectPerspectivePreset("[2P-Normal View]")
```

You can create new perspective presets, export presets to files, and import presets from files. These scripts shows how to export and import presets:

```
Set appRef = CreateObject ("Illustrator.Application")
Rem Create a new document
Set docRef = appRef.Documents.Add()
Rem Export perspective presets to a file
docRef.ExportPerspectiveGridPreset("C:/scripting/PGPresetsExported")

Set appRef = CreateObject ("Illustrator.Application")
Rem Create a new document
Set docRef = appRef.Documents.Add()
Rem Import perspective presets from a file
docRef.ImportPerspectiveGridPreset("C:/scripting/PGPresets")
```

41.2 Show or hide the grid

This script shows or hides the Perspective Grid programmatically:

```
Set appRef = CreateObject ("Illustrator.Application")

Rem Create a new document

Set docRef = appRef.Documents.Add()

Rem Show the Perspective Grid defined in the document
docRef.ShowPerspectiveGrid();

Rem Hide the Perspective Grid defined in the document
docRef.HidePerspectiveGrid();
```

41.3 Set the active plane

The perspective grid plane types are:

Left plane	`aiLEFTPLANE (1)`
Right plane	`aiRIGHTPLANE (2)`
Floor plane	`aiFLOORPLANE (3)`
Invalid plane	`aiNOPLANE (4)`

For a one-point perspective grid, only the left and floor plane are valid.

This script sets the active perspective plane to the left plane:

```
Set appRef = CreateObject ("Illustrator.Application")

Rem Create a new document
Set docRef = appRef.Documents.Add()
```

(continues on next page)

(continued from previous page)

```
Rem Set left plane as the active plane
docRef.SetPerspectiveActivePlane(1) 'aiLEFTPLANE
```

41.4 Draw on a perspective grid

When the Perspective Grid is on, drawing methods allow you to draw or operate on objects in perspective. This script creates a new document, shows a two-point perspective grid, and draws art objects on the left plane

```
Set appRef = CreateObject ("Illustrator.Application")

Rem Create a new document
Set docRef = appRef.Documents.Add()

Rem Select the default two point perspective preset
docRef.SelectPerspectivePreset("[2P-Normal View]")

Rem Display the perspective grid defined in the document
docRef.ShowPerspectiveGrid()

Rem Check if active plane is set to left, otherwise set it to left
If docRef.GetPerspectiveActivePlane() <> 1 Then
  docRef.SetPerspectiveActivePlane(1) 'aiLEFTPLANE
End If

Rem Draw rectangle in perspective, then resize to 200% and move
Set pathItemRect = docRef.PathItems.Rectangle(30, -30, 30, 30, False)

call pathItemRect.Resize(200, 200, True, False, False, False, 100, 2)
call pathItemRect.Translate(-420, 480)

Rem Draw ellipse in perspective
Set pathItemEllipse = docRef.PathItems.Ellipse(60, -60, 30, 30, False, True)

Rem Draw rounded rectangle in perspective
Set pathItemRRect = docRef.PathItems.RoundedRectangle(90, -90, 30, 30, 10, 10, False)

Rem Draw polygon in perspective
Set pathItemPoly = docRef.PathItems.Polygon(-105, 105, 15, 7, False)

Rem Draw star in perspective
Set pathItemStar = docRef.PathItems.Star(-135, 135, 15, 10, 6, False)

Rem Draw path in perspective
Set newPath = docRef.PathItems.Add()
newPath.SetEntirePath(Array(Array(0,0),Array(60,0),Array(30,45),Array(90,110)))
```

41.5 Bring objects into perspective

If an art object is not in perspective, use the `bringInPerspective()` method to bring it into perspective and place it on a plane.

This script creates a new document, draws an art object, and brings it into perspective on a three-point perspective grid

```
Set appRef = CreateObject ("Illustrator.Application")

Rem Create a new document
Set docRef = appRef.Documents.Add()

Rem Draw ellipse
Set pathItemEllipse = docRef.PathItems.Ellipse(60, -60, 30, 30, False, True)

Rem Draw polygon
Set pathItemPoly = docRef.PathItems.Polygon(-105, 105, 15, 7, False)

Rem Draw star
Set pathItemStar = docRef.PathItems.Star(-135, 135, 15, 10, 6, False)

Rem Select the default three-point perspective preset
docRef.SelectPerspectivePreset("[3P-Normal View]")

Rem Display the perspective grid defined in the document
docRef.ShowPerspectiveGrid()

Rem Check if active plane is set to left, otherwise set it to left
If docRef.GetPerspectiveActivePlane() <> 1 Then
  docRef.SetPerspectiveActivePlane(1) 'aiLEFTPLANE
End If

Rem Bring the ellipse to the active plane (left plane)
Call pathItemEllipse.BringInPerspective(100,100, 1) 'aiLEFTPLANE

Rem Bring the polygon to the right plane
Call pathItemPoly.BringInPerspective(100,-100,2) 'aiRIGHTPLANE

Rem Bring the star to the floor plane
Call pathItemStar.BringInPerspective(100,100,3) 'aiFLOORPLANE
```

CHAPTER 42

Scripting Constants

This chapter lists and describes the enumerations defined for use with Illustrator JavaScript properties and methods.

42.1 AlternateGlyphsForm

The alternate glyphs form of text.

Value	Description
`AlternateGlyphsForm.DEFAULTFORM`	Defaultform
`AlternateGlyphsForm.TRADITIONAL`	Traditional
`AlternateGlyphsForm.EXPERT`	Expert
`AlternateGlyphsForm.JIS78FORM`	JIS78FORM
`AlternateGlyphsForm.JIS83FORM`	JIS83FORM
`AlternateGlyphsForm.HALFWIDTH`	Half Width
`AlternateGlyphsForm.THIRDWIDTH`	Third Width
`AlternateGlyphsForm.QUARTERWIDTH`	Quarter Width
`AlternateGlyphsForm.FULLWIDTH`	Full Width
`AlternateGlyphsForm.PROPORTIONALWIDTH`	Proportional Width
`AlternateGlyphsForm.JIS90FORM`	JIS90FORM
`AlternateGlyphsForm.JIS04FORM`	JIS04FORM

Example

```
textRef.textRange.characters[i].characterAttributes.alternateGlyphs ==␣
↪AlternateGlyphsForm.DEFAULTFORM;
textRef.textRange.characters[i].characterAttributes.alternateGlyphs ==␣
↪AlternateGlyphsForm.FULLWIDTH
```

42.2 AntiAliasingMethod

The type of antialiasing method used in the rasterization.

Value	Description
`AntiAliasingMethod.None`	No antialiasing is allowed.
`AntiAliasingMethod.ARTOPTIMIZED`	Optimize for the art object.
`AntiAliasingMethod.TYPEOPTIMIZED`	Optimize for the type object.

42.3 ArtClippingOption

How the art should be clipped during output.

Value	Description
`ArtClippingOption.OUTPUTARTBOUNDS`	Output size is the size of the artwork.
`ArtClippingOption.OUTPUTARTBOARDBOUNDS`	Output size is the size of the artboard.
`ArtClippingOption.OUTPUTCROPRECTBOUNDS`	Output size is the size of the crop area.

42.4 AutoCADColors

Value	Description
`AutoCADColors.Max8Colors`	Max 8 CColors
`AutoCADColors.Max16Colors`	Max 16 Colors
`AutoCADColors.Max256Colors`	Max 25 6Colors
`AutoCADColors.TrueColors`	True Colors

42.5 AutoCADCompatibility

Value	Description
`AutoCADCompatibility.AutoCADRelease13`	Release 13
`AutoCADCompatibility.AutoCADRelease18`	Release 18
`AutoCADCompatibility.AutoCADRelease14`	Release 14
`AutoCADCompatibility.AutoCADRelease21`	Release 21
`AutoCADCompatibility.AutoCADRelease15`	Release 15
`AutoCADCompatibility.AutoCADRelease24`	Release 24

42.6 AutoCADExportFileFormat

Value	Description
`AutoCADExportFileFormat.DXF`	DXF
`AutoCADExportFileFormat.DWG`	DWG

42.7 AutoCADExportOption

Value	Description
`AutoCADExportOption.PreserveAppearance`	Preserve Appearance
`AutoCADExportOption.MaximizeEditability`	Maximize Editability

42.8 AutoCADGlobalScaleOption

Value	Description
`AutoCADGlobalScaleOption.OriginalSize`	Original Size
`AutoCADGlobalScaleOption.ScaleByValue`	Scale by Value
`AutoCADGlobalScaleOption.FitArtboard`	Fit Artboard

42.9 AutoCADRasterFormat

Value	Description
`AutoCADRasterFormat.PNG`	PNG
`AutoCADRasterFormat.JPEG`	JPEG

42.10 AutoCADUnit

Value	Description
`AutoCADUnit.Points`	Points
`AutoCADUnit.Picas`	Picas
`AutoCADUnit.Inches`	Inches
`AutoCADUnit.Millimeters`	Millimeters
`AutoCADUnit.Centimeters`	Centimeters
`AutoCADUnit.Pixels`	Pixels

42.11 AutoKernType

The auto kern type.

Value	Description
`AutoKernType.NOAUTOKERN`	None
`AutoKernType.AUTO`	Auto
`AutoKernType.OPTICAL`	Optical
`AutoKernType.METRICSROMANONLY`	Metrics

42.12 AutoLeadingType

The auto leading type.

Value	Description
`AutoLeadingType.BOTTOMTOBOTTOM`	Bottom to Bottom
`AutoLeadingType.TOPTOTOP`	Top to Top

42.13 BaselineDirectionType

The baseline direction type.

Value	Description
`BaselineDirectionType.Standard`	Standard
`BaselineDirectionType.VerticalRotated`	Vertical Rotated
`BaselineDirectionType.TateChuYoko`	TateChuYoko

42.14 BlendAnimationType

Value	Description
`BlendAnimationType.INBUILD`	In Build
`BlendAnimationType.NOBLENDANIMATION`	None
`BlendAnimationType.INSEQUENCE`	In Sequence

42.15 BlendModes

The blend mode used when compositing an object.

Value	Description
`BlendModes.COLORBLEND`	Color
`BlendModes.COLORBURN`	Color Burn
`BlendModes.COLORDODGE`	Color Dodge
`BlendModes.DARKEN`	Darken
`BlendModes.DIFFERENCE`	Difference
`BlendModes.EXCLUSION`	Exclusion
`BlendModes.HARDLIGHT`	Hard Light
`BlendModes.HUE`	Hue
`BlendModes.LIGHTEN`	Lighten
`BlendModes.LUMINOSITY`	Luminosity
`BlendModes.MULTIPLY`	Multiply
`BlendModes.NORMAL`	Normal
`BlendModes.OVERLAY`	Overlay
`BlendModes.SATURATIONBLEND`	Saturation
`BlendModes.SCREEN`	Screen
`BlendModes.SOFTLIGHT`	Soft Light

42.16 BlendsExpandPolicy

Policy used by FXG file format to expand blends.

Value	Description
`BlendsExpandPolicy.AUTOMATICALLYCONVERTBLENDS`	Automatically convert blends
`BlendsExpandPolicy.RASTERIZEBLENDS`	Rasterize blends

42.17 BurasagariTypeEnum

The Burasagari type.

Value	Description
`BurasagariTypeEnum.Forced`	Forced
`BurasagariTypeEnum.None`	None
`BurasagariTypeEnum.Standard`	Standard

42.18 CaseChangeType

The case change type.

Value	Description
`CaseChangeType.LOWERCASE`	Lowercase (`"hello world"`)
`CaseChangeType.SENTENCECASE`	Sentence case (`"Hello world"`)
`CaseChangeType.TITLECASE`	Title case (`"Hello World"`)
`CaseChangeType.UPPERCASE`	Uppercase (`"HELLO WORLD"`)

42.19 ColorConversion

The color conversion policy.

Value	Description
`ColorConversion.COLORCONVERSIONREPURPOSE`	Color Conversion Repurpose
`ColorConversion.COLORCONVERSIONTODEST`	Color Conversion to Dest
`ColorConversion.None`	None

42.20 ColorConvertPurpose

The purpose of color conversion using the `ConvertSampleColor` method of the `Application` class.

Value	Description
`ColorConvertPurpose.defaultpurpose`	Default
`ColorConvertPurpose.exportpurpose`	Export
`ColorConvertPurpose.previewpurpose`	Preview
`ColorConvertPurpose.dummypurpose`	Dummy

42.21 ColorDestination

Destination profile

Value	Description
`ColorDestination.COLORDESTINATIONDOCCMYK`	Doc CMYK
`ColorDestination.COLORDESTINATIONDOCRGB`	Doc RGB
`ColorDestination.COLORDESTINATIONPROFILE`	Profile
`ColorDestination.COLORDESTINATIONWORKINGCMYK`	Working CMYK
`ColorDestination.COLORDESTINATIONWORKINGRGB`	Working RGB
`ColorDestination.None`	None

42.22 ColorDitherMethod

The method used to dither colors in exported GIF and PNG8 images.

Value	Description
`ColorDitherMethod.DIFFUSION`	Diffusion
`ColorDitherMethod.NOISE`	Noise
`ColorDitherMethod.NOREDUCTION`	No Reduction
`ColorDitherMethod.PATTERNDITHER`	Pattern Dither

42.23 ColorModel

The color model to use.

Value	Description
`ColorModel.PROCESS`	Process
`ColorModel.REGISTRATION`	Registration
`ColorModel.SPOT`	Spot

42.24 ColorProfile

Value	Description
`ColorProfile.INCLUDEALLPROFILE`	Include All Profile
`ColorProfile.INCLUDEDESTPROFILE`	Include Dest Profile
`ColorProfile.INCLUDERGBPROFILE`	Include RGB Profile
`ColorProfile.LEAVEPROFILEUNCHANGED`	Leave Profile Unchanged
`ColorProfile.None`	None

42.25 ColorReductionMethod

The method used to reduce the number of colors in exported GIF and PNG8 images.

Value	Description
`ColorReductionMethod.ADAPTIVE`	Adaptive
`ColorReductionMethod.SELECTIVE`	Selective
`ColorReductionMethod.PERCEPTUAL`	Perceptual
`ColorReductionMethod.WEB`	Web

42.26 ColorType

The color specification for an individual color.

Value	Description
`ColorType.CMYK`	Cmyk
`ColorType.GRADIENT`	Gradient
`ColorType.GRAY`	Gray
`ColorType.PATTERN`	Pattern
`ColorType.RGB`	Rgb
`ColorType.SPOT`	Spot
`ColorType.NONE`	None

42.27 Compatibility

The version of the Illustrator file to create when saving an EPS or Illustrator file

Value	Description
`Compatibility.ILLUSTRATOR8`	Illustrator 8
`Compatibility.ILLUSTRATOR9`	Illustrator 9
`Compatibility.ILLUSTRATOR10`	Illustrator 10
`Compatibility.ILLUSTRATOR11`	Illustrator 11
`Compatibility.ILLUSTRATOR12`	Illustrator 12
`Compatibility.ILLUSTRATOR13`	Illustrator 13
`Compatibility.ILLUSTRATOR14`	Illustrator 14
`Compatibility.ILLUSTRATOR15`	Illustrator 15
`Compatibility.ILLUSTRATOR16`	Illustrator 16
`Compatibility.ILLUSTRATOR17`	Illustrator 17
`Compatibility.ILLUSTRATOR19`	Illustrator 19
`Compatibility.JAPANESEVERSION3`	Japanese Version 3

42.28 CompressionQuality

The quality of bitmap compression used when saving a PDF file

Value	Description
CompressionQuality.AUTOMATICJPEG2000HIGH	todo
CompressionQuality.AUTOMATICJPEG2000LOSSLESS	todo
CompressionQuality.AUTOMATICJPEG2000LOW	todo
CompressionQuality.AUTOMATICJPEG2000MAXIMUM	todo
CompressionQuality.AUTOMATICJPEG2000MEDIUM	todo
CompressionQuality.AUTOMATICJPEG2000MINIMUM	todo
CompressionQuality.AUTOMATICJPEGHIGH	todo
CompressionQuality.AUTOMATICJPEGLOW	todo
CompressionQuality.AUTOMATICJPEGMAXIMUM	todo
CompressionQuality.AUTOMATICJPEGMEDIUM	todo
CompressionQuality.AUTOMATICJPEGMINIMUM	todo
CompressionQuality.JPEG2000HIGH	todo
CompressionQuality.JPEG2000LOSSLESS	todo
CompressionQuality.JPEG2000LOW	todo
CompressionQuality.JPEG2000MAXIMUM	todo
CompressionQuality.JPEG2000MEDIUM	todo
CompressionQuality.JPEG2000MINIMUM	todo
CompressionQuality.JPEGHIGH	todo
CompressionQuality.JPEGLOW	todo
CompressionQuality.JPEGMAXIMUM	todo
CompressionQuality.JPEGMEDIUM	todo
CompressionQuality.JPEGMINIMUM	todo
CompressionQuality.ZIP4BIT	todo
CompressionQuality.ZIP8BIT	todo
CompressionQuality.None	todo

42.29 CoordinateSystem

The coordinate system used by Illustrator

Value	Description
CoordinateSystem.ARTBOARDCOORDINATESYSTEM	todo
CoordinateSystem.DOCUMENTCOORDINATESYSTEM	todo

42.30 CropOptions

The style of a document's cropping box

Value	Description
CropOptions.Japanese	Japanese
CropOptions.Standard	Standard

42.31 DocumentArtboardLayout

The layout of in the new document.

Value	Description
`DocumentArtboardLayout.Column`	todo
`DocumentArtboardLayout.GridByCol`	todo
`DocumentArtboardLayout.GridByRow`	todo
`DocumentArtboardLayout.RLGridByCol`	todo
`DocumentArtboardLayout.RLGridByRow`	todo
`DocumentArtboardLayout.RLRow`	todo
`DocumentArtboardLayout.Row`	todo

42.32 DocumentColorSpace

The color space of a document

Value	Description
`DocumentColorSpace.CMYK`	CMYK
`DocumentColorSpace.RGB`	RGB

42.33 DocumentLayoutStyle

Layout style for the document

Value	Description
`DocumentLayoutStyle.CASCADE`	todo
`DocumentLayoutStyle.CONSOLIDATEALL`	todo
`DocumentLayoutStyle.FLOATALL`	todo
`DocumentLayoutStyle.HORIZONTALTILE`	todo
`DocumentLayoutStyle.VERTICALTILE`	todo

42.34 DocumentPresetType

The preset types available for new documents.

Value	Description
DocumentPresetType.BasicCMYK	Basic CMYK
DocumentPresetType.BasicRGB	Basic RGB
DocumentPresetType.Mobile	Mobile
DocumentPresetType.Print	Print
DocumentPresetType.Video	Video
DocumentPresetType.Web	Web

42.35 DocumentPreviewMode

The document preview mode.

Value	Description
DocumentPreviewMode.DefaultPreview	Default
DocumentPreviewMode.OverprintPreview	Overprint
DocumentPreviewMode.PixelPreview	Pixel

42.36 DocumentRasterResolution

The preset document raster resolution.

Value	Description
DocumentRasterResolution.ScreenResolution	Screen Resolution
DocumentRasterResolution.HighResolution	High Resolution
DocumentRasterResolution.MediumResolution	Medium Resolution

42.37 DocumentTransparencyGrid

Document transparency grid colors.

Value	Description
DocumentTransparencyGrid.TransparencyGridBlue	Blue
DocumentTransparencyGrid.TransparencyGridDark	Dark
DocumentTransparencyGrid.TransparencyGridGreen	Green
DocumentTransparencyGrid.TransparencyGridLight	Light
DocumentTransparencyGrid.TransparencyGridMedium	Medium
DocumentTransparencyGrid.TransparencyGridNone	None
DocumentTransparencyGrid.TransparencyGridOrange	Orange
DocumentTransparencyGrid.TransparencyGridPurple	Purple
DocumentTransparencyGrid.TransparencyGridRed	Red

42.38 DocumentType

The file format used to save a file.

Value	Description
`DocumentType.EPS`	EPS
`DocumentType.FXG`	FXG
`DocumentType.ILLUSTRATOR`	Illustrator
`DocumentType.PDF`	PDF

42.39 DownsampleMethod

Value	Description
`DownsampleMethod.AVERAGEDOWNSAMPLE`	Average Downsample
`DownsampleMethod.BICUBICDOWNSAMPLE`	Bicubic Downsample
`DownsampleMethod.NODOWNSAMPLE`	No Downsample
`DownsampleMethod.SUBSAMPLE`	Subsample

42.40 ElementPlacement

Value	Description
`ElementPlacement.INSIDE`	Inside
`ElementPlacement.PLACEAFTER`	Place After
`ElementPlacement.PLACEATBEGINNING`	Place At Beginning
`ElementPlacement.PLACEATEND`	Place At End
`ElementPlacement.PLACEBEFORE`	Place Before

42.41 EPSPostScriptLevelEnum

Value	Description
`EPSPostScriptLevelEnum.LEVEL2`	Level 2
`EPSPostScriptLevelEnum.LEVEL3`	Level 3

42.42 EPSPreview

The preview image format used when saving an EPS file

Value	Description
`EPSPreview.BWTIFF`	todo
`EPSPreview.COLORTIFF`	todo
`EPSPreview.TRANSPARENTCOLORTIFF`	todo
`EPSPreview.None`	todo

42.43 ExportType

The file format used to export a file

Value	Description
`ExportType.AutoCAD`	AutoCAD
`ExportType.FLASH`	FLASH
`ExportType.GIF`	GIF
`ExportType.JPEG`	JPEG
`ExportType.Photoshop`	Photoshop
`ExportType.PNG24`	PNG24
`ExportType.PNG8`	PNG8
`ExportType.SVG`	SVG
`ExportType.TIFF`	TIFF

42.44 FigureStyleType

Value	Description
`FigureStyleType.DEFAULTFIGURESTYLE`	todo
`FigureStyleType.PROPORTIONAL`	todo
`FigureStyleType.PROPORTIONALOLDSTYLE`	todo
`FigureStyleType.TABULAR`	todo
`FigureStyleType.TABULAROLDSTYLE`	todo

42.45 FiltersPreservePolicy

The filters preserve policy used by the FXG file format.

Value	Description
FiltersPreservePolicy.EXPANDFILTERS	todo
FiltersPreservePolicy.KEEPFILTERSEDITABLE	todo
FiltersPreservePolicy.RASTERIZEFILTERS	todo

42.46 FlashExportStyle

The method used to convert Illustrator images when exporting files

Value	Description
FlashExportStyle.ASFLASHFILE	todo
FlashExportStyle.LAYERSASFILES	todo
FlashExportStyle.LAYERSASFRAMES	todo
FlashExportStyle.LAYERSASSYMBOLS	todo
FlashExportStyle.TOFILES	todo

42.47 FlashExportVersion

Version for exported SWF file.

Value	Description
FlashExportVersion.FlashVersion1	Version 1
FlashExportVersion.FlashVersion2	Version 2
FlashExportVersion.FlashVersion3	Version 3
FlashExportVersion.FlashVersion4	Version 4
FlashExportVersion.FlashVersion5	Version 5
FlashExportVersion.FlashVersion6	Version 6
FlashExportVersion.FlashVersion7	Version 7
FlashExportVersion.FlashVersion8	Version 8
FlashExportVersion.FlashVersion9	Version 9

42.48 FlashImageFormat

The format used to store flash images.

Value	Description
FlashImageFormat.LOSSLESS	Lossless
FlashImageFormat.LOSSY	Lossy

42.49 FlashJPEGMethod

The method used to store JPEG images.

Value	Description
`FlashJPEGMethod.Optimized`	Optimized
`FlashJPEGMethod.Standard`	Standard

42.50 FlashPlaybackSecurity

Value	Description
`FlashPlaybackSecurity.PlaybackLocal`	Local
`FlashPlaybackSecurity.PlaybackNetwork`	Network

42.51 FontBaselineOption

Value	Description
`FontBaselineOption.NORMALBASELINE`	todo
`FontBaselineOption.SUPERSCRIPT`	todo
`FontBaselineOption.SUBSCRIPT`	todo

42.52 FontCapsOption

Value	Description
`FontCapsOption.ALLCAPS`	All Caps
`FontCapsOption.ALLSMALLCAPS`	All Smallcaps
`FontCapsOption.NORMALCAPS`	Normal Caps
`FontCapsOption.SMALLCAPS`	Small Caps

42.53 FontOpenTypePositionOption

Value	Description
FontOpenTypePositionOption.DENOMINATOR	Denominator
FontOpenTypePositionOption.NUMERATOR	Numerator
FontOpenTypePositionOption.OPENTYPEDEFAULT	Opentype Default
FontOpenTypePositionOption.OPENTYPESUBSCRIPT	Opentype Subscript
FontOpenTypePositionOption.OPENTYPESUPERSCRIPT	Opentype Superscript

42.54 FontSubstitutionPolicy

Value	Description
FontSubstitutionPolicy.SUBSTITUTEDEVICE	Device
FontSubstitutionPolicy.SUBSTITUTEOBLIQUE	Oblique
FontSubstitutionPolicy.SUBSTITUTETINT	Tint

42.55 FXGVersion

The FXG file-format version.

Value	Description
FXGVersion.VERSION1PT0	Version 1 PT0
FXGVersion.VERSION2PT0	Version 2 PT0

42.56 GradientsPreservePolicy

The gradients preserve policy used by the FXG file format.

Value	Description
GradientsPreservePolicy.AUTOMATICALLYCONVERTGRADIENTS	Automaticaly Convert Gradients
GradientsPreservePolicy.KEEPGRADIENTSEDITABLE	Keep Gradients Editable

42.57 GradientType

The type of gradient.

Value	Description
`GradientType.LINEAR`	Linear
`GradientType.RADIAL`	Radial

42.58 ImageColorSpace

The color space of a raster item or an exported file

Value	Description
`ImageColorSpace.CMYK`	CMYK
`ImageColorSpace.DeviceN`	DeviceN
`ImageColorSpace.Grayscale`	Grayscale
`ImageColorSpace.Indexed`	Indexed
`ImageColorSpace.LAB`	LAB
`ImageColorSpace.RGB`	RGB
`ImageColorSpace.Separation`	Separation

42.59 InkPrintStatus

Value	Description
`InkPrintStatus.CONVERTINK`	Convert Ink
`InkPrintStatus.ENABLEINK`	Enable Ink
`InkPrintStatus.DISABLEINK`	Disable Ink

42.60 InkType

Value	Description
`InkType.BLACKINK`	Black Ink
`InkType.CUSTOMINK`	Custom Ink
`InkType.CYANINK`	Cyan Ink
`InkType.MAGENTAINK`	Magenta Ink
`InkType.YELLOWINK`	Yellow Ink

42.61 JavaScriptExecutionMode

Value	Description
`JavaScriptExecutionMode.BeforeRunning`	Before Running
`JavaScriptExecutionMode.OnRuntimeError`	On Runtime Error
`JavaScriptExecutionMode.never`	Never

42.62 Justification

The alignment or justification for a paragraph of text.

Value	Description
`Justification.CENTER`	Center
`Justification.FULLJUSTIFY`	Full Justify
`Justification.FULLJUSTIFYLASTLINECENTER`	Full Justify Last Line Center
`Justification.FULLJUSTIFYLASTLINELEFT`	Full Justify Last Line Left
`Justification.FULLJUSTIFYLASTLINERIGHT`	Full Justify Last Line Right
`Justification.LEFT`	Left
`Justification.RIGHT`	Right

42.63 KinsokuOrderEnum

Value	Description
`KinsokuOrderEnum.PUSHIN`	todo
`KinsokuOrderEnum.PUSHOUTFIRST`	todo
`KinsokuOrderEnum.PUSHOUTONLY`	todo

42.64 KnockoutState

The type of knockout to use on a page item.

Value	Description
`KnockoutState.DISABLED`	Disabled
`KnockoutState.ENABLED`	Enabled
`KnockoutState.INHERITED`	Inherited
`KnockoutState.Unknown`	Unknown

42.65 LanguageType

Value	Description
LanguageType.BOKMALNORWEGIAN	todo
LanguageType.BRAZILLIANPORTUGUESE	todo
LanguageType.BULGARIAN	todo
LanguageType.CANADIANFRENCH	todo
LanguageType.CATALAN	todo
LanguageType.CHINESE	todo
LanguageType.CZECH	todo
LanguageType.DANISH	todo
LanguageType.DUTCH	todo
LanguageType.DUTCH2005REFORM	todo
LanguageType.ENGLISH	todo
LanguageType.FINNISH	todo
LanguageType.GERMAN2006REFORM	todo
LanguageType.GREEK	todo
LanguageType.HUNGARIAN	todo
LanguageType.ICELANDIC	todo
LanguageType.ITALIAN	todo
LanguageType.JAPANESE	todo
LanguageType.NYNORSKNORWEGIAN	todo
LanguageType.OLDGERMAN	todo
LanguageType.POLISH	todo
LanguageType.RUMANIAN	todo
LanguageType.RUSSIAN	todo
LanguageType.SERBIAN	todo
LanguageType.SPANISH	todo
LanguageType.STANDARDFRENCH	todo
LanguageType.STANDARDGERMAN	todo
LanguageType.STANDARDPORTUGUESE	todo
LanguageType.SWEDISH	todo
LanguageType.SWISSGERMAN	todo
LanguageType.SWISSGERMAN2006REFORM	todo
LanguageType.TURKISH	todo
LanguageType.UKENGLISH	todo
LanguageType.UKRANIAN	todo

42.66 LayerOrderType

Value	Description
LayerOrderType.TOPDOWN	Top Down
LayerOrderType.BOTTOMUP	Bottom Up

42.67 LibraryType

Illustrator library type.

Value	Description
`LibraryType.Brushes`	Brushes
`LibraryType.GraphicStyles`	Graphic Styles
`LibraryType.IllustratorArtwork`	Illustrator Artwork
`LibraryType.Swatches`	Swatches
`LibraryType.Symbols`	Symbols

42.68 MonochromeCompression

The type of compression to use on a monochrome bitmap item when saving a PDF file.

Value	Description
`MonochromeCompression.CCIT3`	CCIT3
`MonochromeCompression.CCIT4`	CCIT4
`MonochromeCompression.MONOZIP`	MONOZIP
`MonochromeCompression.None`	None
`MonochromeCompression.RUNLENGTH`	RUNLENGTH

42.69 OutputFlattening

How transparency should be flattened when saving EPS and Illustrator file formats with compatibility set to versions of Illustrator earlier than Illustrator 10

Value	Description
`OutputFlattening.PRESERVEAPPEARANCE`	Preserve Appearance
`OutputFlattening.PRESERVEPATHS`	Preserve Paths

42.70 PageMarksTypes

Value	Description
`PageMarksTypes.Japanese`	Japanese
`PageMarksTypes.Roman`	Roman

42.71 PathPointSelection

Which points, if any, of a path are selected.

Value	Description
PathPointSelection.ANCHORPOINT	todo
PathPointSelection.LEFTDIRECTION	todo
PathPointSelection.LEFTRIGHTPOINT	todo
PathPointSelection.NOSELECTION	todo
PathPointSelection.RIGHTDIRECTION	todo

42.72 PDFBoxType

Value	Description
PDFBoxType.PDFARTBOX	todo
PDFBoxType.PDFBLEEDBOX	todo
PDFBoxType.PDFBOUNDINGBOX	todo
PDFBoxType.PDFCROPBOX	todo
PDFBoxType.PDFMEDIABOX	todo
PDFBoxType.PDFTRIMBOX	todo

42.73 PDFChangesAllowedEnum

Value	Description
PDFChangesAllowedEnum.CHANGE128ANYCHANGES	todo
PDFChangesAllowedEnum.CHANGE128COMMENTING	todo
PDFChangesAllowedEnum.CHANGE128EDITPAGE	todo
PDFChangesAllowedEnum.CHANGE128FILLFORM	todo
PDFChangesAllowedEnum.CHANGE128NONE	todo
PDFChangesAllowedEnum.CHANGE40ANYCHANGES	todo
PDFChangesAllowedEnum.CHANGE40COMMENTING	todo
PDFChangesAllowedEnum.CHANGE40NONE	todo
PDFChangesAllowedEnum.CHANGE40PAGELAYOUT	todo

42.74 PDFCompatibility

The version of the Acrobat file format to create when saving a PDF file

Value	Description
PDFCompatibility.ACROBAT4	Acrobat 4
PDFCompatibility.ACROBAT5	Acrobat 5
PDFCompatibility.ACROBAT6	Acrobat 6
PDFCompatibility.ACROBAT7	Acrobat 7
PDFCompatibility.ACROBAT8	Acrobat 8

42.75 PDFOverprint

Value	Description
PDFOverprint.DISCARDPDFOVERPRINT	Discard Pdf Overprint
PDFOverprint.PRESERVEPDFOVERPRINT	Preserve Pdf Overprint

42.76 PDFPrintAllowedEnum

Value	Description
PDFPrintAllowedEnum.PRINT128HIGHRESOLUTION	128 High Resolution
PDFPrintAllowedEnum.PRINT128LOWRESOLUTION	128 Low Resolution
PDFPrintAllowedEnum.PRINT128NONE	128 None
PDFPrintAllowedEnum.PRINT40HIGHRESOLUTION	40 High Resolution
PDFPrintAllowedEnum.PRINT40NONE	40 None

42.77 PDFTrimMarkWeight

Value	Description
PDFTrimMarkWeight.TRIMMARKWEIGHT0125	Weight 0125
PDFTrimMarkWeight.TRIMMARKWEIGHT025	Weight 025
PDFTrimMarkWeight.TRIMMARKWEIGHT05	Weight 05

42.78 PDFXStandard

Value	Description
PDFXStandard.PDFX1A2001	PDFX1A2001
PDFXStandard.PDFX1A2003	PDFX1A2003
PDFXStandard.PDFX32002	PDFX32002
PDFXStandard.PDFX32003	PDFX32003
PDFXStandard.PDFX42007	PDFX42007
PDFXStandard.PDFXNONE	PDFXNONE

42.79 PerspectiveGridType

Value	Description
PerspectiveGridType.OnePointPerspectiveGridType	One Point Perspective Grid Type
PerspectiveGridType.TwoPointPerspectiveGridType	Two Point Perspective Grid Type
PerspectiveGridType.ThreePointPerspectiveGridType	Three Point Perspective Grid Type
PerspectiveGridType.InvalidPerspectiveGridType	Invalid Perspective Grid Type

42.80 PerspectiveGridPlaneType

Value	Description
PerspectiveGridPlaneType.GRIDLEFTPLANETYPE	Grid Left Plane Type
PerspectiveGridPlaneType.GRIDRIGHTPLANETYPE	Grid Right Plane Type
PerspectiveGridPlaneType.GRIDFLOORPLANETYPE	Grid Floor Plane Type
PerspectiveGridPlaneType.INVALIDGRIDPLANETYPE	Invalid Grid Plane Type

42.81 PhotoshopCompatibility

Value	Description
PhotoshopCompatibility.Photoshop6	Photoshop 6
PhotoshopCompatibility.Photoshop8	Photoshop 8

42.82 PointType

The type of path point selected

Value	Description
`PointType.CORNER`	Corner
`PointType.SMOOTH`	Smooth

42.83 PolarityValues

Value	Description
`PolarityValues.NEGATIVE`	Negative
`PolarityValues.POSITIVE`	Positive

42.84 PostScriptImageCompressionType

Value	Description
`PostScriptImageCompressionType.IMAGECOMPRESSIONNONE`	todo
`PostScriptImageCompressionType.JPEG`	todo
`PostScriptImageCompressionType.RLE`	todo

42.85 PrintArtworkDesignation

Value	Description
`PrintArtworkDesignation.ALLLAYERS`	All Layers
`PrintArtworkDesignation.VISIBLELAYERS`	Visible Layers
`PrintArtworkDesignation.VISIBLEPRINTABLELAYERS`	Visible Printable Layers

42.86 PrintColorIntent

Value	Description
`PrintColorIntent.ABSOLUTECOLORIMETRIC`	todo
`PrintColorIntent.PERCEPTUALINTENT`	todo
`PrintColorIntent.RELATIVECOLORIMETRIC`	todo
`PrintColorIntent.SATURATIONINTENT`	todo

42.87 PrintColorProfile

Value	Description
PrintColorProfile.CUSTOMPROFILE	Custom Profile
PrintColorProfile.PRINTERPROFILE	Printer Profile
PrintColorProfile.OLDSTYLEPROFILE	Oldstyle Profile
PrintColorProfile.SOURCEPROFILE	Source Profile

42.88 PrintColorSeparationMode

Value	Description
PrintColorSeparationMode.COMPOSITE	Composite
PrintColorSeparationMode.HOSTBASEDSEPARATION	Host-Based Separation
PrintColorSeparationMode.INRIPSEPARATION	Inrip Separation

42.89 PrinterColorMode

Value	Description
PrinterColorMode.BLACKANDWHITEPRINTER	Black & White
PrinterColorMode.COLORPRINTER	Color
PrinterColorMode.GRAYSCALEPRINTER	Grayscale

42.90 PrinterPostScriptLevelEnum

Value	Description
PrinterPostScriptLevelEnum.PSLEVEL1	PS LEVEL 1
PrinterPostScriptLevelEnum.PSLEVEL2	PS LEVEL 2
PrinterPostScriptLevelEnum.PSLEVEL3	PS LEVEL 3

42.91 PrinterTypeEnum

Value	Description
PrinterTypeEnum.NONPOSTSCRIPTPRINTER	Non Postscript Printer
PrinterTypeEnum.POSTSCRIPTPRINTER	Postscript Printer
PrinterTypeEnum.Unknown	Unknown

42.92 PrintFontDownloadMode

Value	Description
PrintFontDownloadMode.DOWNLOADNONE	Download None
PrintFontDownloadMode.DOWNLOADCOMPLETE	Download Complete
PrintFontDownloadMode.DOWNLOADSUBSET	Download Subset

42.93 PrintingBounds

Value	Description
PrintingBounds.ARTBOARDBOUNDS	Artboard Bounds
PrintingBounds.ARTWORKBOUNDS	Artwork Bounds

42.94 PrintOrientation

The artwork printing orientation.

Value	Description
PrintOrientation.AUTOROTATE	Auto Rotate
PrintOrientation.LANDSCAPE	Landscape
PrintOrientation.PORTRAIT	Portrait
PrintOrientation.REVERSELANDSCAPE	Reverse Landscape
PrintOrientation.REVERSEPORTRAIT	Reverse Portrait

42.95 PrintPosition

Value	Description
PrintPosition.TRANSLATEBOTTOM	Translate Bottom
PrintPosition.TRANSLATEBOTTOMLEFT	Translate Bottom Left
PrintPosition.TRANSLATEBOTTOMRIGHT	Translate Bottom Right
PrintPosition.TRANSLATECENTER	Translate Center
PrintPosition.TRANSLATELEFT	Translate Left
PrintPosition.TRANSLATERIGHT	Translate Right
PrintPosition.TRANSLATETOP	Translate Top
PrintPosition.TRANSLATETOPLEFT	Translate Top Left
PrintPosition.TRANSLATETOPRIGHT	Translate Top Right

42.96 PrintTiling

Value	Description
`PrintTiling.TILEFULLPAGES`	Full Pages
`PrintTiling.TILESINGLEFULLPAGE`	Single Full Page
`PrintTiling.TILEIMAGEABLEAREAS`	Imageable Areas

42.97 RasterizationColorModel

The color model for the rasterization.

Value	Description
`RasterizationColorModel.BITMAP`	Bitmap
`RasterizationColorModel.DEFAULTCOLORMODEL`	Default Color Model
`RasterizationColorModel.GRAYSCALE`	Grayscale

42.98 RasterLinkState

The status of a raster item's linked image if the image is stored externally

Value	Description
`RasterLinkState.DATAFROMFILE`	Data From File
`RasterLinkState.DATAMODIFIED`	Data Modified
`RasterLinkState.NODATA`	No Data

42.99 RulerUnits

The default measurement units for the rulers of a document

Value	Description
`RulerUnits.Centimeters`	Centimeters
`RulerUnits.Qs`	Qs
`RulerUnits.Inches`	Inches
`RulerUnits.Pixels`	Pixels
`RulerUnits.Millimeters`	Millimeters
`RulerUnits.Unknown`	Unknown
`RulerUnits.Picas`	Picas
`RulerUnits.Points`	Points

42.100 SaveOptions

Save options provided when closing a document.

Value	Description
SaveOptions.DONOTSAVECHANGES	Do Not Save Changes
SaveOptions.SAVECHANGES	Save Changes
SaveOptions.PROMPTTOSAVECHANGES	Prompt To Save Changes

42.101 ScreenMode

The mode of display for a view.

Value	Description
ScreenMode.DESKTOP	Desktop
ScreenMode.MULTIWINDOW	Multi Window
ScreenMode.FULLSCREEN	Fullscreen

42.102 SpotColorKind

The custom color kind of a spot color.

Value	Description
SpotColorKind.SpotCMYK	CMYK
SpotColorKind.SpotLAB	LAB
SpotColorKind.SpotRGB	RGB

42.103 StrokeCap

The type of line capping for a path stroke.

Value	Description
StrokeCap.BUTTENDCAP	Butt
StrokeCap.ROUNDENDCAP	Round
StrokeCap.PROJECTINGENDCAP	Projecting

42.104 StrokeJoin

The type of joints for a path stroke.

Value	Description
`StrokeJoin.BEVELENDJOIN`	Bevel
`StrokeJoin.ROUNDENDJOIN`	Round
`StrokeJoin.MITERENDJOIN`	Miter

42.105 StyleRunAlignmentType

Value	Description
`StyleRunAlignmentType.bottom`	Bottom
`StyleRunAlignmentType.icfTop`	ICF Top
`StyleRunAlignmentType.center`	Center
`StyleRunAlignmentType.ROMANBASELINE`	Roman Baseline
`StyleRunAlignmentType.icfBottom`	ICF Bottom
`StyleRunAlignmentType.top`	Top

42.106 SVGCSSPropertyLocation

How should the CSS properties of the document be included in an exported SVG file

Value	Description
`SVGCSSPropertyLocation.ENTITIES`	Entities
`SVGCSSPropertyLocation.STYLEATTRIBUTES`	Style Attributes
`SVGCSSPropertyLocation.PRESENTATIONATTRIBUTES`	Presentation Attributes
`SVGCSSPropertyLocation.STYLEELEMENTS`	Style Elements

42.107 SVGDocumentEncoding

How should the text in the document be encoded when exporting an SVG file

Value	Description
`SVGDocumentEncoding.ASCII`	ASCII
`SVGDocumentEncoding.UTF8`	UTF8
`SVGDocumentEncoding.UTF16`	UTF16

42.108 SVGDTDVersion

SVG version compatibility for exported files

Value	Description
SVGDTDVersion.SVG1_0	SVG1_0
SVGDTDVersion.SVG1_1	SVG1_1
SVGDTDVersion.SVGBASIC1_1	SVGBASIC1_1
SVGDTDVersion.SVGTINY1_1	SVGTINY1_1
SVGDTDVersion.SVGTINY1_1PLUS	SVGTINY1_1PLUS
SVGDTDVersion.SVGTINY1_2	SVGTINY1_2

42.109 SVGFontSubsetting

What font glyphs should be included in exported SVG files

Value	Description
SVGFontSubsetting.ALLGLYPHS	All Glyphs
SVGFontSubsetting.GLYPHSUSEDPLUSENGLISH	Glyphs Used Plus English
SVGFontSubsetting.COMMONENGLISH	Common English
SVGFontSubsetting.GLYPHSUSEDPLUSROMAN	Glyphs Used Plus Roman
SVGFontSubsetting.COMMONROMAN	Common Roman
SVGFontSubsetting.GLYPHSUSED	Glyphs Used
SVGFontSubsetting.None	None

42.110 SVGFontType

Types for fonts included in exported SVG files

Value	Description
SVGFontType.CEFFONT	CEF Font
SVGFontType.SVGFONT	SVG Font
SVGFontType.OUTLINEFONT	Outline Font

42.111 SymbolRegistrationPoint

Registration points for symbols.

Value	Description
SymbolRegistrationPoint.SYMBOLBOTTOMLEFTPOINT	Bottom Left Point
SymbolRegistrationPoint.SYMBOLBOTTOMMIDDLEPOINT	Bottom Middle Point
SymbolRegistrationPoint.SYMBOLBOTTOMRIGHTPOINT	Bottom Right Point
SymbolRegistrationPoint.SYMBOLCENTERPOINT	Center Point
SymbolRegistrationPoint.SYMBOLMIDDLELEFTPOINT	Middle Left Point
SymbolRegistrationPoint.SYMBOLMIDDLERIGHTPOINT	Middle Right Point
SymbolRegistrationPoint.SYMBOLTOPLEFTPOINT	Top Left Point
SymbolRegistrationPoint.SYMBOLTOPMIDDLEPOINT	Top Middle Point
SymbolRegistrationPoint.SYMBOLTOPRIGHTPOINT	Top Right Point

42.112 TabStopAlignment

The alignment of a tab stop.

Value	Description
TabStopAlignment.Center	Center
TabStopAlignment.Decimal	Decimal
TabStopAlignment.Left	Left
TabStopAlignment.Right	Right

42.113 TextAntialias

The type of text anti-aliasing in a text art item.

Value	Description
TextAntialias.CRISP	Crisp
TextAntialias.NONE	None
TextAntialias.SHARP	Sharp
TextAntialias.STRONG	Strong

42.114 TextOrientation

The orientation of text in a text art item.

Value	Description
TextOrientation.HORIZONTAL	Horizontal
TextOrientation.VERTICAL	Vertical

42.115 TextPreservePolicy

The text-preserve policy used by the FXG file format.

Value	Description
`TextPreservePolicy.AUTOMATICALLYCONVERTTEXT`	Automatically Convert Text
`TextPreservePolicy.OUTLINETEXT`	Outline Text
`TextPreservePolicy.KEEPTEXTEDITABLE`	Keep Text Editable
`TextPreservePolicy.RASTERIZETEXT`	Rasterize Text

42.116 TextType

The type of text art displayed by this object.

Value	Description
`TextType.AREATEXT`	Area Text
`TextType.POINTTEXT`	Point Text
`TextType.PATHTEXT`	Path Text

42.117 TIFFByteOrder

The byte order to use for an exported TIFF file.

Value	Description
`TIFFByteOrder.IBMPC`	IBM PC
`TIFFByteOrder.MACINTOSH`	Macintosh

42.118 TracingModeType

Value	Description
`TracingModeType.TRACINGMODEBLACKANDWHITE`	Black & White
`TracingModeType.TRACINGMODECOLOR`	Color
`TracingModeType.TRACINGMODEGRAY`	Gray

42.119 Transformation

The point to use as the anchor point about which an object is rotated, resized, or transformed.

Value	Description
`Transformation.BOTTOM`	Bottom
`Transformation.BOTTOMLEFT`	Bottom Left
`Transformation.BOTTOMRIGHT`	Bottom Right
`Transformation.CENTER`	Center
`Transformation.DOCUMENTORIGIN`	Document Origin
`Transformation.LEFT`	Left
`Transformation.RIGHT`	Right
`Transformation.TOP`	Top
`Transformation.TOPLEFT`	Top Left
`Transformation.TOPRIGHT`	Top Right

42.120 TrappingType

Value	Description
`TrappingType.IGNOREOPAQUE`	todo
`TrappingType.OPAQUE`	todo
`TrappingType.NORMALTRAPPING`	todo
`TrappingType.TRANSPARENT`	todo

42.121 UserInteractionLevel

User interface settings

Value	Description
`UserInteractionLevel.DISPLAYALERTS`	Display Alerts
`UserInteractionLevel.DONTDISPLAYALERTS`	Don't Display Alerts

42.122 VariableKind

What type of variables are included in the document.

Value	Description
`VariableKind.GRAPH`	Graph
`VariableKind.IMAGE`	Image
`VariableKind.VISIBILITY`	Visibility
`VariableKind.TEXTUAL`	Textual
`VariableKind.Unknown`	Unknown

42.123 ViewRasterType

The raster visualization mode for tracing.

Value	Description
`ViewRasterType.TRACINGVIEWRASTERADJUSTEDIMAGE`	Adjusted Image
`ViewRasterType.TRACINGVIEWRASTERNOIMAGE`	No Image
`ViewRasterType.TRACINGVIEWRASTERORIGINALIMAGE`	Original Image
`ViewRasterType.TRACINGVIEWRASTERTRANSPARENTIMAGE`	Transparent Image

42.124 ViewVectorType

The vector visualization mode for tracing.

Value	Description
`ViewVectorType.TRACINGVIEWVECTORNOTRACINGRESULT`	No Tracing Result
`ViewVectorType.TRACINGVIEWVECTOROUTLINES`	Outlines
`ViewVectorType.TRACINGVIEWVECTOROUTLINESWITHTRACING`	Outlines With Tracing
`ViewVectorType.TRACINGVIEWVECTORTRACINGRESULT`	Tracing Result

42.125 WariChuJustificationType

Value	Description
`WariChuJustificationType.Center`	Center
`WariChuJustificationType.Left`	Left
`WariChuJustificationType.Right`	Right
`WariChuJustificationType.WARICHUAUTOJUSTIFY`	Warichu Auto Justify
`WariChuJustificationType.WARICHUFULLJUSTIFY`	Warichu Full Justify
`WariChuJustificationType.WARICHUFULLJUSTIFYLASTLINECENTER`	Warichu Full Justify Last Line Center
`WariChuJustificationType.WARICHUFULLJUSTIFYLASTLINELEFT`	Warichu Full Justify Last Line Left
`WariChuJustificationType.WARICHUFULLJUSTIFYLASTLINERIGHT`	Warichu Full Justify Last Line Right

42.126 ZOrderMethod

The method used to arrange an art item's position in the stacking order of its parent group or layer, as specified with the zOrder method

Value	Description
`ZOrderMethod.BRINGFORWARD`	Bring Forward
`ZOrderMethod.SENDBACKWARD`	Send Backward
`ZOrderMethod.BRINGTOFRONT`	Bring To Front
`ZOrderMethod.SENDTOBACK`	Send To Back

CHAPTER 43

JavaScript Object Reference

This section presents all of the object classes in the type library.

Each class listing includes the following:

- Properties of the class, including value type, read-only status, and an explanation.
- Methods for the class. Constants and value types needed by the method are shown in bold face. Required terms are shown in plain face. All items surrounded by brackets [ ] are optional.
- Notes to explain special issues.
- Sample code to help illustrate the syntax and typical workflow usage of the object class.

These examples are intended to be clear demonstrations of syntax, and do not show the best or most efficient way to construct a JavaScript script.

Error checking, for instance, is generally brief or missing.

However, the examples can be combined and expanded to make scripts with greater functionality.

CHAPTER 44

Application

`app`

Description

The Adobe® Illustrator® application object, referenced using the pre-defined global app object, which contains all other Illustrator objects.

44.1 Properties

44.1.1 Application.activeDocument

`app.activeDocument`

Description

The active (frontmost) document in Illustrator.

Type

Document

44.1.2 Application.browserAvailable

`app.browserAvailable`

Description

If `true`, a web browser is available.

Type

Boolean; read-only.

44.1.3 Application.buildNumber

`app.buildNumber`

Description

The application's build number.

Type

String; read-only.

44.1.4 Application.colorSettingsList

`app.colorSettingsList`

Description

The list of color-settings files currently available for use.

Type

Object; read-only.

44.1.5 Application.coordinateSystem

`app.coordinateSystem`

Description

The coordinate system currently in use, document or artboard.

Type

CoordinateSystem

44.1.6 Application.defaultColorSettings

`app.defaultColorSettings`

Description

The default color-settings file for the current application locale.

Type

File; read-only.

44.1.7 Application.documents

`app.documents`

Description

The documents in the application.

Type

Documents

44.1.8 Application.flattenerPresetList

`app.flattenerPresetList`

Description

The list of flattener style names currently available for use.

Type

Object; read-only.

44.1.9 Application.freeMemory

`app.freeMemory`

Description

The amount of unused memory (in bytes) within the Illustrator partition.

Type

Number (long); read-only.

44.1.10 Application.locale

`app.locale`

Description

The application's locale.

Type

String; read-only.

44.1.11 Application.name

`app.name`

Description

The application's name (not related to the filename of the application file).

Type

String; read-only.

44.1.12 Application.pasteRememberLayers

`app.pasteRememberLayers`

Description

If `true`, the paste operation maintains the layer structure.

Type

Boolean; read-only.

44.1.13 Application.path

`app.path`

Description

The file path to the application.

Type

File; read-only.

44.1.14 Application.PDFPresetsList

`app.PDFPresetsList`

Description

The list of preset PDF-options names available for use.

Type

Object; read-only.

44.1.15 Application.PPDFileList

`app.PPDFileList`

Description

The list of PPD files currently available for use.

Type

Object; read-only.

44.1.16 Application.preferences

`app.preferences`

Description

Illustrator's preference settings.

Type

Preferences

44.1.17 Application.printerList

`app.printerList`

Description

The list of installed printers.

Type

Array of *Printer*

44.1.18 Application.printPresetsList

`app.printPresetsList`

Description

The list of preset printing-options names available for use.

Type

Object; read-only.

44.1.19 Application.scriptingVersion

`app.scriptingVersion`

Description

The version of the Scripting plug-in.

Type

String; read-only.

44.1.20 Application.selection

`app.selection`

Description

All currently selected objects in the active (frontmost) document.

Type

Array of Objects; read-only.

44.1.21 Application.startupPresetsList

`app.startupPresetsList`

Description

The list of presets available for creating a new document.

Type

Object; read-only.

44.1.22 Application.textFonts

`app.textFonts`

Description

The installed fonts.

Type

TextFonts

44.1.23 Application.tracingPresetList

`app.tracingPresetList`

Description

The list of preset tracing-options names available for use.

Type

Array of Strings; read-only.

44.1.24 Application.typename

`app.typename`

Description

The class name of the referenced object.

Type

String; read-only.

44.1.25 Application.userInteractionLevel

`app.userInteractionLevel`

Description

What level of interaction with the user should be allowed when handling script commands.

Type

UserInteractionLevel

44.1.26 Application.version

`app.version`

Description

The application's version.

Type

String; read-only.

44.1.27 Application.visible

`app.visible`

Description

If `true`, the application is visible.

Type

Boolean; read-only.

44.2 Methods

44.2.1 Application.beep()

`app.beep()`

Description

Alerts the user.

Returns

Nothing.

44.2.2 Application.concatenateMatrix()

`app.concatenateMatrix(matrix, secondMatrix)`

Description

Joins two matrices together.

Parameters

Parameter	Type	Description
`matrix`	*Matrix*	First matrix
`secondMatrix`	*Matrix*	Second matrix

Returns

jsobjref/Matrix.

44.2.3 Application.concatenateRotationMatrix()

```
app.concatenateRotationMatrix(matrix, angle)
```

Description

Joins a rotation translation to a transformation matrix.

Parameters

Parameter	Type	Description
`matrix`	Matrix	Matrix
`angle`	Number (double)	Angle

Returns

jsobjref/Matrix.

44.2.4 Application.concatenateScaleMatrix()

```
app.concatenateScaleMatrix(matrix[, scaleX][, scaleY])
```

Description

Concatenates a scale translation to a transformation matrix.

Parameters

Parameter	Type	Description
`matrix`	Matrix	Matrix
`scaleX`	Number (double), optional	X Scale
`scaleY`	Number (dobule), optional	Y Scale

Returns

Matrix

44.2.5 Application.concatenateTranslationMatrix()

```
app.concatenateTranslationMatrix(matrix[, deltaX][, deltaY])
```

Description

Joins a translation to a transformation matrix.

Parameters

Parameter	Type	Description
`matrix`	Matrix	Matrix
`deltaX`	Number (double), optional	X Delta
`deltaY`	Number (dobule), optional	Y Delta

Returns

Matrix

44.2.6 Application.convertSampleColor()

```
app.convertSampleColor(sourceColorSpace, sourceColor, destColorSpace,
colorConvertPurpose[, sourceHasAlpha][, destHasAlpha])
```

Description

Converts a sample-component color from one color space to another.

Parameters

Parameter	Type	Description
`sourceColorSpace`	*ImageColorSpace*	Color space of source color
`sourceColor`	ColorComponents	Source color to convert
`destColorSpace`	*ImageColorSpace*	Destination color space
`colorConvertPurpose`	*ColorConvertPurpose*	The purpose of the convert
`sourceHasAlpha`	Boolean, optional	Whether the source has alpha
`destHasAlpha`	Boolean, optional	Whether the destination has alpha

Returns

Array of ColorComponents

44.2.7 Application.copy()

```
app.copy()
```

Description

Copies current selection to the clipboard.

Returns

Nothing.

44.2.8 Application.cut()

```
app.cut()
```

Description

Cuts current selection to the clipboard.

Returns

Nothing.

44.2.9 Application.deleteWorkspace()

```
app.deleteWorkspace(workspaceName)
```

Description

Deletes an existing workspace.

Parameters

Parameter	Type	Description
`workspaceName`	String	Name of workspace to delete

Returns

Boolean

44.2.10 Application.getIdentityMatrix()

```
app.getIdentityMatrix()
```

Description

Returns an identity matrix.

Returns

Matrix

44.2.11 Application.getIsFileOpen()

```
app.getIsFileOpen(filePath)
```

Note: This functionality was added in Illustrator XX.X (CC2017)

Description

Returns whether the specified filePath is open

Parameters

Parameter	Type	Description
`filePath`	String	File path to check

Returns

Boolean

44.2.12 Application.getPPDFileInfo()

```
app.getPPDFileInfo(name)
```

Description

Gets detailed file information for specified PPD file.

Parameters

Parameter	Type	Description
`name`	String	File name to get info for

Returns

PPDFileInfo

44.2.13 Application.getPresetFileOfType()

```
app.getPresetFileOfType(presetType)
```

Description

Returns the full path to the application's default document profile for the specified preset type.

Parameters

Parameter	Type	Description
`presetType`	*DocumentPresetType*	Preset type to get file of

Returns

File

44.2.14 Application.getPresetSettings()

```
app.getPresetSettings(preset)
```

Description

Retrieves the tracing-option settings from the template with a given preset name.

Parameters

Parameter	Type	Description
`preset`	String	Preset name to get settings from

Returns

DocumentPreset

44.2.15 Application.getRotationMatrix()

```
app.getRotationMatrix([angle])
```

Description

Returns a transformation matrix containing a single rotation.

Note: Requires a value in degrees.

For example, `30` rotates the object 30 degrees counterclockwise; `-30` rotates the object 30 degrees clockwise.

Parameters

Parameter	Type	Description
`angle`	Number (double), optional	Angle to get matrix of

Returns

Matrix

44.2.16 Application.getScaleMatrix()

```
app.getScaleMatrix([scaleX][, scaleY])
```

Description

Returns a transformation matrix containing a single scale.

Note: Requires a value in percentage.

For example, `60` scales the object to 60% of its original size; `200` doubles the object's bounds.

Parameters

Parameter	Type	Description
`scaleX`	Number (double), optional	X scale to get matrix of
`scaleY`	Number (double), optional	Y scale to get matrix of

Returns

Matrix

44.2.17 Application.getScriptableHelpGroup()

```
app.getScriptableHelpGroup()
```

Description

Gets the scriptable help group object that represents the search widget in the app bar.

Returns

Variant

44.2.18 Application.getTranslationMatrix()

```
app.getTranslationMatrix([deltaX][, deltaY])
```

Description

Returns a transformation matrix containing a single translation.

Note: Requires a value in points.

For example, *(100, 200)* moves the object 100 pt. to the right and 200 pt. up; a minus before each number moves the object left and down.

Parameters

Parameter	Type	Description
`deltaX`	Number (double), optional	X Delta
`deltaY`	Number (dobule), optional	Y Delta

Returns

Matrix

44.2.19 Application.invertMatrix()

```
app.invertMatrix(matrix)
```

Description

Inverts a matrix.

Parameters

Parameter	Type	Description
`matrix`	*Matrix*	Matrix to invert

Returns

Matrix

44.2.20 Application.isEqualMatrix()

```
app.isEqualMatrix(matrix, secondMatrix)
```

Description

Checks whether the two matrices are equal.

Parameters

Parameter	Type	Description
`matrix`	*Matrix*	First matrix to check
`secondMatrix`	*Matrix*	Second matrix to check

Returns

Boolean

44.2.21 Application.isSingularMatrix()

```
app.isSingularMatrix(matrix)
```

Description

Checks whether a matrix is singular and cannot be inverted.

Parameters

Parameter	Type	Description
`matrix`	*Matrix*	Matrix to check

Returns

Boolean

44.2.22 Application.loadColorSettings()

```
app.loadColorSettings(fileSpec)
```

Description

Loads color settings from specified file, or, if file is empty, turns color management off.

Parameters

Parameter	Type	Description
`fileSpec`	File	File to load settings from

Returns

Nothing.

44.2.23 Application.open()

```
app.open(file[, documentColorSpace][, options])
```

Description

Opens the specified document file.

Note: If you open a pre-Illustrator 9 document that contains both RGB and CMYK colors and *documentColorSpace* is supplied, all colors are converted to the specified color space.

If the parameter is not supplied, Illustrator opens a dialog so the user can choose the color space.

Parameters

Parameter	Type	Description
`file`	File	File to open
`documentColorSpace`	*DocumentColorSpace*, optional	Color space of document
`options`	anything	todo

Returns

Document

44.2.24 Application.paste()

```
app.paste()
```

Description

Pastes current clipboard content into the current document.

Returns

Nothing.

44.2.25 Application.quit()

```
app.quit()
```

Description

Quits Illustrator.

Note: If the clipboard contains data, Illustrator may show a dialog prompting the user to save the data for other applications.

Returns

Nothing.

44.2.26 Application.redo()

```
app.redo()
```

Description

Redoes the most recently undone transaction.

Returns

Nothing.

44.2.27 Application.redraw()

```
app.redraw()
```

Description

Forces Illustrator to redraw all its windows.

Returns

Nothing.

44.2.28 Application.resetWorkspace()

```
app.resetWorkspace()
```

Description

Resets the current workspace.

Returns

Boolean

44.2.29 Application.saveWorkspace()

```
app.saveWorkspace(workspaceName)
```

Description

Saves a new workspace.

Parameters

Parameter	Type	Description
`workspaceName`	String	Name of workspace to save as

Returns

Boolean

44.2.30 Application.sendScriptMessage()

```
app.sendScriptMessage(pluginName, messageSelector, inputString)
```

Description

Sends a plug-in-defined command message to a plug-in with given input arguments, and returns the plug-in-defined result string.

Parameters

Parameter	Type	Description
`pluginName`	String	Name of plugin to send message to
`messageSelector`	String	Message to send to the plugin
`inputString`	String	Data to pass into the command

Returns

String

44.2.31 Application.showPresets()

```
app.showPresets(fileSpec)
```

Description

Gets presets from the file.

Parameters

Parameter	Type	Description
`fileSpec`	File	File to get presets from

Returns

PrintPresetList

44.2.32 Application.switchWorkspace()

```
app.switchWorkspace(workspaceName)
```

Description

Switches to the specified workspace.

Parameters

Parameter	Type	Description
`workspaceName`	String	Name to switch to

Returns

Boolean

44.2.33 Application.translatePlaceholderText()

`app.translatePlaceholderText(text)`

Description

Translates the placeholder text to regular text (a way to enter Unicode points in hex values).

Parameters

Parameter	Type	Description
`text`	String	String to translate

Returns

String

44.2.34 Application.undo()

`app.undo()`

Description

Undoes the most recent transaction.

Returns

Nothing.

44.3 Example

44.3.1 Duplicating the Active Document

```
// Duplicates any selected items from
// the active document into a new document.

var newItem;
var docSelected = app.activeDocument.selection;

if (docSelected.length > 0) {
  // Create a new document and move the selected items to it.
  var newDoc = app.documents.add();
  if (docSelected.length > 0) {
    for (var i = 0; i < docSelected.length; i++) {
      docSelected[i].selected = false;
      newItem = docSelected[i].duplicate(newDoc, ElementPlacement.PLACEATEND);
    }
  } else {
    docSelected.selected = false;
    newItem = docSelected.parent.duplicate(newDoc, ElementPlacement.PLACEATEND);
  }
```

(continues on next page)

(continued from previous page)

```
} else {
  alert("Please select one or more art objects");
}
```

CHAPTER 45

Artboard

`artboard`

Description

An Artboard object represents a single artboard in a document. There can be between 1 to 100 artboards in one document.

45.1 Properties

45.1.1 Artboard.artboardRect

`artboard.artboardRect`

Description

Size and position of the artboard.

Type

Rect

45.1.2 Artboard.name

`artboard.name`

Description

The unique identifying name of the artboard.

Type

String

45.1.3 Artboard.parent

`artboard.parent`

Description

The parent of this object.

Type

Document; read-only.

45.1.4 Artboard.rulerOrigin

`artboard.rulerOrigin`

Description

Ruler origin of the artboard, relative to the top left corner of the artboard.

Type

Point

45.1.5 Artboard.rulerPAR

`artboard.rulerPAR`

Description

Pixel aspect ratio, used in ruler visualization if the units are pixels.

Range: 0.1 to 10.

Type

Number (double)

45.1.6 Artboard.showCenter

`artboard.showCenter`

Description

Show center mark.

Type

Boolean

45.1.7 Artboard.showCrossHairs

`artboard.showCrossHairs`

Description

Show cross hairs.

Type

Boolean

45.1.8 Artboard.showSafeAreas

`artboard.showSafeAreas`

Description

Show title and action safe areas (for video).

Type

Boolean

45.1.9 Artboard.typename

`artboard.typename`

Description

Read-only. The class name of this object.

Type

String

45.2 Methods

45.2.1 Artboard.remove()

`artboard.remove()`

Description

Deletes this artboard object. You cannot remove the last artboard in a document.

Returns

Nothing.

CHAPTER 46

Artboards

`artboards`

Description

A collection of Artboard objects.

46.1 Properties

46.1.1 Artboards.length

`artboards.length`

Description

The number of datasets in the collection

Type

Number; read-only.

46.1.2 Artboards.parent

`artboards.parent`

Description

The name of the object that contains this dataset

Type

Artboard; read-only.

46.1.3 Artboards.typename

`artboards.typename`

Description

The class name of the referenced object.

Type

String; read-only.

46.2 Methods

46.2.1 Artboards.add()

`artboards.add(artboardRect)`

Description

Creates a new Artboard object.

Parameters

Parameter	Type	Description
`artboardRect`	Rect	Artboard dimensions

Returns

Artboard

46.2.2 Artboards.getActiveArtboardIndex()

`artboards.getActiveArtboardIndex()`

Description

Retrieves the index position of the active artboard in the document's list.

Returns the 0-based index.

Returns

Number (long)

46.2.3 Artboards.getByName()

```
artboards.getByName(name)
```

Description

Gets the first element in the collection with the specified name.

Parameters

Parameter	Type	Description
`name`	String	Name of element to get

Returns

Artboard

46.2.4 Artboards.insert()

```
artboards.insert(artboardRect, index)
```

Description

Creates a new Artboard object and inserts it at the given index in the list.

Parameters

Parameter	Type	Description
`artboardRect`	Rect	Artboard dimensions
`index`	Number (long)	Index to insert artboard at

Returns

Nothing.

46.2.5 Artboards.remove()

```
artboards.remove(index)
```

Description

Deletes an artboard object. You cannot remove the last artboard in a document.

Parameters

Parameter	Type	Description
`index`	Number (long)	Index of artboard to remove

Returns

Nothing.

46.2.6 Artboards.setActiveArtboardIndex()

```
artboards.setActiveArtboardIndex(index)
```

Description

Makes a specific artboard active and makes it current in the iteration order.

Parameters

Parameter	Type	Description
`index`	Number (long)	Index of artboard to set active

Returns

Nothing.

CHAPTER 47

Brush

`app.activeDocument.brushes[index`

Description

A brush in an Illustrator document. Brushes are contained in documents. Additional brushes may be created by the user within Illustrator. You can access brushes within a script, but you cannot create them.

47.1 Properties

47.1.1 Brush.name

`app.activeDocument.brushes[index].name`

Description

The name of the brush

Type

String

47.1.2 Brush.parent

`app.activeDocument.brushes[index].parent`

Description

The document that contains this brush.

Type

Document; read-only.

47.1.3 Brush.typename

`app.activeDocument.brushes[index].typename`

Description

The class name of the referenced object.

Type

String; read-only.

47.2 Methods

47.2.1 Brush.applyTo()

`app.activeDocument.brushes[index].applyTo(artItem)`

Description

Applies the `brush` to a specific art item.

Parameters

Parameter	Type	Description
`artItem`	*PageItem*	Art item to apply brush to

Returns

Nothing.

47.3 Example

47.3.1 Applying a Brush

```
// Duplicates and groups all items in the current selection,
// then applies the same brush to each item in the group

if (app.documents.length > 0) {
  var docSelection = app.activeDocument.selection;
  if (docSelection.length > 0) {
    var newGroup = app.activeDocument.groupItems.add();

    for (var i = 0; i < docSelection.length; i++) {
      var newItem = docSelection[i].duplicate();
```

(continues on next page)

(continued from previous page)

```
      newItem.moveToBeginning(newGroup);
    }

    var brush = app.activeDocument.brushes[1];
    brush.applyTo(newGroup);
  }
}
```

CHAPTER 48

Brushes

`app.activeDocument.brushes`

Description

A collection of brush objects in a document.

48.1 Properties

48.1.1 Brushes.length

`app.activeDocument.brushes.length`

Description

The number of objects in the collection

Type

Number; read-only.

48.1.2 Brushes.parent

`app.activeDocument.brushes.parent`

Description

The document that contains this brushes collection.

Type

Object; read-only.

48.1.3 Brushes.typename

```
app.activeDocument.brushes.typename
```

Description

The class name of the referenced object.

Type

String; read-only.

48.2 Methods

48.2.1 Brushes.getByName()

```
app.activeDocument.brushes.getByName(name)
```

Description

Gets the first element in the collection with the specified name.

Parameters

Parameter	Type	Description
`name`	String	Name of element to get

Returns

Brush

48.2.2 Brushes.index()

```
app.activeDocument.brushes.index(itemKey)
```

Description

Gets an element from the collection.

Parameters

Parameter	Type	Description
`itemKey`	String, Number	String or number key

Returns

Brush

48.3 Example

48.3.1 Counting brushes

```
// Counts all brushes in the active document

if (app.documents.length > 0) {
  var numberOfBrushes = app.activeDocument.brushes.length;
}
```

CHAPTER 49

CharacterAttributes

`characterAttributes`

Description

Specifies the properties of a character contained in a text frame. A `CharacterStyle` object associates these attributes with a specific text range through its `characterAttributes` property.

Note: Character attributes do not have default values, and are undefined until explicitly set.

// todo: get the absolute path to characterAttributes.. `document.textFrames.textRef.textRange.characters[index].characterAttributes` ?

49.1 Properties

49.1.1 CharacterAttributes.akiLeft

`characterAttributes.akiLeft`

Description

The amount of inter-character spacing to be added to the left side of the character, in thousandths of an em (that amount will not compress or expand during full-justification).

Type

Number (double)

49.1.2 CharacterAttributes.akiRight

`characterAttributes.akiRight`

Description

The amount of inter-character spacing to be added to the right side of the character, in thousandths of an em (that amount will not compress or expand during full-justification).

Type

Number (double)

49.1.3 CharacterAttributes.alignment

`characterAttributes.alignment`

Description

The character alignment type.

Type

StyleRunAlignmentType

49.1.4 CharacterAttributes.alternateGlyphs

`characterAttributes.alternateGlyphs`

Description

The alternate glyphs form.

Type

AlternateGlyphsForm

49.1.5 CharacterAttributes.autoLeading

`characterAttributes.autoLeading`

Description

If `true`, the automatic leading should be used.

Type

Boolean

49.1.6 CharacterAttributes.baselineDirection

`characterAttributes.baselineDirection`

Description

The Japanese text baseline direction.

Type

BaselineDirectionType

49.1.7 CharacterAttributes.baselinePosition

`characterAttributes.baselinePosition`

Description

The baseline position of text.

Type

FontBaselineOption

49.1.8 CharacterAttributes.baselineShift

`characterAttributes.baselineShift`

Description

The amount of shift in points of the text baseline.

Type

Number (double)

49.1.9 CharacterAttributes.capitalization

`characterAttributes.capitalization`

Description

The case of text.

Type

FontCapsOption

49.1.10 CharacterAttributes.connectionForms

`characterAttributes.connectionForms`

Description

If `true`, the OpenType® connection forms should be used.

Type

Boolean

49.1.11 CharacterAttributes.contextualLigature

`characterAttributes.contextualLigature`

Description

If `true`, the contextual ligature should be used.

Type

Boolean

49.1.12 CharacterAttributes.discretionaryLigature

`characterAttributes.discretionaryLigature`

Description

If `true`, the discretionary ligature should be used.

Type

Boolean

49.1.13 CharacterAttributes.figureStyle

`characterAttributes.figureStyle`

Description

The number style in a OpenType font.

Type

FigureStyleType

49.1.14 CharacterAttributes.fillColor

`characterAttributes.fillColor`

Description

The color of the text fill.

Type

Color

49.1.15 CharacterAttributes.fractions

`characterAttributes.fractions`

Description

If `true`, the OpenType fractions should be used.

Type

Boolean

49.1.16 CharacterAttributes.horizontalScale

`characterAttributes.horizontalScale`

Description

The character horizontal scaling factor expressed as a percentage (100 = 100%).

Type

Number (double)

49.1.17 CharacterAttributes.italics

`characterAttributes.italics`

Description

If `true`, the Japanese OpenType font supports italics.

Type

Boolean

49.1.18 CharacterAttributes.kerningMethod

`characterAttributes.kerningMethod`

Description

The automatic kerning method to use.

Type

AutoKernType

49.1.19 CharacterAttributes.language

`characterAttributes.language`

Description

The language of text.

Type

LanguageType

49.1.20 CharacterAttributes.leading

`characterAttributes.leading`

Description

The amount of space between two lines of text, in points.

Type

Number (double)

49.1.21 CharacterAttributes.ligature

`characterAttributes.ligature`

Description

If `true`, the ligature should be used.

Type

Boolean

49.1.22 CharacterAttributes.noBreak

`characterAttributes.noBreak`

Description

If `true`, line breaks are not allowed.

Type

Boolean

49.1.23 CharacterAttributes.openTypePosition

`characterAttributes.openTypePosition`

Description

The OpenType baseline position.

Type

FontOpenTypePositionOption

49.1.24 CharacterAttributes.ordinals

`characterAttributes.ordinals`

Description

If `true`, the OpenType ordinals should be used.

Type

Boolean

49.1.25 CharacterAttributes.ornaments

`characterAttributes.ornaments`

Description

If `true`, the OpenType ornaments should be used.

Type

Boolean

49.1.26 CharacterAttributes.overprintFill

`characterAttributes.overprintFill`

Description

If `true`, the fill of the text should be overprinted.

Type

Boolean

49.1.27 CharacterAttributes.overprintStroke

`characterAttributes.overprintStroke`

Description

If `true`, the stroke of the text should be overprinted.

Type

Boolean

49.1.28 CharacterAttributes.parent

`characterAttributes.parent`

Description

The object's container.

Type

Object, read-only.

49.1.29 CharacterAttributes.proportionalMetrics

`characterAttributes.proportionalMetrics`

Description

If `true`, the Japanese OpenType font supports proportional glyphs.

Type

Boolean

49.1.30 CharacterAttributes.rotation

`characterAttributes.rotation`

Description

The character rotation angle in degrees.

Type

Number (double)

49.1.31 CharacterAttributes.size

`characterAttributes.size`

Description

Font size in points.

Type

Number (double)

49.1.32 CharacterAttributes.strikeThrough

`characterAttributes.strikeThrough`

Description

If `true`, characters use strike-through style.

Type

Boolean

49.1.33 CharacterAttributes.strokeColor

`characterAttributes.strokeColor`

Description

The color of the text stroke.

Type

Color

49.1.34 CharacterAttributes.strokeWeight

`characterAttributes.strokeWeight`

Description

Line width of stroke.

Type

Number (double)

49.1.35 CharacterAttributes.stylisticAlternates

`characterAttributes.stylisticAlternates`

Description

If `true`, the OpenType stylistic alternates should be used.

Type

Boolean

49.1.36 CharacterAttributes.swash

`characterAttributes.swash`

Description

If `true`, the OpenType swash should be used.

Type

Boolean

49.1.37 CharacterAttributes.tateChuYokoHorizontal

`characterAttributes.tateChuYokoHorizontal`

Description

The Tate-Chu-Yoko horizontal adjustment in points.

Type

Number (long)

49.1.38 CharacterAttributes.tateChuYokoVertical

`characterAttributes.tateChuYokoVertical`

Description

The Tate-Chu-Yoko vertical adjustment in points.

Type

Number (long)

49.1.39 CharacterAttributes.textFont

`characterAttributes.textFont`

Description

The text font.

Type

TextFont

49.1.40 CharacterAttributes.titling

`characterAttributes.titling`

Description

If `true`, the OpenType titling alternates should be used.

Type

Boolean

49.1.41 CharacterAttributes.tracking

`characterAttributes.tracking`

Description

The tracking or range kerning amount, in thousandths of an em.

Type

Number (long)

49.1.42 CharacterAttributes.Tsume

`characterAttributes.Tsume`

Description

The percentage of space reduction around a Japanese character.

Type

Number (double)

49.1.43 CharacterAttributes.typename

`characterAttributes.typename`

Description

The class name of the object.

Type

String, read-only.

49.1.44 CharacterAttributes.underline

`characterAttributes.underline`

Description

If `true`, characters are underlined.

Type

Boolean

49.1.45 CharacterAttributes.verticalScale

`characterAttributes.verticalScale`

Description

Character vertical scaling factor expressed as a percentage (= 100%).

Type

Number (double)

49.1.46 CharacterAttributes.wariChuCharactersAfterBreak

`characterAttributes.wariChuCharactersAfterBreak`

Description

Specifies how the characters in Wari-Chu text (an inset note in Japanese text) are divided into two or more lines.

Type

Number (long)

49.1.47 CharacterAttributes.wariChuCharactersBeforeBreak

`characterAttributes.wariChuCharactersBeforeBreak`

Description

Specifies how the characters in Wari-Chu text (an inset note in Japanese text) are divided into two or more lines.

Type

Number (long)

49.1.48 CharacterAttributes.waiChuEnabled

`characterAttributes.waiChuEnabled`

Description

If `true`, Wari-Chu is enabled.

Type

Boolean

49.1.49 CharacterAttributes.wariChuJustification

`characterAttributes.wariChuJustification`

Description

The Wari-Chu justification.

Type

WariChuJustificationType

49.1.50 CharacterAttributes.wariChuLineGap

`characterAttributes.wariChuLineGap`

Description

The Wari-Chu line gap.

Type

Number (long)

49.1.51 CharacterAttributes.wariChuLines

`characterAttributes.wariChuLines`

Description

The number of Wari-Chu (multiple text lines fit into a space meant for one) lines.

Type

Number (long)

49.1.52 CharacterAttributes.wariChuScale

`characterAttributes.wariChuScale`

Description

The Wari-Chu scale.

Type

Number (double)

49.2 Example

49.2.1 Setting character attributes

```
// Creates a new document, adds a simple text item
// then incrementally increases the horizontal and
// vertical scale attributes of each character

var docRef = documents.add();
var textRef = docRef.textFrames.add();
textRef.contents = "I Love Scripting!";
textRef.top = 400;
textRef.left = 100;

// incrementally increase the scale of each character
```

(continues on next page)

(continued from previous page)

```
var charCount = textRef.textRange.characters.length;
var size = 100;
for (var i = 0; i < charCount; i++, size *= 1.2) {
  textRef.textRange.characters[i].characterAttributes.horizontalScale = size;
  textRef.textRange.characters[i].characterAttributes.verticalScale = size;
}
```

CHAPTER 50

Characters

`app.activeDocument.textFrames[index].contents`

Description

A collection of characters (`TextRange` objects of length 1).

The elements are not named; you must access them by index.

50.1 Properties

50.1.1 Characters.length

`app.activeDocument.textFrames[index].contents.length`

Description

The number of characters in the collection.

Type

Number; read-only.

50.1.2 Characters.parent

`app.activeDocument.textFrames[index].contents.parent`

Description

The text art item that contains this character.

Type

Object; read-only.

50.1.3 Characters.typename

```
app.activeDocument.textFrames[index].contents.typename
```

Description

The class name of the referenced object.

Type

String; read-only.

50.2 Methods

50.2.1 Characters.add()

```
app.activeDocument.textFrames[index].contents.add(contents[,relativeObject][,
insertionLocation])
```

Description

Adds a new character with specified text contents at the specified location in the current document.

If a location is not specified, adds the new character to the containing text frame after the current text selection or insertion point.

Parameters

Parameter	Type	Description
`contents`	String	Text contents to add
`relativeObject`	*TextFrameItem*, optional	Object to add item to
`insertionLocation`	*ElementPlacement*, optional	Location to place text

Returns

TextRange

50.2.2 Characters.addBefore()

```
app.activeDocument.textFrames[index].contents.addBefore(contents)
```

Description

Adds a character before the specified text selection.

Parameters

Parameter	Type	Description
`contents`	String	Text contents to add

Returns

TextRange

50.2.3 Characters.index()

`app.activeDocument.textFrames[index].contents.index(itemKey)`

Description

Gets an element from the collection.

Parameters

Parameter	Type	Description
`itemKey`	String, Number	String or number key

Returns

TextRange

50.2.4 Characters.removeAll()

`app.activeDocument.textFrames[index].contents.removeAll()`

Description

Deletes all elements in the collection.

Returns

Nothing.

50.3 Example

50.3.1 Counting characters

```
// Counts all characters in the active document,
// including whitespace, and stores in numChars

if (app.documents.length > 0) {
  var doc = app.activeDocument;
  var numChars = 0;
  for (var i = 0; i < doc.textFrames.length; i++) {
    var textArtRange = doc.textFrames[i].contents;
    numChars += textArtRange.length;
  }
}
```

CHAPTER 51

CharacterStyle

`characterStyle`

Description

Associates character attributes with characters. For an example, see *CharacterStyles*.

51.1 Properties

51.1.1 CharacterStyle.characterAttributes

`characterStyle.characterAttributes`

Description

The character properties for the style.

Type

CharacterAttributes, read-only.

51.1.2 CharacterStyle.name

`characterStyle.name`

Description

The character style's name.

Type

String

51.1.3 CharacterStyle.parent

```
characterStyle.parent
```

Description

The object's container.

Type

Object, read-only.

51.1.4 CharacterStyle.typename

```
characterStyle.typename
```

Description

The class name of the object.

Type

String, read-only.

51.2 Methods

51.2.1 CharacterStyle.applyTo()

```
characterStyle.applyTo(textItem [,clearingOverrides])
```

Description

Applies the character style to the text object or objects.

Paramaters

Parameter	Type	Description
`textItem`	Object	Text item to apply style to
`clearingOverrides`	Boolean, optional	Whether to clear overrides

Returns

Nothing

51.2.2 CharacterStyle.remove()

```
characterStyle.remove()
```

Description

Deletes the object.

Returns

Nothing.

CHAPTER 52

CharacterStyles

`app.activeDocument.characterStyles`

Description

A collection of CharacterStyle objects.

52.1 Properties

52.1.1 CharacterStyles.length

`app.activeDocument.characterStyles.length`

Description

The number of characters in the collection.

Type

Number; read-only.

52.1.2 CharacterStyles.parent

`app.activeDocument.characterStyles.parent`

Description

The object's container.

Type

Object; read-only.

52.1.3 CharacterStyles.typename

```
app.activeDocument.characterStyles.typename
```

Description

The class name of the object.

Type

String; read-only.

52.2 Methods

52.2.1 CharacterStyles.add()

```
add(name)
```

Description

Creates a named character style.

Parameters

Parameter	Type	Description
`name`	String	Element name to create

Returns

CharacterStyle

52.2.2 CharacterStyles.getByName()

```
getByName(name)
```

Description

Gets the first element in the collection with the provided name.

Parameters

Parameter	Type	Description
`name`	String	Name of element to get

Returns

CharacterStyle

52.2.3 CharacterStyles.index()

`index(itemKey)`

Description

Gets an element from the collection.

Parameters

Parameter	Type	Description
`itemKey`	String, Number	String or number key

Returns

CharacterStyle

52.2.4 CharacterStyles.removeAll()

`removeAll()`

Description

Deletes all elements in this collection.

Returns

Nothing.

52.3 Example

52.3.1 Using characters styles

```
// Creates 3 text frames in a new document then creates
// a character style and applies it to each text frame.

var docRef = documents.add();
var textRef1 = docRef.textFrames.add();
textRef1.contents = "Scripting is fun!";
textRef1.top = 700;
textRef1.left = 50;

var textRef2 = docRef.textFrames.add();
textRef2.contents = "Scripting is easy!";
textRef2.top = 625;
textRef2.left = 100;

var textRef3 = docRef.textFrames.add();
textRef3.contents = "Everyone should script!";
textRef3.top = 550;
textRef3.left = 150;
```

(continues on next page)

(continued from previous page)

```
redraw();

// Create a new character style
var charStyle = docRef.characterStyles.add("BigRed");

// set character attributes
var charAttr = charStyle.characterAttributes;
charAttr.size = 40;
charAttr.tracking = -50;
charAttr.capitalization = FontCapsOption.ALLCAPS;

var redColor = new RGBColor();
redColor.red = 255;
redColor.green = 0;
redColor.blue = 0;
charAttr.fillColor = redColor;

// apply to each textFrame in the document
charStyle.applyTo(textRef1.textRange);
charStyle.applyTo(textRef2.textRange);
charStyle.applyTo(textRef3.textRange);
```

CHAPTER 53

CMYKColor

`new cmykColor()`

Description

A CMYK color specification, used where a `color` object is required.

If the color space of a document is `RGB` and you specify the color value for a page item in that document using CMYK, Illustrator will translate the CMYK color specification into an RGB color specification. The same thing happens if the document's color space is CMYK and you specify colors using RGB. Since this translation can lose information, you should specify colors using the class that matches the document's actual color space.

53.1 Properties

53.1.1 CMYKColor.black

`cmykColor.black`

Description

The black color value. Range 0.0–100.0. Default: 0.0

Type

Number (double)

53.1.2 CMYKColor.cyan

`cmykColor.cyan`

Description

The cyan color value. Range 0.0–100.0. Default: 0.0

Type

Number (double)

53.1.3 CMYKColor.magenta

`cmykColor.magenta`

Description

The magenta color value. Range 0.0–100.0. Default: 0.0

Type

Number (double)

53.1.4 CMYKColor.typename

`cmykColor.typename`

Description

The class name of the referenced object.

Type

String; read-only.

53.1.5 CMYKColor.yellow

`cmykColor.yellow`

Description

The yellow color value. Range 0.0–100.0. Default: 0.0

Type

Number (double)

53.2 Example

53.2.1 Setting a CMYK color

```
// Sets the fill color of the frontmost path item in
// the current document to a light purple CMYK color

if (app.documents.length > 0 && app.activeDocument.pathItems.length > 0) {
  var frontPath = app.activeDocument.pathItems[0];

  // Set color values for the CMYK object
  var newCMYKColor = new cmykColor();
  newCMYKColor.black = 0;
  newCMYKColor.cyan = 30.4;
  newCMYKColor.magenta = 32;
  newCMYKColor.yellow = 0;

  // Use the color object in the path item
  frontPath.filled = true;
  frontPath.fillColor = newCMYKColor;
}
```

CHAPTER 54

Color

`color`

Description

An abstract parent class for all color classes used in Illustrator.

Subclasses are:

54.1 GradientColor

`gradientColor`

Description

A gradient color specification in a Gradient object. A script can create a new gradient color using a reference to an existing gradient in the document. If no existing gradient object is referenced, a default gradient is supplied.

54.1.1 Properties

GradientColor.angle

`gradientColor.angle`

Description

The gradient vector angle in degrees. Default: 0.0.

Type

Number (double).

GradientColor.gradient

`gradientColor.gradient`

Description

Reference to the object defining the gradient.

Type

Gradient

GradientColor.hiliteAngle

`gradientColor.hiliteAngle`

Description

The gradient highlight vector angle in degrees.

Type

Number (double).

GradientColor.hiliteLength

`gradientColor.hiliteLength`

Description

The gradient highlight vector length.

Type

Number (double).

GradientColor.length

`gradientColor.length`

Description

The gradient vector length.

Type

Number (double).

GradientColor.matrix

`gradientColor.matrix`

Description

An additional transformation matrix to manipulate the gradient path.

Type

Matrix.

GradientColor.origin

`gradientColor.origin`

Description

The gradient vector origin, the center point of the gradient in this color.

Type

Array of 2 numbers.

GradientColor.typename

`gradientColor.typename`

Description

The class name of the referenced object.

Type

String, read-only.

54.1.2 Example

Changing a gradient stop color

```
// Creates a new RGB document, then changes the color of the first gradient stop of␣
↪an indexed gradient
app.documents.add(DocumentColorSpace.RGB);

// Get a reference to the gradient that you want to change
var gradientRef = app.activeDocument.gradients[1];

// Create the new color
var startColor = new RGBColor();
startColor.red = 255;
startColor.green = 238;
startColor.blue = 98;
```

(continues on next page)

(continued from previous page)

```
// apply new color to the first gradient stop
gradientRef.gradientStops[0].color = startColor;
```

54.2 GrayColor

```
new GrayColor()
```

Description

A grayscale color specification, used where a `color` object is required.

54.2.1 Properties

GrayColor.gray

```
grayColor.gray
```

Description

The tint of the gray. Range: 0.0 to 100.0, where 0.0 is black and 100.0 is white.

Type

Number (double).

GrayColor.typename

```
grayColor.typename
```

Description

The class name of the referenced object.

Type

String, read-only.

54.2.2 Example

Changing a color to gray

```
// Sets the color of the first word in the active document to a shade of gray

if (app.documents.length > 0 && app.activeDocument.textFrames.length > 0) {
  var text = app.activeDocument.textFrames[0].textRange;
  var firstWord = text.words[0];

  // Create the new color
  var textColor = new GrayColor();
  textColor.gray = 45;

  firstWord.filled = true;
  firstWord.fillColor = textColor;
}
```

54.3 LabColor

`labColor`

Description

A color specification in the CIE Lab color space, used where a `color` object is required.

54.3.1 Properties

LabColor.a

`labColor.a`

Description

The a (red-green) color value. Range -128.0–128.0. Default: 0.0.

Type

Number (double).

LabColor.b

`labColor.b`

Description

The b (yellow-blue) color value. Range -128.0–128.0. Default: 0.0.

Type

Number (double).

LabColor.l

`labColor.l`

Description

The l (lightness) color value. Range -128.0–128.0. Default: 0.0.

Type

Number (double).

54.4 NoColor

`new NoColor()`

Description

Represents the "none" color. Assigning a `NoColor` object to the fill or stroke color of an art item is equivalent to setting the `filled` or `stroked` property to `false`.

54.4.1 Properties

NoColor.typename

`noColor.typename`

Description

The class name of the object.

Type

String, read-only.

54.4.2 Example

Using NoColor to remove a fill color

```
// Creates 2 overlapping objects with different fill colors.
// Assign the top object a fill color of "NoColor"
// allowing the bottom object to become visible.

// Create 2 overlapping objects one blue, one red;
var docRef = documents.add();
var itemRef1 = docRef.pathItems.rectangle(500, 200, 200, 100);
var itemRef2 = docRef.pathItems.rectangle(550, 150, 200, 200);
var rgbColor = new RGBColor();
rgbColor.red = 255;
itemRef2.fillColor = rgbColor;

rgbColor.blue = 255;
```

(continues on next page)

(continued from previous page)

```
rgbColor.red = 0;
itemRef1.fillColor = rgbColor;
redraw();

// create a nocolor and assign it to the top object
var noColor = new NoColor();
itemRef2.fillColor = noColor;
redraw();
```

54.5 PatternColor

`patternColor`

Description

A pattern color specification. You can create a new pattern color by modifying an existing pattern in the document. Any modification you make to a pattern affects that pattern in the Palette.

`PatternColor` objects can be used in any property that takes a color object, such as `fillColor` or `strokeColor`.

54.5.1 Properties

PatternColor.matrix

`patternColor.matrix`

Description

Additional transformation arising from manipulating the path.

Type

Matrix

PatternColor.pattern

`patternColor.pattern`

Description

A reference to the pattern object that defines the pattern to use in this color definition.

Type

Pattern

PatternColor.reflect

`patternColor.reflect`

Description

If `true`, the prototype should be reflected before filling.

Default: `false`

Type

Boolean

PatternColor.reflectAngle

`patternColor.reflectAngle`

Description

The axis around which to reflect, in points.

Default: 0.0

Type

Number (double)

PatternColor.rotation

`patternColor.rotation`

Description

The angle in radians to rotate the prototype pattern before filling.

Default: 0.0

Type

Number (double)

PatternColor.scaleFactor

`patternColor.scaleFactor`

Description

The fraction to which to scale the prototype pattern before filling, represented as a point containing horizontal and vertical scaling percentages.

Type

Array of 2 numbers

PatternColor.shearAngle

`patternColor.shearAngle`

Description

The angle in radians by which to slant the shear.

Default: 0.0

Type

Number (double)

PatternColor.shearAxis

`patternColor.shearAxis`

Description

The axis to shear with respect to, in points.

Default: 0.0

Type

Number (double)

PatternColor.shiftAngle

`patternColor.shiftAngle`

Description

The angle in radians to which to translate the unscaled prototype pattern before filling.

Default: 0.0

Type

Number (double)

PatternColor.shiftDistance

`patternColor.shiftDistance`

Description

The distance in points to which to translate the unscaled prototype pattern before filling.

Default: 0.0

Type

Number (double)

PatternColor.typename

`patternColor.typename`

Description

The class name of the referenced object.

Type

String; read-only.

54.5.2 Example

Modifying and applying pattern colors

```
// Rotates the color of each pattern in the current document,
// then applies the last pattern to the first path item
if (app.documents.length > 0 && app.activeDocument.pathItems.length > 0) {
  var doc = app.activeDocument;
  var swatchIndex = 0;

  for (i = 0; i < doc.swatches.length; i++) {
    // Get the generic color object of the swatch
    var currentSwatch = doc.swatches[i];
    var swatchColor = currentSwatch.color;

    // Only operate on patterns
    if (currentSwatch.color.typename == "PatternColor") {

      // Change a pattern property
      currentSwatch.color.rotation = 10;
      swatchIndex = i;
    }
  }

  // Apply the last pattern color swatch to the frontmost path
  var firstPath = app.activeDocument.pathItems[0];
  firstPath.filled = true;
  firstPath.fillColor = doc.swatches[swatchIndex].color;
}
```

54.6 RGBColor

`new RGBColor()`

Description

An RGB color specification, used to apply an RGB color to a layer or art item.

If the color space of a document is RGB and you specify the color value for a page item in that document using CMYK, Illustrator will translate the CMYK color specification into an RGB color specification. The same thing happens if the document's color space is CMYK and you specify colors using RGB. Since this translation can lose information, you should specify colors using the class that matches the document's actual color space.

54.6.1 Properties

RGBColor.blue

`rgbColor.blue`

Description

The blue color value. Range: 0.0 to 255.0.

Type

Number (double).

RGBColor.green

`rgbColor.green`

Description

The green color value. Range: 0.0 to 255.0.

Type

Number (double).

RGBColor.red

`rgbColor.red`

Description

The red color value. Range: 0.0 to 255.0.

Type

Number (double).

RGBColor.typename

`rgbColor.typename`

Description

The class name of the referenced object.

Type

String, read-only.

54.6.2 Example

Setting an RGB color

```
// Sets the default fill color in the current document to yellow.

if (app.documents.length > 0) {
  // Define the new color
  var newRGBColor = new RGBColor();
  newRGBColor.red = 255;
  newRGBColor.green = 255;
  newRGBColor.blue = 0;

  app.activeDocument.defaultFillColor = newRGBColor;
}
```

54.7 SpotColor

`new SpotColor()`

Description

Color class used to apply the color value of a spot at a specified tint value. Can be used in any property that takes a color object.

54.7.1 Properties

SpotColor.spot

`spotColor.spot`

Description

A reference to the spot color object that defines the color.

Type

Spot

SpotColor.tint

`spotColor.tint`

Description

The tint of the color. Range: 0.0 to 100.0

Type

Number (double)

SpotColor.typename

```
spotColor.typename
```

Description

The class name of the referenced object.

Type

String; read-only.

CHAPTER 55

CompoundPathItem

`app.activeDocument.activeLayer.compoundPathItems[index`

Description

A compound path. These objects are composed of multiple intersecting paths, resulting in transparent interior spaces where the component paths overlap. The pathItems property provides access to the paths that make up the compound path.

Paths contained within a compound path or group in a document are returned as individual paths when a script asks for the paths contained in the document. However, paths contained in a compound path or group are not returned when a script asks for the paths in a layer that contains the compound path or group.

All paths within a compound path share property values. Therefore, if you set the value of a property of any one of the paths in the compound path, the properties of all other component paths are updated with the new value.

55.1 Properties

55.1.1 CompoundPathItem.artworkKnockout

`app.activeDocument.activeLayer.compoundPathItems[index].artworkKnockout`

Description

Is this object used to create a knockout, and if so, what kind of knockout.

Type

KnockoutState

55.1.2 CompoundPathItem.blendingMode

`app.activeDocument.activeLayer.compoundPathItems[index].blendingMode`

Description

The mode used when compositing an object.

Type

BlendModes

55.1.3 CompoundPathItem.controlBounds

`app.activeDocument.activeLayer.compoundPathItems[index].controlBounds`

Description

The bounds of the object including stroke width and controls.

Type

Array of 4 numbers, read-only.

55.1.4 CompoundPathItem.editable

`app.activeDocument.activeLayer.compoundPathItems[index].editable`

Description

If `true`, this item is editable.

Type

Boolean, read-only.

55.1.5 CompoundPathItem.geometricBounds

`app.activeDocument.activeLayer.compoundPathItems[index].geometricBounds`

Description

The bounds of the object excluding stroke width.

Type

Array of 4 numbers, read-only.

55.1.6 CompoundPathItem.height

`app.activeDocument.activeLayer.compoundPathItems[index].height`

Description

The height of the compound path item excluding stroke width.

Type

Number (double).

55.1.7 CompoundPathItem.hidden

`app.activeDocument.activeLayer.compoundPathItems[index].hidden`

Description

If `true`, this compound path item is hidden.

Type

Boolean.

55.1.8 CompoundPathItem.isIsolated

`app.activeDocument.activeLayer.compoundPathItems[index].isIsolated`

Description

If `true`, this object is isolated.

Type

Boolean.

55.1.9 CompoundPathItem.layer

`app.activeDocument.activeLayer.compoundPathItems[index].layer`

Description

The layer to which this compound path item belongs.

Type

Layer, read-only.

55.1.10 CompoundPathItem.left

`app.activeDocument.activeLayer.compoundPathItems[index].left`

Description

The position of the left side of the item (in points, measured from the left side of the page).

Type

Number (double).

55.1.11 CompoundPathItem.locked

`app.activeDocument.activeLayer.compoundPathItems[index].locked`

Description

If `true`, this compound path item is locked.

Type

Boolean.

55.1.12 CompoundPathItem.name

`app.activeDocument.activeLayer.compoundPathItems[index].name`

Description

The name of this compound path item.

Type

String.

55.1.13 CompoundPathItem.note

`app.activeDocument.activeLayer.compoundPathItems[index].note`

Description

The note assigned to this item.

Type

String.

55.1.14 CompoundPathItem.opacity

`app.activeDocument.activeLayer.compoundPathItems[index].opacity`

Description

The opacity of the object. Range: 0.0 to 100.0

Type

Number (double).

55.1.15 CompoundPathItem.parent

`app.activeDocument.activeLayer.compoundPathItems[index].parent`

Description

The parent of this object.

Type

Layer or *GroupItem*, read-only.

55.1.16 CompoundPathItem.pathItems

`app.activeDocument.activeLayer.compoundPathItems[index].pathItems`

Description

The path art items in this compound path.

Type

PathItems, read-only.

55.1.17 CompoundPathItem.position

`app.activeDocument.activeLayer.compoundPathItems[index].position`

Description

The position (in points) of the top left corner of the `compoundPathItem` object in the format [x, y]. Does not include stroke weight.

Type

Array of 2 numbers.

55.1.18 CompoundPathItem.selected

`app.activeDocument.activeLayer.compoundPathItems[index].selected`

Description

If `true`, this compound path item is selected.

Type

Boolean.

55.1.19 CompoundPathItem.sliced

`app.activeDocument.activeLayer.compoundPathItems[index].sliced`

Description

If `true`, the item is sliced. Default: `false`

Type

Boolean.

55.1.20 CompoundPathItem.tags

`app.activeDocument.activeLayer.compoundPathItems[index].tags`

Description

The tags contained in this object.

Type

Tags, read-only.

55.1.21 CompoundPathItem.top

`app.activeDocument.activeLayer.compoundPathItems[index].top`

Description

The position of the top of the item (in points, measured from the bottom of the page).

Type

Number (double).

55.1.22 CompoundPathItem.typename

`app.activeDocument.activeLayer.compoundPathItems[index].typename`

Description

The class name of the referenced object.

Type

String, read-only.

55.1.23 CompoundPathItem.uRL

`app.activeDocument.activeLayer.compoundPathItems[index].uRL`

Description

The value of the Adobe URL tag assigned to this compound path item.

Type

String.

55.1.24 CompoundPathItem.visibilityVariable

`app.activeDocument.activeLayer.compoundPathItems[index].visibilityVariable`

Description

The visibility variable bound to the item.

Type

Variant.

55.1.25 CompoundPathItem.visibleBounds

`app.activeDocument.activeLayer.compoundPathItems[index].visibleBounds`

Description

The visible bounds of the compound path item including stroke width.

Type

Array of 4 numbers, read-only.

55.1.26 CompoundPathItem.width

`app.activeDocument.activeLayer.compoundPathItems[index].width`

Description

The width of the compound path item excluding stroke width.

Type

Number (double).

55.1.27 CompoundPathItem.wrapInside

`app.activeDocument.activeLayer.compoundPathItems[index].wrapInside`

Description

If `true`, the text frame object should be wrapped inside this object.

Type

Boolean.

55.1.28 CompoundPathItem.wrapOffset

`app.activeDocument.activeLayer.compoundPathItems[index].wrapOffset`

Description

The offset to use when wrapping text around this object.

Type

Number (double).

55.1.29 CompoundPathItem.wrapped

`app.activeDocument.activeLayer.compoundPathItems[index].wrapped`

Description

If `true`, wrap text frame objects around this object (text frame must be above the object).

Type

Boolean.

55.1.30 CompoundPathItem.zOrderPosition

`app.activeDocument.activeLayer.compoundPathItems[index].zOrderPosition`

Description

The position of this art item within the stacking order of the group or layer (`Parent`) that contains the art item.

Type

Number (long), read-only.

55.2 Methods

55.2.1 CompoundPathItem.duplicate()

`app.activeDocument.activeLayer.compoundPathItems[index].duplicate([relativeObject][, insertionLocation])`

Description

Creates a duplicate of the selected object.

Parameters

Parameter	Type	Description
`relativeObject`	Object, optional	Object to duplicate to
`insertionLocation`	*ElementPlacement*, optional	Location to insert element

Returns

CompoundPathItem

55.2.2 CompoundPathItem.move()

`app.activeDocument.activeLayer.compoundPathItems[index].move(relativeObject, insertionLocation)`

Description

Moves the object.

Parameters

Parameter	Type	Description
`relativeObject`	Object	Object to move element within
`insertionLocation`	*ElementPlacement*, optional	Location to move element to

Returns

Nothing.

55.2.3 CompoundPathItem.remove()

```
app.activeDocument.activeLayer.compoundPathItems[index].remove()
```

Description

Deletes this object.

Returns

Nothing.

55.2.4 CompoundPathItem.resize()

```
app.activeDocument.activeLayer.compoundPathItems[index].resize(
    scaleX, scaleY [,changePositions] [,changeFillPatterns] [,changeFillGradients]
    [,changeStrokePattern] [,changeLineWidths] [,scaleAbout]
)
```

Description

Scales the art item where `scaleX` is the horizontal scaling factor and `scaleY` is the vertical scaling factor. 100.0 = 100%.

Parameters

Parameter	Type	Description
`scaleX`	Number (double)	Horizontal scaling factor
`scaleY`	Number (double)	Vertical scaling factor
`changePositions`	Boolean, optional	Whether to effect art object positions and orientations
`changeFillPatterns`	Boolean, optional	Whether to transform fill patterns
`changeFillGradients`	Boolean, optional	Whether to transform fill gradients
`changeStrokePattern`	Boolean, optional	Whether to transform stroke patterns
`changeLineWidths`	Number (double), optional	The amount to scale line widths
`scaleAbout`	*Transformation*, optional	The point to use as anchor, to transform about

Returns

Nothing.

55.2.5 CompoundPathItem.rotate()

```
app.activeDocument.activeLayer.compoundPathItems[index].rotate(
    angle [,changePositions] [,changeFillPatterns]
    [,changeFillGradients] [,changeStrokePattern] [,rotateAbout]
)
```

Description

Rotates the art item relative to the current rotation. The object is rotated counter-clockwise if the `angle` value is positive, clockwise if the value is negative.

Parameters

Parameter	Type	Description
`angle`	Number (double)	The angle amount to rotate the element
`changePositions`	Boolean, optional	Whether to effect art object positions and orientations
`changeFillPatterns`	Boolean, optional	Whether to transform fill patterns
`changeFillGradients`	Boolean, optional	Whether to transform fill gradients
`changeStrokePattern`	Boolean, optional	Whether to transform stroke patterns
`rotateAbout`	*Transformation*, optional	The point to use as anchor, to transform about

Returns

Nothing.

55.2.6 compoundPathItem.transform()

```
app.activeDocument.activeLayer.compoundPathItems[index].transform(
    transformationMatrix [,changePositions] [,changeFillPatterns] [,
↪changeFillGradients]
    [,changeStrokePattern] [,changeLineWidths] [,transformAbout]
)
```

Description

Transforms the art item by applying a transformation matrix.

Parameters

Parameter	Type	Description
`transformationMatrix`	*Matrix*	Transformation matrix to apply
`changePositions`	Boolean, optional	Whether to change Positions
`changeFillPatterns`	Boolean, optional	Whether to change Fill Patterns
`changeFillGradients`	Boolean, optional	Whether to change Fill Gradients
`changeStrokePattern`	Boolean, optional	Whether to change Stroke Pattern
`changeLineWidths`	Number (double), optional	The amount to scale line widths
`transformAbout`	*Transformation*, optional	The point to use as anchor, to transform about

Returns

Nothing.

55.2.7 CompoundPathItem.translate()

```
app.activeDocument.activeLayer.compoundPathItems[index].translate(
    deltaX [,deltaY] [,transformObjects] [,transformFillPatterns]
    [,transformFillGradients] [,transformStrokePatterns]
)
```

Description

Repositions the art item relative to the current position, where `deltaX` is the horizontal offset and `deltaY` is the vertical offset.

Parameters

Parameter	Type	Description
`deltaX`	Number (double), optional	Horizontal offset
`deltaY`	Number (double), optional	Vertical offset
`transformObjects`	Boolean, optional	Whether to transform Objects
`transformFillPatterns`	Boolean, optional	Whether to transform Fill Patterns
`transformFillGradients`	Boolean, optional	Whether to transform Fill Gradients
`transformStrokePatterns`	Boolean, optional	Whether to transform Stroke Patterns

Returns

Nothing.

55.2.8 CompoundPathItem.zOrder()

`app.activeDocument.activeLayer.compoundPathItems[index].zOrder(zOrderCmd)`

Description

Arranges the art item's position in the stacking order of the group or layer (parent) of this object.

Parameters

Parameter	Type	Description
`zOrderCmd`	*ZOrderMethod*	Stacking order arrangement method

Returns

Nothing.

55.3 Example

55.3.1 Selecting paths in a document

```
// Selects all paths not part of a compound path
if ( app.documents.length > 0 ) {
    var doc = app.activeDocument;
    var count = 0;
    if ( doc.pathItems.length > 0 ) {
        var thePaths = doc.pathItems;
        var numPaths = thePaths.length;
        for ( var i = 0; i < doc.pathItems.length; i++ ) {
        var pathArt = doc.pathItems[i];
        if ( pathArt.parent.typename != "compoundPathItem" ) {
            pathArt.selected = true;
            count++;
        }
    }
}
```

55.3.2 Creating and modifying a compound path item

```
// Creates a new compound path item containing 3 path
// items, then sets the width and the color of the stroke
// to all items in the compound path

if (app.documents.length > 0) {
  var doc = app.activeDocument;
  var newCompoundPath = doc.activeLayer.compoundPathItems.add();

  // Create the path items
  var newPath = newCompoundPath.pathItems.add();
  newPath.setEntirePath(Array(Array(30, 50), Array(30, 100)));

  newPath = newCompoundPath.pathItems.add();
  newPath.setEntirePath(Array(Array(40, 100), Array(100, 100)));

  newPath = newCompoundPath.pathItems.add();
  newPath.setEntirePath(Array(Array(100, 110), Array(100, 300)));

  // Set stroke and width properties of the compound path
  newPath.stroked = true;
  newPath.strokeWidth = 3.5;
  newPath.strokeColor = app.activeDocument.swatches[3].color;
}
```

CHAPTER 56

CompoundPathItems

`app.activeDocument.activeLayer.compoundPathItems`

Description

A collection of *CompoundPathItem* objects.

56.1 Properties

56.1.1 CompoundPathItems.length

`app.activeDocument.activeLayer.compoundPathItems.length`

Description

The number of objects in the collection.

Type

Number, read-only.

56.1.2 CompoundPathItems.parent

`app.activeDocument.activeLayer.compoundPathItems.parent`

Description

The parent of this collection (either a `Layer` or a `GroupItem`).

Type

Object, read-only.

56.1.3 CompoundPathItems.typename

`app.activeDocument.activeLayer.compoundPathItems.typename`

Description

The class name of the referenced object.

Type

String, read-only.

56.2 Methods

56.2.1 CompoundPathItems.add()

`app.activeDocument.activeLayer.compoundPathItems.add()`

Description

Creates a new `CompoundPathItem`.

Returns

CompoundPathItem

56.2.2 CompoundPathItems.getByName()

`app.activeDocument.activeLayer.compoundPathItems.getByName(name)`

Description

Gets the first element in the collection with the specified name.

Parameters

Parameter	Type	Description
`name`	String	Name of element to get

Returns

CompoundPathItem

56.2.3 CompoundPathItems.index()

`app.activeDocument.activeLayer.compoundPathItems.index(itemKey)`

Description

Gets an element from the collection.

Parameters

Parameter	Type	Description
`itemKey`	String, Number	String or number key

Returns

CompoundPathItem

56.2.4 CompoundPathItems.removeAll()

`app.activeDocument.activeLayer.compoundPathItems.removeAll()`

Description

Deletes all elements in this collection.

Returns

Nothing.

56.3 Example

56.3.1 Counting compound paths

```
// Counts all compound path items in layer 1 of the current document
if (app.documents.length > 0) {
  var doc = app.activeDocument;
  var numCompoundPaths = doc.layers[0].compoundPathItems.length;
}
```

CHAPTER 57

Dataset

`app.activeDocument.dataSets[index`

Description

A set of data used for dynamic publishing. A dataset allows you to collect a number of variables and their dynamic data into one object. You must have at least one variable bound to an art item in order to create a dataset. See the class *Variable*.

57.1 Properties

57.1.1 Dataset.name

`app.activeDocument.dataSets[index].name`

Description

Then name of the dataset.

Type

String.

57.1.2 Dataset.parent

`app.activeDocument.dataSets[index].parent`

Description

The name of the object that contains this dataset.

Type

Document, read-only.

57.1.3 Dataset.typename

`app.activeDocument.dataSets[index].typename`

Description

The class name of the referenced object.

Type

String.

57.2 Methods

57.2.1 Dataset.display()

`app.activeDocument.dataSets[index].display()`

Description

Displays the dataset.

Returns

Nothing.

57.2.2 Dataset.remove()

`app.activeDocument.dataSets[index].remove()`

Description

Deletes this object.

Returns

Nothing.

57.2.3 Dataset.update()

`app.activeDocument.dataSets[index].update()`

Description

Updates the dataset.

Returns

Nothing.

57.3 Example

57.3.1 Using variables and datasets

```
// Creates two variables, 1 visibility and 1 text,
// creates two datasets each with different values for the variables,
// then displays both datasets

var docRef = documents.add();

// Create visibility variable
var itemRef = docRef.pathItems.rectangle(600, 200, 150, 150);
var colorRef = new RGBColor;
colorRef.red = 255;
itemRef.fillColor = colorRef;

var visibilityVar = docRef.variables.add();
visibilityVar.kind = VariableKind.VISIBILITY;
itemRef.visibilityVariable = visibilityVar;

// Create text variable
var textRef = docRef.textFrames.add();
textRef.contents = "Text Variable, dataset 1";
textRef.top = 400;
textRef.left = 200;

var textVar = docRef.variables.add();
textVar.kind = VariableKind.TEXTUAL;
textRef.contentVariable = textVar;
redraw();

// Create dataset 1
var ds1 = docRef.dataSets.add();

// Change variable values and create dataset 2
itemRef.hidden = true;
textRef.contents = "Text Variable, dataset 2";
redraw();
var ds2 = docRef.dataSets.add();

// display each dataset
ds1.display();
redraw();
ds2.display();
redraw();
```

CHAPTER 58

Datasets

`app.activeDocument.dataSets`

Description

A collection of *Dataset* objects.

58.1 Properties

58.1.1 Datasets.length

`app.activeDocument.dataSets.length`

Description

The number of datasets in the collection.

Type

Number, read-only.

58.1.2 Datasets.parent

`app.activeDocument.dataSets.parent`

Description

The name of the object that contains this dataset.

Type

Document, read-only.

58.1.3 Datasets.typename

`app.activeDocument.dataSets.typename`

Description

The class name of the referenced object.

Type

String, read-only.

58.2 Methods

58.2.1 Datasets.add()

`app.activeDocument.dataSets.add()`

Description

Creates a new dataset object.

Returns

Dataset

58.2.2 Datasets.getByName()

`app.activeDocument.dataSets.getByName(name)`

Description

Gets the first element in the collection with the specified name.

Parameters

Parameter	Type	Description
`name`	String	Name of element to get

Returns

Dataset

58.2.3 Datasets.index()

`app.activeDocument.dataSets.index(itemKey)`

Description

Gets an element from the collection.

Parameters

Parameter	Type	Description
`itemKey`	String, Number	String or number key

Returns

Dataset

58.2.4 Datasets.removeAll()

```
app.activeDocument.dataSets.removeAll()
```

Description

Removes all elements in the collection.

Returns

Nothing.

CHAPTER 59

Document

`app.activeDocument`

Description

An Illustrator document. Documents are contained in the *Application* object.

The default document settings—those properties starting with the word "default"—are global settings that affect the current document. Be sure to modify these default properties only when a document is open. Note that if you set default properties to desired values before creating new objects, you can streamline your scripts, eliminating the need to specify specific properties such as `fillColor` and `stroked` that have default properties.

A document's color space, height, and width can only be set when the document is created. You cannot modify these properties in an existing document. See *Application.open()* for more information on how document color spaces are handled.

59.1 Properties

59.1.1 Document.activeDataset

`app.activeDocument.activeDataset`

Description

The currently opened dataset.

Type

Dataset

59.1.2 Document.activeLayer

`app.activeDocument.activeLayer`

Description

The active layer in the document.

Type

Layer

59.1.3 Document.activeView

`app.activeDocument.activeView`

Description

The document's current view.

Type

View, read-only.

59.1.4 Document.artboards

`app.activeDocument.artboards`

Description

All artboards in the document.

Type

Artboards, read-only.

59.1.5 Document.brushes

`app.activeDocument.brushes`

Description

The brushes contained in the document.

Type

Brushes, read-only.

59.1.6 Document.characterStyles

`app.activeDocument.characterStyles`

Description

The list of character styles in this document.

Type

CharacterStyles, read-only.

59.1.7 Document.compoundPathItems

`app.activeDocument.compoundPathItems`

Description

The compound path items contained in the document.

Type

CompoundPathItems, read-only.

59.1.8 Document.cropBox

`app.activeDocument.cropBox`

Description

The boundary of the document's cropping box for output, or `null` if no value has been set.

Type

Array of 4 numbers.

59.1.9 Document.cropStyle

`app.activeDocument.cropStyle`

Description

The style of the document's cropping box.

Type

CropOptions

59.1.10 Document.dataSets

`app.activeDocument.dataSets`

Description

The datasets contained in the document.

Type

Datasets, read-only.

59.1.11 Document.defaultFillColor

`app.activeDocument.defaultFillColor`

Description

The color to use to fill new paths if `defaultFilled` is `true`.

Type

Color

59.1.12 Document.defaultFilled

`app.activeDocument.defaultFilled`

Description

If `true`, a new path should be filled.

Type

Boolean.

59.1.13 Document.defaultFillOverprint

`app.activeDocument.defaultFillOverprint`

Description

If `true`, the art beneath a filled object should be overprinted by default.

Type

Boolean.

59.1.14 Document.defaultStrokeCap

`app.activeDocument.defaultStrokeCap`

Description

Default type of line capping for paths created.

Type

StrokeCap

59.1.15 Document.defaultStrokeColor

`app.activeDocument.defaultStrokeColor`

Description

The stroke color for new paths if default stroked is `true`.

Type

Color

59.1.16 Document.defaultStroked

`app.activeDocument.defaultStroked`

Description

If `true`, a new path should be stroked.

Type

Boolean.

59.1.17 Document.defaultStrokeDashes

`app.activeDocument.defaultStrokeDashes`

Description

Default lengths for dashes and gaps in dashed lines, starting with the first dash length, followed by the first gap length, and so on. Set to an empty object, {}, for solid line.

Type

Object.

59.1.18 Document.defaultStrokeDashOffset

`app.activeDocument.defaultStrokeDashOffset`

Description

The default distance into the dash pattern at which the pattern should be started for new paths.

Type

Number (double).

59.1.19 Document.defaultStrokeJoin

`app.activeDocument.defaultStrokeJoin`

Description

Default type of joints in new paths.

Type

StrokeJoin

59.1.20 Document.defaultStrokeMiterLimit

`app.activeDocument.defaultStrokeMiterLimit`

Description

When a default stroke join is set to `mitered`, this property specifies when the join will be converted to beveled (squared-off) by default. The default miter limit of 4 means that when the length of the point reaches four times the stroke weight, the join switches from a miter join to a bevel join. Range: 1 to 500; a value of 1 specifies a bevel join.

Type

Number (double).

59.1.21 Document.defaultStrokeOverprint

`app.activeDocument.defaultStrokeOverprint`

Description

If `true`, the art beneath a stroked object should be overprinted by default.

Type

Boolean.

59.1.22 Document.defaultStrokeWidth

`app.activeDocument.defaultStrokeWidth`

Description

Default width of stroke for new paths.

Type

Number (double).

59.1.23 Document.documentColorSpace

`app.activeDocument.documentColorSpace`

Description

The color specification system to use for this document's color space.

Type

DocumentColorSpace

59.1.24 Document.fullName

`app.activeDocument.fullName`

Description

The file associated with the document, which includes the complete path to the file.

Type

File, read-only.

59.1.25 Document.geometricBounds

`app.activeDocument.geometricBounds`

Description

The bounds of the illustration excluding the stroke width of any objects in the document.

Type

Array of 4 numbers, read-only.

59.1.26 Document.gradients

```
app.activeDocument.gradients
```

Description

The gradients contained in the document.

Type

Gradients, read-only.

59.1.27 Document.graphicStyles

```
app.activeDocument.graphicStyles
```

Description

The graphic styles defined in this document.

Type

GraphicStyles, read-only.

59.1.28 Document.graphItems

```
app.activeDocument.graphItems
```

Description

The graph art items in this document.

Type

GraphItems, read-only.

59.1.29 Document.groupItems

```
app.activeDocument.groupItems
```

Description

The group items contained in the document.

Type

GroupItems, read-only.

59.1.30 Document.height

`app.activeDocument.height`

Description

The height of the document.

Type

Number (double), read-only.

59.1.31 Document.inkList

`app.activeDocument.inkList`

Description

The list of inks in this document.

Type

Object, read-only.

59.1.32 Document.kinsokuSet

`app.activeDocument.kinsokuSet`

Description

The Kinsoku set of characters that cannot begin or end a line of Japanese text.

Type

Object, read-only.

59.1.33 Document.layers

`app.activeDocument.layers`

Description

The layers contained in the document.

Type

Layers, read-only.

59.1.34 Document.legacyTextItems

`app.activeDocument.legacyTextItems`

Description

The legacy text items in the document.

Type

LegacyTextItems, read-only.

59.1.35 Document.meshItems

`app.activeDocument.meshItems`

Description

The mesh art items contained in the document.

Type

MeshItems, read-only.

59.1.36 Document.mojikumiSet

`app.activeDocument.mojikumiSet`

Description

A list of names of predefined Mojikumi sets which specify the spacing for the layout and composition of Japanese text.

Type

Object, read-only.

59.1.37 Document.name

`app.activeDocument.name`

Description

The document's name (not the complete file path to the document).

Type

String, read-only.

59.1.38 Document.nonNativeItems

`app.activeDocument.nonNativeItems`

Description

The non-native art items in this document.

Type

NonNativeItems, read-only.

59.1.39 Document.outputResolution

`app.activeDocument.outputResolution`

Description

The current output resolution for the document in dots per inch (dpi).

Type

Number (double), read-only.

59.1.40 Document.pageItems

`app.activeDocument.pageItems`

Description

The page items (all art item classes) contained in the document.

Type

PageItems, read-only.

59.1.41 Document.pageOrigin

`app.activeDocument.pageOrigin`

Description

The zero-point of the page in the document without margins, relative to the overall height and width.

Type

Array of 2 numbers.

59.1.42 Document.paragraphStyles

`app.activeDocument.paragraphStyles`

Description

The list of paragraph styles in this document.

Type

ParagraphStyles, read-only.

59.1.43 Document.parent

`app.activeDocument.parent`

Description

The application that contains this document.

Type

Application, read-only.

59.1.44 Document.path

`app.activeDocument.path`

Description

The file associated with the document, which includes the complete path to the file.

Type

File, read-only.

59.1.45 Document.pathItems

`app.activeDocument.pathItems`

Description

The path items contained in this document.

Type

PathItems, read-only.

59.1.46 Document.patterns

`app.activeDocument.patterns`

Description

The patterns contained in this document.

Type

Patterns, read-only.

59.1.47 Document.placedItems

`app.activeDocument.placedItems`

Description

The placed items contained in this document.

Type

PlacedItems, read-only.

59.1.48 Document.pluginItems

`app.activeDocument.pluginItems`

Description

The plug-in items contained in this document.

Type

PluginItems, read-only.

59.1.49 Document.printTiles

`app.activeDocument.printTiles`

Description

If `true`, this document should be printed as tiled output.

Type

Boolean, read-only.

59.1.50 Document.rasterEffectSettings

`app.activeDocument.rasterEffectSettings`

Description

The document's raster effect settings.

Type

RasterEffectOptions, read-only.

59.1.51 Document.rasterItems

`app.activeDocument.rasterItems`

Description

The raster items contained in this document.

Type

RasterItems, read-only.

59.1.52 Document.rulerOrigin

`app.activeDocument.rulerOrigin`

Description

The zero-point of the rulers in the document relative to the bottom left of the document.

Type

Array of 2 numbers.

59.1.53 Document.rulerUnits

`app.activeDocument.rulerUnits`

Description

The default measurement units for the rulers in the document.

Type

RulerUnits, read-only.

59.1.54 Document.saved

`app.activeDocument.saved`

Description

If `true`, the document has not been changed since last time it was saved.

Type

Boolean.

59.1.55 Document.selection

`app.activeDocument.selection`

Description

References to the objects in this document's current selection, or `null` when nothing is selected.

A reference to an insertion point is returned when there is an active insertion point in the contents of a selected text art item. Similarly, a reference to a range of text is returned when characters are selected in the contents of a text art item.

Type

Array of objects.

59.1.56 Document.showPlacedImages

`app.activeDocument.showPlacedImages`

Description

If `true`, placed images should be displayed in the document.

Type

Boolean, read-only.

59.1.57 Document.splitLongPaths

`app.activeDocument.splitLongPaths`

Description

If `true`, long paths should be split when printing.

Type

Boolean, read-only.

59.1.58 Document.spots

`app.activeDocument.spots`

Description

The spot colors contained in this document.

Type

Spots, read-only.

59.1.59 Document.stationery

`app.activeDocument.stationery`

Description

If `true`, the file is a stationery file.

Type

Boolean, read-only.

59.1.60 Document.stories

`app.activeDocument.stories`

Description

The story items in this document.

Type

Stories, read-only.

59.1.61 Document.swatches

`app.activeDocument.swatches`

Description

The swatches in this document.

Type

Swatches, read-only.

59.1.62 Document.swatchGroups

`app.activeDocument.swatchGroups`

Description

The swatch groups in this document.

Type

SwatchGroups, read-only.

59.1.63 Document.symbolItems

`app.activeDocument.symbolItems`

Description

The art items in the document linked to symbols.

Type

SymbolItems, read-only.

59.1.64 Document.symbols

`app.activeDocument.symbols`

Description

The symbols in this document.

Type

Symbols, read-only.

59.1.65 Document.tags

`app.activeDocument.tags`

Description

The tags in this document.

Type

Tags, read-only.

59.1.66 Document.textFrames

`app.activeDocument.textFrames`

Description

The text frames in this document.

Type

TextFrameItems, read-only.

59.1.67 Document.tileFullPages

`app.activeDocument.tileFullPages`

Description

If `true`, full pages should be tiled when printing this document.

Type

Boolean, read-only.

59.1.68 Document.typename

`app.activeDocument.typename`

Description

The class name of the referenced object.

Type

String, read-only.

59.1.69 Document.useDefaultScreen

`app.activeDocument.useDefaultScreen`

Description

If `true`, the printer's default screen should be used when printing this document.

Type

Boolean, read-only.

59.1.70 Document.variables

`app.activeDocument.variables`

Description

The variables defined in this document.

Type

Variables, read-only.

59.1.71 Document.variablesLocked

`app.activeDocument.variablesLocked`

Description

If true, the variables are locked.

Type

Boolean.

59.1.72 Document.views

`app.activeDocument.views`

Description

The views contained in this document.

Type

Views, read-only.

59.1.73 Document.visibleBounds

`app.activeDocument.visibleBounds`

Description

The visible bounds of the document, including stroke width of any objects in the illustration.

Type

Array of 4 numbers, read-only.

59.1.74 Document.width

```
app.activeDocument.width
```

Description

The width of this document.

Type

Number (double), read-only.

59.1.75 Document.XMPString

```
app.activeDocument.XMPString
```

Description

The XMP metadata packet associated with this document.

Type

String.

59.2 Methods

59.2.1 Document.activate()

```
app.activeDocument.activate()
```

Description

Brings the first window associated with the document to the front.

Returns

Nothing.

59.2.2 Document.arrange()

```
app.activeDocument.arrange([layoutStyle])
```

Description

Arranges multiple documents in the given layout style.

Parameters

Parameter	Type	Description
`layoutStyle`	*DocumentLayoutStyle*, optional	The layout style to arrange documents in

Returns

Boolean.

59.2.3 Document.close()

```
app.activeDocument.close([saveOptions])
```

Description

Closes a document using specified save options.

When you close a document, you should set your document reference to `null` to prevent your script from accidentally trying to access closed documents.

Parameters

Parameter	Type	Description
`saveOptions`	*SaveOptions*	Save options to close with

Returns

Nothing.

59.2.4 Document.closeNoUI()

```
app.activeDocument.closeNoUI()
```

Description

Closes the specified non-UI document.

Returns

Nothing.

59.2.5 Document.convertCoordinate()

```
app.activeDocument.convertCoordinate(coordinate, source, destination)
```

Description

Converts the given point between artboard and document coordinate systems. Returns the converted point coordinates.

Parameters

Parameter	Type	Description
`coordinate`	Point	Point to convert
`source`	*CoordinateSystem*	Source coordinate system
`destination`	*CoordinateSystem*	Destination coordinate system

Returns

Point.

59.2.6 Document.exportFile()

```
app.activeDocument.exportFile(exportFile, exportFormat [,options])
```

Description

Exports the document to the specified file using one of the predefined export file formats. The appropriate file extension is automatically appended to the file name, except for Photoshop® documents. For these, you must include the file extension (PSD) in the file specification.

Parameters

Parameter	Type	Description
`exportFile`	File	File to save
`exportFormat`	*ExportType*	Export file format
`options`	*Variable*, optional	todo

Returns

Nothing.

59.2.7 Document.exportPDFPreset()

```
app.activeDocument.exportPDFPreset(file)
```

Description

Exports the current PDF preset values to the file.

Parameters

Parameter	Type	Description
`file`	File	Preset file to export to

Returns

Nothing.

59.2.8 Document.exportPerspectiveGridPreset()

```
app.activeDocument.exportPerspectiveGridPreset(file)
```

Description

Exports the current perspective grid preset values to the file.

Parameters

Parameter	Type	Description
`file`	File	Preset file to export to

Returns

Nothing.

59.2.9 Document.exportPrintPreset()

`app.activeDocument.exportPrintPreset(file)`

Description

Exports the current print preset values to the file.

Parameters

Parameter	Type	Description
`file`	File	Preset file to export to

Returns

Nothing.

59.2.10 Document.exportVariables()

`app.activeDocument.exportVariables(fileSpec)`

Description

Saves datasets into an XML library. The datasets contain variables and their associated dynamic data.

Parameters

Parameter	Type	Description
`fileSpec`	File	XML Library file to export to

Returns

Nothing.

59.2.11 Document.fitArtboardToSelectedArt()

`app.activeDocument.fitArtboardToSelectedArt([index])`

Description

Resizes the artboard at the given index to fit currently selected art. Index default is 0. Returns `true` on success.

Parameters

Parameter	Type	Description
`index`	Number (long), optional	Artboard index to resize

Returns

Boolean.

59.2.12 Document.getPageItemFromUuid()

```
app.activeDocument.getPageItemFromUuid(uuid)
```

Note: This functionality was added in Illustrator 24.0. (CC2020)

Description

Retrieves the pageitem using Uuid.

Parameters

Parameter	Type	Description
`uuid`	String	uuid of PageItem

Returns

PageItem.

59.2.13 Document.getPerspectiveActivePlane()

```
app.activeDocument.getPerspectiveActivePlane()
```

Description

Retrieves the active plane of the active perspective grid of the document.

Returns

PerspectiveGridPlaneType

59.2.14 Document.hidePerspectiveGrid()

```
app.activeDocument.hidePerspectiveGrid()
```

Description

Hides the current active grid for the document. If no grid is visible, does nothing. Returns `true` if a grid is hidden.

Returns

Boolean.

59.2.15 Document.imageCapture()

```
app.activeDocument.imageCapture(imageFile [,clipBounds] [,options])
```

Description

Captures the artwork content within the clipping boundaries in this document as a raster image, and writes the image data to a specified file.

If the bounds parameter is omitted, captures the entire artwork.

Parameters

Parameter	Type	Description
`imageFile`	File	Image file to write to
`clipBounds`	Rect, optional	Clipping bounds
`options`	*ImageCaptureOptions*, optional	todo

Returns

Nothing.

59.2.16 Document.importCharacterStyles()

```
app.activeDocument.importCharacterStyles(fileSpec)
```

Description

Loads the character styles from the Illustrator file.

Parameters

Parameter	Type	Description
`fileSpec`	File	File to load character styles from

Returns

Nothing.

59.2.17 Document.importParagraphStyles()

```
app.activeDocument.importParagraphStyles(fileSpec)
```

Description

Loads the paragraph styles from the Illustrator file.

Parameters

Parameter	Type	Description
`fileSpec`	File	File to load paragraph styles from

Returns

Nothing.

59.2.18 Document.importPDFPreset()

```
app.activeDocument.importPDFPreset(fileSpec [, replacingPreset])
```

Description

Loads all PDF presets from a file.

Parameters

Parameter	Type	Description
`fileSpec`	File	File to load PDF presets from
`replacingPreset`	String, optional	Whether to replace existing presets

Returns

Nothing.

59.2.19 Document.importPrintPreset()

```
app.activeDocument.importPrintPreset(printPreset, fileSpec)
```

Description

Loads the named print preset from the file.

Parameters

Parameter	Type	Description
`printPreset`	String	Name of preset to load
`fileSpec`	File	File to load print presets from

Returns

Nothing.

59.2.20 Document.importVariables()

```
app.activeDocument.importVariables(fileSpec)
```

Description

Imports a library containing datasets, variables, and their associated dynamic data. Importing variables overwrites existing variables and datasets.

Parameters

Parameter	Type	Description
`fileSpec`	File	File to import variables from

Returns

Nothing.

59.2.21 Document.print()

```
app.activeDocument.print([options])
```

Description

Prints the document.

Parameters

Parameter	Type	Description
`options`	*PrintOptions*, optional	todo

Returns

Nothing.

59.2.22 Document.rasterize()

```
app.activeDocument.rasterize(sourceArt [, clipBounds] [, options])
```

Description

Rasterizes the source art(s) within the specified clip bounds. The source art(s) is disposed of as a result of the rasterization.

Parameters

Parameter	Type	Description
`sourceArt`	*Variable*	Source art to rasterize
`clipBounds`	Rect, optional	Clipping bounds
`options`	*RasterizeOptions*, optional	todo

Returns

RasterItem

59.2.23 Document.rearrangeArboards()

```
app.activeDocument.rearrangeArboards([artboardLayout] [, artboardRowsOrCols]
[, artboardSpacing] [, artboardMoveArtwork])
```

Description

Rearranges artboards in the document. All arguments are optional.

Default layout style is `DocumentArtboard Layout.GridByRow`.

The second argument specifies the number of rows or columns, as appropriate for the chosen layout style, in the range `1..docNumArtboards-1`, or 1 (the default) for single row/column layouts.

Spacing is a number of pixels, default 20.

When last argument is true (the default), artwork is moved with the artboards.

Parameters

Parameter	Type	Description
`artboardLayout`	*DocumentArtboardLayout*, optional	Artboard layout
`artboardRowsOrCols`	Integer, optional	Number of rows or columns
`artboardSpacing`	Number, optional	Number of pixels for spacing
`artboardMoveArtwork`	Boolean, optional	Whether to move artwork with the artboards

Returns

Boolean.

59.2.24 Document.save()

```
app.activeDocument.save()
```

Description

Saves the document in it current location.

Returns

Nothing.

59.2.25 Document.saveAs()

```
app.activeDocument.saveAs(saveIn [, options])
```

Description

Saves the document in the specified file as an Illustrator, EPS, or PDF file.

Parameters

Parameter	Type	Description
`saveIn`	File	File to save the document as
`options`	*SaveOptions*, optional	Save options to close with

Returns

Nothing.

59.2.26 Document.saveNoUI()

```
app.activeDocument.saveNoUI(saveIn)
```

Description

Saves the non-UI document at the specified path

Parameters

Parameter	Type	Description
`saveIn`	File	File to save the document as

Returns

Nothing.

59.2.27 Document.selectObjectsOnActiveArtboard()

```
app.activeDocument.selectObjectsOnActiveArtboard()
```

Description

Selects the objects on the currently active artboard. Returns `true` on success.

Returns

Boolean.

59.2.28 Document.setActivePlane()

```
app.activeDocument.setActivePlane(gridPlane)
```

Description

Sets the active plane of the active perspective grid of the document. Returns `true` on success.

Parameters

Parameter	Type	Description
`gridPlane`	*PerspectiveGridPlaneType*	Grid plane type

Returns

Boolean.

59.2.29 Document.selectPerspectivePreset()

```
app.activeDocument.selectPerspectivePreset(gridType, presetName)
```

Description

Selects a predefined preset to define grid for the current document. Returns `true` on success.

Parameters

Parameter	Type	Description
`gridType`	*PerspectiveGridType*	Grid type
`presetName`	String	Preset name to select

Returns

Boolean.

59.2.30 Document.showPerspectiveGrid()

```
app.activeDocument.showPerspectiveGrid()
```

Description

Shows the current active grid for the document, or if no grid is active, shows the default grid. Returns `true` on success.

Returns

Boolean.

59.2.31 Document.windowCapture()

```
app.activeDocument.windowCapture(imageFile, windowSize)
```

Description

Captures the current document window to the target TIFF image file.

Parameters

Parameter	Type	Description
`imageFile`	File	Image file to save as
`windowSize`	Array of 2 numbers	Window size

Returns

Nothing.

59.3 Example

59.3.1 Deselecting all objects in the current document

Note: The frontmost document can be referred to as either `activeDocument` or `documents[0]`.

```
var docRef = activeDocument;
docRef.selection = null;
```

59.3.2 Closing a document

```
// Closes the active document without saving changes
if ( app.documents.length > 0 ) {
    var aiDocument = app.activeDocument;
    aiDocument.close( SaveOptions.DONOTSAVECHANGES );
    aiDocument = null;
}
```

59.3.3 Creating a document with defaults

```
// Creates a new document if none exists then sets fill and stroke defaults to true
var doc;
if (app.documents.length == 0) {
  doc = app.documents.add();
} else {
  doc = app.activeDocument;
}

doc.defaultFilled = true;
doc.defaultStroked = true;
```

CHAPTER 60

DocumentPreset

`documentPreset`

Description

A preset document template to use when creating a new document. See *Documents.addDocument()*.

60.1 Properties

60.1.1 DocumentPreset.artboardLayout

`documentPreset.artboardLayout`

Description

The layout of artboards in the new document. Default: `GridByRow`.

Type

DocumentArtboardLayout

60.1.2 DocumentPreset.artboardRowsOrCols

`documentPreset.artboardRowsOrCols`

Description

The number of rows (for rows layout) or columns (for column layout) of artboards. Range: 1 to (`numArtboards` - 1) or 1 for single row or column layouts. Default: 1

Type

Number (long).

60.1.3 DocumentPreset.artboardSpacing

`documentPreset.artboardSpacing`

Description

The spacing between artboards in the new document. Default: 20.0

Type

Number (double).

60.1.4 DocumentPreset.colorMode

`documentPreset.colorMode`

Description

The color space for the new document.

Type

DocumentColorSpace

60.1.5 DocumentPreset.documentBleedLink

`documentPreset.documentBleedLink`

Description

The document link for bleed values.

Type

Boolean.

60.1.6 DocumentPreset.documentBleedOffsetRect

`documentPreset.documentBleedOffsetRect`

Description

The document bleed offset rectangle.

Type

Rectangle.

60.1.7 DocumentPreset.height

`documentPreset.height`

Description

The height in document points. Default: 792.0

Type

Number (double).

60.1.8 DocumentPreset.numArtboards

`documentPreset.numArtboards`

Description

The number of artboards for the new document. Range: 1 to 100. Default: 1.

Type

Number (long).

60.1.9 DocumentPreset.previewMode

`documentPreset.previewMode`

Description

The preview mode for the new document.

Type

DocumentPreviewMode

60.1.10 DocumentPreset.rasterResolution

`documentPreset.rasterResolution`

Description

The raster resolution for the new document.

Type

DocumentRasterResolution

60.1.11 DocumentPreset.title

`documentPreset.title`

Description

The document title.

Type

String.

60.1.12 DocumentPreset.transparencyGrid

`documentPreset.transparencyGrid`

Description

The transparency grid color for the new document.

Type

DocumentTransparencyGrid

60.1.13 DocumentPreset.typename

`documentPreset.typename`

Description

The class name of the referenced object.

Type

String, read-only.

60.1.14 DocumentPreset.units

`documentPreset.units`

Description

The ruler units for the new document.

Type

RulerUnits

60.1.15 DocumentPreset.width

`documentPreset.width`

Description

The width in document points. Default: 612.0

Type

Number (double).

CHAPTER 61

Documents

`app.documents`

Description

A collection of *Document* objects.

61.1 Properties

61.1.1 Documents.length

`app.documents.length`

Description

The number of objects in the collection.

Type

Number, read-only.

61.1.2 Documents.parent

`app.documents.parent`

Description

The parent of this object.

Type

Object, read-only.

61.1.3 Documents.typename

`app.documents.typename`

Description

The class name of the referenced object.

Type

String, read-only.

61.2 Methods

61.2.1 Documents.add()

```
app.documents.add([documentColorSpace] [, width] [, height] [, numArtBoards]
    [, artboardLayout] [, artboardSpacing] [, artboardRowsOrCols]
)
```

Description

Creates a new document using optional parameters and returns a reference to the new document.

Parameters

Parameter	Type	Description
`documentColorSpace`	*DocumentColorSpace*, optional	Color space of document
`width`	Number (double), optional	Width of document to add
`height`	Number (double), optional	Height of document to add
`numArtBoards`	Number (long), optional	Number of artboards to create
`artboardLayout`	*DocumentArtboardLayout*, optional	Artboard layout
`artboardSpacing`	Number, optional	Number of pixels for spacing
`artboardRowsOrCols`	Integer, optional	Number of rows or columns

Returns

Document

61.2.2 Documents.addDocument()

`app.documents.addDocument(startupPreset [, presetSettings] [, showOptionsDialog])`

Description

Creates a document from the preset, replacing any provided setting values, and returns a reference to the new document.

Parameters

Parameter	Type	Description
`startupPreset`	String	Startup preset to use
`presetSettings`	*DocumentPreset*, optional	Preset document template
`showOptionsDialog`	Boolean, optional	Whether to show options dialog

Returns

Document

61.2.3 Documents.addDocumentNoUI()

```
app.documents.addDocumentNoUI(startupPreset)
```

Description

Creates a document without showing in UI.

Parameters

Parameter	Type	Description
`startupPreset`	String	Startup preset to use

Returns

Document

61.2.4 Documents.getByName()

```
app.documents.getByName(name)
```

Description

Gets the first element in the collection with the specified name.

Parameters

Parameter	Type	Description
`name`	String	Name of element to get

Returns

Document

61.2.5 Documents.index()

`app.documents.index(itemKey)`

Description

Gets an element from the collection.

Parameters

Parameter	Type	Description
`itemKey`	String, Number	String or number key

Returns

Document

61.3 Example

61.3.1 Creating a new document

```
// Creates a new document with an RGB color space

app.documents.add(DocumentColorSpace.RGB);
```

CHAPTER 62

EPSSaveOptions

`epsSaveOptions`

Description

Options for saving a document as an Illustrator EPS file, used with the *Document.saveAs()* method.

All properties are optional.

62.1 Properties

62.1.1 EPSSaveOptions.artboardRange

`epsSaveOptions.artboardRange`

Description

If `saveMultipleArtboards` is `true`, this is considered for multi-asset extraction, which specifies the artboard range. An empty string extracts all the artboards. Default: empty string

Type

String.

62.1.2 EPSSaveOptions.cmykPostScript

`epsSaveOptions.cmykPostScript`

Description

If `true`, use CMYK PostScript.

Type

Boolean.

62.1.3 EPSSaveOptions.compatibility

`epsSaveOptions.compatibility`

Description

Specifies the version of the EPS file format to save.

Default: `Compatibility.ILLUSTRATOR1719`.

Type

Compatibility

62.1.4 EPSSaveOptions.compatibleGradientPrinting

`epsSaveOptions.compatibleGradientPrinting`

Description

If `true`, create a raster item of the gradient or gradient mesh so that PostScript Level 2 printers can print the object.

Default: `false`.

Type

Boolean.

62.1.5 EPSSaveOptions.embedAllFonts

`epsSaveOptions.embedAllFonts`

Description

If `true`, all fonts used by the document should be embedded in the saved file (version 7 or later).

Default: `false`.

Type

Boolean.

62.1.6 EPSSaveOptions.embedLinkedFiles

`epsSaveOptions.embedLinkedFiles`

Description

If `true`, linked image files are to be included in the saved document.

Type

Boolean.

62.1.7 EPSSaveOptions.flattenOuput

`epsSaveOptions.flattenOuput`

Description

How should transparency be flattened for file formats older than Illustrator 9.

Type

OutputFlattening

62.1.8 EPSSaveOptions.includeDocumentThumbnails

`epsSaveOptions.includeDocumentThumbnails`

Description

If `true`, thumbnail image of the EPS artwork should be included.

Type

Boolean.

62.1.9 EPSSaveOptions.overprint

`epsSaveOptions.overprint`

Description

Whether to preserve, discard, or simulate the overprint.

Default: `PDFOverprint.PRESERVEPDFOVERPRINT`.

Type

PDFOverprint

62.1.10 EPSSaveOptions.postScript

`epsSaveOptions.postScript`

Description

PostScript Language Level to use (Level 1 valid for file format version 8 or older).

Default: `EPSPostScriptLevelEnum.LEVEL2`.

Type

EPSPostScriptLevelEnum

62.1.11 EPSSaveOptions.preview

`epsSaveOptions.preview`

Description

The format for the EPS preview image.

Type

EPSPreview

62.1.12 EPSSaveOptions.saveMultipleArtboards

`epsSaveOptions.saveMultipleArtboards`

Description

If `true`, all artboards or range of artboards are saved.

Default: `false`.

Type

Boolean.

62.1.13 EPSSaveOptions.typename

`epsSaveOptions.typename`

Description

The class name of the referenced object.

Type

String, read-only.

62.2 Example

62.2.1 Exporting to EPS format

```
// Exports current document to destFile as an EPS file with specified options,
// destFile contains the full path including the file name

function exportFileAsEPS(destFile) {
  var newFile = new File(destFile);
  var saveDoc;
  if (app.documents.length == 0) {
    saveDoc = app.documents.add();
  } else {
    saveDoc = app.activeDocument;
  }

  var saveOpts = new ePSSaveOptions();
  saveOpts.cmykPostScript = true;
  saveOpts.embedAllFonts = true;

  saveDoc.saveAs(newFile, saveOpts);
}
```

CHAPTER 63

ExportOptionsAutoCAD

`exportOptionsAutoCAD`

Description

Options for exporting a document as an AutoCAD file, used with the *Document.exportFile()* method. All properties are optional.

When you export a document, a file extension is appended automatically. You should not include any file extension in the file specification.

To override the default AutoCAD export format (DWG), use the *ExportOptionsAutoCAD.exportFileFormat* property.

63.1 Properties

63.1.1 ExportOptionsAutoCAD.alterPathsForAppearance

`exportOptionsAutoCAD.alterPathsForAppearance`

Description

If `true`, paths are altered if needed to maintain appearance.

Default: `false`.

Type

Boolean.

63.1.2 ExportOptionsAutoCAD.colors

`exportOptionsAutoCAD.colors`

Description

The colors exported into the AutoCAD file.

Type

AutoCADColors

63.1.3 ExportOptionsAutoCAD.convertTextToOutlines

`exportOptionsAutoCAD.convertTextToOutlines`

Description

If `true`, text is converted to vector paths; preserves the visual appearance of type.

Default: `false`.

Type

Boolean.

63.1.4 ExportOptionsAutoCAD.exportFileFormat

`exportOptionsAutoCAD.exportFileFormat`

Description

The format to which the file is exported.

Default: `AutoCADExportFileFormat.DWG`.

Type

AutoCADExportFileFormat

63.1.5 ExportOptionsAutoCAD.exportOption

`exportOptionsAutoCAD.exportOption`

Description

Specifies whether to preserve appearance or editability during export.

Default: `AutoCADExportOption.MaximizeEditability`.

Type

AutoCADExportOption

63.1.6 ExportOptionsAutoCAD.exportSelectedArtOnly

`exportOptionsAutoCAD.exportSelectedArtOnly`

Description

If `true`, only selected artwork is exported.

Default: `false`.

Type

Boolean.

63.1.7 ExportOptionsAutoCAD.rasterFormat

`exportOptionsAutoCAD.rasterFormat`

Description

The format in which raster art is exported.

Type

AutoCADRasterFormat

63.1.8 ExportOptionsAutoCAD.scaleLineweights

`exportOptionsAutoCAD.scaleLineweights`

Description

If `true`, line weights are scaled by the same scaling factor as the rest of the drawing.

Default: `false`.

Type

Boolean.

63.1.9 ExportOptionsAutoCAD.typename

`exportOptionsAutoCAD.typename`

Description

The class name of the referenced object.

Type

String, read-only.

63.1.10 ExportOptionsAutoCAD.unit

`exportOptionsAutoCAD.unit`

Description

The measurement units from which to map.

Type

AutoCADUnit

63.1.11 ExportOptionsAutoCAD.unitScaleRatio

`exportOptionsAutoCAD.unitScaleRatio`

Description

The ratio (as a percentage) by which output is scaled.

Range: 0 to 1000

Type

Number (double).

63.1.12 ExportOptionsAutoCAD.version

`exportOptionsAutoCAD.version`

Description

The release of AutoCAD to which the file is exported.

Default: `AutoCADCompatibility.AutoCADRelease24`.

Type

AutoCADCompatibility

CHAPTER 64

ExportOptionsFlash

`exportOptionsFlash`

Description

Options for exporting a document as a Macromedia® Flash® (SWF) file, used with the *Document.exportFile()* method. All properties are optional.

When you export a document, the appropriate file extension is appended automatically. You should not include any file extension in the file specification.

64.1 Properties

64.1.1 ExportOptionsFlash.artClipping

`exportOptionsFlash.artClipping`

Description

How the art should be clipped during output. Default: `ArtClippingOption.OUTPUTARTBOUNDS`.

Type

ArtClippingOption

64.1.2 ExportOptionsFlash.artboardRange

`exportOptionsFlash.artboardRange`

Description

If `saveMultipleArtboards` is `true`, this is considered for multi-asset extraction, which specifies the artboard range. An empty string extracts all the artboards. Default: empty String.

Type

String.

64.1.3 ExportOptionsFlash.backgroundColor

`exportOptionsFlash.backgroundColor`

Description

The background color of the exported Flash frames.

Type

RGBColor

64.1.4 ExportOptionsFlash.backgroundLayers

`exportOptionsFlash.backgroundLayers`

Description

A list of layers to be included as the static background of the exported Flash frames.

Type

Array of *Layers*

64.1.5 ExportOptionsFlash.blendAnimation

`exportOptionsFlash.blendAnimation`

Description

The animation type for blended objects. Default: `BlendAnimationType.NOBLENDANIMATION`.

Type

BlendAnimationType

64.1.6 ExportOptionsFlash.compressed

`exportOptionsFlash.compressed`

Description

If `true`, the exported file should be exported compressed. Default: `false`.

Type

Boolean.

64.1.7 ExportOptionsFlash.convertTextToOutlines

`exportOptionsFlash.convertTextToOutlines`

Description

If `true`, all text is converted to vector paths; preserves the visual appearance of type in all Flash players. Default: `false`.

Type

Boolean.

64.1.8 ExportOptionsFlash.curveQuality

`exportOptionsFlash.curveQuality`

Description

The amount of curve information that should be presented. Default: 7.

Type

Number (long).

64.1.9 ExportOptionsFlash.exportAllSymbols

`exportOptionsFlash.exportAllSymbols`

Description

If `true`, export all symbols defined in the palette. Default: `false`.

Type

Boolean.

64.1.10 ExportOptionsFlash.exportStyle

`exportOptionsFlash.exportStyle`

Description

The style in which the exported data should be created in Flash. Default: `FlashExportStyle.ASFLASHFILE`.

Type

FlashExportStyle

64.1.11 ExportOptionsFlash.exportVersion

`exportOptionsFlash.exportVersion`

Description

The version of the exported SWF file. Default: `FlashExportVersion.FlashVersion9`.

Type

FlashExportVersion

64.1.12 ExportOptionsFlash.frameRate

`exportOptionsFlash.frameRate`

Description

The display rate in frames per second. Range: 0.01–120.0. Default: 12.0.

Type

Number (double).

64.1.13 ExportOptionsFlash.ignoreTextKerning

`exportOptionsFlash.ignoreTextKerning`

Description

If `true`, ignore kerning information in text objects. Default: `false`.

Type

Boolean.

64.1.14 ExportOptionsFlash.imageFormat

`exportOptionsFlash.imageFormat`

Description

How should the image in the exported Flash file be compressed. Default: `FlashImageFormat.LOSSLESS`.

Type

FlashImageFormat

64.1.15 ExportOptionsFlash.includeMetadata

`exportOptionsFlash.includeMetadata`

Description

If `true`, include minimal XMP metadata in the SWF file. Default: `false`.

Type

Boolean.

64.1.16 ExportOptionsFlash.jpegMethod

`exportOptionsFlash.jpegMethod`

Description

Specifies the JPEG method to use. Default: `FlashJPEGMethod.Standard`.

Type

FlashJPEGMethod

64.1.17 ExportOptionsFlash.jpegQuality

`exportOptionsFlash.jpegQuality`

Description

Level of compression to use. Range 1 to 10. Default: 3.

Type

Number (long).

64.1.18 ExportOptionsFlash.layerOrder

`exportOptionsFlash.layerOrder`

Description

The order in which layers are exported to Flash frames. Default: `LayerOrderType.BOTTOMUP`.

Type

LayerOrderType

64.1.19 ExportOptionsFlash.looping

`exportOptionsFlash.looping`

Description

If `true`, the Flash file is set to loop when run. Default: `false`.

Type

Boolean.

64.1.20 ExportOptionsFlash.playbackAccess

`exportOptionsFlash.playbackAccess`

Description

The access level for the exported SWF file. Default: `FlashPlaybackSecurity.PlaybackLocal`.

Type

FlashPlaybackSecurity

64.1.21 ExportOptionsFlash.preserveAppearance

`exportOptionsFlash.preserveAppearance`

Description

If `true`, preserve appearance. If `false`, preserve editability. Default: `false`.

Type

Boolean.

64.1.22 ExportOptionsFlash.readOnly

`exportOptionsFlash.readOnly`

Description

If `true`, export as read-only file. Default: `false`.

Type

Boolean.

64.1.23 ExportOptionsFlash.replacing

`exportOptionsFlash.replacing`

Description

If a file with the same name already exists, should it be replaced. Default: `SaveOptions.PROMPTTOSAVECHANGES`.

Type

SaveOptions

64.1.24 ExportOptionsFlash.resolution

`exportOptionsFlash.resolution`

Description

The resolution in pixels per inch. Range: 72–2400. Default: 72.

Type

Number (double).

64.1.25 ExportOptionsFlash.saveMultipleArtboards

`exportOptionsFlash.saveMultipleArtboards`

Description

If `true`, all artboards or range of artboards are saved. Default: `false`.

Type

Boolean.

64.1.26 ExportOptionsFlash.typename

`exportOptionsFlash.typename`

Description

The class name of the referenced object.

Type

String, read-only.

64.2 Example

64.2.1 Exporting to Flash format

```
// Exports current document to destFile as a flash file with specified options,
// destFile contains the full path including the file name

function exportToFlashFile(destFile) {
  if (app.documents.length > 0) {
    var exportOptions = new ExportOptionsFlash();
    exportOptions.resolution = 150;

    var type = ExportType.FLASH;
    var fileSpec = new File(destFile);

    app.activeDocument.exportFile(fileSpec, type, exportOptions);
  }
}
```

CHAPTER 65

ExportOptionsGIF

`exportOptionsGIF`

Description

Options for exporting a document as a GIF file, used with the *Document.exportFile()* method. All properties are optional.

When you export a document, the appropriate file extension is appended automatically. You should not include any file extension in the file specification.

65.1 Properties

65.1.1 ExportOptionsGIF.antiAliasing

`exportOptionsGIF.antiAliasing`

Description

If `true`, the exported image should be anti-aliased. Default: `true`.

Type

Boolean.

65.1.2 ExportOptionsGIF.artBoardClipping

`exportOptionsGIF.artBoardClipping`

Description

If `true`, the exported image should be clipped to the art board. Default: `false`.

Type

Boolean.

65.1.3 ExportOptionsGIF.colorCount

`exportOptionsGIF.colorCount`

Description

The number of colors in the exported image's color table. Range: 2 to 256. Default: 128.

Type

Number (long).

65.1.4 ExportOptionsGIF.colorDither

`exportOptionsGIF.colorDither`

Description

The method used to dither colors in the exported image. Default: `ColorDitherMethod.DIFFUSION`.

Type

ColorDitherMethod

65.1.5 ExportOptionsGIF.colorReduction

`exportOptionsGIF.colorReduction`

Description

The method used to reduce the number of colors in the exported image. `Default: ColorReductionMethod.SELECTIVE`.

Type

ColorReductionMethod

65.1.6 ExportOptionsGIF.ditherPercent

`exportOptionsGIF.ditherPercent`

Description

How much should the colors of the exported image be dithered, where 100.0 is 100%.

Type

Number (long).

65.1.7 ExportOptionsGIF.horizontalScale

`exportOptionsGIF.horizontalScale`

Description

The horizontal scaling factor to apply to the exported image, where 100.0 is 100%. Default: 100.0.

Type

Number (double).

65.1.8 ExportOptionsGIF.infoLossPercent

`exportOptionsGIF.infoLossPercent`

Description

The level of information loss allowed during compression, where 100.0 is 100%.

Type

Number (long).

65.1.9 ExportOptionsGIF.interlaced

`exportOptionsGIF.interlaced`

Description

If `true`, the exported image should be interlaced. Default: `false`.

Type

Boolean.

65.1.10 ExportOptionsGIF.matte

`exportOptionsGIF.matte`

Description

If `true`, the art board should be matted with a color. Default: `true`.

Type

Boolean.

65.1.11 ExportOptionsGIF.matteColor

`exportOptionsGIF.matteColor`

Description

The color to use when matting the art board. Default: `WHITE`.

Type

RGBColor

65.1.12 ExportOptionsGIF.saveAsHTML

`exportOptionsGIF.saveAsHTML`

Description

If `true`, the exported image should be saved with an accompanying HTML file. Default: `false`.

Type

Boolean.

65.1.13 ExportOptionsGIF.transparency

`exportOptionsGIF.transparency`

Description

If `true`, the exported image should use transparency. Default: `true`.

Type

Boolean.

65.1.14 ExportOptionsGIF.typename

`exportOptionsGIF.typename`

Description

The class name of the referenced object.

Type

String, read-only.

65.1.15 ExportOptionsGIF.verticalScale

`exportOptionsGIF.verticalScale`

Description

The vertical scaling factor to apply to the exported image, where 100.0 is 100%. Default: 100.0.

Type

Number (double).

65.1.16 ExportOptionsGIF.webSnap

`exportOptionsGIF.webSnap`

Description

Howmuchshouldthecolortablebechangedtomatch the web palette, where 100 is maximum. Default: 0.

Type

Number (long).

65.2 Example

65.2.1 Exporting to GIF format

```
// Exports current document to dest as a GIF file with specified options,
// dest contains the full path including the file name

function exportToGIFFile(dest) {
  if (app.documents.length > 0) {
    var exportOptions = new ExportOptionsGIF();
    exportOptions.antiAliasing = false;
    exportOptions.colorCount = 64;
    exportOptions.colorDither = ColorDitherMethod.DIFFUSION;

    var type = ExportType.GIF;
    var fileSpec = new File(dest);

    app.activeDocument.exportFile(fileSpec, type, exportOptions);
  }
}
```

CHAPTER 66

ExportOptionsJPEG

`exportOptionsJPEG`

Description

Options for exporting a document as a JPEG file, used with the *Document.exportFile()* method. All properties are optional.

When you export a document, the appropriate file extension is appended automatically. You should not include any file extension in the file specification.

66.1 Properties

66.1.1 ExportOptionsJPEG.antiAliasing

`exportOptionsJPEG.antiAliasing`

Description

If `true`, the exported image should be anti-aliased. Default: `true`.

Type

Boolean.

66.1.2 ExportOptionsJPEG.artBoardClipping

`exportOptionsJPEG.artBoardClipping`

Description

If `true`, the exported image should be clipped to the art board.

Type

Boolean.

66.1.3 ExportOptionsJPEG.blurAmount

`exportOptionsJPEG.blurAmount`

Description

The amount of blur to apply to the exported image. Range: 0.0 to 2.0. Default: 0.0.

Type

Number (double).

66.1.4 ExportOptionsJPEG.horizontalScale

`exportOptionsJPEG.horizontalScale`

Description

The horizontal scaling factor to apply to the exported image, where 100.0 is 100%. Default: 100.0.

Type

Number (double).

66.1.5 ExportOptionsJPEG.matte

`exportOptionsJPEG.matte`

Description

If `true`, the art board should be matted with a color. Default: `true`.

Type

Boolean.

66.1.6 ExportOptionsJPEG.matteColor

`exportOptionsJPEG.matteColor`

Description

The color to use when matting the art board. Default: `white`.

Type

RGBColor

66.1.7 ExportOptionsJPEG.optimization

`exportOptionsJPEG.optimization`

Description

If `true`, the exported image should be optimized for web viewing. Default: `true`.

Type

Boolean.

66.1.8 ExportOptionsJPEG.qualitySetting

`exportOptionsJPEG.qualitySetting`

Description

The quality of the exported image. Range: 0 to 100. Default: 30.

Type

Number (long).

66.1.9 ExportOptionsJPEG.saveAsHTML

`exportOptionsJPEG.saveAsHTML`

Description

If `true`, the exported image should be saved with an accompanying HTML file. Default: `false`.

Type

Boolean.

66.1.10 ExportOptionsJPEG.typename

`exportOptionsJPEG.typename`

Description

The class name of the referenced object.

Type

String, read-only.

66.1.11 ExportOptionsJPEG.verticalScale

`exportOptionsJPEG.verticalScale`

Description

The vertical scaling factor to apply to the exported image. Range: 0.0 to 776.19. Default: 100.0.

Type

Number (double)

66.2 Example

66.2.1 Exporting to JPEG format

```
// Exports current document to dest as a JPEG file with specified options,
// dest contains the full path including the file name

function exportFileToJPEG(dest) {
  if (app.documents.length > 0) {
    var exportOptions = new ExportOptionsJPEG();
    exportOptions.antiAliasing = false;
    exportOptions.qualitySetting = 70;

    var type = ExportType.JPEG;
    var fileSpec = new File(dest);

    app.activeDocument.exportFile(fileSpec, type, exportOptions);
  }
}
```

CHAPTER 67

ExportOptionsPhotoshop

`exportOptionsPhotoshop`

Description

Options for exporting a document as a Photoshop file, used with the *Document.exportFile()* method. All properties are optional.

When you export a document, the appropriate file extension is appended automatically. You should not include any file extension in the file specification.

67.1 Properties

67.1.1 ExportOptionsPhotoshop.antiAliasing

`exportOptionsPhotoshop.antiAliasing`

Description

If `true`, the exported image should be anti-aliased. Default: `true`.

Type

Boolean.

67.1.2 ExportOptionsPhotoshop.artboardRange

`exportOptionsPhotoshop.artboardRange`

Description

If `saveMultipleArtboards` is true, this is considered for multi-asset extraction, which specifies the artboard range. An empty string extracts all the artboards. Default: empty String.

Type

String.

67.1.3 ExportOptionsPhotoshop.editableText

`exportOptionsPhotoshop.editableText`

Description

If `true`, text objects should be exported as editable text layers. Default: `true`.

Type

Boolean.

67.1.4 ExportOptionsPhotoshop.embedICCProfile

`exportOptionsPhotoshop.embedICCProfile`

Description

If `true`, an ICC profile should be embedded in the exported file. Default: `false`.

Type

Boolean.

67.1.5 ExportOptionsPhotoshop.imageColorSpace

`exportOptionsPhotoshop.imageColorSpace`

Description

The color space of the exported file. Default: `ImageColorSpace.RGB`.

Type

ImageColorSpace

67.1.6 ExportOptionsPhotoshop.maximumEditability

`exportOptionsPhotoshop.maximumEditability`

Description

Preserve as much of the original document's structure as possible when exporting. Default: `true`.

Type

Boolean.

67.1.7 ExportOptionsPhotoshop.resolution

`exportOptionsPhotoshop.resolution`

Description

Resolution of the exported file in dots per inch (dpi). Range: 72.0 to 2400.0. Default: 150.0.

Type

Number (double).

67.1.8 ExportOptionsPhotoshop.saveMultipleArtboards

`exportOptionsPhotoshop.saveMultipleArtboards`

Description

If `true`, all artboards or range of artboards are saved. Default: `false`.

Type

Boolean.

67.1.9 ExportOptionsPhotoshop.typename

`exportOptionsPhotoshop.typename`

Description

The class name of the referenced object.

Type

String, read-only.

67.1.10 ExportOptionsPhotoshop.warnings

`exportOptionsPhotoshop.warnings`

Description

If `true`, a warning dialog should be displayed in case of conflicts in the export settings. Default: `true`.

Type

Boolean.

67.1.11 ExportOptionsPhotoshop.writeLayers

`exportOptionsPhotoshop.writeLayers`

Description

If `true`, the document layers should be presented in the exported document. Default: `true`.

Type

Boolean.

67.2 Example

67.2.1 Exporting to Photoshop format

```
// Exports current document to dest as a PSD file with specified options,
// dest contains the full path including the file name

function exportFileToPSD(dest) {
  if (app.documents.length > 0) {
    var exportOptions = new ExportOptionsPhotoshop();
    exportOptions.resolution = 150;

    var type = ExportType.PHOTOSHOP;
    var fileSpec = new File(dest);

    app.activeDocument.exportFile(fileSpec, type, exportOptions);
  }
}
```

CHAPTER 68

ExportOptionsPNG24

`exportOptionsPNG24`

Description

Options for exporting a document as a 24-bit PNG file, used with the *Document.exportFile()* method. All properties are optional.

When you export a document, the appropriate file extension is appended automatically. You should not include any file extension in the file specification.

68.1 Properties

68.1.1 ExportOptionsPNG24.antiAliasing

`exportOptionsPNG24.antiAliasing`

Description

If `true`, the exported image be anti-aliased. Default: `true`.

Type

Boolean.

68.1.2 ExportOptionsPNG24.artBoardClipping

`exportOptionsPNG24.artBoardClipping`

Description

If `true`, the exported image be clipped to the art board. Default: `false`.

Type

Boolean.

68.1.3 ExportOptionsPNG24.horizontalScale

`exportOptionsPNG24.horizontalScale`

Description

The horizontal scaling factor to apply to the exported image, where 100.0 is 100%. Default: 100.0.

Type

Number (double).

68.1.4 ExportOptionsPNG24.matte

`exportOptionsPNG24.matte`

Description

If `true`, the art board be matted with a color. Default: `true`.

Type

Boolean.

68.1.5 ExportOptionsPNG24.matteColor

`exportOptionsPNG24.matteColor`

Description

The color to use when matting the art board. Default: `white`.

Type

RGBColor

68.1.6 ExportOptionsPNG24.saveAsHTML

`exportOptionsPNG24.saveAsHTML`

Description

If `true`, the exported image be saved with an accompanying HTML file. Default: `false`.

Type

Boolean.

68.1.7 ExportOptionsPNG24.transparency

`exportOptionsPNG24.transparency`

Description

If `true`, the exported image use transparency. Default: `true`.

Type

Boolean.

68.1.8 ExportOptionsPNG24.typename

`exportOptionsPNG24.typename`

Description

The class name of the referenced object.

Type

String, read-only.

68.1.9 ExportOptionsPNG24.verticalScale

`exportOptionsPNG24.verticalScale`

Description

The vertical scaling factor to apply to the exported image, where 100.0 is 100. Default: 100.0.

Type

Number (double).

68.2 Example

68.2.1 Exporting to PNG24 format

```
// Exports current document to dest as a PNG24 file with specified options,
// dest contains the full path including the file name,
// saveAsHTML option creates an HTML version with the PNG file in an images folder

function exportFileToPNG24(dest) {
  if (app.documents.length > 0) {
    var exportOptions = new ExportOptionsPNG24();
    exportOptions.antiAliasing = false;
    exportOptions.transparency = false;
    exportOptions.saveAsHTML = true;
```

(continues on next page)

(continued from previous page)

```
    var type = ExportType.PNG24;
    var fileSpec = new File(dest);

    app.activeDocument.exportFile(fileSpec, type, exportOptions);
  }
}
```

CHAPTER 69

ExportOptionsPNG8

`exportOptionsPNG8`

Description

Options for exporting a document as an 8-bit PNG file, used with the *Document.exportFile()* method. All properties are optional.

When you export a document, the appropriate file extension is appended automatically. You should not include any file extension in the file specification.

69.1 Properties

69.1.1 ExportOptionsPNG8.antiAliasing

`exportOptionsPNG8.antiAliasing`

Description

If `true`, the exported image should be anti-aliased. Default: `true`.

Type

Boolean.

69.1.2 ExportOptionsPNG8.artBoardClipping

`exportOptionsPNG8.artBoardClipping`

Description

If `true`, the exported image should be clipped to the art board. Default: `false`.

Type

Boolean.

69.1.3 ExportOptionsPNG8.colorCount

`exportOptionsPNG8.colorCount`

Description

The number of colors in the exported image's color table. Range: 2 to 256. Default: 128.

Type

Number (long).

69.1.4 ExportOptionsPNG8.colorDither

`exportOptionsPNG8.colorDither`

Description

The method used to dither colors in the exported image. Default: `ColorDitherMethod.Diffusion`.

Type

ColorDitherMethod

69.1.5 ExportOptionsPNG8.colorReduction

`exportOptionsPNG8.colorReduction`

Description

The method used to reduce the number of colors in the exported image. Default: `ColorReductionMethod.SELECTIVE`.

Type

ColorReductionMethod

69.1.6 ExportOptionsPNG8.ditherPercent

`exportOptionsPNG8.ditherPercent`

Description

The amount (as a percentage) that the colors of the exported image are dithered, where 100.0 is 100%. Range: 0 to 100. Default: 88.

Type

Number (long).

69.1.7 ExportOptionsPNG8.horizontalScale

`exportOptionsPNG8.horizontalScale`

Description

The horizontal scaling factor to apply to the exported image, where 100.0 is 100%. Default: 100.0.

Type

Number (double).

69.1.8 ExportOptionsPNG8.interlaced

`exportOptionsPNG8.interlaced`

Description

If `true`, the exported image should be interlaced. Default: `false`.

Type

Boolean.

69.1.9 ExportOptionsPNG8.matte

`exportOptionsPNG8.matte`

Description

If `true`, the art board should be matted with a color. Default: `true`.

Type

Boolean.

69.1.10 ExportOptionsPNG8.matteColor

`exportOptionsPNG8.matteColor`

Description

The color to use when matting the art board. Default: `white`.

Type

RGBColor

69.1.11 ExportOptionsPNG8.saveAsHTML

`exportOptionsPNG8.saveAsHTML`

Description

If `true`, the exported image be saved with an accompanying HTML file. Default: `false`.

Type

Boolean.

69.1.12 ExportOptionsPNG8.transparency

`exportOptionsPNG8.transparency`

Description

If `true`, the exported image use transparency. Default: `true`.

Type

Boolean.

69.1.13 ExportOptionsPNG8.typename

`exportOptionsPNG8.typename`

Description

The class name of the referenced object.

Type

String, read-only.

69.1.14 ExportOptionsPNG8.verticalScale

`exportOptionsPNG8.verticalScale`

Description

The vertical scaling factor to apply to the exported image, where 100.0 is 100. Default: 100.0.

Type

Number (double).

69.1.15 ExportOptionsPNG8.webSnap

`exportOptionsPNG8.webSnap`

Description

Specifies how much the color table should be changed to match the web palette, where 100 is maximum. Default: 0.

Type

Number (long).

69.2 Example

69.2.1 Exporting to PNG8 format

```
// Exports current document to dest as a PNG8 file with specified options,
// dest contains the full path including the file name

function exportFileToPNG8(dest) {
  if (app.documents.length > 0) {
    var exportOptions = new ExportOptionsPNG8();
    exportOptions.colorCount = 8;
    exportOptions.transparency = false;

    var type = ExportType.PNG8;
    var fileSpec = new File(dest);

    app.activeDocument.exportFile(fileSpec, type, exportOptions);
  }
}
```

CHAPTER 70

ExportOptionsSVG

`exportOptionsSVG`

Description

Options for exporting a document as a SVG file, used with the *Document.exportFile()* method. All properties are optional.

When you export a document, the appropriate file extension is appended automatically. You should not include any file extension in the file specification.

70.1 Properties

70.1.1 ExportOptionsSVG.artboardRange

`exportOptionsSVG.artboardRange`

Description

A range of artboards to save, if `saveMultipleArtboards` is `true`. A comma-delimited list of artboard names., or the empty string to save all artboards. Default: empty String.

Type

String.

70.1.2 ExportOptionsSVG.compressed

`exportOptionsSVG.compressed`

Description

If `true`, the exported file should be compressed. Default: `false`.

Type

Boolean.

70.1.3 ExportOptionsSVG.coordinatePrecision

`exportOptionsSVG.coordinatePrecision`

Description

The decimal precision for element coordinate values. Range: 1 to 7. Default: 3.

Type

Number (long)

70.1.4 ExportOptionsSVG.cssProperties

`exportOptionsSVG.cssProperties`

Description

How the CSS properties of the document should be included in the exported file. Default: `SVGCSSPropertyLocation.STYLEATTRIBUTES`.

Type

SVGCSSPropertyLocation

70.1.5 ExportOptionsSVG.documentEncoding

`exportOptionsSVG.documentEncoding`

Description

How the text in the document should be encoded. Default: `SVGDocumentEncoding.ASCII`.

Type

SVGDocumentEncoding

70.1.6 ExportOptionsSVG.DTD

`exportOptionsSVG.DTD`

Description

The SVG version to which the file should conform. Default: `SVGDTDVersion.SVG1_1`.

Type

SVGDTDVersion

70.1.7 ExportOptionsSVG.embedRasterImages

`exportOptionsSVG.embedRasterImages`

Description

If `true`, the raster images contained in the document should be embedded in the exported file. Default: `false`.

Type

Boolean.

70.1.8 ExportOptionsSVG.fontSubsetting

`exportOptionsSVG.fontSubsetting`

Description

Which font glyphs should be included in the exported file. Default: `SVGFontSubsetting.ALLGLYPHS`.

Type

SVGFontSubsetting

70.1.9 ExportOptionsSVG.fontType

`exportOptionsSVG.fontType`

Description

The type of font to included in the exported file. Default: `SVGFontType.CEFFONT`.

Type

SVGFontType

70.1.10 ExportOptionsSVG.includeFileInfo

`exportOptionsSVG.includeFileInfo`

Description

If `true`, file information should be saved in the exported file. Default: `false`.

Type

Boolean.

70.1.11 ExportOptionsSVG.includeUnusedStyles

`exportOptionsSVG.includeUnusedStyles`

Description

If `true`, save unused styles in the exported file. Default: `false`.

Type

Boolean.

70.1.12 ExportOptionsSVG.includeVariablesAndDatasets

`exportOptionsSVG.includeVariablesAndDatasets`

Description

If `true`, variables and datasets should be saved in the exported file. Default: `false`.

Type

Boolean.

70.1.13 ExportOptionsSVG.optimizeForSVGViewer

`exportOptionsSVG.optimizeForSVGViewer`

Description

If `true`, the exported file should be optimized for the SVG Viewer. Default: `false`.

Type

Boolean.

70.1.14 ExportOptionsSVG.preserveEditability

`exportOptionsSVG.preserveEditability`

Description

If `true`, Illustrator editing capabilities should be preserved when exporting the document. Default: `false`.

Type

Boolean.

70.1.15 ExportOptionsSVG.saveMultipleArtboards

`exportOptionsSVG.saveMultipleArtboards`

Description

If `true`, save the artboards specified by artboardRange in the exported file. Default: `false`.

Type

Boolean.

70.1.16 ExportOptionsSVG.slices

`exportOptionsSVG.slices`

Description

If `true`, slice data should be exported with the file. Default: `false`.

Type

Boolean.

70.1.17 ExportOptionsSVG.sVGAutoKerning

`exportOptionsSVG.sVGAutoKerning`

Description

If `true`, SVG automatic kerning is allowed in the file. Default: `false`.

Type

Boolean.

70.1.18 ExportOptionsSVG.sVGTextOnPath

`exportOptionsSVG.sVGTextOnPath`

Description

If `true`, the SVG text-on-path construct is allowed in the file. Default: `false`.

Type

Boolean.

70.1.19 ExportOptionsSVG.typename

`exportOptionsSVG.typename`

Description

The class name of the referenced object.

Type

String, read-only.

70.2 Example

70.2.1 Exporting to SVG format

```
// Exports current document to dest as an SVG file with specified options,
// dest contains the full path including the file name

function exportFileToSVG(dest) {
  if (app.documents.length > 0) {
    var exportOptions = new ExportOptionsSVG();
    exportOptions.embedRasterImages = true;
    exportOptions.embedAllFonts = false;
    exportOptions.fontSubsetting = SVGFontSubsetting.GLYPHSUSED;

    var type = ExportType.SVG;
    var fileSpec = new File(dest);

    app.activeDocument.exportFile(fileSpec, type, exportOptions);
  }
}
```

CHAPTER 71

ExportOptionsTIFF

`exportOptionsTIFF`

Description

Options for exporting a document as a TIFF file, used with the *Document.exportFile()* method. All properties are optional.

When you export a document, the appropriate file extension is appended automatically. You should not include any file extension in the file specification.

71.1 Properties

71.1.1 ExportOptionsTIFF.antiAliasing

`exportOptionsTIFF.antiAliasing`

Description

If `true`, the exported image should be anti-aliased. Default: `true`.

Type

Boolean.

71.1.2 ExportOptionsTIFF.artboardRange

`exportOptionsTIFF.artboardRange`

Description

If `saveMultipleArtboards` is `true`, this is considered for multi-asset extraction, which specifies the artboard range. An empty string extracts all the artboards. Default: empty String.

Type

String.

71.1.3 ExportOptionsTIFF.byteOrder

`exportOptionsTIFF.byteOrder`

Description

The byte order to use in the new file.

Type

TIFFByteOrder

71.1.4 ExportOptionsTIFF.imageColorSpace

`exportOptionsTIFF.imageColorSpace`

Description

The color space of the exported file. Default: `ImageColorSpace.RGB`.

Type

ImageColorSpace

71.1.5 ExportOptionsTIFF.IZWCompression

`exportOptionsTIFF.IZWCompression`

Description

If `true`, use IZW compression in the new file.

Type

Boolean.

71.1.6 ExportOptionsTIFF.resolution

`exportOptionsTIFF.resolution`

Description

Resolution of the exported file in dots per inch (dpi). Range: 72.0 to 2400.0. Default: 150.0.

Type

Number (double).

71.1.7 ExportOptionsTIFF.saveMultipleArtboards

`exportOptionsTIFF.saveMultipleArtboards`

Description

If `true`, all artboards or range of artboards are saved. Default: `false`.

Type

Number (double).

71.2 Example

71.2.1 Exporting to TIFF format

```
// Exports current document to dest as a TIFF file with specified options,
// dest contains the full path including the file name

function exportFileToPSD(dest) {
  if (app.documents.length > 0) {
    var exportOptions = new ExportOptionsTIFF();
    exportOptions.resolution = 150;
    exportOptions.byteOrder = TIFFByteOrder.IBMPC;
    exportOptions.IZWCompression = false;

    var type = ExportType.TIFF;
    var fileSpec = new File(dest);

    app.activeDocument.exportFile(fileSpec, type, exportOptions);
  }
}
```

CHAPTER 72

FXGSaveOptions

`fxgSaveOptions`

Description

Specifies options which may be supplied when saving a document as an FXG file. All properties are optional.

72.1 Properties

72.1.1 FXGSaveOptions.artboardRange

`fxgSaveOptions.artboardRange`

Description

If saveMultipleArtboards is true, this is considered for multi-asset extraction, which specifies the artboard range. An empty string extracts all the artboards. Default: empty String.

Type

String.

72.1.2 FXGSaveOptions.blendsPolicy

`fxgSaveOptions.blendsPolicy`

Description

The policy used by FXG to expand blends. Default: `BlendsExpandPolicy.AUTOMATICALLYCONVERTBLENDS`.

Type

BlendsExpandPolicy

72.1.3 FXGSaveOptions.downsampleLinkedImages

`fxgSaveOptions.downsampleLinkedImages`

Description

If `true`, linked images are downsampled (at 72 dpi). Default: `false`.

Type

Boolean.

72.1.4 FXGSaveOptions.filtersPolicy

`fxgSaveOptions.filtersPolicy`

Description

The policy used by FXG to preserve filters. Default: `FiltersPreservePolicy.KEEPFILTERSEDITABLE`.

Type

FiltersPreservePolicy

72.1.5 FXGSaveOptions.gradientsPolicy

`fxgSaveOptions.gradientsPolicy`

Description

The policy used by FXG to preserve gradients. Default: `GradientsPreservePolicy.AUTOMATICALLYCONVERTGRADIENTS`.

Type

GradientsPreservePolicy

72.1.6 FXGSaveOptions.includeUnusedSymbols

`fxgSaveOptions.includeUnusedSymbols`

Description

If `true`, unused symbols are included. Default: `false`.

Type

Boolean.

72.1.7 FXGSaveOptions.preserveEditingCapabilities

`fxgSaveOptions.preserveEditingCapabilities`

Description

If `true`, the editing capabilities of FXG are preserved. Default: `true`.

Type

Boolean.

72.1.8 FXGSaveOptions.saveMultipleArtboards

`fxgSaveOptions.saveMultipleArtboards`

Description

If `true`, all artboards or range of artboards are saved. Default: `false`.

Type

Boolean.

72.1.9 FXGSaveOptions.textPolicy

`fxgSaveOptions.textPolicy`

Description

The policy used by FXG to preserve text. Default: `TextPreservePolicy.AUTOMATICALLYCONVERTTEXT`.

Type

TextPreservePolicy

72.1.10 FXGSaveOptions.version

`fxgSaveOptions.version`

Description

The version of the FXG file format to create. Default `FXGVersion.VERSION2PT0`.

Type

FXGVersion

CHAPTER 73

Gradient

`gradient`

Description

A gradient definition contained in a document. Scripts can create new gradients.

73.1 Properties

73.1.1 Gradient.gradientStops

`gradient.gradientStops`

Description

The gradient stops contained in this gradient.

Type

GradientStops, read-only.

73.1.2 Gradient.name

`gradient.name`

Description

The gradient's name.

Type

String.

73.1.3 Gradient.parent

`gradient.parent`

Description

The document that contains this gradient.

Type

Document, read-only.

73.1.4 Gradient.type

`gradient.type`

Description

The kind of the gradient, either radial or linear.

Type

GradientType

73.1.5 Gradient.typename

`gradient.typename`

Description

The class name of the referenced object.

Type

String, read-only.

73.2 Methods

73.2.1 Gradient.remove()

`app.activeDocument.gradients[index].remove()`

Description

Removes the referenced object from the document.

Returns

Nothing.

73.3 Example

73.3.1 Creating and applying a gradient

```
// Creates a new gradient in current document then applies the gradient to the␣
↪frontmost path item

if (app.documents.length > 0) {
  // Create a color for both ends of the gradient
  var startColor = new RGBColor();
  startColor.red = 0;
  startColor.green = 100;
  startColor.blue = 255;

  var endColor = new RGBColor();
  endColor.red = 220;
  endColor.green = 0;
  endColor.blue = 100;

  // Create a new gradient
  // A new gradient always has 2 stops
  var newGradient = app.activeDocument.gradients.add();
  newGradient.name = "NewGradient";
  newGradient.type = GradientType.LINEAR;

  // Modify the first gradient stop
  newGradient.gradientStops[0].rampPoint = 30;
  newGradient.gradientStops[0].midPoint = 60;
  newGradient.gradientStops[0].color = startColor;

  // Modify the last gradient stop
  newGradient.gradientStops[1].rampPoint = 80;
  newGradient.gradientStops[1].color = endColor;

  // construct an Illustrator.GradientColor object referring to the newly created␣
↪gradient
  var colorOfGradient = new GradientColor();
  colorOfGradient.gradient = newGradient;

  // get first path item, apply new gradient as its fill
  var topPath = app.activeDocument.pathItems[0];
  topPath.filled = true;
  topPath.fillColor = colorOfGradient;
}
```

CHAPTER 74

Gradients

`app.activeDocument.gradients`

Description

A collection of *Gradient* objects in a document.

74.1 Properties

74.1.1 Gradients.length

`app.activeDocument.gradients.length`

Description

The number of objects in the collection.

Type

Number, read-only.

74.1.2 Gradients.parent

`app.activeDocument.gradients.parent`

Description

The parent of this object.

Type

Object, read-only.

74.1.3 Gradients.typename

`app.activeDocument.gradients.typename`

Description

The class name of the referenced object.

Type

String, read-only.

74.2 Methods

74.2.1 Gradients.add()

`app.activeDocument.gradients.add()`

Description

Creates a new `Gradient` object.

Returns

Gradient

74.2.2 Gradients.getByName()

`app.activeDocument.gradients.getByName(name)`

Description

Gets the first element in the collection with the specified name.

Parameters

Parameter	Type	Description
`name`	String	Name of element to get

Returns

Gradient

74.2.3 Gradients.index()

`app.activeDocument.gradients.index(itemKey)`

Description

Gets an element from the collection.

Parameters

Parameter	Type	Description
`itemKey`	String, Number	String or number key

Returns

Gradient

74.2.4 Gradients.removeAll()

`app.activeDocument.gradients.removeAll()`

Description

Deletes all elements in this collection.

Returns

Nothing.

74.3 Example

74.3.1 Removing a gradient

```
// Deletes the first gradient from the current document
if (app.documents.length > 0) {
  app.activeDocument.gradients[0].remove();
}
```

CHAPTER 75

GradientStop

`app.activeDocument.gradients[index].gradientStops[index`

Description

A gradient stop definition that represents a point on a specific gradient defined in the document. Each gradient stop specifies a color change in the containing gradient. See *Changing a gradient stop color* for an example.

75.1 Properties

75.1.1 GradientStop.color

`app.activeDocument.gradients[index].gradientStops[index].color`

Description

The color linked to this gradient stop.

Type

Color

75.1.2 GradientStop.midPoint

`app.activeDocument.gradients[index].gradientStops[index].midPoint`

Description

The midpoint key value, specified as a percentage from 13.0 to 87.0.

Type

Number (double).

75.1.3 GradientStop.opacity

`app.activeDocument.gradients[index].gradientStops[index].opacity`

Description

The opacity value for the gradient stop. Range: 0.0 to 100.0

Type

Number (double).

75.1.4 GradientStop.parent

`app.activeDocument.gradients[index].gradientStops[index].parent`

Description

The gradient that contains this gradient stop.

Type

Gradient, read-only.

75.1.5 GradientStop.rampPoint

`app.activeDocument.gradients[index].gradientStops[index].rampPoint`

Description

The location of the color in the blend in a range from 0.0 to 100.0, where 100.0 is 100%.

Type

Number (double).

75.1.6 GradientStop.typename

`app.activeDocument.gradients[index].gradientStops[index].typename`

Description

The class name of the referenced object.

Type

String, read-only.

75.2 Methods

75.2.1 GradientStop.remove()

```
app.activeDocument.gradients[index].gradientStops[index].remove()
```

Description

Deletes this object.

Returns

Nothing.

CHAPTER 76

GradientStops

`app.activeDocument.gradients[index].gradientStops`

Description

A collection of *GradientStop* objects in a specific gradient. The elements are not named; you must access them by index.

76.1 Properties

76.1.1 GradientStops.length

`app.activeDocument.gradients[index].gradientStops.length`

Description

The number of objects in the collection.

Type

Number, read-only.

76.1.2 GradientStops.parent

`app.activeDocument.gradients[index].gradientStops.parent`

Description

The parent of this object.

Type

Object, read-only.

76.1.3 GradientStops.typename

`app.activeDocument.gradients[index].gradientStops.typename`

Description

The class name of the referenced object.

Type

String, read-only.

76.2 Methods

76.2.1 GradientStops.add()

`app.activeDocument.gradients[index].gradientStops.add()`

Description

Creates a new object.

Returns

GradientStop

76.2.2 GradientStops.getByName()

`app.activeDocument.gradients[index].gradientStops.getByName(name)`

Description

Gets the first element in the collection with the specified name.

Parameters

Parameter	Type	Description
`name`	String	Name of element to get

Returns

GradientStop

76.2.3 GradientStops.index()

`app.activeDocument.gradients[index].gradientStops.index(itemKey)`

Description

Gets an element from the collection.

Parameters

Parameter	Type	Description
`itemKey`	String, Number	String or number key

Returns

GradientStop

76.2.4 GradientStops.removeAll()

`app.activeDocument.gradients[index].gradientStops.removeAll()`

Description

Deletes all objects in this collection.

Returns

Nothing.

76.3 Example

76.3.1 Adding a new gradient stop

```
// Adds a new gradient stop to a gradient, color of new stop is 70% gray
if (app.documents.length > 0 && app.activeDocument.gradients.length > 0) {
  // Get a reference to the gradient to change
  var changeGradient = app.activeDocument.gradients[0];

  // Get a reference to the last gradient stop
  var origCount = changeGradient.gradientStops.length;
  var lastStop = changeGradient.gradientStops[origCount - 1];

  // add the new gradient stop
  var newStop = changeGradient.gradientStops.add();

  // Set the values of the new gradient stop.
  // Move the original last gradient stop a bit to the left and insert the new␣
↪gradient stop at the old position
  newStop.rampPoint = lastStop.rampPoint;
  lastStop.rampPoint = lastStop.rampPoint - 10;
```

(continues on next page)

(continued from previous page)

```
  // Create a new color to apply to the newly created gradient stop
  var newStopColor = new GrayColor();
  newStopColor.gray = 70.0;
  newStop.color = newStopColor;
}
```

CHAPTER 77

GraphicStyle

`app.activeDocument.graphicStyles[index`

Description

A graphic style. Each graphic style defines a set of appearance attributes that you can apply non-destructively to page items. Graphic styles are contained in documents. Scripts cannot create new graphic styles.

77.1 Properties

77.1.1 GraphicStyle.name

`app.activeDocument.graphicStyles[index].name`

Description

The graphic style name.

Type

String.

77.1.2 GraphicStyle.parent

`app.activeDocument.graphicStyles[index].parent`

Description

The document that contains this graphic style.

Type

Document, read-only.

77.1.3 GraphicStyle.typename

`app.activeDocument.graphicStyles[index].typename`

Description

The class name of the referenced object.

Type

String, read-only.

77.2 Methods

77.2.1 GraphicStyle.applyTo()

`app.activeDocument.graphicStyles[index].applyTo(artItem)`

Description

Applies this art style to a specified art item.

Parameters

Parameter	Type	Description
`artItem`	*PageItem*	Target art item

Returns

Nothing.

77.2.2 GraphicStyle.mergeTo()

`app.activeDocument.graphicStyles[index].mergeTo(artItem)`

Description

Merges this art style into the current styles of a specified art item.

Parameters

Parameter	Type	Description
`artItem`	*PageItem*	Target art item

Returns

Nothing.

77.2.3 GraphicStyle.remove()

`app.activeDocument.graphicStyles[index].remove()`

Description

Deletes this object.

Returns

Nothing.

77.3 Example

77.3.1 Applying a graphic style

```
// Duplicates each path item in the selection, places the duplicate into a new group,
// then applies a graphic style to the new groups items

if (app.documents.length > 0) {
  var doc = app.activeDocument;
  var selected = doc.selection;
  var newGroup = doc.groupItems.add();
  newGroup.name = "NewGroup";
  newGroup.move(doc, ElementPlacement.PLACEATEND);

  var endIndex = selected.length;
  for (var i = 0; i < endIndex; i++) {
    if (selected[i].typename == "PathItem")
      selected[i].duplicate(newGroup, ElementPlacement.PLACEATEND);
  }

  for (i = 0; i < newGroup.pageItems.length; i++) {
    doc.graphicStyles[1].applyTo(newGroup.pageItems[i]);
  }
}
```

CHAPTER 78

GraphicStyles

`app.activeDocument.graphicStyles`

Description

A collection of GraphicStyle objects in a document.

78.1 Properties

78.1.1 GraphicStyles.length

`app.activeDocument.graphicStyles.length`

Description

The number of graphic styles in the document.

Type

Number, read-only.

78.1.2 GraphicStyles.parent

`app.activeDocument.graphicStyles.parent`

Description

The document that contains this graphic styles collection.

Type

Object, read-only.

78.1.3 GraphicStyles.typename

```
app.activeDocument.graphicStyles.typename
```

Description

The class name of the referenced object.

Type

String, read-only.

78.2 Methods

78.2.1 GraphicStyles.getByName()

```
app.activeDocument.graphicStyles.getByName(name)
```

Description

Gets the first element in the collection with the specified name.

Parameters

Parameter	Type	Description
`name`	String	Name of element to get

Returns

String.

78.2.2 GraphicStyles.index()

```
app.activeDocument.graphicStyles.index(itemKey)
```

Description

Gets an element from the collection.

Parameters

Parameter	Type	Description
`itemKey`	String, Number	String or number key

Returns

String, Number.

78.2.3 GraphicStyles.removeAll()

`app.activeDocument.graphicStyles.removeAll()`

Description

Removes all elements in the referenced collection.

Returns

Nothing.

78.3 Example

78.3.1 Counting graphics styles

```
// Counts the number of graphic styles in the active document
// and stores result in numberOfStyles

if (app.documents.length > 0) {
  var numberOfStyles = app.activeDocument.graphicStyles.length;
}
```

CHAPTER 79

GraphItem

`app.activeDocument.graphItems[index`

Description

Any graph artwork object. See example *Rotating graph items*.

79.1 Properties

79.1.1 GraphItem.artworkKnockout

`app.activeDocument.graphItems[index].artworkKnockout`

Description

Is this object used to create a knockout, and if so, what kind of knockout. You cannot set this value to `KnockoutState.Unknown`.

Type

KnockoutState

79.1.2 GraphItem.blendingMode

`app.activeDocument.graphItems[index].blendingMode`

Description

The mode used when compositing an object.

Type

BlendModes

79.1.3 GraphItem.contentVariable

`app.activeDocument.graphItems[index].contentVariable`

Description

The content variable bound to the graph item.

It is not necessary to set the type of the `contentVariable` before binding. Illustrator automatically set the type to `GRAPH`.

Type

Variable

79.1.4 GraphItem.controlBounds

`app.activeDocument.graphItems[index].controlBounds`

Description

The content variable bound to the graph item.

The bounds of the object including stroke width and controls.

Type

Array of 4 numbers, read-only.

79.1.5 GraphItem.editable

`app.activeDocument.graphItems[index].editable`

Description

If `true`, this graph item is editable.

Type

Boolean, read-only.

79.1.6 GraphItem.geometricBounds

`app.activeDocument.graphItems[index].geometricBounds`

Description

The bounds of the object excluding stroke width.

Type

Array of 4 numbers, read-only.

79.1.7 GraphItem.height

`app.activeDocument.graphItems[index].height`

Description

The height of the graph item.

Type

Number (double), read-only.

79.1.8 GraphItem.hidden

`app.activeDocument.graphItems[index].hidden`

Description

If `true`, this graph item is hidden.

Type

Boolean.

79.1.9 GraphItem.isIsolated

`app.activeDocument.graphItems[index].isIsolated`

Description

If `true`, this object is isolated.

Type

Boolean.

79.1.10 GraphItem.layer

`app.activeDocument.graphItems[index].layer`

Description

The layer to which this graph item belongs.

Type

Layer, read-only.

79.1.11 GraphItem.left

`app.activeDocument.graphItems[index].left`

Description

The offset (in points) of the left side of the graph item from the left side of the page.

Type

Number.

79.1.12 GraphItem.locked

`app.activeDocument.graphItems[index].locked`

Description

If `true`, this graph item is locked.

Type

Boolean.

79.1.13 GraphItem.name

`app.activeDocument.graphItems[index].name`

Description

The name of this graph item.

Type

String.

79.1.14 GraphItem.note

`app.activeDocument.graphItems[index].note`

Description

The note assigned to this item.

Type

String.

79.1.15 GraphItem.opacity

`app.activeDocument.graphItems[index].opacity`

Description

The opacity of the object; the value is between 0.0 and 100.0.

Type

Number (double)

79.1.16 GraphItem.parent

`app.activeDocument.graphItems[index].parent`

Description

The parent of this object.

Type

Layer or *GroupItem*

79.1.17 GraphItem.position

`app.activeDocument.graphItems[index].position`

Description

The position (in points) of the top left corner of the `graphItem` object in the format [x, y]. Does not include stroke weight.

Type

Array of 2 numbers.

79.1.18 GraphItem.selected

`app.activeDocument.graphItems[index].selected`

Description

If `true`, this object is selected.

Type

Boolean.

79.1.19 GraphItem.sliced

`app.activeDocument.graphItems[index].sliced`

Description

If `true`, the graph item is sliced. Default: `false`.

Type

Boolean.

79.1.20 GraphItem.tags

`app.activeDocument.graphItems[index].tags`

Description

The tags contained in this graph item.

Type

Tags, read-only.

79.1.21 GraphItem.top

`app.activeDocument.graphItems[index].top`

Description

The offset (in points) of the top of the graph item from the bottom of the page.

Type

Number (double).

79.1.22 GraphItem.typename

`app.activeDocument.graphItems[index].typename`

Description

The type of the graph item.

Type

String, read-only.

79.1.23 GraphItem.uRL

`app.activeDocument.graphItems[index].uRL`

Description

The value of the Adobe URL tag assigned to this graph item.

Type

String.

79.1.24 GraphItem.visibilityVariable

`app.activeDocument.graphItems[index].visibilityVariable`

Description

The visibility variable bound to the graph item.

It is not necessary to set the type of the `visibilityVariable` before binding. Illustrator automatically set the type to `VISIBILITY`.

Type

Variable

79.1.25 GraphItem.visibleBounds

`app.activeDocument.graphItems[index].visibleBounds`

Description

The visible bounds of the graph item including stroke width.

Type

Array of 4 numbers, read-only.

79.1.26 GraphItem.width

`app.activeDocument.graphItems[index].width`

Description

The width of the graph item. Range: 0.0 to 16348.0.

Type

Number (double).

79.1.27 GraphItem.wrapInside

`app.activeDocument.graphItems[index].wrapInside`

Description

If `true`, the text frame object should be wrapped inside this object.

Type

Boolean.

79.1.28 GraphItem.wrapOffset

`app.activeDocument.graphItems[index].wrapOffset`

Description

The offset to use when wrapping text around this object.

Type

Number (double).

79.1.29 GraphItem.wrapped

`app.activeDocument.graphItems[index].wrapped`

Description

If `true`, wrap text frame objects around this object. (Text frame must be above the object.)

Type

Boolean.

79.1.30 GraphItem.zOrderPosition

`app.activeDocument.graphItems[index].zOrderPosition`

Description

The position of this art item within the stacking order of the group or layer (parent) that contains the art item.

Type

Number (long).

79.2 Methods

79.2.1 GraphItem.duplicate()

```
app.activeDocument.graphItems[index].duplicate([relativeObject] [,
insertionLocation])
```

Description

Creates a duplicate of the selected object.

Parameters

Parameter	Type	Description
`relativeObject`	Object, optional	Object to duplicate to
`insertionLocation`	*ElementPlacement*, optional	Location to insert element

Returns

GraphItem

79.2.2 GraphItem.move()

```
app.activeDocument.graphItems[index].move(relativeObject, insertionLocation)
```

Description

Moves the object.

Parameters

Parameter	Type	Description
`relativeObject`	Object	Object to move element within
`insertionLocation`	*ElementPlacement*, optional	Location to move element to

Returns

GraphItem

79.2.3 GraphItem.remove()

```
app.activeDocument.graphItems[index].remove()
```

Description

Deletes this object.

Returns

Nothing.

79.2.4 GraphItem.resize()

```
app.activeDocument.graphItems[index].resize(scaleX, scaleY
  [,changePositions] [,changeFillPatterns] [,changeFillGradients]
  [,changeStrokePattern] [,changeLineWidths] [,scaleAbout])
)
```

Description

Scales the art item where `scaleX` is the horizontal scaling factor and `scaleY` is the vertical scaling factor. 100.0 = 100%.

Parameters

Parameter	Type	Description
`scaleX`	Number (double)	Horizontal scaling factor
`scaleY`	Number (double)	Vertical scaling factor
`changePositions`	Boolean, optional	Whether to effect art object positions and orientations
`changeFillPatterns`	Boolean, optional	Whether to transform fill patterns
`changeFillGradients`	Boolean, optional	Whether to transform fill gradients
`changeStrokePattern`	Boolean, optional	Whether to transform stroke patterns
`changeLineWidths`	Number (double), optional	The amount to scale line widths
`scaleAbout`	*Transformation*, optional	The point to use as anchor, to transform about

Returns

Nothing.

79.2.5 GraphItem.rotate()

```
app.activeDocument.graphItems[index].rotate(angle
  [,changePositions] [,changeFillPatterns] [,changeFillGradients]
  [,changeStrokePattern] [,rotateAbout]
)
```

Description

Rotates the art item relative to the current rotation. The object is rotated counter-clockwise if the `angle` value is positive, clockwise if the value is negative.

Parameters

Parameter	Type	Description
`angle`	Number (double)	The angle amount to rotate the element
`changePositions`	Boolean, optional	Whether to effect art object positions and orientations
`changeFillPatterns`	Boolean, optional	Whether to transform fill patterns
`changeFillGradients`	Boolean, optional	Whether to transform fill gradients
`changeStrokePattern`	Boolean, optional	Whether to transform stroke patterns
`rotateAbout`	*Transformation*, optional	The point to use as anchor, to transform about

Returns

Nothing.

79.2.6 GraphItem.transform()

```
app.activeDocument.graphItems[index].transform(transformationMatrix
  [,changePositions] [,changeFillPatterns] [,changeFillGradients]
  [,changeStrokePattern] [,changeLineWidths] [,transformAbout]
)
```

Description

Transforms the art item by applying a transformation matrix.

Parameters

Parameter	Type	Description
`transformationMatrix`	*Matrix*	Transformation matrix to apply
`changePositions`	Boolean, optional	Whether to change Positions
`changeFillPatterns`	Boolean, optional	Whether to change Fill Patterns
`changeFillGradients`	Boolean, optional	Whether to change Fill Gradients
`changeStrokePattern`	Boolean, optional	Whether to change Stroke Pattern
`changeLineWidths`	Number (double), optional	The amount to scale line widths
`transformAbout`	*Transformation*, optional	The point to use as anchor, to transform about

Returns

Nothing.

79.2.7 GraphItem.translate()

```
app.activeDocument.graphItems[index].translate([deltaX] [,deltaY]
  [,transformObjects] [,transformFillPatterns]
  [,transformFillGradients] [,transformStrokePatterns]
)
```

Description

Repositions the art item relative to the current position, where `deltaX` is the horizontal offset and `deltaY` is the vertical offset.

Parameters

Parameter	Type	Description
`deltaX`	Number (double), optional	Horizontal offset
`deltaY`	Number (double), optional	Vertical offset
`transformObjects`	Boolean, optional	Whether to transform Objects
`transformFillPatterns`	Boolean, optional	Whether to transform Fill Patterns
`transformFillGradients`	Boolean, optional	Whether to transform Fill Gradients
`transformStrokePatterns`	Boolean, optional	Whether to transform Stroke Patterns

Returns

Nothing.

79.2.8 GraphItem.zOrder()

```
app.activeDocument.graphItems[index].zOrder(zOrderCmd)
```

Description

Arranges the art item's position in the stacking order of the group or layer (parent) of this object.

Parameters

Parameter	Type	Description
`zOrderCmd`	*ZOrderMethod*	Stacking order arrangement method

Returns

Nothing.

CHAPTER 80

GraphItems

`app.activeDocument.graphItems`

Description

A collection `GraphItems` objects, which gives you access to all the graph art items in an Illustrator document.

80.1 Properties

80.1.1 GraphItems.length

`app.activeDocument.graphItems.length`

Description

The number of objects in the collection.

Type

Number, read-only.

80.1.2 GraphItems.parent

`app.activeDocument.graphItems.parent`

Description

The parent of this object.

Type

Object, read-only.

80.1.3 GraphItems.typename

```
app.activeDocument.graphItems.typename
```

Description

The class name of the referenced object.

Type

String, read-only.

80.2 Methods

80.2.1 GraphItems.getByName()

```
app.activeDocument.graphItems.getByName(name)
```

Description

Gets the first element in the collection with the specified name.

Parameters

Parameter	Type	Description
`name`	String	Name of element to get

Returns

GraphItems

80.2.2 GraphItems.index()

```
app.activeDocument.graphItems.index(itemKey)
```

Description

Gets an element from the collection.

Parameters

Parameter	Type	Description
`itemKey`	String, Number	String or number key

Returns

GraphItems

80.2.3 GraphItems.removeAll()

`app.activeDocument.graphItems.removeAll()`

Description

Deletes all elements in the collection.

Returns

Nothing.

80.3 Example

80.3.1 Rotating graph items

```
// Rotates each graph item in the current document 90 degrees.

// Verify a document with a graph item is open
var ok = false;

if (documents.length > 0) {
  var docRef = activeDocument;
  var iCount = docRef.graphItems.length;
  if (iCount > 0) {
    ok = true;
    for (var i = 0; i < iCount; i++) {
      var graphRef = docRef.graphItems[i];
      graphRef.selected = true;
      graphRef.rotate(90); //rotate clockwise 90 degrees
    }
    redraw();
  }
}
```

CHAPTER 81

GroupItem

`app.activeDocument.groupItems[index`

Description

A grouped set of art items. Group items can contain all of the same page items that a layer can contain, including other nested groups.

Paths contained in a group or compound path in a document are returned as individual paths when a script asks for the paths contained in the document. However, paths contained in a group or compound path are not returned when a script asks for the paths in a layer which contains the group or compound path.

81.1 Properties

81.1.1 GroupItem.artworkKnockout

`app.activeDocument.groupItems[index].artworkKnockout`

Description

Is this object used to create a knockout, and if so, what kind of knockout.

Type

KnockoutState

81.1.2 GroupItem.blendingMode

`app.activeDocument.groupItems[index].blendingMode`

Description

The blend mode used when compositing an object.

Type

BlendModes

81.1.3 GroupItem.clipped

`app.activeDocument.groupItems[index].clipped`

Description

If `true`, the group is clipped to the clipping mask.

Type

Boolean.

81.1.4 GroupItem.compoundPathItems

`app.activeDocument.groupItems[index].compoundPathItems`

Description

The compound path items contained in this group.

Type

CompoundPathItems, read-only.

81.1.5 GroupItem.controlBounds

`app.activeDocument.groupItems[index].controlBounds`

Description

The bounds of the object including stroke width and controls.

Type

Array of 4 numbers, read-only.

81.1.6 GroupItem.editable

`app.activeDocument.groupItems[index].editable`

Description

If true, this item is editable.

Type

Boolean, read-only.

81.1.7 GroupItem.geometricBounds

`app.activeDocument.groupItems[index].geometricBounds`

Description

The bounds of the object excluding stroke width.

Type

Array of 4 numbers, read-only.

81.1.8 GroupItem.graphItems

`app.activeDocument.groupItems[index].graphItems`

Description

he graph items contained in this group.

Type

GraphItems, read-only.

81.1.9 GroupItem.groupItems

`app.activeDocument.groupItems[index].groupItems`

Description

The group items contained in this group.

Type

GroupItems, read-only.

81.1.10 GroupItem.height

`app.activeDocument.groupItems[index].height`

Description

The height of the group item.

Type

Number (double).

81.1.11 GroupItem.hidden

`app.activeDocument.groupItems[index].hidden`

Description

If `true`, this group item is hidden.

Type

Boolean.

81.1.12 GroupItem.isIsolated

`app.activeDocument.groupItems[index].isIsolated`

Description

If `true`, this object is isolated.

Type

Boolean.

81.1.13 GroupItem.layer

`app.activeDocument.groupItems[index].layer`

Description

The layer to which this group item belongs.

Type

Layer, read-only.

81.1.14 GroupItem.left

`app.activeDocument.groupItems[index].left`

Description

The position of the left side of the item (in points, measured from the left side of the page).

Type

Number (double).

81.1.15 GroupItem.legacyTextItems

`app.activeDocument.groupItems[index].legacyTextItems`

Description

The legacy text items in the group.

Type

LegacyTextItems, read-only.

81.1.16 GroupItem.locked

`app.activeDocument.groupItems[index].locked`

Description

If `true`, this group item is locked.

Type

Boolean.

81.1.17 GroupItem.meshItems

`app.activeDocument.groupItems[index].meshItems`

Description

The mesh items contained in this group.

Type

MeshItems, read-only.

81.1.18 GroupItem.name

`app.activeDocument.groupItems[index].name`

Description

The name of this group item.

Type

String.

81.1.19 GroupItem.nonNativeItems

`app.activeDocument.groupItems[index].nonNativeItems`

Description

The non-native art items in this group.

Type

NonNativeItems

81.1.20 GroupItem.note

`app.activeDocument.groupItems[index].note`

Description

The note assigned to this item.

Type

String.

81.1.21 GroupItem.opacity

`app.activeDocument.groupItems[index].opacity`

Description

The opacity of the object. Range: 0.0 to 100.0.

Type

Number (double).

81.1.22 GroupItem.pageItems

`app.activeDocument.groupItems[index].pageItems`

Description

The page items (all art item classes) contained in this group.

Type

PageItems, read-only.

81.1.23 GroupItem.parent

```
app.activeDocument.groupItems[index].parent
```

Description

The parent of this object.

Type

Layer or *GroupItem*, read-only.

81.1.24 GroupItem.pathItems

```
app.activeDocument.groupItems[index].pathItems
```

Description

The path items contained in this group.

Type

PathItems, read-only.

81.1.25 GroupItem.placedItems

```
app.activeDocument.groupItems[index].placedItems
```

Description

The placed items contained in this group.

Type

PlacedItems, read-only.

81.1.26 GroupItem.pluginItems

```
app.activeDocument.groupItems[index].pluginItems
```

Description

The plug-in items contained in this group.

Type

PluginItems, read-only.

81.1.27 GroupItem.position

`app.activeDocument.groupItems[index].position`

Description

The position (in points) of the top left corner of the `groupItem` object in the format [x, y]. Does not include stroke weight.

Type

Array of 2 numbers.

81.1.28 GroupItem.rasterItems

`app.activeDocument.groupItems[index].rasterItems`

Description

The raster items contained in this group.

Type

RasterItems, read-only.

81.1.29 GroupItem.selected

`app.activeDocument.groupItems[index].selected`

Description

If `true`, this group item is selected.

Type

Boolean.

81.1.30 GroupItem.sliced

`app.activeDocument.groupItems[index].sliced`

Description

If `true`, the item sliced. Default: `false`.

Type

Boolean.

81.1.31 GroupItem.symbolItems

`app.activeDocument.groupItems[index].symbolItems`

Description

The symbol item objects in this group.

Type

SymbolItems, read-only.

81.1.32 GroupItem.tags

`app.activeDocument.groupItems[index].tags`

Description

The tags contained in this group.

Type

Tags, read-only.

81.1.33 GroupItem.textFrames

`app.activeDocument.groupItems[index].textFrames`

Description

The text art items contained in this group.

Type

TextFrameItems, read-only.

81.1.34 GroupItem.top

`app.activeDocument.groupItems[index].top`

Description

The position of the top of the item (in points, measured from the bottom of the page).

Type

Number (double).

81.1.35 GroupItem.typename

`app.activeDocument.groupItems[index].typename`

Description

The class name of the referenced object.

Type

String, read-only.

81.1.36 GroupItem.uRL

`app.activeDocument.groupItems[index].uRL`

Description

The value of the Adobe URL tag assigned to this group item.

Type

String.

81.1.37 GroupItem.visibilityVariable

`app.activeDocument.groupItems[index].visibilityVariable`

Description

The visibility variable bound to the item.

Type

Variable

81.1.38 GroupItem.visibleBounds

`app.activeDocument.groupItems[index].visibleBounds`

Description

The visible bounds of the group item including stroke width.

Type

Array of 4 numbers, read-only.

81.1.39 GroupItem.width

`app.activeDocument.groupItems[index].width`

Description

The width of the group item.

Type

Number (double).

81.1.40 GroupItem.wrapInside

`app.activeDocument.groupItems[index].wrapInside`

Description

If `true`, the text frame object should be wrapped inside this object.

Type

Boolean.

81.1.41 GroupItem.wrapOffset

`app.activeDocument.groupItems[index].wrapOffset`

Description

The offset to use when wrapping text around this object.

Type

Number (double).

81.1.42 GroupItem.wrapped

`app.activeDocument.groupItems[index].wrapped`

Description

If `true`, wrap text frame objects around this object (text frame must be above the object).

Type

Boolean.

81.1.43 GroupItem.zOrderPosition

```
app.activeDocument.groupItems[index].zOrderPosition
```

Description

The position of this group object within the stacking order of the group or layer (`parent`) that contains the group object.

Type

Number (long).

81.2 Methods

81.2.1 GroupItem.duplicate()

```
app.activeDocument.groupItems[index].duplicate([relativeObject] [,
insertionLocation])
```

Description

Creates a duplicate of the selected object.

Parameters

Parameter	Type	Description
`relativeObject`	Object, optional	Object to duplicate to
`insertionLocation`	*ElementPlacement*, optional	Location to insert element

Returns

GroupItem

81.2.2 GroupItem.move()

```
app.activeDocument.groupItems[index].move(relativeObject, insertionLocation)
```

Description

Moves the object.

Parameters

Parameter	Type	Description
`relativeObject`	Object	Object to move element within
`insertionLocation`	*ElementPlacement*, optional	Location to move element to

Returns

GroupItem

81.2.3 GroupItem.remove()

```
app.activeDocument.groupItems[index].remove()
```

Description

Deletes this object.

Returns

Nothing.

81.2.4 GroupItem.resize()

```
app.activeDocument.groupItems[index].resize(scaleX, scaleY
  [,changePositions] [,changeFillPatterns] [,changeFillGradients]
  [,changeStrokePattern] [,changeLineWidths] [,scaleAbout]
)
```

Description

Scales the art item where `scaleX` is the horizontal scaling factor and `scaleY` is the vertical scaling factor. 100.0 = 100%.

Parameters

Parameter	Type	Description
`scaleX`	Number (double)	Horizontal scaling factor
`scaleY`	Number (double)	Vertical scaling factor
`changePositions`	Boolean, optional	Whether to effect art object positions and orientations
`changeFillPatterns`	Boolean, optional	Whether to transform fill patterns
`changeFillGradients`	Boolean, optional	Whether to transform fill gradients
`changeStrokePattern`	Boolean, optional	Whether to transform stroke patterns
`changeLineWidths`	Number (double), optional	The amount to scale line widths
`scaleAbout`	*Transformation*, optional	The point to use as anchor, to transform about

Returns

Nothing.

81.2.5 GroupItem.rotate()

```
app.activeDocument.groupItems[index].rotate(angle
  [,changePositions] [,changeFillPatterns] [,changeFillGradients]
  [,changeStrokePattern] [,rotateAbout]
)
```

Description

Rotates the art item relative to the current rotation. The object is rotated counter-clockwise if the `angle` value is positive, clockwise if the value is negative.

Parameters

Parameter	Type	Description
`angle`	Number (double)	The angle amount to rotate the element
`changePositions`	Boolean, optional	Whether to effect art object positions and orientations
`changeFillPatterns`	Boolean, optional	Whether to transform fill patterns
`changeFillGradients`	Boolean, optional	Whether to transform fill gradients
`changeStrokePattern`	Boolean, optional	Whether to transform stroke patterns
`rotateAbout`	*Transformation*, optional	The point to use as anchor, to transform about

Returns

Nothing.

81.2.6 GroupItem.transform()

```
app.activeDocument.groupItems[index].transform(transformationMatrix
  [,changePositions] [,changeFillPatterns] [,changeFillGradients]
  [,changeStrokePattern] [,changeLineWidths] [,transformAbout]
)
```

Description

Transforms the art item by applying a transformation matrix.

Parameters

Parameter	Type	Description
`transformationMatrix`	*Matrix*	Transformation matrix to apply
`changePositions`	Boolean, optional	Whether to change Positions
`changeFillPatterns`	Boolean, optional	Whether to change Fill Patterns
`changeFillGradients`	Boolean, optional	Whether to change Fill Gradients
`changeStrokePattern`	Boolean, optional	Whether to change Stroke Pattern
`changeLineWidths`	Number (double), optional	The amount to scale line widths
`transformAbout`	*Transformation*, optional	The point to use as anchor, to transform about

Returns

Nothing.

81.2.7 GroupItem.translate()

```
app.activeDocument.groupItems[index].translate([deltaX] [,deltaY]
  [,transformObjects] [,transformFillPatterns]
  [,transformFillGradients] [,transformStrokePatterns]
)
```

Description

Repositions the art item relative to the current position, where `deltaX` is the horizontal offset and `deltaY` is the vertical offset.

Parameters

Parameter	Type	Description
`deltaX`	Number (double), optional	Horizontal offset
`deltaY`	Number (double), optional	Vertical offset
`transformObjects`	Boolean, optional	Whether to transform Objects
`transformFillPatterns`	Boolean, optional	Whether to transform Fill Patterns
`transformFillGradients`	Boolean, optional	Whether to transform Fill Gradients
`transformStrokePatterns`	Boolean, optional	Whether to transform Stroke Patterns

Returns

Nothing.

81.2.8 GroupItem.zOrder()

`app.activeDocument.groupItems[index].zOrder(zOrderCmd)`

Description

Arranges the art item's position in the stacking order of the group or layer (parent) of this object.

Parameters

Parameter	Type	Description
`zOrderCmd`	*ZOrderMethod*	Stacking order arrangement method

Returns

Nothing.

81.3 Example

81.3.1 Modifying all objects in a group

It is easy to modify all of the objects contained in a group. This example demonstrates how to simplify your operations on multiple objects by creating group to contain them.

```
// Creates a new group item, adds a new path item, of triangle shape, to the group,
// then adds a new text item to the group and sets the fill color of the text to red

if (app.documents.length > 0) {
  var triangleGroup = app.activeDocument.groupItems.add();

  // Create a triangle and add text, the new art is created inside the group
  var trianglePath = triangleGroup.pathItems.add();
  trianglePath.setEntirePath(Array(Array(100, 100), Array(300, 100), Array(200, Math.
↪tan(1.0471975) * 100 + 100)));
  trianglePath.closed = true;
```

(continues on next page)

(continued from previous page)

```
  trianglePath.stroked = true;
  trianglePath.filled = false;
  trianglePath.strokeWidth = 3;

  var captionText = triangleGroup.textFrames.add();
  captionText.position = Array(100, 150);
  captionText.textRange.size = 48;
  captionText.contents = "A triangle";

  var fillColor = new RGBColor();
  fillColor.red = 255;
  fillColor.green = 0;
  fillColor.blue = 0;
  captionText.characters.fillColor = fillColor;
}
```

CHAPTER 82

GroupItems

`app.activeDocument.groupItems`

Description

The collection of grouped art items in a document.

82.1 Properties

82.1.1 GroupItems.length

`app.activeDocument.groupItems.length`

Description

The number of objects in the collection.

Type

Number, read-only.

82.1.2 GroupItems.parent

`app.activeDocument.groupItems.parent`

Description

The parent of this object.

Type

Object, read-only.

82.1.3 GroupItems.typename

```
app.activeDocument.groupItems.typename
```

Description

The class name of the referenced object.

Type

String, read-only.

82.2 Methods

82.2.1 GroupItems.add()

```
app.activeDocument.groupItems.add()
```

Description

Creates a new object.

Returns

GroupItem

82.2.2 GroupItems.createFromFile()

```
app.activeDocument.groupItems.createFromFile(imageFile)
```

Description

Places an external vector art file as a group item in the document.

Parameters

Parameter	Type	Description
`imageFile`	File	Vector art file to place

Returns

GroupItem

82.2.3 GroupItems.getByName()

```
app.activeDocument.groupItems.getByName(name)
```

Description

Gets the first element in the collection with the specified name.

Parameters

Parameter	Type	Description
`name`	String	Name of element to get

Returns

GroupItem

82.2.4 GroupItems.index()

```
app.activeDocument.groupItems.index(itemKey)
```

Description

Gets an element from the collection.

Parameters

Parameter	Type	Description
`itemKey`	String, Number	String or number key

Returns

GroupItem

82.2.5 GroupItems.removeAll()

```
app.activeDocument.groupItems.removeAll()
```

Description

Deletes all elements in this collection.

Returns

Nothing.

82.3 Example

82.3.1 Importing a PDF as a group item

The following script shows how you can import a PDF document using the *GroupItems.createFromFile()* function.

Note: Before running this script you must create a one page PDF file and put it in the location `/temp/testfile1.pdf`.

```
// Embeds a new group item in to the current document from a file specified by dest
// dest should contain the full path and file name

function embedPDF(dest) {
  var embedDoc = new File(dest);
  if (app.documents.length > 0 && embedDoc.exists) {
    var doc = app.activeDocument;
    var placed = doc.groupItems.createFromFile(embedDoc);
  }
}
```

CHAPTER 83

IllustratorSaveOptions

`illustratorSaveOptions`

Description

Options for saving a document as an Illustrator file, used with the *Document.saveAs()* method. All properties are optional.

83.1 Properties

83.1.1 IllustratorSaveOptions.artboardRange

`illustratorSaveOptions.artboardRange`

Description

If `saveMultipleArtboards` is `true` (which is valid only for Illustrator 13 or earlier), the document is considered for multi-asset extraction, which specifies an artboard range. An empty string extracts all artboards. Default: empty String.

Type

String.

83.1.2 IllustratorSaveOptions.compatibility

`illustratorSaveOptions.compatibility`

Description

Specifies the version of Illustrator file format to create. Default: `Compatibility.ILLUSTRATOR19`.

Type

Compatibility

83.1.3 IllustratorSaveOptions.compressed

`illustratorSaveOptions.compressed`

Description

(Illustrator version 10 or later.) If `true`, the saved file is compressed. Default: `true`.

Type

Boolean.

83.1.4 IllustratorSaveOptions.embedICCProfile

`illustratorSaveOptions.embedICCProfile`

Description

(Illustrator version 9 or later.) If `true`, the document's ICC profile is embedded in the saved file. Default: `false`.

Type

Boolean.

83.1.5 IllustratorSaveOptions.embedLinkedFiles

`illustratorSaveOptions.embedLinkedFiles`

Description

(Illustrator version 7 or later.) If `true`, the linked image files is embedded in the saved file. Default: `false`.

Type

Boolean.

83.1.6 IllustratorSaveOptions.flattenOutput

`illustratorSaveOptions.flattenOutput`

Description

(Versions before Illustrator 9.) How transparency should be flattened for older file format versions. Default: `OutputFlattening.PRESERVEAPPEARANCE`.

Type

OutputFlattening

83.1.7 IllustratorSaveOptions.fontSubsetThreshold

`illustratorSaveOptions.fontSubsetThreshold`

Description

(Illustrator version 9 or later.) Include a subset of fonts when less than this percentage of characters is used in the document. Range: 0.0 to 100.0. Default: 100.0.

Type

Number (double).

83.1.8 IllustratorSaveOptions.pdfCompatible

`illustratorSaveOptions.pdfCompatible`

Description

(Illustrator version 10 or later.) If `true`, the file is saved as a PDF compatible file. Default: `true`.

Type

Boolean.

83.1.9 IllustratorSaveOptions.saveMultipleArtboards

`illustratorSaveOptions.saveMultipleArtboards`

Description

If `true`, all artboards or range of the artboards are saved. Valid for Illustrator 13 or earlier.

Type

Boolean.

83.1.10 IllustratorSaveOptions.typename

`illustratorSaveOptions.typename`

Description

The class name of the referenced object.

Type

String, read-only.

83.2 Example

83.2.1 Saving with options

```
// Saves the current document to dest as an AI file with specified options,
// dest specifies the full path and file name of the new file

function exportFileToAI(dest) {
  if (app.documents.length > 0) {
    var ai8Doc = new File(dest);
    var saveOptions = new IllustratorSaveOptions();
    saveOptions.compatibility = Compatibility.ILLUSTRATOR8;
    saveOptions.flattenOutput = OutputFlattening.PRESERVEAPPEARANCE;

    app.activeDocument.saveAs(ai8Doc, saveOptions);
  }
}
```

CHAPTER 84

ImageCaptureOptions

`imageCaptureOptions`

Description

Options for image capture, used with the *Document.imageCapture()* method. All properties are optional.

84.1 Properties

84.1.1 ImageCaptureOptions.antiAliasing

`imageCaptureOptions.antiAliasing`

Description

If `true`, the image result is anti-aliased. Default: `false`.

Type

Boolean

84.1.2 ImageCaptureOptions.matte

`imageCaptureOptions.matte`

Description

If `true`, the artboard is matted with a color. Default: `false`.

Type

Boolean

84.1.3 ImageCaptureOptions.matteColor

`imageCaptureOptions.matteColor`

Description

The color to use for the artboard matte. Default: white.

Type

RGBColor

84.1.4 ImageCaptureOptions.resolution

`imageCaptureOptions.resolution`

Description

The resolution of the captured image file in points-per-inch (PPI), in the range [72.0 . . . 2400.0]. Default: 150.

Type

Number (double).

84.1.5 ImageCaptureOptions.transparency

`imageCaptureOptions.transparency`

Description

If `true`, the image result is transparent. Default: `false`.

Type

Boolean.

84.1.6 ImageCaptureOptions.typename

`imageCaptureOptions.typename`

Description

The class name of the referenced object.

Type

String, read-only.

CHAPTER 85

Ink

`app.activeDocument.inkLisk[index`

Description

Associates a document ink name with ink information.

85.1 Properties

85.1.1 Ink.inkInfo

`app.activeDocument.inkLisk[index].inkInfo`

Description

The ink information

Type

InkInfo

85.1.2 Ink.name

`app.activeDocument.inkLisk[index].name`

Description

The ink's name.

Type

String.

85.1.3 Ink.typename

```
app.activeDocument.inkLisk[index].typename
```

Description

The class name of the object.

Type

String, read-only.

CHAPTER 86

InkInfo

`app.activeDocument.inkList[index].inkInfo`

Description

Ink information for printing a document.

86.1 Properties

86.1.1 InkInfo.angle

`app.activeDocument.inkList[index].inkInfo.angle`

Description

The ink's screen angle in degrees. Range: -360 to 360.

Type

Number (double).

86.1.2 InkInfo.customColor

`app.activeDocument.inkList[index].inkInfo.customColor`

Description

The color of the custom ink.

Type

Color

86.1.3 InkInfo.density

`app.activeDocument.inkList[index].inkInfo.density`

Description

The neutral density. Minimum: 0.0.

Type

Number (double).

86.1.4 InkInfo.dotShape

`app.activeDocument.inkList[index].inkInfo.dotShape`

Description

The dot shape name.

Type

String.

86.1.5 InkInfo.frequency

`app.activeDocument.inkList[index].inkInfo.frequency`

Description

The ink's frequency. Range: 0.0 to 1000.0.

Type

Number (double).

86.1.6 InkInfo.kind

`app.activeDocument.inkList[index].inkInfo.kind`

Description

The ink type.

Type

InkType

86.1.7 InkInfo.printingStatus

`app.activeDocument.inkList[index].inkInfo.printingStatus`

Description

The ink printing status.

Type

InkPrintStatus

86.1.8 InkInfo.trapping

`app.activeDocument.inkList[index].inkInfo.trapping`

Description

The trapping type.

Type

TrappingType

86.1.9 InkInfo.trappingOrder

`app.activeDocument.inkList[index].inkInfo.trappingOrder`

Description

The order of trapping for the ink. Range: 1 to 4 for CMYK.

Type

Number (long).

86.1.10 InkInfo.typename

`app.activeDocument.inkList[index].inkInfo.typename`

Description

The class name of the object.

Type

String, read-only.

86.2 Example

86.2.1 Getting ink information

```
// Displays the current documents inks in a text frame

var docRef = documents.add();

// assemble a string of the inks in this document
var sInks = "";
var iLength = activeDocument.inkList.length;
for (var i = 0; i < iLength; i++) {
  sInks += docRef.inkList[i].name;
  sInks += "\r\t";
  sInks += "Frequency = " + docRef.inkList[i].inkInfo.frequency;
  sInks += "\r\t";
  sInks += "Density = " + docRef.inkList[i].inkInfo.density;
  sInks += "\r";
}

var textRef = docRef.textFrames.add();
textRef.contents = sInks;
textRef.top = 600;
textRef.left = 200;

redraw();
```

CHAPTER 87

InsertionPoint

`app.activeDocument.textFrames[index].insertionPoints[index`

Description

A location between characters that is used to insert new text objects. An insertion point is contained in an `InsertionPoints` collection.

87.1 Properties

87.1.1 InsertionPoint.characters

`app.activeDocument.textFrames[index].insertionPoints[index].characters`

Description

All the characters in this text range.

Type

Characters, read-only.

87.1.2 InsertionPoint.lines

`app.activeDocument.textFrames[index].insertionPoints[index].lines`

Description

All the lines in this text range.

Type

Lines, read-only.

87.1.3 InsertionPoint.paragraphs

`app.activeDocument.textFrames[index].insertionPoints[index].paragraphs`

Description

All the paragraphs in this text range.

Type

Paragraphs, read-only.

87.1.4 InsertionPoint.parent

`app.activeDocument.textFrames[index].insertionPoints[index].parent`

Description

The object's container.

Type

TextRange, read-only.

87.1.5 InsertionPoint.story

`app.activeDocument.textFrames[index].insertionPoints[index].story`

Description

The story to which the text range belongs.

Type

Story, read-only.

87.1.6 InsertionPoint.textRanges

`app.activeDocument.textFrames[index].insertionPoints[index].textRanges`

Description

All of the text in this text range.

Type

TextRanges, read-only.

87.1.7 InsertionPoint.typename

```
app.activeDocument.textFrames[index].insertionPoints[index].typename
```

Description

The class name of the object.

Type

String, read-only.

87.1.8 InsertionPoint.words

```
app.activeDocument.textFrames[index].insertionPoints[index].words
```

Description

All the words contained in this text range.

Type

Words, read-only.

CHAPTER 88

InsertionPoints

`app.activeDocument.textFrames[index].insertionPoints`

Description

A collection of `InsertionPoint` objects.

88.1 Properties

88.1.1 InsertionPoints.length

`app.activeDocument.textFrames[index].insertionPoints.length`

Description

Number of elements in the collection.

Type

Number, read-only.

88.1.2 InsertionPoints.parent

`app.activeDocument.textFrames[index].insertionPoints.parent`

Description

The object's container.

Type

Object, read-only.

88.1.3 InsertionPoints.typename

`app.activeDocument.textFrames[index].insertionPoints.typename`

Description

The class name of the object.

Type

String, read-only.

88.2 Methods

88.2.1 InsertionPoints.index()

`app.activeDocument.textFrames[index].insertionPoints.index(itemKey)`

Description

Gets an element from the collection.

Parameters

Parameter	Type	Description
`itemKey`	String, Number	String or number key

Returns

InsertionPoint

88.3 Example

88.3.1 Using insertion points to add spaces

```
// Creates a new document, adds text then inserts a
// space between each character using insertion points

var docRef = documents.add();
var textRef = docRef.textFrames.add();
textRef.contents = "Wouldn't you rather be scripting?";
textRef.top = 400;
textRef.left = 100;
textRef.textRange.characterAttributes.size = 20;

redraw();

// Add a space between each character using insertion points.
var ip;
for (var i = 0; i < textRef.insertionPoints.length; i += 2) {
```

(continues on next page)

(continued from previous page)

```
  ip = textRef.insertionPoints[i];
  ip.characters.add(" ");
}
```

CHAPTER 89

Layer

`app.activeDocument.layers[index`

Description

A layer in an Illustrator document. Layers may contain nested layers, which are called sublayers in the user interface.

The `layer` object contains all of the page items in the specific layer as elements. Your script can access page items as elements of either the Layer object or as elements of the `Document` object. When accessing page items as elements of a layer, only objects in that layer can be accessed. To access page items throughout the entire document, be sure to refer to them as contained by the document.

89.1 Properties

89.1.1 Layer.artworkKnockout

`app.activeDocument.layers[index].artworkKnockout`

Description

Is this object used to create a knockout, and if so, what kind of knockout. You cannot set this value to `KnockoutState.Unknown`.

Type

KnockoutState

89.1.2 Layer.blendingMode

`app.activeDocument.layers[index].blendingMode`

Description

The mode used when compositing an object.

Type

BlendModes

89.1.3 Layer.color

`app.activeDocument.layers[index].color`

Description

The layer's selection mark color.

Type

RGBColor

89.1.4 Layer.compoundPathItems

`app.activeDocument.layers[index].compoundPathItems`

Description

The compound path items contained in this layer.

Type

CompoundPathItems, read-only.

89.1.5 Layer.dimPlacedImages

`app.activeDocument.layers[index].dimPlacedImages`

Description

If `true`, placed images should be rendered as dimmed in this layer.

Type

Boolean.

89.1.6 Layer.graphItems

`app.activeDocument.layers[index].graphItems`

Description

The graph items contained in this layer.

Type

GraphItems, read-only.

89.1.7 Layer.groupItems

`app.activeDocument.layers[index].groupItems`

Description

The group items contained in this layer.

Type

GroupItems, read-only.

89.1.8 Layer.hasSelectedArtwork

`app.activeDocument.layers[index].hasSelectedArtwork`

Description

If `true`, an object in this layer has been selected; set to `false` to deselect all objects in the layer.

Type

Boolean.

89.1.9 Layer.isIsolated

`app.activeDocument.layers[index].isIsolated`

Description

If `true`, this object is isolated.

Type

Boolean.

89.1.10 Layer.layers

`app.activeDocument.layers[index].layers`

Description

The layers contained in this layer.

Type

Layers, read-only.

89.1.11 Layer.legacyTextItems

`app.activeDocument.layers[index].legacyTextItems`

Description

The legacy text items in this layer.

Type

LegacyTextItems, read-only.

89.1.12 Layer.locked

`app.activeDocument.layers[index].locked`

Description

If `true`, this layer is editable; set to false to lock the layer.

Type

Boolean.

89.1.13 Layer.meshItems

`app.activeDocument.layers[index].meshItems`

Description

The mesh items contained in this layer.

Type

MeshItems, read-only.

89.1.14 Layer.name

`app.activeDocument.layers[index].name`

Description

The name of this layer.

Type

String.

89.1.15 Layer.nonNativeItems

`app.activeDocument.layers[index].nonNativeItems`

Description

The non-native art items in this layer.

Type

NonNativeItems

89.1.16 Layer.opacity

`app.activeDocument.layers[index].opacity`

Description

The opacity of the layer. Range: 0.0 to 100.0.

Type

Number (double).

89.1.17 Layer.pageItems

`app.activeDocument.layers[index].pageItems`

Description

The page items (all art item classes) contained in this layer.

Type

PageItems

89.1.18 Layer.parent

`app.activeDocument.layers[index].parent`

Description

The document or layer that contains this layer.

Type

Document or *Layer*, read-only.

89.1.19 Layer.pathItems

`app.activeDocument.layers[index].pathItems`

Description

The path items contained in this layer.

Type

PathItems, read-only.

89.1.20 Layer.placedItems

`app.activeDocument.layers[index].placedItems`

Description

The placed items contained in this layer.

Type

PlacedItems, read-only.

89.1.21 Layer.pluginItems

`app.activeDocument.layers[index].pluginItems`

Description

The plug-in items contained in this layer.

Type

PluginItems, read-only.

89.1.22 Layer.preview

`app.activeDocument.layers[index].preview`

Description

If `true`, this layer should be displayed using preview mode.

Type

Boolean.

89.1.23 Layer.printable

`app.activeDocument.layers[index].printable`

Description

If `true`, this layer should be printed when printing the document.

Type

Boolean.

89.1.24 Layer.rasterItems

`app.activeDocument.layers[index].rasterItems`

Description

The raster items contained in this layer.

Type

RasterItems, read-only.

89.1.25 Layer.sliced

`app.activeDocument.layers[index].sliced`

Description

If `true`, the layer item is sliced. Default: `false`.

Type

Boolean.

89.1.26 Layer.symbolItems

`app.activeDocument.layers[index].symbolItems`

Description

The symbol items contained in the layer.

Type

SymbolItems, read-only.

89.1.27 Layer.textFrames

`app.activeDocument.layers[index].textFrames`

Description

The text art items contained in this layer.

Type

TextFrameItems, read-only.

89.1.28 Layer.typename

`app.activeDocument.layers[index].typename`

Description

The class name of the referenced object.

Type

String, read-only.

89.1.29 Layer.visible

`app.activeDocument.layers[index].visible`

Description

If `true`, this layer is visible.

Type

Boolean.

89.1.30 Layer.zOrderPosition

```
app.activeDocument.layers[index].zOrderPosition
```

Description

The position of this layer within the stacking order of layers in the document.

Type

Number (long), read-only.

89.2 Methods

89.2.1 Layer.move()

```
app.activeDocument.layers[index].move(relativeObject, insertionLocation)
```

Description

Moves the object.

Parameters

Parameter	Type	Description
`relativeObject`	Object	Object to move element within
`insertionLocation`	*ElementPlacement*, optional	Location to move element to

Returns

Layer

89.2.2 Layer.remove()

```
app.activeDocument.layers[index].remove()
```

Description

Deletes this object.

Returns

Nothing.

89.2.3 Layer.zOrder()

`app.activeDocument.layers[index].zOrder(ZOrderCmd)`

Description

Arranges the layer's position in the stacking order of the containing layer or document (`parent`) of this object.

Parameters

Parameter	Type	Description
`zOrderCmd`	*ZOrderMethod*	Stacking order arrangement method

Returns

Nothing.

89.3 Example

89.3.1 Bringing a layer to the front

```
// Moves the bottom layer to become the topmost layer

if (documents.length > 0) {
  var countOfLayers = activeDocument.layers.length;
  if (countOfLayers > 1) {
    var bottomLayer = activeDocument.layers[countOfLayers - 1];
    bottomLayer.zOrder(ZOrderMethod.BRINGTOFRONT);
  } else {
    alert("The active document only has only 1 layer");
  }
}
```

CHAPTER 90

Layers

`app.activeDocument.layers`

Description

The collection of layers in the document.

90.1 Properties

90.1.1 Layers.length

`app.activeDocument.layers.length`

Description

The number of objects in the collection.

Type

Number, read-only.

90.1.2 Layers.parent

`app.activeDocument.layers.parent`

Description

The parent of this object.

Type

Object, read-only.

90.1.3 Layers.typename

```
app.activeDocument.layers.typename
```

Description

The class name of the referenced object.

Type

String, read-only.

90.2 Methods

90.2.1 Layers.add()

```
app.activeDocument.layers.add()
```

Description

Creates a new layer in the document.

Returns

Layer

90.2.2 Layers.getByName()

```
app.activeDocument.layers.getByName(name)
```

Description

Gets the first element in the collection with the specified name.

Parameters

Parameter	Type	Description
`name`	String	Name of element to get

Returns

Layer

90.2.3 Layers.index()

`app.activeDocument.layers.index(itemKey)`

Description

Gets an element from the collection.

Parameters

Parameter	Type	Description
`itemKey`	String, Number	String or number key

Returns

Layer

90.2.4 Layers.removeAll()

`app.activeDocument.layers.removeAll()`

Description

Deletes all elements in this collection.

Returns

Nothing.

90.3 Example

90.3.1 Finding and deleting layers

```
// Deletes all layers whose name begins with "Temp" in all open documents

var layersDeleted = 0;
for (var i = 0; i < app.documents.length; i++) {
  var targetDocument = app.documents[i];
  var layerCount = targetDocument.layers.length;

  // Loop through layers from the back, to preserve index
  // of remaining layers when we remove one
  for (var ii = layerCount - 1; ii >= 0; ii--) {
    var targetLayer = targetDocument.layers[ii];
    var layerName = new String(targetLayer.name);
    if (layerName.indexOf("Temp") == 0) {
      targetDocument.layers[ii].remove();
      layersDeleted++;
    }
  }
}
```

CHAPTER 91

LegacyTextItem

`legacyTextItems[index`

Description

A text object created in Illustrator CS (version 10) or earlier, which is uneditable until converted. To convert legacy text, see *LegacyTextItems.convertToNative()*.

You can view, move, and print legacy text, but you cant edit it. Legacy text has an "x" through its bounding box when selected.

91.1 Properties

91.1.1 LegacyTextItem.artworkKnockout

`legacyTextItems[index].artworkKnockout`

Description

Is this object used to create a knockout, and if so, what kind of knockout.

Type

KnockoutState

91.1.2 LegacyTextItem.blendingMode

`legacyTextItems[index].blendingMode`

Description

The blend mode used when compositing an object.

Type

BlendModes

91.1.3 LegacyTextItem.controlBounds

`legacyTextItems[index].controlBounds`

Description

The bounds of the object including stroke width and controls.

Type

Array of 4 numbers, read-only.

91.1.4 LegacyTextItem.converted

`legacyTextItems[index].converted`

Description

If `true`, the legacy text item has been updated to a native text frame item.

Type

Boolean, read-only.

91.1.5 LegacyTextItem.editable

`legacyTextItems[index].editable`

Description

If `true`, this item is editable.

Type

Boolean, read-only.

91.1.6 LegacyTextItem.geometricBounds

`legacyTextItems[index].geometricBounds`

Description

The bounds of the object excluding stroke width.

Type

Array of 4 numbers, read-only.

91.1.7 LegacyTextItem.height

`legacyTextItems[index].height`

Description

The height of the group item.

Type

Number (double).

91.1.8 LegacyTextItem.hidden

`legacyTextItems[index].hidden`

Description

If `true`, this item is hidden.

Type

Boolean.

91.1.9 LegacyTextItem.isIsolated

`legacyTextItems[index].isIsolated`

Description

If `true`, this object is isolated.

Type

Boolean.

91.1.10 LegacyTextItem.layer

`legacyTextItems[index].layer`

Description

The layer to which this item belongs.

Type

Layer, read-only.

91.1.11 LegacyTextItem.left

`legacyTextItems[index].left`

Description

The position of the left side of the item (in points, measured from the left side of the page).

Type

Number (double).

91.1.12 LegacyTextItem.locked

`legacyTextItems[index].locked`

Description

If `true`, this item is locked.

Type

Boolean.

91.1.13 LegacyTextItem.name

`legacyTextItems[index].name`

Description

The name of this item.

Type

String.

91.1.14 LegacyTextItem.note

`legacyTextItems[index].note`

Description

The note assigned to this item.

Type

String.

91.1.15 LegacyTextItem.opacity

`legacyTextItems[index].opacity`

Description

The opacity of the object. Range: 0.0 to 100.0.

Type

Number (double).

91.1.16 LegacyTextItem.parent

`legacyTextItems[index].parent`

Description

The parent of this object.

Type

Layer or *GroupItem*, read-only.

91.1.17 LegacyTextItem.position

`legacyTextItems[index].position`

Description

The position (in points) of the top left corner of the `legacyTextItems[index` object in the format [x, y]. Does not include stroke weight.

Type

Array of 2 numbers.

91.1.18 LegacyTextItem.selected

`legacyTextItems[index].selected`

Description

If `true`, this item is selected.

Type

Boolean.

91.1.19 LegacyTextItem.sliced

`legacyTextItems[index].sliced`

Description

If `true`, the item sliced. Default: `false`.

Type

Boolean.

91.1.20 LegacyTextItem.tags

`legacyTextItems[index].tags`

Description

The tags contained in this item.

Type

Tags, read-only.

91.1.21 LegacyTextItem.top

`legacyTextItems[index].top`

Description

The position of the top of the item (in points, measured from the bottom of the page).

Type

Number (double).

91.1.22 LegacyTextItem.typename

`legacyTextItems[index].typename`

Description

The class name of the referenced object.

Type

String, read-only.

91.1.23 LegacyTextItem.uRL

`legacyTextItems[index].uRL`

Description

The value of the Adobe URL tag assigned to this item.

Type

String.

91.1.24 LegacyTextItem.visibilityVariable

`legacyTextItems[index].visibilityVariable`

Description

The visibility variable bound to the item.

Type

Variable

91.1.25 LegacyTextItem.visibleBounds

`legacyTextItems[index].visibleBounds`

Description

The visible bounds of the item including stroke width.

Type

Array of 4 numbers, read-only.

91.1.26 LegacyTextItem.width

`legacyTextItems[index].width`

Description

The width of the item.

Type

Number (double).

91.1.27 LegacyTextItem.wrapInside

`legacyTextItems[index].wrapInside`

Description

If `true`, the text frame object should be wrapped inside this object.

Type

Boolean.

91.1.28 LegacyTextItem.wrapOffset

`legacyTextItems[index].wrapOffset`

Description

The offset to use when wrapping text around this object.

Type

Number (double).

91.1.29 LegacyTextItem.wrapped

`legacyTextItems[index].wrapped`

Description

If `true`, wrap text frame objects around this object (text frame must be above the object).

Type

Boolean.

91.1.30 LegacyTextItem.zOrderPosition

`legacyTextItems[index].zOrderPosition`

Description

The position of this item within the stacking order of the group or layer (`parent`) that contains the item.

Type

Number (long), read-only.

91.2 Methods

91.2.1 LegacyTextItem.convertToNative()

```
legacyTextItems[index].convertToNative()
```

Description

Converts the legacy text item to a text frame and deletes the original legacy text.

Returns

GroupItem

91.2.2 LegacyTextItem.duplicate()

```
legacyTextItems[index].duplicate([relativeObject] [,insertionLocation])
```

Description

Creates a duplicate of the selected object.

Parameters

Parameter	Type	Description
`relativeObject`	Object, optional	Object to duplicate to
`insertionLocation`	*ElementPlacement*, optional	Location to insert element

Returns

LegacyTextItem

91.2.3 LegacyTextItem.move()

```
legacyTextItems[index].move(relativeObject, insertionLocation)
```

Description

Moves the object.

Parameters

Parameter	Type	Description
`relativeObject`	Object	Object to move element within
`insertionLocation`	*ElementPlacement*, optional	Location to move element to

Returns

LegacyTextItem

91.2.4 LegacyTextItem.remove()

```
legacyTextItems[index].remove()
```

Description

Deletes this object.

Returns

Nothing.

91.2.5 LegacyTextItem.resize()

```
legacyTextItem.resize(scaleX, scaleY
  [,changePositions] [,changeFillPatterns] [,changeFillGradients]
  [,changeStrokePattern] [,changeLineWidths] [,scaleAbout]
)
```

Description

Scales the art item where scaleX is the horizontal scaling factor and scaleY is the vertical scaling factor. 100.0 = 100%.

Parameters

Parameter	Type	Description
`scaleX`	Number (double)	Horizontal scaling factor
`scaleY`	Number (double)	Vertical scaling factor
`changePositions`	Boolean, optional	Whether to effect art object positions and orientations
`changeFillPatterns`	Boolean, optional	Whether to transform fill patterns
`changeFillGradients`	Boolean, optional	Whether to transform fill gradients
`changeStrokePattern`	Boolean, optional	Whether to transform stroke patterns
`changeLineWidths`	Number (double), optional	The amount to scale line widths
`scaleAbout`	*Transformation*, optional	The point to use as anchor, to transform about

Returns

Nothing.

91.2.6 LegacyTextItem.rotate()

```
legacyTextItem.rotate(angle [,changePositions] [,changeFillPatterns]
  [,changeFillGradients] [,changeStrokePattern] [,rotateAbout]
)
```

Description

Rotates the art item relative to the current rotation. The object is rotated counter-clockwise if the `angle` value is positive, clockwise if the value is negative.

Parameters

Parameter	Type	Description
`angle`	Number (double)	The angle amount to rotate the element
`changePositions`	Boolean, optional	Whether to effect art object positions and orientations
`changeFillPatterns`	Boolean, optional	Whether to transform fill patterns
`changeFillGradients`	Boolean, optional	Whether to transform fill gradients
`changeStrokePattern`	Boolean, optional	Whether to transform stroke patterns
`rotateAbout`	*Transformation*, optional	The point to use as anchor, to transform about

Returns

Nothing.

91.2.7 LegacyTextItem.transform()

```
legacyTextItem.transform(transformationMatrix
  [,changePositions] [,changeFillPatterns] [,changeFillGradients]
  [,changeStrokePattern] [,changeLineWidths] [,transformAbout]
)
```

Description

Transforms the art item by applying a transformation matrix.

Parameters

Parameter	Type	Description
`transformationMatrix`	*Matrix*	Transformation matrix to apply
`changePositions`	Boolean, optional	Whether to change Positions
`changeFillPatterns`	Boolean, optional	Whether to change Fill Patterns
`changeFillGradients`	Boolean, optional	Whether to change Fill Gradients
`changeStrokePattern`	Boolean, optional	Whether to change Stroke Pattern
`changeLineWidths`	Number (double), optional	The amount to scale line widths
`transformAbout`	*Transformation*, optional	The point to use as anchor, to transform about

Returns

Nothing.

91.2.8 LegacyTextItem.translate()

```
legacyTextItem.translate([deltaX] [,deltaY]
  [,transformObjects] [,transformFillPatterns]
  [,transformFillGradients] [,transformStrokePatterns]
)
```

Description

Repositions the art item relative to the current position, where `deltaX` is the horizontal offset and `deltaY` is the vertical offset.

Parameters

Parameter	Type	Description
`deltaX`	Number (double), optional	Horizontal offset
`deltaY`	Number (double), optional	Vertical offset
`transformObjects`	Boolean, optional	Whether to transform Objects
`transformFillPatterns`	Boolean, optional	Whether to transform Fill Patterns
`transformFillGradients`	Boolean, optional	Whether to transform Fill Gradients
`transformStrokePatterns`	Boolean, optional	Whether to transform Stroke Patterns

Returns

Nothing.

91.2.9 LegacyTextItem.zOrder()

`legacyTextItems[index].zOrder(zOrderCmd)`

Description

Arranges the art item's position in the stacking order of the group or layer (parent) of this object.

Parameters

Parameter	Type	Description
`zOrderCmd`	*ZOrderMethod*	Stacking order arrangement method

Returns

Nothing.

CHAPTER 92

LegacyTextItems

`legacyTextItems`

Description

A collection of *LegacyTextItem* objects.

92.1 Properties

92.1.1 LegacyTextItems.length

`legacyTextItems.length`

Description

Number of elements in the collection.

Type

Number, read-only.

92.1.2 LegacyTextItems.parent

`legacyTextItems.parent`

Description

The object's container.

Type

Object, read-only.

92.1.3 LegacyTextItems.typename

`legacyTextItems.typename`

Description

The class name of the object.

Type

String, read-only.

92.2 Methods

92.2.1 LegacyTextItems.convertToNative()

`legacyTextItems.convertToNative()`

Description

Creates text frames from all legacy text items; the original legacy text items are deleted. Returns `true` on success.

Returns

Boolean.

92.2.2 LegacyTextItems.getByName()

`legacyTextItems.getByName(name)`

Description

Get the first element in the collection with the specified name.

Parameters

Parameter	Type	Description
`name`	String	Name of element to get

Returns

LegacyTextItem

92.2.3 LegacyTextItems.index()

```
legacyTextItems.index(itemKey)
```

Description

Gets an element from the collection.

Parameters

Parameter	Type	Description
`itemKey`	String, Number	String or number key

Returns

LegacyTextItem

92.2.4 LegacyTextItems.removeAll()

```
legacyTextItems.removeAll()
```

Description

Deletes all elements in this collection.

Returns

Nothing.

CHAPTER 93

Lines

`lines`

Description

A collection of `TextRange` objects representing lines of text in a text frame. The elements are not named; you must access them by index.

93.1 Properties

93.1.1 Lines.length

`lines.length`

Description

Number of elements in the collection.

Type

Number, read-only.

93.1.2 Lines.parent

`lines.parent`

Description

The object's container.

Type

Object, read-only.

93.1.3 Lines.typename

`lines.typename`

Description

The class name of the object.

Type

String, read-only.

93.2 Methods

93.2.1 Lines.index()

`lines.index(itemKey)`

Description

Gets an element from the collection.

Parameters

Parameter	Type	Description
`itemKey`	String, Number	String or number key

Returns

TextRange

93.2.2 Lines.removeAll()

`lines.removeAll()`

Description

Deletes all elements in this collection.

Returns

Nothing.

CHAPTER 94

Matrix

`matrix`

Description

A transformation matrix specification, used to transform the geometry of objects. Use it to specify and retrieve matrix information from an Illustrator document or from page items in a document.

Matrices are used in conjunction with the `transform` method and as a property of a number of objects. A matrix specifies how to transform the geometry of an object. You can generate an original matrix using the *Application* object methods *Application.getTranslationMatrix()*, *Application.getScaleMatrix()*, or *Application.getRotationMatrix()*.

A `Matrix` is a record containing the matrix values, not a reference to a matrix object. The matrix commands operate on the values of a matrix record. If a command modifies a matrix, a modified matrix record is returned as the result of the command. The original matrix record passed to the command is not modified.

94.1 Properties

94.1.1 Matrix.mValueA

`matrix.mValueA`

Description

Matrix property `a`.

Type

Number (double).

94.1.2 Matrix.mValueB

`matrix.mValueB`

Description

Matrix property `b`.

Type

Number (double).

94.1.3 Matrix.mValueC

`matrix.mValueC`

Description

Matrix property `c`.

Type

Number (double).

94.1.4 Matrix.mValueD

`matrix.mValueD`

Description

Matrix property `d`.

Type

Number (double).

94.1.5 Matrix.mValueTX

`matrix.mValueTX`

Description

Matrix property `tx`.

Type

Number (double).

94.1.6 Matrix.mValueTY

`matrix.mValueTY`

Description

Matrix property `ty`.

Type

Number (double).

94.1.7 Matrix.typename

`matrix.typename`

Description

The class name of the referenced object.

Type

String, read-only.

94.2 Example

94.2.1 Combining matrices to apply multiple transformations

To apply multiple transformations to objects, it is more efficient to use the matrix suite than to apply the transformations one at a time. The following script demonstrates how to combine multiple matrices.

```
// Tranforms all art in a document using translation and rotation matrices,
// moves art half an inch to the right and 1.5 inches up on the page
if (app.documents.length > 0) {
  var moveMatrix = app.getTranslationMatrix(0.5, 1.5);

  // Add a rotation to the translation, 10 degrees counter clockwise
  var totalMatrix = concatenateRotationMatrix(moveMatrix, 10);

  // apply the transformation to all art in the document
  var doc = app.activeDocument;
  for (var i = 0; i < doc.pageItems.length; i++) {
    doc.pageItems[i].transform(totalMatrix);
  }
}
```

CHAPTER 95

MeshItem

`app.activeDocument.meshItems[index`

Description

A gradient mesh art item. You cannot create mesh items from a script. However, you can copy an existing mesh item with the `duplicate` method, then use the one of the move methods to place the copy at the proper location.

95.1 Properties

95.1.1 MeshItem.artworkKnockout

`app.activeDocument.meshItems[index].artworkKnockout`

Description

Is this object used to create a knockout, and if so, what kind of knockout.

Type

KnockoutState

95.1.2 MeshItem.blendingMode

`app.activeDocument.meshItems[index].blendingMode`

Description

The blend mode used when compositing an object.

Type

BlendModes

95.1.3 MeshItem.controlBounds

`app.activeDocument.meshItems[index].controlBounds`

Description

The bounds of the object including stroke width and controls.

Type

Array of 4 numbers, read-only.

95.1.4 MeshItem.editable

`app.activeDocument.meshItems[index].editable`

Description

If `true`, this item is editable.

Type

Boolean, read-only.

95.1.5 MeshItem.geometricBounds

`app.activeDocument.meshItems[index].geometricBounds`

Description

The bounds of the object excluding stroke width.

Type

Array of 4 numbers, read-only.

95.1.6 MeshItem.height

`app.activeDocument.meshItems[index].height`

Description

The height of the group item.

Type

Number (double).

95.1.7 MeshItem.hidden

`app.activeDocument.meshItems[index].hidden`

Description

If `true`, this item is hidden.

Type

Boolean.

95.1.8 MeshItem.isIsolated

`app.activeDocument.meshItems[index].isIsolated`

Description

If `true`, this object is isolated.

Type

Boolean.

95.1.9 MeshItem.layer

`app.activeDocument.meshItems[index].layer`

Description

The layer to which this item belongs.

Type

Layer, read-only.

95.1.10 MeshItem.left

`app.activeDocument.meshItems[index].left`

Description

The position of the left side of the item (in points, measured from the left side of the page).

Type

Number (double).

95.1.11 MeshItem.locked

`app.activeDocument.meshItems[index].locked`

Description

If `true`, this item is locked.

Type

Boolean.

95.1.12 MeshItem.name

`app.activeDocument.meshItems[index].name`

Description

The name of this item.

Type

String.

95.1.13 MeshItem.note

`app.activeDocument.meshItems[index].note`

Description

The note assigned to this item.

Type

String.

95.1.14 MeshItem.opacity

`app.activeDocument.meshItems[index].opacity`

Description

The opacity of the object. Range: 0.0 to 100.0.

Type

Number (double).

95.1.15 MeshItem.parent

`app.activeDocument.meshItems[index].parent`

Description

The parent of this object.

Type

Layer or *GroupItem*, read-only.

95.1.16 MeshItem.position

`app.activeDocument.meshItems[index].position`

Description

The position (in points) of the top left corner of the *MeshItem* object in the format [x, y]. Does not include stroke weight.

Type

Array of 2 numbers.

95.1.17 MeshItem.selected

`app.activeDocument.meshItems[index].selected`

Description

If `true`, this item is selected.

Type

Boolean.

95.1.18 MeshItem.sliced

`app.activeDocument.meshItems[index].sliced`

Description

If `true`, the item sliced. Default: `false`.

Type

Boolean.

95.1.19 MeshItem.tags

`app.activeDocument.meshItems[index].tags`

Description

The tags contained in this item.

Type

Tags, read-only.

95.1.20 MeshItem.top

`app.activeDocument.meshItems[index].top`

Description

The position of the top of the item (in points, measured from the bottom of the page).

Type

Number (double).

95.1.21 MeshItem.typename

`app.activeDocument.meshItems[index].typename`

Description

The class name of the referenced object.

Type

String, read-only.

95.1.22 MeshItem.uRL

`app.activeDocument.meshItems[index].uRL`

Description

The value of the Adobe URL tag assigned to this item.

Type

String.

95.1.23 MeshItem.visibilityVariable

`app.activeDocument.meshItems[index].visibilityVariable`

Description

The visibility variable bound to the item.

Type

Variable

95.1.24 MeshItem.visibleBounds

`app.activeDocument.meshItems[index].visibleBounds`

Description

The visible bounds of the item including stroke width.

Type

Array of 4 numbers, read-only.

95.1.25 MeshItem.width

`app.activeDocument.meshItems[index].width`

Description

The width of the item.

Type

Number (double).

95.1.26 MeshItem.wrapInside

`app.activeDocument.meshItems[index].wrapInside`

Description

If `true`, the text frame object should be wrapped inside this object.

Type

Boolean.

95.1.27 MeshItem.wrapOffset

`app.activeDocument.meshItems[index].wrapOffset`

Description

The offset to use when wrapping text around this object.

Type

Number (double).

95.1.28 MeshItem.wrapped

`app.activeDocument.meshItems[index].wrapped`

Description

If `true`, wrap text frame objects around this object (text frame must be above the object).

Type

Boolean.

95.1.29 MeshItem.zOrderPosition

`app.activeDocument.meshItems[index].zOrderPosition`

Description

The position of this item within the stacking order of the group or layer (`parent`) that contains the item.

Type

Number (long), read-only.

95.2 Methods

95.2.1 MeshItem.duplicate()

`app.activeDocument.meshItems[index].duplicate([relativeObject] [, insertionLocation])`

Description

Creates a duplicate of the selected object.

Parameters

Parameter	Type	Description
`relativeObject`	Object, optional	Object to duplicate to
`insertionLocation`	*ElementPlacement*, optional	Location to insert element

Returns

MeshItem

95.2.2 MeshItem.move()

```
app.activeDocument.meshItems[index].move(relativeObject, insertionLocation)
```

Description

Moves the object.

Parameters

Parameter	Type	Description
`relativeObject`	Object	Object to move element within
`insertionLocation`	*ElementPlacement*, optional	Location to move element to

Returns

MeshItem

95.2.3 MeshItem.remove()

```
app.activeDocument.meshItems[index].move()
```

Description

Deletes this object.

Returns

Nothing.

95.2.4 MeshItem.resize()

```
app.activeDocument.meshItems[index].resize(scaleX, scaleY
  [,changePositions] [,changeFillPatterns] [,changeFillGradients]
  [,changeStrokePattern] [,changeLineWidths] [,scaleAbout]
)
```

Description

Scales the art item where `scaleX` is the horizontal scaling factor and `scaleY` is the vertical scaling factor. 100.0 = 100%.

Parameters

Parameter	Type	Description
`scaleX`	Number (double)	Horizontal scaling factor
`scaleY`	Number (double)	Vertical scaling factor
`changePositions`	Boolean, optional	Whether to effect art object positions and orientations
`changeFillPatterns`	Boolean, optional	Whether to transform fill patterns
`changeFillGradients`	Boolean, optional	Whether to transform fill gradients
`changeStrokePattern`	Boolean, optional	Whether to transform stroke patterns
`changeLineWidths`	Number (double), optional	The amount to scale line widths
`scaleAbout`	*Transformation*, optional	The point to use as anchor, to transform about

Returns

Nothing.

95.2.5 MeshItem.rotate()

```
app.activeDocument.meshItems[index].rotate(angle [,changePositions]
  [,changeFillPatterns] [,changeFillGradients]
  [,changeStrokePattern] [,rotateAbout]
)
```

Description

Rotates the art item relative to the current rotation. The object is rotated counter-clockwise if the `angle` value is positive, clockwise if the value is negative.

Parameters

Parameter	Type	Description
`angle`	Number (double)	The angle amount to rotate the element
`changePositions`	Boolean, optional	Whether to effect art object positions and orientations
`changeFillPatterns`	Boolean, optional	Whether to transform fill patterns
`changeFillGradients`	Boolean, optional	Whether to transform fill gradients
`changeStrokePattern`	Boolean, optional	Whether to transform stroke patterns
`rotateAbout`	*Transformation*, optional	The point to use as anchor, to transform about

Returns

Nothing.

95.2.6 MeshItem.transform()

```
app.activeDocument.meshItems[index].transform(transformationMatrix
  [,changePositions] [,changeFillPatterns] [,changeFillGradients]
  [,changeStrokePattern] [,changeLineWidths] [,transformAbout]
)
```

Description

Transforms the art item by applying a transformation matrix.

Parameters

Parameter	Type	Description
`transformationMatrix`	*Matrix*	Transformation matrix to apply
`changePositions`	Boolean, optional	Whether to change Positions
`changeFillPatterns`	Boolean, optional	Whether to change Fill Patterns
`changeFillGradients`	Boolean, optional	Whether to change Fill Gradients
`changeStrokePattern`	Boolean, optional	Whether to change Stroke Pattern
`changeLineWidths`	Number (double), optional	The amount to scale line widths
`transformAbout`	*Transformation*, optional	The point to use as anchor, to transform about

Returns

Nothing.

95.2.7 MeshItem.translate()

```
app.activeDocument.meshItems[index].translate([deltaX] [,deltaY]
  [,transformObjects] [,transformFillPatterns]
  [,transformFillGradients] [,transformStrokePatterns]
)
```

Description

Repositions the art item relative to the current position, where `deltaX` is the horizontal offset and `deltaY` is the vertical offset.

Parameters

Parameter	Type	Description
`deltaX`	Number (double), optional	Horizontal offset
`deltaY`	Number (double), optional	Vertical offset
`transformObjects`	Boolean, optional	Whether to transform Objects
`transformFillPatterns`	Boolean, optional	Whether to transform Fill Patterns
`transformFillGradients`	Boolean, optional	Whether to transform Fill Gradients
`transformStrokePatterns`	Boolean, optional	Whether to transform Stroke Patterns

Returns

Nothing.

95.2.8 MeshItem.zOrder()

`app.activeDocument.meshItems[index].zOrder(zOrderCmd)`

Description

Arranges the art item's position in the stacking order of the group or layer (parent) of this object.

Parameters

Parameter	Type	Description
`zOrderCmd`	*ZOrderMethod*	Stacking order arrangement method

Returns

Nothing.

95.3 Example

95.3.1 Example name

```
// Locks all mesh items in the current document
if (app.documents.length > 0) {
  var doc = app.activeDocument;
  for (var i = 0; i < doc.meshItems.length; i++) {
    doc.meshItems[i].locked = true;
  }
}
```

CHAPTER 96

MeshItems

`app.activeDocument.meshItems`

Description

A collection of *MeshItem* objects.

96.1 Properties

96.1.1 MeshItems.length

`app.activeDocument.meshItems.length`

Description

The number of objects in the collection.

Type

Number, read-only.

96.1.2 MeshItems.parent

`app.activeDocument.meshItems.parent`

Description

The parent of this object.

Type

Object, read-only.

96.1.3 MeshItems.typename

`app.activeDocument.meshItems.typename`

Description

The class name of the referenced object.

Type

String, read-only.

96.2 Methods

96.2.1 MeshItems.getByName()

`app.activeDocument.meshItems.getByName(name)`

Description

Gets the first element in the collection with the specified name.

Parameters

Parameter	Type	Description
`name`	String	Name of element to get

Returns

MeshItem

96.2.2 MeshItems.index()

`app.activeDocument.meshItems.index(itemKey)`

Description

Gets an element from the collection.

Parameters

Parameter	Type	Description
`itemKey`	String, Number	String or number key

Returns

MeshItem

96.2.3 MeshItems.removeAll()

`app.activeDocument.meshItems.removeAll()`

Description

Deletes all elements in this collection.

Returns

Nothing.

96.3 Example

96.3.1 Copying mesh items to another document

To run this script, have two open documents. One document should contain at least one mesh item, the other document can be empty. Make the empty document the frontmost before running the script.

```
// Copies all mesh items from one document to a new document
if (app.documents.length > 0) {
  var srcDoc = documents[0];
  var locationOffset = 0;
  var targetDoc = documents.add();
  for (var i = 0; i < srcDoc.meshItems.length; i++) {
    var srcItem = srcDoc.meshItems[i];
    var dupItem = srcDoc.meshItems[i].duplicate(targetDoc, ElementPlacement.
↪PLACEATEND);

    // offset the copied items' position on the y axis
    dupItem.position = Array(100, 50 + locationOffset);
    locationOffset += 50;
  }
}
```

CHAPTER 97

NonNativeItem

`nonNativeItems[index`

Description

A non-native artwork item.

97.1 Properties

97.1.1 NonNativeItem.artworkKnockout

`nonNativeItems[index].artworkKnockout`

Description

Is this object used to create a knockout, and if so, what kind of knockout.

Type

KnockoutState

97.1.2 NonNativeItem.blendingMode

`nonNativeItems[index].blendingMode`

Description

The blend mode used when compositing an object.

Type

BlendModes

97.1.3 NonNativeItem.controlBounds

`nonNativeItems[index].controlBounds`

Description

The bounds of the object including stroke width and controls.

Type

Array of 4 numbers, read-only.

97.1.4 NonNativeItem.editable

`nonNativeItems[index].editable`

Description

If `true`, this item is editable.

Type

Boolean, read-only.

97.1.5 NonNativeItem.geometricBounds

`nonNativeItems[index].geometricBounds`

Description

The bounds of the object excluding stroke width.

Type

Array of 4 numbers, read-only.

97.1.6 NonNativeItem.height

`nonNativeItems[index].height`

Description

The height of the group item.

Type

Number (double).

97.1.7 NonNativeItem.hidden

`nonNativeItems[index].hidden`

Description

If `true`, this item is hidden.

Type

Boolean.

97.1.8 NonNativeItem.isIsolated

`nonNativeItems[index].isIsolated`

Description

If `true`, this object is isolated.

Type

Boolean.

97.1.9 NonNativeItem.layer

`nonNativeItems[index].layer`

Description

The layer to which this item belongs.

Type

Layer, read-only.

97.1.10 NonNativeItem.left

`nonNativeItems[index].left`

Description

The position of the left side of the item (in points, measured from the left side of the page).

Type

Number (double).

97.1.11 NonNativeItem.locked

`nonNativeItems[index].locked`

Description

If `true`, this item is locked.

Type

Boolean.

97.1.12 NonNativeItem.name

`nonNativeItems[index].name`

Description

The name of this item.

Type

String.

97.1.13 NonNativeItem.note

`nonNativeItems[index].note`

Description

The note assigned to this item.

Type

String.

97.1.14 NonNativeItem.opacity

`nonNativeItems[index].opacity`

Description

The opacity of the object. Range: 0.0 to 100.0.

Type

Number (double).

97.1.15 NonNativeItem.parent

`nonNativeItems[index].parent`

Description

The parent of this object.

Type

Document, *Layer* or *GroupItem*, read-only.

97.1.16 NonNativeItem.position

`nonNativeItems[index].position`

Description

The position (in points) of the top left corner of the `NonNativeItems[index` object in the format [x, y]. Does not include stroke weight.

Type

Array of 2 numbers.

97.1.17 NonNativeItem.selected

`nonNativeItems[index].selected`

Description

If `true`, this item is selected.

Type

Boolean.

97.1.18 NonNativeItem.sliced

`nonNativeItems[index].sliced`

Description

If `true`, the item sliced. Default: `false`.

Type

Boolean.

97.1.19 NonNativeItem.tags

`nonNativeItems[index].tags`

Description

The tags contained in this item.

Type

Tags, read-only.

97.1.20 NonNativeItem.top

`nonNativeItems[index].top`

Description

The position of the top of the item (in points, measured from the bottom of the page).

Type

Number (double).

97.1.21 NonNativeItem.typename

`nonNativeItems[index].typename`

Description

The class name of the referenced object.

Type

String, read-only.

97.1.22 NonNativeItem.uRL

`nonNativeItems[index].uRL`

Description

The value of the Adobe URL tag assigned to this item.

Type

String.

97.1.23 NonNativeItem.visibilityVariable

`nonNativeItems[index].visibilityVariable`

Description

The visibility variable bound to the item.

Type

Variable

97.1.24 NonNativeItem.visibleBounds

`nonNativeItems[index].visibleBounds`

Description

The visible bounds of the item including stroke width.

Type

Array of 4 numbers, read-only.

97.1.25 NonNativeItem.width

`nonNativeItems[index].width`

Description

The width of the item.

Type

Number (double).

97.1.26 NonNativeItem.wrapInside

`nonNativeItems[index].wrapInside`

Description

If `true`, the non-native-item object should be wrapped inside this object.

Type

Boolean.

97.1.27 NonNativeItem.wrapOffset

`nonNativeItems[index].wrapOffset`

Description

The offset to use when wrapping text around this object.

Type

Number (double).

97.1.28 NonNativeItem.wrapped

`nonNativeItems[index].wrapped`

Description

If `true`, wrap non-native-item objects around this object (non-native-item object must be above the object).

Type

Boolean.

97.1.29 NonNativeItem.zOrderPosition

`nonNativeItems[index].zOrderPosition`

Description

The position of this item within the stacking order of the group or layer (`parent`) that contains the item.

Type

Number, read-only.

97.2 Methods

97.2.1 NonNativeItem.duplicate()

`nonNativeItems[index].duplicate([relativeObject] [,insertionLocation])`

Description

Creates a duplicate of the selected object.

Parameters

Parameter	Type	Description
`relativeObject`	Object, optional	Object to duplicate to
`insertionLocation`	*ElementPlacement*, optional	Location to insert element

Returns

NonNativeItem

97.2.2 NonNativeItem.move()

```
nonNativeItems[index].move(relativeObject, insertionLocation)
```

Description

Moves the object.

Parameters

Parameter	Type	Description
`relativeObject`	Object	Object to move element within
`insertionLocation`	*ElementPlacement*, optional	Location to move element to

Returns

NonNativeItem

97.2.3 NonNativeItem.remove()

```
nonNativeItems[index].remove()
```

Description

Deletes this object.

Returns

Nothing.

97.2.4 NonNativeItem.removeAll()

```
nonNativeItems[index].removeAll()
```

Description

Deletes all elements in this collection.

Returns

Nothing.

97.2.5 NonNativeItem.resize()

```
nonNativeItem.resize(scaleX, scaleY
  [,changePositions] [,changeFillPatterns] [,changeFillGradients]
  [,changeStrokePattern] [,changeLineWidths] [,scaleAbout]
)
```

Description

Scales the art item where `scaleX` is the horizontal scaling factor and `scaleY` is the vertical scaling factor. 100.0 = 100%.

Parameters

Parameter	Type	Description
`scaleX`	Number (double)	Horizontal scaling factor
`scaleY`	Number (double)	Vertical scaling factor
`changePositions`	Boolean, optional	Whether to effect art object positions and orientations
`changeFillPatterns`	Boolean, optional	Whether to transform fill patterns
`changeFillGradients`	Boolean, optional	Whether to transform fill gradients
`changeStrokePattern`	Boolean, optional	Whether to transform stroke patterns
`changeLineWidths`	Number (double), optional	The amount to scale line widths
`scaleAbout`	*Transformation*, optional	The point to use as anchor, to transform about

Returns

Nothing.

97.2.6 NonNativeItem.rotate()

```
nonNativeItem.rotate(angle
  [,changePositions] [,changeFillPatterns]
  [,changeFillGradients] [,changeStrokePattern] [,rotateAbout]
)
```

Description

Rotates the art item relative to the current rotation. The object is rotated counter-clockwise if the `angle` value is positive, clockwise if the value is negative.

Parameters

Parameter	Type	Description
`angle`	Number (double)	The angle amount to rotate the element
`changePositions`	Boolean, optional	Whether to effect art object positions and orientations
`changeFillPatterns`	Boolean, optional	Whether to transform fill patterns
`changeFillGradients`	Boolean, optional	Whether to transform fill gradients
`changeStrokePattern`	Boolean, optional	Whether to transform stroke patterns
`rotateAbout`	*Transformation*, optional	The point to use as anchor, to transform about

Returns

Nothing.

97.2.7 NonNativeItem.transform()

```
nonNativeItem.transform(transformationMatrix
  [,changePositions] [,changeFillPatterns] [,changeFillGradients]
  [,changeStrokePattern] [,changeLineWidths] [,transformAbout]
)
```

Description

Transforms the art item by applying a transformation matrix.

Parameters

Parameter	Type	Description
`transformationMatrix`	*Matrix*	Transformation matrix to apply
`changePositions`	Boolean, optional	Whether to change Positions
`changeFillPatterns`	Boolean, optional	Whether to change Fill Patterns
`changeFillGradients`	Boolean, optional	Whether to change Fill Gradients
`changeStrokePattern`	Boolean, optional	Whether to change Stroke Pattern
`changeLineWidths`	Number (double), optional	The amount to scale line widths
`transformAbout`	*Transformation*, optional	The point to use as anchor, to transform about

Returns

Nothing.

97.2.8 NonNativeItem.translate()

```
nonNativeItem.translate([deltaX] [,deltaY]
  [,transformObjects] [,transformFillPatterns]
  [,transformFillGradients] [,transformStrokePatterns]
)
```

Description

Repositions the art item relative to the current position, where `deltaX` is the horizontal offset and `deltaY` is the vertical offset.

Parameters

Parameter	Type	Description
`deltaX`	Number (double), optional	Horizontal offset
`deltaY`	Number (double), optional	Vertical offset
`transformObjects`	Boolean, optional	Whether to transform Objects
`transformFillPatterns`	Boolean, optional	Whether to transform Fill Patterns
`transformFillGradients`	Boolean, optional	Whether to transform Fill Gradients
`transformStrokePatterns`	Boolean, optional	Whether to transform Stroke Patterns

Returns

Nothing.

97.2.9 NonNativeItem.zOrder()

```
nonNativeItems[index].zOrder(zOrderCmd)
```

Description

Arranges the art item's position in the stacking order of the group or layer (parent) of this object.

Parameters

Parameter	Type	Description
`zOrderCmd`	*ZOrderMethod*	Stacking order arrangement method

Returns

Nothing.

CHAPTER 98

NonNativeItems

`nonNativeItems`

Description

A collection of *NonNativeItem* objects.

98.1 Properties

98.1.1 NonNativeItems.length

`nonNativeItems.length`

Description

The number of objects in the collection.

Type

Number, read-only.

98.1.2 NonNativeItems.parent

`nonNativeItems.parent`

Description

The parent of this object.

Type

Object, read-only.

98.1.3 NonNativeItems.typename

`nonNativeItems.typename`

Description

The class name of the referenced object.

Type

String, read-only.

98.2 Methods

98.2.1 NonNativeItems.getByName()

`nonNativeItems.getByName(name)`

Description

Gets the first element in the collection with the specified name.

Parameters

Parameter	Type	Description
`name`	String	Name of element to get

Returns

NonNativeItem, *SymbolItem*

CHAPTER 99

OpenOptions

`openOptions`

Description

Options for opening a document, used with the *Application.open()* method.

99.1 Properties

99.1.1 OpenOptions.convertCropAreaToArboard

`openOptions.convertCropAreaToArboard`

Description

Optional. Convert crop areas to artboards when opening a legacy document in Illustrator CS4 or later. When `false`, crop areas are discarded. Default: `true`.

Type

Boolean.

99.1.2 OpenOptions.convertTilesToArboard

`openOptions.convertTilesToArboard`

Description

Optional. Convert print tiles to artboards when opening a legacy document in Illustrator CS4 or later. Default: `false`.

Type

Boolean.

99.1.3 OpenOptions.createArtboardWithArtworkBoundingBox

`openOptions.createArtboardWithArtworkBoundingBox`

Description

Optional. Create an artboard with the dimensions of the bounding box of the artwork when opening a legacy document in Illustrator CS4 or later. Default: `false`.

Type

Boolean.

99.1.4 OpenOptions.openAs

`openOptions.openAs`

Description

Optional. Open the file as an Illustrator library of this type. Default: `LibraryType.IllustratorArtwork`.

Type

LibraryType

99.1.5 OpenOptions.preserveLegacyArtboard

`openOptions.preserveLegacyArtboard`

Description

Optional. Preserve legacy artboards when opening a legacy document in Illustrator CS4 or later. Default: `true`.

Type

Boolean.

99.1.6 OpenOptions.updateLegacyGradientMesh

`openOptions.updateLegacyGradientMesh`

Description

If `true`, preserves the spot colors in the gradient mesh objects for legacy documents (pre-Illustrator CS4). Default: `true`.

Type

Boolean.

99.1.7 OpenOptions.updateLegacyText

`openOptions.updateLegacyText`

Description

Optional. If `true`, update all legacy text items (from previous versions of Illustrator). Default: `false`.

Type

Boolean.

99.2 Example

99.2.1 Automatically updating legacy text on open

```
// Opens a file with legacy text (AI 10 or older), using
// OpenOptions to automatically update the legacy text.

var fileRef = filePath;
if (fileRef != null) {
  var openOptions = new OpenOptions();
  openOptions.updateLegacyText = true;

  var docRef = open(fileRef, DocumentColorSpace.RGB, openOptions);
}
```

CHAPTER 100

OpenOptionsAutoCAD

`openOptionsAutoCAD`

Description

Options for opening an AutoCAD drawing, used with the *Application.open()* method.

100.1 Properties

100.1.1 OpenOptionsAutoCAD.centerArtwork

`openOptionsAutoCAD.centerArtwork`

Description

If `true`, the artwork is centered on the artboard. Default: `true`.

Type

Boolean.

100.1.2 OpenOptionsAutoCAD.globalScaleOption

`openOptionsAutoCAD.globalScaleOption`

Description

How to scale the drawing on import. Default: `AutoCADGlobalScaleOption.FitArtboard`.

Type

AutoCADGlobalScaleOption

100.1.3 OpenOptionsAutoCAD.globalScalePercent

`openOptionsAutoCAD.globalScalePercent`

Description

The value when `globalScaleOption` is `AutoCADGlobalScaleOption.ScaleByValue`, expressed as a percentage. Range: 0.0 to 100.0. Default is 100.0.

Type

Number (double).

100.1.4 OpenOptionsAutoCAD.mergeLayers

`openOptionsAutoCAD.mergeLayers`

Description

If `true`, the layers of the artwork are merged. Default: `false`.

Type

Boolean.

100.1.5 OpenOptionsAutoCAD.parent

`openOptionsAutoCAD.parent`

Description

The object's container.

Type

Object, read-only.

100.1.6 OpenOptionsAutoCAD.scaleLineweights

`openOptionsAutoCAD.scaleLineweights`

Description

If `true`, line weights are scaled by the same factor as the rest of the drawing. Default: `false`.

Type

Boolean.

100.1.7 OpenOptionsAutoCAD.selectedLayoutName

`openOptionsAutoCAD.selectedLayoutName`

Description

The name of the layout in the drawing to import.

Type

String.

100.1.8 OpenOptionsAutoCAD.typename

`openOptionsAutoCAD.typename`

Description

The class name of the object.

Type

String, read-only.

100.1.9 OpenOptionsAutoCAD.unit

`openOptionsAutoCAD.unit`

Description

The unit to map to. Default: `AutoCADUnit.Millimeters`.

Type

AutoCADUnit

100.1.10 OpenOptionsAutoCAD.unitScaleRatio

`openOptionsAutoCAD.unitScaleRatio`

Description

The ratio by which to scale while mapping units. Default: 1.0.

Type

Number (double).

CHAPTER 101

OpenOptionsFreeHand

`openOptionsFreeHand`

Description

Options for opening a FreeHand file.

101.1 Properties

101.1.1 OpenOptionsFreeHand.convertTextToOutlines

`openOptionsFreeHand.convertTextToOutlines`

Description

If `true`, all text is converted to vector paths; preserves the visual appearance of type. Default: `false`.

Type

Boolean.

101.1.2 OpenOptionsFreeHand.importSinglePage

`openOptionsFreeHand.importSinglePage`

Description

If `true`, imports only the page specified in the `pageToOpen` property. Default: `true`.

Type

Boolean.

101.1.3 OpenOptionsFreeHand.pageToOpen

`openOptionsFreeHand.pageToOpen`

Description

The number of the page to import when opening a multipage document. Valid only when `importSinglePage` is `true`.

Type

Number (long).

101.1.4 OpenOptionsFreeHand.parent

`openOptionsFreeHand.parent`

Description

The parent of this object.

Type

Object, read-only.

101.1.5 OpenOptionsFreeHand.typename

`openOptionsFreeHand.typename`

Description

The class name of the referenced object.

Type

String, read-only.

CHAPTER 102

OpenOptionsPhotoshop

`openOptionsPhotoshop`

Description

Options for opening a Photoshop document, used with the *Application.open()* method.

102.1 Properties

102.1.1 OpenOptionsPhotoshop.layerComp

`openOptionsPhotoshop.layerComp`

Description

The name of the layer comp to use when the document is converted.

Type

String.

102.1.2 OpenOptionsPhotoshop.preserveHiddenLayers

`openOptionsPhotoshop.preserveHiddenLayers`

Description

If `true`, preserve hidden layers when the document is converted. Default: `false`.

Type

Boolean.

102.1.3 OpenOptionsPhotoshop.preserveImageMaps

`openOptionsPhotoshop.preserveImageMaps`

Description

If `true`, preserve image maps when the document is converted. Default: `true`.

Type

Boolean.

102.1.4 OpenOptionsPhotoshop.preserveLayers

`openOptionsPhotoshop.preserveLayers`

Description

If `true`, preserve layers when the document is converted. Default: `true`.

Type

Boolean.

102.1.5 OpenOptionsPhotoshop.preserveSlices

`openOptionsPhotoshop.preserveSlices`

Description

If `true`, preserve slices when the document is converted. Default: `true`.

Type

Boolean.

102.1.6 OpenOptionsPhotoshop.typename

`openOptionsPhotoshop.typename`

Description

The class name of the object.

Type

String, read-only.

CHAPTER 103

PageItem

`app.activeDocument.pageItems[index`

Description

Any art item. Every art item and group in a document is a page item. You may refer to a page item as an element of a document, layer, or group item.

The `PageItem` class gives you complete access to every art item contained in an Illustrator document. The `PageItem` class is the superclass of all artwork objects in a document. The *CompoundPathItem*, *GroupItem*, *MeshItem*, *PathItem*, *PlacedItem*, *PluginItem*, *RasterItem*, and *TextFrameItem* classes each inherit a set of properties from the `PageItem` class.

You cannot create a `PageItem` directly, you must create one of the specific `PageItem` subclasses, such as *PathItem*.

103.1 Properties

103.1.1 PageItem.artworkKnockout

`app.activeDocument.pageItems[index].artworkKnockout`

Description

Is this object used to create a knockout.

Type

KnockoutState

103.1.2 PageItem.blendingMode

`app.activeDocument.pageItems[index].blendingMode`

Description

The mode to use when compositing this object. An object is considered composited when its opacity is set to less than 100.0 (100%).

Type

BlendModes

103.1.3 PageItem.controlBounds

`app.activeDocument.pageItems[index].controlBounds`

Description

The bounds of the object including stroke width and controls.

Type

Rect, read-only.

103.1.4 PageItem.editable

`app.activeDocument.pageItems[index].editable`

Description

If `true`, this page item is editable.

Type

Boolean, read-only.

103.1.5 PageItem.geometricBounds

`app.activeDocument.pageItems[index].geometricBounds`

Description

The object's bounds excluding the stroke width.

Type

Array of 4 numbers, read-only.

103.1.6 PageItem.height

`app.activeDocument.pageItems[index].height`

Description

The height of the page item, calculated from the geometric bounds. Range: 0.0 to 16348.0.

Type

Number (double).

103.1.7 PageItem.hidden

`app.activeDocument.pageItems[index].hidden`

Description

If `true`, this page item is hidden.

Type

Boolean.

103.1.8 PageItem.isIsolated

`app.activeDocument.pageItems[index].isIsolated`

Description

If `true`, this object is isolated.

Type

Boolean.

103.1.9 PageItem.layer

`app.activeDocument.pageItems[index].layer`

Description

The layer to which this page item belongs.

Type

Layer, read-only.

103.1.10 PageItem.left

`app.activeDocument.pageItems[index].left`

Description

The left position of the art item.

Type

Number (double).

103.1.11 PageItem.locked

`app.activeDocument.pageItems[index].locked`

Description

If `true`, this page item is locked.

Type

Boolean.

103.1.12 PageItem.name

`app.activeDocument.pageItems[index].name`

Description

The name of this page item.

Type

String.

103.1.13 PageItem.note

`app.activeDocument.pageItems[index].note`

Description

The note assigned to this item.

Type

String.

103.1.14 PageItem.opacity

`app.activeDocument.pageItems[index].opacity`

Description

The opacity of this object, where 100.0 is completely opaque and 0.0 is completely transparent.

Type

Number (double).

103.1.15 PageItem.parent

`app.activeDocument.pageItems[index].parent`

Description

The parent of this object.

Type

Object, read-only.

103.1.16 PageItem.pixelAligned

`app.activeDocument.pageItems[index].pixelAligned`

Description

`True` if this item is aligned to the pixel grid.

Type

Boolean.

103.1.17 PageItem.position

`app.activeDocument.pageItems[index].position`

Description

The position (in points) of the top left corner of the item in the format {x, y}. Does not include stroke weight.

Type

Array of 2 numbers.

103.1.18 PageItem.selected

`app.activeDocument.pageItems[index].selected`

Description

If `true`, this object is selected.

Type

Boolean.

103.1.19 PageItem.sliced

`app.activeDocument.pageItems[index].sliced`

Description

If `true`, preserve slices.

Type

Boolean.

103.1.20 PageItem.tags

`app.activeDocument.pageItems[index].tags`

Description

The collection of tags associated with this page item.

Type

Tags

103.1.21 PageItem.top

`app.activeDocument.pageItems[index].top`

Description

The top position of the art item.

Type

Number (double).

103.1.22 PageItem.typename

`app.activeDocument.pageItems[index].typename`

Description

The class name of the object.

Type

String, read-only.

103.1.23 PageItem.uRL

`app.activeDocument.pageItems[index].uRL`

Description

The value of the Adobe URL tag assigned to this page item.

Type

String.

103.1.24 PageItem.uuid

`app.activeDocument.pageItems[index].uuid`

Note: This functionality was added in Illustrator 24.0. (CC2020)

Description

The unique identifier for this pageItem

Type

String, read-only.

103.1.25 PageItem.visibilityVariable

`app.activeDocument.pageItems[index].visibilityVariable`

Description

The visibility variable to which this page item path is bound.

Type

Variable

103.1.26 PageItem.visibleBounds

`app.activeDocument.pageItems[index].visibleBounds`

Description

The object's visible bounds, including stroke width of any objects in the illustration.

Type

Array of 4 numbers, read-only.

103.1.27 PageItem.width

`app.activeDocument.pageItems[index].width`

Description

The width of the page item, calculated from the geometric bounds. Range: 0.0 to 16348.0.

Type

Number (double).

103.1.28 PageItem.wrapInside

`app.activeDocument.pageItems[index].wrapInside`

Description

If `true`, the text frame object should be wrapped inside this object.

Type

Boolean.

103.1.29 PageItem.wrapOffset

`app.activeDocument.pageItems[index].wrapOffset`

Description

The offset to use when wrapping text around this object.

Type

Number (double).

103.1.30 PageItem.wrapped

```
app.activeDocument.pageItems[index].wrapped
```

Description

If `true`, wrap text frame objects around this object (text frame must be above the object).

Type

Boolean.

103.1.31 PageItem.zOrderPosition

```
app.activeDocument.pageItems[index].zOrderPosition
```

Description

The drawing order of the art within its group or layer.

Type

Number (long), read-only.

103.2 Methods

103.2.1 PageItem.bringInPerspective()

```
app.activeDocument.pageItems[index].bringInPerspective(posX, posY,
perspectiveGridPlane)
```

Description

Places art object(s) in a perspective grid at a specified position and grid plane.

Parameters

Parameter	Type	Description
`posX`	Number	X position to place art at
`posY`	Number	Y position to place art at
`perspectiveGridPlane`	*PerspectiveGridPlaneType*	Perspective grid plane type to use

Returns

Returns.

103.2.2 PageItem.resize()

```
app.activeDocument.pageItems[index].resize(
    scaleX, scaleY [,changePositions] [,changeFillPatterns] [,changeFillGradients]
    [,changeStrokePattern] [,changeLineWidths] [,scaleAbout]
)
```

Description

Scales the art item where `scaleX` is the horizontal scaling factor and `scaleY` is the vertical scaling factor. 100.0 = 100%.

Parameters

Parameter	Type	Description
`scaleX`	Number (double)	Horizontal scaling factor
`scaleY`	Number (double)	Vertical scaling factor
`changePositions`	Boolean, optional	Whether to effect art object positions and orientations
`changeFillPatterns`	Boolean, optional	Whether to transform fill patterns
`changeFillGradients`	Boolean, optional	Whether to transform fill gradients
`changeStrokePattern`	Boolean, optional	Whether to transform stroke patterns
`changeLineWidths`	Number (double), optional	The amount to scale line widths
`scaleAbout`	*Transformation*, optional	The point to use as anchor, to transform about

Returns

Nothing.

103.2.3 PageItem.rotate()

```
app.activeDocument.pageItems[index].rotate(
    angle [,changePositions] [,changeFillPatterns]
    [,changeFillGradients] [,changeStrokePattern] [,rotateAbout]
)
```

Description

Rotates the art item relative to the current rotation. The object is rotated counter-clockwise if the `angle` value is positive, clockwise if the value is negative.

Parameters

Parameter	Type	Description
`angle`	Number (double)	The angle amount to rotate the element
`changePositions`	Boolean, optional	Whether to effect art object positions and orientations
`changeFillPatterns`	Boolean, optional	Whether to transform fill patterns
`changeFillGradients`	Boolean, optional	Whether to transform fill gradients
`changeStrokePattern`	Boolean, optional	Whether to transform stroke patterns
`rotateAbout`	*Transformation*, optional	The point to use as anchor, to transform about

Returns

Nothing.

103.2.4 PageItem.transform()

```
app.activeDocument.pageItems[index].transform(
    transformationMatrix [,changePositions] [,changeFillPatterns] [,
↪changeFillGradients]
    [,changeStrokePattern] [,changeLineWidths] [,transformAbout]
)
```

Description

Transforms the art item by applying a transformation matrix.

Parameters

Parameter	Type	Description
`transformationMatrix`	*Matrix*	Transformation matrix to apply
`changePositions`	Boolean, optional	Whether to change Positions
`changeFillPatterns`	Boolean, optional	Whether to change Fill Patterns
`changeFillGradients`	Boolean, optional	Whether to change Fill Gradients
`changeStrokePattern`	Boolean, optional	Whether to change Stroke Pattern
`changeLineWidths`	Number (double), optional	The amount to scale line widths
`transformAbout`	*Transformation*, optional	The point to use as anchor, to transform about

Returns

Nothing.

103.2.5 PageItem.translate()

```
app.activeDocument.pageItems[index].translate(
    deltaX [,deltaY] [,transformObjects] [,transformFillPatterns]
    [,transformFillGradients] [,transformStrokePatterns]
)
```

Description

Repositions the art item relative to the current position, where `deltaX` is the horizontal offset and `deltaY` is the vertical offset.

Parameters

Parameter	Type	Description
`deltaX`	Number (double), optional	Horizontal offset
`deltaY`	Number (double), optional	Vertical offset
`transformObjects`	Boolean, optional	Whether to transform Objects
`transformFillPatterns`	Boolean, optional	Whether to transform Fill Patterns
`transformFillGradients`	Boolean, optional	Whether to transform Fill Gradients
`transformStrokePatterns`	Boolean, optional	Whether to transform Stroke Patterns

Returns

Nothing.

103.2.6 PageItem.zOrder()

```
app.activeDocument.pageItems[index].zOrder(zOrderCmd)
```

Description

Arranges the art item's position in the stacking order of the group or layer (parent) of this object.

Parameters

Parameter	Type	Description
`zOrderCmd`	*ZOrderMethod*	Stacking order arrangement method

Returns

Nothing.

CHAPTER 104

PageItems

`app.activeDocument.pageItems`

Description

A collection of *PageItem* objects. Provides complete access to all the art items in an Illustrator document in the following classes:

104.1 PathItem

`app.activeDocument.pathItems[index]`

Description

Specifies a path item, which contains *PathPoint* objects that define its geometry.

The `PathItem` class gives you complete access to paths in Illustrator.

The `setEntirePath` method provides an extremely efficient way to create paths comprised of straight lines.

104.1.1 Properties

PathItem.area

`app.activeDocument.pathItems[index].area`

Description

The area of this path in square points.

If the area is negative, the path is wound counterclockwise.

Self-intersecting paths can contain sub-areas that cancel each other out, which makes this value zero even though the path has apparent area.

Type

Number (double); read-only.

PathItem.artworkKnockout

`app.activeDocument.pathItems[index].artworkKnockout`

Description

Is this object used to create a knockout, and if so, what kind of knockout.

Type

KnockoutState

PathItem.blendingMode

`app.activeDocument.pathItems[index].blendingMode`

Description

The blend mode used when compositing an object.

Type

BlendModes

PathItem.clipping

`app.activeDocument.pathItems[index].clipping`

Description

If `true`, this path should be used as a clipping path.

Type

Boolean

PathItem.closed

`app.activeDocument.pathItems[index].closed`

Description

If `true`, this path is closed.

Type

Boolean

PathItem.controlBounds

```
app.activeDocument.pathItems[index].controlBounds
```

Description

The bounds of the object including stroke width and controls.

Type

Array of 4 numbers; read-only.

PathItem.editable

```
app.activeDocument.pathItems[index].editable
```

Description

If `true`, this item is editable.

Type

Boolean; read-only.

PathItem.evenodd

```
app.activeDocument.pathItems[index].evenodd
```

Description

If `true`, the even-odd rule should be used to determine "insideness."

Type

Boolean

PathItem.fillColor

```
app.activeDocument.pathItems[index].fillColor
```

Description

The fill color of the path.

Type

Color

PathItem.filled

```
app.activeDocument.pathItems[index].filled
```

Description

If `true`, the path is filled.

Type

Boolean

PathItem.fillOverprint

```
app.activeDocument.pathItems[index].fillOverprint
```

Description

If `true`, the art beneath a filled object should be overprinted.

Type

Boolean

PathItem.geometricBounds

```
app.activeDocument.pathItems[index].geometricBounds
```

Description

The bounds of the object excluding stroke width.

Type

Array of 4 numbers; read-only.

PathItem.guides

```
app.activeDocument.pathItems[index].guides
```

Description

If `true`, this path is a guide object.

Type

Boolean

PathItem.height

`app.activeDocument.pathItems[index].height`

Description

The height of the group item.

Type

Number (double)

PathItem.hidden

`app.activeDocument.pathItems[index].hidden`

Description

If `true`, this item is hidden.

Type

Boolean

PathItem.isIsolated

`app.activeDocument.pathItems[index].isIsolated`

Description

If `true`, this object is isolated.

Type

Boolean

PathItem.layer

`app.activeDocument.pathItems[index].layer`

Description

The layer to which this item belongs.

Type

Layer; read-only.

PathItem.left

`app.activeDocument.pathItems[index].left`

Description

The position of the left side of the item (in points, measured from the left side of the page).

Type

Number (double)

PathItem.length

`app.activeDocument.pathItems[index].length`

Description

The length of this path in points.

Type

Number (double)

PathItem.locked

`app.activeDocument.pathItems[index].locked`

Description

If `true`, this item is locked.

Type

Boolean

PathItem.name

`app.activeDocument.pathItems[index].name`

Description

The name of this item.

Type

String

PathItem.note

`app.activeDocument.pathItems[index].note`

Description

The note assigned to this item.

Type

String

PathItem.opacity

`app.activeDocument.pathItems[index].opacity`

Description

The opacity of the object. Range: 0.0 to 100.0

Type

Number (double)

PathItem.parent

`app.activeDocument.pathItems[index].parent`

Description

The parent of this object.

Type

Layer or *GroupItem*

PathItem.pathPoints

`app.activeDocument.pathItems[index].pathPoints`

Description

The path points contained in this path item.

Type

PathPoints; read-only.

PathItem.pixelAligned

`app.activeDocument.pathItems[index].pixelAligned`

Description

`true` if this item is aligned to the pixel grid.

Type

Boolean

PathItem.polarity

`app.activeDocument.pathItems[index].polarity`

Description

The polarity of the path.

Type

PolarityValues

PathItem.position

`app.activeDocument.pathItems[index].position`

Description

The position (in points) of the top left corner of the `pluginItem` object in the format [x, y]. Does not include stroke weight.

Type

Array of 2 numbers; read-only.

PathItem.resolution

`app.activeDocument.pathItems[index].resolution`

Description

The resolution of the path in dots per inch (dpi).

Type

Number (double)

PathItem.selected

`app.activeDocument.pathItems[index].selected`

Description

If `true`, this item is selected.

Type

Boolean

PathItem.selectedPathPoints

`app.activeDocument.pathItems[index].selectedPathPoints`

Description

All of the selected path points in the path.

Type

PathPoints; read-only.

PathItem.sliced

`app.activeDocument.pathItems[index].sliced`

Description

If `true`, the item sliced.

Default: `false`

Type

Boolean

PathItem.strokeCap

`app.activeDocument.pathItems[index].strokeCap`

Description

The type of line capping.

Type

StrokeCap

PathItem.strokeColor

`app.activeDocument.pathItems[index].strokeColor`

Description

The stroke color for the path.

Type

Color

PathItem.stroked

`app.activeDocument.pathItems[index].stroked`

Description

If `true`, the path should be stroked.

Type

Boolean

PathItem.strokeDashes

`app.activeDocument.pathItems[index].strokeDashes`

Description

Dash lengths. Set to an empty object, {}, for a solid line.

Type

Object

PathItem.strokeDashOffset

`app.activeDocument.pathItems[index].strokeDashOffset`

Description

The default distance into the dash pattern at which the pattern should be started.

Type

Number (double)

PathItem.strokeJoin

```
app.activeDocument.pathItems[index].strokeJoin
```

Description

Type of joints for the path.

Type

StrokeJoin

PathItem.strokeMiterLimit

```
app.activeDocument.pathItems[index].strokeMiterLimit
```

Description

When a default stroke join is set to mitered, this property specifies when the join will be converted to beveled (squared-off) by default. The default miter limit of 4 means that when the length of the point reaches four times the stroke weight, the join switches from a miter join to a bevel join. A value of 1 specifies a bevel join. Range: 1 to 500. Default: 4

Type

Number (double)

PathItem.strokeOverprint

```
app.activeDocument.pathItems[index].strokeOverprint
```

Description

If `true`, the art beneath a stroked object should be overprinted.

Type

Boolean

PathItem.strokeWidth

```
app.activeDocument.pathItems[index].strokeWidth
```

Description

The width of the stroke (in points).

Type

Number (double)

PathItem.tags

`app.activeDocument.pathItems[index].tags`

Description

The tags contained in this item.

Type

Tags; read-only.

PathItem.top

`app.activeDocument.pathItems[index].top`

Description

The position of the top of the item (in points, measured from the bottom of the page).

Type

Number (double)

PathItem.typename

`app.activeDocument.pathItems[index].typename`

Description

The class name of the referenced object.

Type

String; read-only.

PathItem.uRL

`app.activeDocument.pathItems[index].uRL`

Description

The value of the Adobe URL tag assigned to this item.

Type

String

PathItem.visibilityVariable

```
app.activeDocument.pathItems[index].visibilityVariable
```

Description

The visibility variable bound to the item.

Type

Variable

PathItem.visibleBounds

```
app.activeDocument.pathItems[index].visibleBounds
```

Description

The visible bounds of the item including stroke width.

Type

Array of 4 numbers; read-only.

PathItem.width

```
app.activeDocument.pathItems[index].width
```

Description

The width of the item.

Type

Number (double)

PathItem.wrapInside

```
app.activeDocument.pathItems[index].wrapInside
```

Description

If `true`, the text frame object should be wrapped inside this object.

Type

Boolean

PathItem.wrapOffset

```
app.activeDocument.pathItems[index].wrapOffset
```

Description

The offset to use when wrapping text around this object.

Type

Number (double)

PathItem.wrapped

```
app.activeDocument.pathItems[index].wrapped
```

Description

If `true`, wrap text frame objects around this object (text frame must be above the object).

Type

Boolean

PathItem.zOrderPosition

```
app.activeDocument.pathItems[index].zOrderPosition
```

Description

The position of this item within the stacking order of the group or layer (`parent`) that contains the item.

Type

Number; read-only.

104.1.2 Methods

PathItem.duplicate()

```
app.activeDocument.pathItems[index].duplicate([relativeObject][,
insertionLocation])
```

Description

Creates a duplicate of the selected object.

Parameters

Parameter	Type	Description
`relativeObject`	Object, optional	Object to duplicate to
`insertionLocation`	*ElementPlacement*, optional	Location to insert element

Returns

PathItem

PathItem.move()

```
app.activeDocument.pathItems[index].move(relativeObject, insertionLocation)
```

Description

Moves the object.

Parameters

Parameter	Type	Description
`relativeObject`	Object	Object to move element within
`insertionLocation`	*ElementPlacement*, optional	Location to move element to

Returns

PathItem

PathItem.remove()

```
app.activeDocument.pathItems[index].remove()
```

Description

Deletes this object.

Returns

Nothing.

PathItem.resize()

```
app.activeDocument.pathItems[index].resize(scaleX, scaleY[,changePositions][,
changeFillPatterns][,changeFillGradients][,changeStrokePattern][,
changeLineWidths][,scaleAbout])
```

Description

Scales the art item where `scaleX` is the horizontal scaling factor and `scaleY` is the vertical scaling factor. 100.0 = 100%.

Parameters

Parameter	Type	Description
`scaleX`	Number (double)	Horizontal scaling factor
`scaleY`	Number (double)	Vertical scaling factor
`changePositions`	Boolean, optional	Whether to effect art object positions and orientations
`changeFillPatterns`	Boolean, optional	Whether to transform fill patterns
`changeFillGradients`	Boolean, optional	Whether to transform fill gradients
`changeStrokePattern`	Boolean, optional	Whether to transform stroke patterns
`changeLineWidths`	Number (double), optional	The amount to scale line widths
`scaleAbout`	*Transformation*, optional	The point to use as anchor, to transform about

Returns

Nothing.

PathItem.rotate()

```
app.activeDocument.pathItems[index].rotate(angle[,changePositions][,
changeFillPatterns][,changeFillGradients][,changeStrokePattern][,rotateAbout])
```

Description

Rotates the art item relative to the current rotation.

The object is rotated counter-clockwise if the `angle` value is positive, clockwise if the value is negative.

Parameters

Parameter	Type	Description
`angle`	Number (double)	The angle amount to rotate the element
`changePositions`	Boolean, optional	Whether to effect art object positions and orientations
`changeFillPatterns`	Boolean, optional	Whether to transform fill patterns
`changeFillGradients`	Boolean, optional	Whether to transform fill gradients
`changeStrokePattern`	Boolean, optional	Whether to transform stroke patterns
`rotateAbout`	*Transformation*, optional	The point to use as anchor, to transform about

Returns

Nothing.

PathItem.setEntirePath()

```
app.activeDocument.pathItems[index].setEntirePath(pathPoints)
```

Description

Sets the path using an array of [x, y] coordinate pairs.

Parameters

Parameter	Type	Description
`pathPoints`	Array of [x, y] coordinate pairs	Array of point coordinates to set

Returns

Nothing.

PathItem.transform()

```
app.activeDocument.pathItems[index].transform(transformationMatrix[,
changePositions][, changeFillPatterns][, changeFillGradients][,
changeStrokePattern][, changeLineWidths][, transformAbout])
```

Description

Transforms the art item by applying a transformation matrix.

Parameters

Parameter	Type	Description
`transformationMatrix`	*Matrix*	Transformation matrix to apply
`changePositions`	Boolean, optional	Whether to change Positions
`changeFillPatterns`	Boolean, optional	Whether to change Fill Patterns
`changeFillGradients`	Boolean, optional	Whether to change Fill Gradients
`changeStrokePattern`	Boolean, optional	Whether to change Stroke Pattern
`changeLineWidths`	Number (double), optional	The amount to scale line widths
`transformAbout`	*Transformation*, optional	The point to use as anchor, to transform about

Returns

Nothing.

PathItem.translate()

```
app.activeDocument.pathItems[index].translate([deltaX][, deltaY][,
transformObjects][, transformFillPatterns][, transformFillGradients][,
transformStrokePatterns])
```

Description

Repositions the art item relative to the current position, where `deltaX` is the horizontal offset and `deltaY` is the vertical offset.

Parameters

Parameter	Type	Description
`deltaX`	Number (double), optional	Horizontal offset
`deltaY`	Number (double), optional	Vertical offset
`transformObjects`	Boolean, optional	Whether to transform Objects
`transformFillPatterns`	Boolean, optional	Whether to transform Fill Patterns
`transformFillGradients`	Boolean, optional	Whether to transform Fill Gradients
`transformStrokePatterns`	Boolean, optional	Whether to transform Stroke Patterns

Returns

Nothing.

PathItem.zOrder()

`app.activeDocument.pathItems[index].zOrder(zOrderCmd)`

Description

Arranges the art item's position in the stacking order of the group or layer (parent) of this object.

Parameters

Parameter	Type	Description
`zOrderCmd`	*ZOrderMethod*	Stacking order arrangement method

Returns

Nothing.

104.1.3 Example

Setting colors in a path

```
// Sets the stroke and fill of a path item to colors of a randomly selected swatch
if (app.documents.length > 0 && app.activeDocument.pathItems.length > 0) {
  var doc = app.activeDocument;

  for (var i = 0; i < doc.pathItems.length; i++) {
    var pathRef = doc.pathItems[i];
    pathRef.filled = true;
    pathRef.stroked = true;

    var swatchIndex = Math.round(Math.random() * (doc.swatches.length - 1));
    pathRef.fillColor = doc.swatches[swatchIndex].color;
    pathRef.strokeColor = doc.swatches[swatchIndex].color;
  }
}
```

Creating a path from straight lines

```
// This script illustrates the use of the setEntirePath method.
// Creates a new open path consisting of 10 straight lines
if (app.documents.length > 0) {
  var lineList = [];

  for (i = 0; i < lineList.length; i++) {
    lineList.push([i * 10 + 50, ((i - 5) ^ 2) * 5 + 50];
  }
```

(continues on next page)

(continued from previous page)

```
  app.defaultStroked = true;
  newPath = app.activeDocument.pathItems.add();
  newPath.setEntirePath(lineList);
}
```

104.2 PlacedItem

`app.activeDocument.placedItems[index`

Description

An artwork item placed in a document as a linked file.

For example, an artwork object created using the `File > Place` command in Illustrator or using the `add()` method of the `placedItems` collection object is a placed item.

For information, see *PlacedItems*.

104.2.1 Properties

PlacedItem.artworkKnockout

`app.activeDocument.placedItems[index].artworkKnockout`

Description

Is this object used to create a knockout, and if so, what kind of knockout.

Type

KnockoutState

PlacedItem.blendingMode

`app.activeDocument.placedItems[index].blendingMode`

Description

The blend mode used when compositing an object.

Type

BlendModes

PlacedItem.boundingBox

`app.activeDocument.placedItems[index].boundingBox`

Description

The dimensions of the placed art item regardless of transformations.

Type

Array of 4 numbers

PlacedItem.contentVariable

`app.activeDocument.placedItems[index].contentVariable`

Description

The content variable bound to the item.

Type

Variable

PlacedItem.controlBounds

`app.activeDocument.placedItems[index].controlBounds`

Description

The bounds of the object including stroke width and controls.

Type

Array of 4 numbers; read-only.

PlacedItem.editable

`app.activeDocument.placedItems[index].editable`

Description

If `true`, this item is editable.

Type

Boolean; read-only.

PlacedItem.file

`app.activeDocument.placedItems[index].file`

Description

The file containing the artwork.

Type

File; read-only.

PlacedItem.geometricBounds

`app.activeDocument.placedItems[index].geometricBounds`

Description

The bounds of the object excluding stroke width.

Type

Array of 4 numbers; read-only.

PlacedItem.height

`app.activeDocument.placedItems[index].height`

Description

The height of the group item.

Type

Number (double)

PlacedItem.hidden

`app.activeDocument.placedItems[index].hidden`

Description

If `true`, this item is hidden.

Type

Boolean

PlacedItem.isIsolated

```
app.activeDocument.placedItems[index].isIsolated
```

Description

If `true`, this object is isolated.

Type

Boolean

PlacedItem.layer

```
app.activeDocument.placedItems[index].layer
```

Description

The layer to which this item belongs.

Type

Layer; read-only.

PlacedItem.left

```
app.activeDocument.placedItems[index].left
```

Description

The position of the left side of the item (in points, measured from the left side of the page).

Type

Number (double)

PlacedItem.locked

```
app.activeDocument.placedItems[index].locked
```

Description

If `true`, this item is locked.

Type

Boolean

PlacedItem.matrix

```
app.activeDocument.placedItems[index].matrix
```

Description

The transformation matrix of the placed artwork.

Type

Matrix

PlacedItem.name

```
app.activeDocument.placedItems[index].name
```

Description

The name of this item.

Type

String

PlacedItem.note

```
app.activeDocument.placedItems[index].note
```

Description

The note assigned to this item.

Type

String

PlacedItem.opacity

```
app.activeDocument.placedItems[index].opacity
```

Description

The opacity of the object. Range: 0.0 to 100.0

Type

Number (double)

PlacedItem.parent

`app.activeDocument.placedItems[index].parent`

Description

The parent of this object.

Type

Layer or *GroupItem*

PlacedItem.position

`app.activeDocument.placedItems[index].position`

Description

The position (in points) of the top left corner of the `pluginItem` object in the format [x, y]. Does not include stroke weight.

Type

Array of 2 numbers; read-only.

PlacedItem.selected

`app.activeDocument.placedItems[index].selected`

Description

If `true`, this item is selected.

Type

Boolean

PlacedItem.sliced

`app.activeDocument.placedItems[index].sliced`

Description

If `true`, the item sliced.

Default: `false`

Type

Boolean

PlacedItem.tags

```
app.activeDocument.placedItems[index].tags
```

Description

The tags contained in this item.

Type

Tags; read-only.

PlacedItem.top

```
app.activeDocument.placedItems[index].top
```

Description

The position of the top of the item (in points, measured from the bottom of the page).

Type

Number (double)

PlacedItem.typename

```
app.activeDocument.placedItems[index].typename
```

Description

The class name of the referenced object.

Type

String; read-only.

PlacedItem.uRL

```
app.activeDocument.placedItems[index].uRL
```

Description

The value of the Adobe URL tag assigned to this item.

Type

String

PlacedItem.visibilityVariable

```
app.activeDocument.placedItems[index].visibilityVariable
```

Description

The visibility variable bound to the item.

Type

Variable

PlacedItem.visibleBounds

```
app.activeDocument.placedItems[index].visibleBounds
```

Description

The visible bounds of the item including stroke width.

Type

Array of 4 numbers; read-only.

PlacedItem.width

```
app.activeDocument.placedItems[index].width
```

Description

The width of the item.

Type

Number (double)

PlacedItem.wrapInside

```
app.activeDocument.placedItems[index].wrapInside
```

Description

If `true`, the text frame object should be wrapped inside this object.

Type

Boolean

PlacedItem.wrapOffset

```
app.activeDocument.placedItems[index].wrapOffset
```

Description

The offset to use when wrapping text around this object.

Type

Number (double)

PlacedItem.wrapped

```
app.activeDocument.placedItems[index].wrapped
```

Description

If `true`, wrap text frame objects around this object (text frame must be above the object).

Type

Boolean

PlacedItem.zOrderPosition

```
app.activeDocument.placedItems[index].zOrderPosition
```

Description

The position of this item within the stacking order of the group or layer (`parent`) that contains the item.

Type

Number; read-only.

104.2.2 Methods

PlacedItem.duplicate()

```
app.activeDocument.placedItems[index].duplicate([relativeObject][,
insertionLocation])
```

Description

Creates a duplicate of the selected object.

Parameters

Parameter	Type	Description
`relativeObject`	Object, optional	Object to duplicate to
`insertionLocation`	*ElementPlacement*, optional	Location to insert element

Returns

PlacedItem

PlacedItem.embed()

```
app.activeDocument.placedItems[index].embed()
```

Description

Embeds this art in the document. Converts the art to art item objects as needed and deletes this object.

Returns

Nothing.

PlacedItem.move()

```
app.activeDocument.placedItems[index].move(relativeObject, insertionLocation)
```

Description

Moves the object.

Parameters

Parameter	Type	Description
`relativeObject`	Object	Object to move element within
`insertionLocation`	*ElementPlacement*, optional	Location to move element to

Returns

PlacedItem

PlacedItem.relink()

```
app.activeDocument.placedItems[index].relink(linkFile)
```

Description

Relinks the art object with the file that defines its content.

Parameters

Parameter	Type	Description
`linkFile`	File object	File to relink

Returns

Nothing.

PlacedItem.remove()

```
app.activeDocument.placedItems[index].remove()
```

Description

Deletes this object.

Returns

Nothing.

PlacedItem.resize()

```
app.activeDocument.placedItems[index].resize(scaleX, scaleY[,
changePositions][,changeFillPatterns][,changeFillGradients][,
changeStrokePattern][,changeLineWidths][,scaleAbout])
```

Description

Scales the art item where `scaleX` is the horizontal scaling factor and `scaleY` is the vertical scaling factor. 100.0 = 100%.

Parameters

Parameter	Type	Description
`scaleX`	Number (double)	Horizontal scaling factor
`scaleY`	Number (double)	Vertical scaling factor
`changePositions`	Boolean, optional	Whether to effect art object positions and orientations
`changeFillPatterns`	Boolean, optional	Whether to transform fill patterns
`changeFillGradients`	Boolean, optional	Whether to transform fill gradients
`changeStrokePattern`	Boolean, optional	Whether to transform stroke patterns
`changeLineWidths`	Number (double), optional	The amount to scale line widths
`scaleAbout`	*Transformation*, optional	The point to use as anchor, to transform about

Returns

Nothing.

PlacedItem.rotate()

```
app.activeDocument.placedItems[index].rotate(angle[,changePositions][,
changeFillPatterns][,changeFillGradients][,changeStrokePattern][,rotateAbout])
```

Description

Rotates the art item relative to the current rotation.

The object is rotated counter-clockwise if the `angle` value is positive, clockwise if the value is negative.

Parameters

Parameter	Type	Description
`angle`	Number (double)	The angle amount to rotate the element
`changePositions`	Boolean, optional	Whether to effect art object positions and orientations
`changeFillPatterns`	Boolean, optional	Whether to transform fill patterns
`changeFillGradients`	Boolean, optional	Whether to transform fill gradients
`changeStrokePattern`	Boolean, optional	Whether to transform stroke patterns
`rotateAbout`	*Transformation*, optional	The point to use as anchor, to transform about

Returns

Nothing.

PlacedItem.trace

```
app.activeDocument.placedItems[index].trace()
```

Description

Converts the raster art for this object to vector art, using default options.

Reorders the raster art into the source art of a plug-in group, and converts it into a group of filled and/or stroked paths that resemble the original image.

Creates and returns a *PluginItem* object that references a *TracingObject* object.

Returns

PluginItem

PlacedItem.transform()

```
app.activeDocument.placedItems[index].transform(transformationMatrix[,
changePositions][, changeFillPatterns][, changeFillGradients][,
changeStrokePattern][, changeLineWidths][, transformAbout])
```

Description

Transforms the art item by applying a transformation matrix.

Parameters

Parameter	Type	Description
`transformationMatrix`	*Matrix*	Transformation matrix to apply
`changePositions`	Boolean, optional	Whether to change Positions
`changeFillPatterns`	Boolean, optional	Whether to change Fill Patterns
`changeFillGradients`	Boolean, optional	Whether to change Fill Gradients
`changeStrokePattern`	Boolean, optional	Whether to change Stroke Pattern
`changeLineWidths`	Number (double), optional	The amount to scale line widths
`transformAbout`	*Transformation*, optional	The point to use as anchor, to transform about

Returns

Nothing.

PlacedItem.translate()

```
app.activeDocument.placedItems[index].translate([deltaX][, deltaY][,
transformObjects][, transformFillPatterns][, transformFillGradients][,
transformStrokePatterns])
```

Description

Repositions the art item relative to the current position, where `deltaX` is the horizontal offset and `deltaY` is the vertical offset.

Parameters

Parameter	Type	Description
`deltaX`	Number (double), optional	Horizontal offset
`deltaY`	Number (double), optional	Vertical offset
`transformObjects`	Boolean, optional	Whether to transform Objects
`transformFillPatterns`	Boolean, optional	Whether to transform Fill Patterns
`transformFillGradients`	Boolean, optional	Whether to transform Fill Gradients
`transformStrokePatterns`	Boolean, optional	Whether to transform Stroke Patterns

Returns

Nothing.

PlacedItem.zOrder()

```
app.activeDocument.placedItems[index].zOrder(zOrderCmd)
```

Description

Arranges the art item's position in the stacking order of the group or layer (parent) of this object.

Parameters

Parameter	Type	Description
`zOrderCmd`	*ZOrderMethod*	Stacking order arrangement method

Returns

Nothing.

104.2.3 Example

Changing the selection state of placed items

```
// Toggles the selection state of all placed items.
if (app.documents.length > 0) {
  for (i = 0; i < app.activeDocument.placedItems.length; i++) {
    var placedArt = app.activeDocument.placedItems[i];
    placedArt.selected = !(placedArt.selected);
  }
}
```

104.3 PluginItem

`app.activeDocument.pluginItems[index`

Description

An art item created by an Illustrator plug-in.

Scripts can create a plug-in item using *PlacedItem.trace* or *RasterItem.trace()*, and can copy existing plug-in items using the `duplicate` method, but cannot create `PluginItem` objects directly.

104.3.1 Properties

PluginItem.artworkKnockout

`app.activeDocument.pluginItems[index].artworkKnockout`

Description

Is this object used to create a knockout, and if so, what kind of knockout.

Type

KnockoutState

PluginItem.blendingMode

`app.activeDocument.pluginItems[index].blendingMode`

Description

The blend mode used when compositing an object.

Type

BlendModes

PluginItem.controlBounds

```
app.activeDocument.pluginItems[index].controlBounds
```

Description

The bounds of the object including stroke width and controls.

Type

Array of 4 numbers; read-only.

PluginItem.editable

```
app.activeDocument.pluginItems[index].editable
```

Description

If `true`, this item is editable.

Type

Boolean; read-only.

PluginItem.geometricBounds

```
app.activeDocument.pluginItems[index].geometricBounds
```

Description

The bounds of the object excluding stroke width.

Type

Array of 4 numbers; read-only.

PluginItem.height

```
app.activeDocument.pluginItems[index].height
```

Description

The height of the group item.

Type

Number (double)

PluginItem.hidden

`app.activeDocument.pluginItems[index].hidden`

Description

If `true`, this item is hidden.

Type

Boolean

PluginItem.isIsolated

`app.activeDocument.pluginItems[index].isIsolated`

Description

If `true`, this object is isolated.

Type

Boolean

PluginItem.isTracing

`app.activeDocument.pluginItems[index].isTracing`

Description

If `true`, this plug-in group represents a vector art item created by tracing a raster art item.

The `tracing` property contains the tracing object associated with the options used to create it.

Type

Boolean

PluginItem.layer

`app.activeDocument.pluginItems[index].layer`

Description

The layer to which this item belongs.

Type

Layer; read-only.

PluginItem.left

`app.activeDocument.pluginItems[index].left`

Description

The position of the left side of the item (in points, measured from the left side of the page).

Type

Number (double)

PluginItem.locked

`app.activeDocument.pluginItems[index].locked`

Description

If `true`, this item is locked.

Type

Boolean

PluginItem.name

`app.activeDocument.pluginItems[index].name`

Description

The name of this item.

Type

String

PluginItem.note

`app.activeDocument.pluginItems[index].note`

Description

The note assigned to this item.

Type

String

PluginItem.opacity

`app.activeDocument.pluginItems[index].opacity`

Description

The opacity of the object. Range: 0.0 to 100.0

Type

Number (double)

PluginItem.parent

`app.activeDocument.pluginItems[index].parent`

Description

The parent of this object.

Type

Layer or *GroupItem*

PluginItem.position

`app.activeDocument.pluginItems[index].position`

Description

The position (in points) of the top left corner of the `pluginItem` object in the format [x, y]. Does not include stroke weight.

Type

Array of 2 numbers; read-only.

PluginItem.selected

`app.activeDocument.pluginItems[index].selected`

Description

If `true`, this item is selected.

Type

Boolean

PluginItem.sliced

`app.activeDocument.pluginItems[index].sliced`

Description

If `true`, the item sliced.

Default: `false`

Type

Boolean

PluginItem.tags

`app.activeDocument.pluginItems[index].tags`

Description

The tags contained in this item.

Type

Tags; read-only.

PluginItem.top

`app.activeDocument.pluginItems[index].top`

Description

The position of the top of the item (in points, measured from the bottom of the page).

Type

Number (double)

PluginItem.tracing

`app.activeDocument.pluginItems[index].tracing`

Description

When this plug-in group was created by tracing (`isTracing` is `true`), the tracing object associated with the options used to create it.

Type

TracingObject

PluginItem.typename

```
app.activeDocument.pluginItems[index].typename
```

Description

The class name of the referenced object.

Type

String; read-only.

PluginItem.uRL

```
app.activeDocument.pluginItems[index].uRL
```

Description

The value of the Adobe URL tag assigned to this item.

Type

String

PluginItem.visibilityVariable

```
app.activeDocument.pluginItems[index].visibilityVariable
```

Description

The visibility variable bound to the item.

Type

Variable

PluginItem.visibleBounds

```
app.activeDocument.pluginItems[index].visibleBounds
```

Description

The visible bounds of the item including stroke width.

Type

Array of 4 numbers; read-only.

PluginItem.width

`app.activeDocument.pluginItems[index].width`

Description

The width of the item.

Type

Number (double)

PluginItem.wrapInside

`app.activeDocument.pluginItems[index].wrapInside`

Description

If `true`, the text frame object should be wrapped inside this object.

Type

Boolean

PluginItem.wrapOffset

`app.activeDocument.pluginItems[index].wrapOffset`

Description

The offset to use when wrapping text around this object.

Type

Number (double)

PluginItem.wrapped

`app.activeDocument.pluginItems[index].wrapped`

Description

If `true`, wrap text frame objects around this object (text frame must be above the object).

Type

Boolean

PluginItem.zOrderPosition

```
app.activeDocument.pluginItems[index].zOrderPosition
```

Description

The position of this item within the stacking order of the group or layer (`parent`) that contains the item.

Type

Number; read-only.

104.3.2 Methods

PluginItem.duplicate()

```
app.activeDocument.pluginItems[index].duplicate([relativeObject][,
insertionLocation])
```

Description

Creates a duplicate of the selected object.

Parameters

Parameter	Type	Description
`relativeObject`	Object, optional	Object to duplicate to
`insertionLocation`	*ElementPlacement*, optional	Location to insert element

Returns

PluginItem

PluginItem.move()

```
app.activeDocument.pluginItems[index].move(relativeObject, insertionLocation)
```

Description

Moves the object.

Parameters

Parameter	Type	Description
`relativeObject`	Object	Object to move element within
`insertionLocation`	*ElementPlacement*, optional	Location to move element to

Returns

PluginItem

PluginItem.remove()

```
app.activeDocument.pluginItems[index].remove()
```

Description

Deletes this object.

Returns

Nothing.

PluginItem.resize()

```
app.activeDocument.pluginItems[index].resize(scaleX, scaleY[,
changePositions][,changeFillPatterns][,changeFillGradients][,
changeStrokePattern][,changeLineWidths][,scaleAbout])
```

Description

Scales the art item where `scaleX` is the horizontal scaling factor and `scaleY` is the vertical scaling factor. 100.0 = 100%.

Parameters

Parameter	Type	Description
`scaleX`	Number (double)	Horizontal scaling factor
`scaleY`	Number (double)	Vertical scaling factor
`changePositions`	Boolean, optional	Whether to effect art object positions and orientations
`changeFillPatterns`	Boolean, optional	Whether to transform fill patterns
`changeFillGradients`	Boolean, optional	Whether to transform fill gradients
`changeStrokePattern`	Boolean, optional	Whether to transform stroke patterns
`changeLineWidths`	Number (double), optional	The amount to scale line widths
`scaleAbout`	*Transformation*, optional	The point to use as anchor, to transform about

Returns

Nothing.

PluginItem.rotate()

```
app.activeDocument.pluginItems[index].rotate(angle[,changePositions][,
changeFillPatterns][,changeFillGradients][,changeStrokePattern][,rotateAbout])
```

Description

Rotates the art item relative to the current rotation.

The object is rotated counter-clockwise if the `angle` value is positive, clockwise if the value is negative.

Parameters

Parameter	Type	Description
`angle`	Number (double)	The angle amount to rotate the element
`changePositions`	Boolean, optional	Whether to effect art object positions and orientations
`changeFillPatterns`	Boolean, optional	Whether to transform fill patterns
`changeFillGradients`	Boolean, optional	Whether to transform fill gradients
`changeStrokePattern`	Boolean, optional	Whether to transform stroke patterns
`rotateAbout`	*Transformation*, optional	The point to use as anchor, to transform about

Returns

Nothing.

PluginItem.transform()

```
app.activeDocument.pluginItems[index].transform(transformationMatrix[,
changePositions][, changeFillPatterns][, changeFillGradients][,
changeStrokePattern][, changeLineWidths][, transformAbout])
```

Description

Transforms the art item by applying a transformation matrix.

Parameters

Parameter	Type	Description
`transformationMatrix`	*Matrix*	Transformation matrix to apply
`changePositions`	Boolean, optional	Whether to change Positions
`changeFillPatterns`	Boolean, optional	Whether to change Fill Patterns
`changeFillGradients`	Boolean, optional	Whether to change Fill Gradients
`changeStrokePattern`	Boolean, optional	Whether to change Stroke Pattern
`changeLineWidths`	Number (double), optional	The amount to scale line widths
`transformAbout`	*Transformation*, optional	The point to use as anchor, to transform about

Returns

Nothing.

PluginItem.translate()

```
app.activeDocument.pluginItems[index].translate([deltaX][, deltaY][,
transformObjects][, transformFillPatterns][, transformFillGradients][,
transformStrokePatterns])
```

Description

Repositions the art item relative to the current position, where `deltaX` is the horizontal offset and `deltaY` is the vertical offset.

Parameters

Parameter	Type	Description
`deltaX`	Number (double), optional	Horizontal offset
`deltaY`	Number (double), optional	Vertical offset
`transformObjects`	Boolean, optional	Whether to transform Objects
`transformFillPatterns`	Boolean, optional	Whether to transform Fill Patterns
`transformFillGradients`	Boolean, optional	Whether to transform Fill Gradients
`transformStrokePatterns`	Boolean, optional	Whether to transform Stroke Patterns

Returns

Nothing.

PluginItem.zOrder()

`app.activeDocument.pluginItems[index].zOrder(zOrderCmd)`

Description

Arranges the art item's position in the stacking order of the group or layer (parent) of this object.

Parameters

Parameter	Type	Description
`zOrderCmd`	*ZOrderMethod*	Stacking order arrangement method

Returns

Nothing.

104.3.3 Example

Copying a plug-in item

```
// Creates new plug-in art by copying an existing plug-in art item
if (app.documents.length > 0 && app.activeDocument.pluginItems.length > 0) {
  var doc = app.activeDocument;
  var pluginArt = doc.pluginItems[0];
  pluginArt.duplicate(pluginArt.parent, ElementPlacement.PLACEATBEGINNING);
}
```

104.4 RasterItem

`app.activeDocument.rasterItems[index`

Description

A bitmap art item in a document. A script can create a raster item from an external file, or by copying an existing raster item with the `duplicate` method.

104.4.1 Properties

RasterItem.artworkKnockout

```
app.activeDocument.rasterItems[index].artworkKnockout
```

Description

Is this object used to create a knockout, and if so, what kind of knockout.

Type

KnockoutState

RasterItem.bitsPerChannel

```
app.activeDocument.rasterItems[index].bitsPerChannel
```

Description

The number of bits per channel.

Type

Number (long); read-only.

RasterItem.blendingMode

```
app.activeDocument.rasterItems[index].blendingMode
```

Description

The blend mode used when compositing an object.

Type

BlendModes

RasterItem.boundingBox

```
app.activeDocument.rasterItems[index].boundingBox
```

Description

The dimensions of the placed art item regardless of transformations.

Type

Array of 4 numbers

RasterItem.channels

`app.activeDocument.rasterItems[index].channels`

Description

The number of channels.

Type

Number (long); read-only.

RasterItem.colorants

`app.activeDocument.rasterItems[index].colorants`

Description

The colorants used in the raster art.

Type

Array of string; read-only.

RasterItem.colorizedGrayscale

`app.activeDocument.rasterItems[index].colorizedGrayscale`

Description

If `true`, the raster art is a colorized grayscale image.

Type

Boolean; read-only.

RasterItem.contentVariable

`app.activeDocument.rasterItems[index].contentVariable`

Description

The content variable bound to the item.

Type

Variable

RasterItem.controlBounds

```
app.activeDocument.rasterItems[index].controlBounds
```

Description

The bounds of the object including stroke width and controls.

Type

Array of 4 numbers; read-only.

RasterItem.editable

```
app.activeDocument.rasterItems[index].editable
```

Description

If `true`, this item is editable.

Type

Boolean; read-only.

RasterItem.embedded

```
app.activeDocument.rasterItems[index].embedded
```

Description

If `true`, the raster art item is embedded in the illustration.

Type

Boolean

RasterItem.file

```
app.activeDocument.rasterItems[index].file
```

Description

The file containing the artwork.

Type

File; read-only.

RasterItem.geometricBounds

```
app.activeDocument.rasterItems[index].geometricBounds
```

Description

The bounds of the object excluding stroke width.

Type

Array of 4 numbers; read-only.

RasterItem.height

```
app.activeDocument.rasterItems[index].height
```

Description

The height of the group item.

Type

Number (double)

RasterItem.hidden

```
app.activeDocument.rasterItems[index].hidden
```

Description

If `true`, this item is hidden.

Type

Boolean

RasterItem.imageColorSpace

```
app.activeDocument.rasterItems[index].imageColorSpace
```

Description

The color space of the raster image.

Type

ImageColorSpace; read-only.

RasterItem.isIsolated

```
app.activeDocument.rasterItems[index].isIsolated
```

Description

If `true`, this object is isolated.

Type

Boolean

RasterItem.layer

```
app.activeDocument.rasterItems[index].layer
```

Description

The layer to which this item belongs.

Type

Layer; read-only.

RasterItem.left

```
app.activeDocument.rasterItems[index].left
```

Description

The position of the left side of the item (in points, measured from the left side of the page).

Type

Number (double)

RasterItem.locked

```
app.activeDocument.rasterItems[index].locked
```

Description

If `true`, this item is locked.

Type

Boolean

RasterItem.matrix

```
app.activeDocument.rasterItems[index].matrix
```

Description

The transformation matrix of the placed artwork.

Type

Matrix

RasterItem.name

```
app.activeDocument.rasterItems[index].name
```

Description

The name of this item.

Type

String

RasterItem.note

```
app.activeDocument.rasterItems[index].note
```

Description

The note assigned to this item.

Type

String

RasterItem.opacity

```
app.activeDocument.rasterItems[index].opacity
```

Description

The opacity of the object. Range: 0.0 to 100.0

Type

Number (double)

RasterItem.overprint

`app.activeDocument.rasterItems[index].overprint`

Description

If `true`, the raster art overprints.

Type

Boolean

RasterItem.parent

`app.activeDocument.rasterItems[index].parent`

Description

The parent of this object.

Type

Layer or *GroupItem*

RasterItem.position

`app.activeDocument.rasterItems[index].position`

Description

The position (in points) of the top left corner of the `rasterItem` object in the format [x, y]. Does not include stroke weight.

Type

Array of 2 numbers; read-only.

RasterItem.selected

`app.activeDocument.rasterItems[index].selected`

Description

If `true`, this item is selected.

Type

Boolean

RasterItem.sliced

`app.activeDocument.rasterItems[index].sliced`

Description

If `true`, the item sliced.

Default: `false`

Type

Boolean

RasterItem.status

`app.activeDocument.rasterItems[index].status`

Description

Status of the linked image.

Type

RasterLinkState

RasterItem.tags

`app.activeDocument.rasterItems[index].tags`

Description

The tags contained in this item.

Type

Tags; read-only.

RasterItem.top

`app.activeDocument.rasterItems[index].top`

Description

The position of the top of the item (in points, measured from the bottom of the page).

Type

Number (double)

RasterItem.transparent

```
app.activeDocument.rasterItems[index].transparent
```

Description

If `true`, the raster art is transparent.

Type

Boolean; read-only.

RasterItem.typename

```
app.activeDocument.rasterItems[index].typename
```

Description

The class name of the referenced object.

Type

String; read-only.

RasterItem.uRL

```
app.activeDocument.rasterItems[index].uRL
```

Description

The value of the Adobe URL tag assigned to this item.

Type

String

RasterItem.visibilityVariable

```
app.activeDocument.rasterItems[index].visibilityVariable
```

Description

The visibility variable bound to the item.

Type

Variable

RasterItem.visibleBounds

`app.activeDocument.rasterItems[index].visibleBounds`

Description

The visible bounds of the item including stroke width.

Type

Array of 4 numbers; read-only.

RasterItem.width

`app.activeDocument.rasterItems[index].width`

Description

The width of the item.

Type

Number (double)

RasterItem.wrapInside

`app.activeDocument.rasterItems[index].wrapInside`

Description

If `true`, the text frame object should be wrapped inside this object.

Type

Boolean

RasterItem.wrapOffset

`app.activeDocument.rasterItems[index].wrapOffset`

Description

The offset to use when wrapping text around this object.

Type

Number (double)

RasterItem.wrapped

```
app.activeDocument.rasterItems[index].wrapped
```

Description

If `true`, wrap text frame objects around this object (text frame must be above the object).

Type

Boolean

RasterItem.zOrderPosition

```
app.activeDocument.rasterItems[index].zOrderPosition
```

Description

The position of this item within the stacking order of the group or layer (`parent`) that contains the item.

Type

Number; read-only.

104.4.2 Methods

RasterItem.colorize()

```
app.activeDocument.rasterItems[index].colorize(rasterizeColor)
```

Description

Colorizes the raster item with a CMYK or RGB Color.

Parameters

Parameter	Type	Description
`rasterizeColor`	*Color*	CMYK or RGB Color to rasterize with

Returns

Nothing.

RasterItem.duplicate()

```
app.activeDocument.rasterItems[index].duplicate([relativeObject][,
insertionLocation])
```

Description

Creates a duplicate of the selected object.

Parameters

Parameter	Type	Description
`relativeObject`	Object, optional	Object to duplicate to
`insertionLocation`	*ElementPlacement*, optional	Location to insert element

Returns

RasterItem

RasterItem.move()

```
app.activeDocument.rasterItems[index].move(relativeObject, insertionLocation)
```

Description

Moves the object.

Parameters

Parameter	Type	Description
`relativeObject`	Object	Object to move element within
`insertionLocation`	*ElementPlacement*, optional	Location to move element to

Returns

RasterItem

RasterItem.remove()

```
app.activeDocument.rasterItems[index].remove()
```

Description

Deletes this object.

Returns

Nothing.

RasterItem.resize()

```
app.activeDocument.rasterItems[index].resize(scaleX, scaleY[,
changePositions][,changeFillPatterns][,changeFillGradients][,
changeStrokePattern][,changeLineWidths][,scaleAbout])
```

Description

Scales the art item where `scaleX` is the horizontal scaling factor and `scaleY` is the vertical scaling factor. 100.0 = 100%.

Parameters

Parameter	Type	Description
`scaleX`	Number (double)	Horizontal scaling factor
`scaleY`	Number (double)	Vertical scaling factor
`changePositions`	Boolean, optional	Whether to effect art object positions and orientations
`changeFillPatterns`	Boolean, optional	Whether to transform fill patterns
`changeFillGradients`	Boolean, optional	Whether to transform fill gradients
`changeStrokePattern`	Boolean, optional	Whether to transform stroke patterns
`changeLineWidths`	Number (double), optional	The amount to scale line widths
`scaleAbout`	*Transformation*, optional	The point to use as anchor, to transform about

Returns

Nothing.

RasterItem.rotate()

```
app.activeDocument.rasterItems[index].rotate(angle[,changePositions][,
changeFillPatterns][,changeFillGradients][,changeStrokePattern][,rotateAbout])
```

Description

Rotates the art item relative to the current rotation.

The object is rotated counter-clockwise if the `angle` value is positive, clockwise if the value is negative.

Parameters

Parameter	Type	Description
`angle`	Number (double)	The angle amount to rotate the element
`changePositions`	Boolean, optional	Whether to effect art object positions and orientations
`changeFillPatterns`	Boolean, optional	Whether to transform fill patterns
`changeFillGradients`	Boolean, optional	Whether to transform fill gradients
`changeStrokePattern`	Boolean, optional	Whether to transform stroke patterns
`rotateAbout`	*Transformation*, optional	The point to use as anchor, to transform about

Returns

Nothing.

RasterItem.trace()

```
app.activeDocument.rasterItems[index].trace()
```

Description

Converts the raster art for this object to vector art, using default options.

Reorders the raster art into the source art of a plug-in group, and converts it into a group of filled and/or stroked paths that resemble the original image.

Creates and returns a *PluginItem* object that references a *TracingObject* object.

Returns

PluginItem

RasterItem.transform()

```
app.activeDocument.rasterItems[index].transform(transformationMatrix[,
changePositions][, changeFillPatterns][, changeFillGradients][,
changeStrokePattern][, changeLineWidths][, transformAbout])
```

Description

Transforms the art item by applying a transformation matrix.

Parameters

Parameter	Type	Description
`transformationMatrix`	*Matrix*	Transformation matrix to apply
`changePositions`	Boolean, optional	Whether to change Positions
`changeFillPatterns`	Boolean, optional	Whether to change Fill Patterns
`changeFillGradients`	Boolean, optional	Whether to change Fill Gradients
`changeStrokePattern`	Boolean, optional	Whether to change Stroke Pattern
`changeLineWidths`	Number (double), optional	The amount to scale line widths
`transformAbout`	*Transformation*, optional	The point to use as anchor, to transform about

Returns

Nothing.

RasterItem.translate()

```
app.activeDocument.rasterItems[index].translate([deltaX][, deltaY][,
transformObjects][, transformFillPatterns][, transformFillGradients][,
transformStrokePatterns])
```

Description

Repositions the art item relative to the current position, where `deltaX` is the horizontal offset and `deltaY` is the vertical offset.

Parameters

Parameter	Type	Description
`deltaX`	Number (double), optional	Horizontal offset
`deltaY`	Number (double), optional	Vertical offset
`transformObjects`	Boolean, optional	Whether to transform Objects
`transformFillPatterns`	Boolean, optional	Whether to transform Fill Patterns
`transformFillGradients`	Boolean, optional	Whether to transform Fill Gradients
`transformStrokePatterns`	Boolean, optional	Whether to transform Stroke Patterns

Returns

Nothing.

RasterItem.zOrder()

```
app.activeDocument.rasterItems[index].zOrder(zOrderCmd)
```

Description

Arranges the art item's position in the stacking order of the group or layer (parent) of this object.

Parameters

Parameter	Type	Description
`zOrderCmd`	*ZOrderMethod*	Stacking order arrangement method

Returns

Nothing.

104.5 SymbolItem

```
app.activeDocument.symbolItems[index
```

Description

An art item made reusable by adding it to the Symbols palette.

A `SymbolItem` is linked to the *Symbol* from which it was created and changes if you modify the associated `Symbol` object.

104.5.1 Properties

SymbolItem.artworkKnockout

```
app.activeDocument.symbolItems[index].artworkKnockout
```

Description

Is this object used to create a knockout, and if so, what kind of knockout.

Type

KnockoutState

SymbolItem.blendingMode

```
app.activeDocument.symbolItems[index].blendingMode
```

Description

The blend mode used when compositing an object.

Type

BlendModes

SymbolItem.controlBounds

```
app.activeDocument.symbolItems[index].controlBounds
```

Description

The bounds of the object including stroke width and controls.

Type

Array of 4 Numbers; read-only.

SymbolItem.editable

```
app.activeDocument.symbolItems[index].editable
```

Description

If `true`, this item is editable.

Type

Boolean; read-only.

SymbolItem.geometricBounds

```
app.activeDocument.symbolItems[index].geometricBounds
```

Description

The bounds of the object excluding stroke width.

Type

Array of 4 Numbers; read-only.

SymbolItem.height

```
app.activeDocument.symbolItems[index].height
```

Description

The height of the group item.

Type

Number (double)

SymbolItem.hidden

`app.activeDocument.symbolItems[index].hidden`

Description

If `true`, this item is hidden.

Type

Boolean

SymbolItem.isIsolated

`app.activeDocument.symbolItems[index].isIsolated`

Description

If `true`, this object is isolated.

Type

Boolean

SymbolItem.layer

`app.activeDocument.symbolItems[index].layer`

Description

The layer to which this item belongs.

Type

Layer; read-only.

SymbolItem.left

`app.activeDocument.symbolItems[index].left`

Description

The position of the left side of the item (in points, measured from the left side of the page).

Type

Number (double)

SymbolItem.locked

`app.activeDocument.symbolItems[index].locked`

Description

If `true`, this item is locked.

Type

Boolean

SymbolItem.name

`app.activeDocument.symbolItems[index].name`

Description

The name of this item.

Type

String

SymbolItem.note

`app.activeDocument.symbolItems[index].note`

Description

The note assigned to this item.

Type

String

SymbolItem.opacity

`app.activeDocument.symbolItems[index].opacity`

Description

The opacity of the object. Range: 0.0 to 100.0

Type

Number (double)

SymbolItem.parent

`app.activeDocument.symbolItems[index].parent`

Description

The parent of this object.

Type

Layer or *GroupItem*; read-only.

SymbolItem.position

`app.activeDocument.symbolItems[index].position`

Description

The position (in points) of the top left corner of the `symbolItem` object in the format [x, y]. Does not include stroke weight.

Type

Array of 2 Numbers

SymbolItem.selected

`app.activeDocument.symbolItems[index].selected`

Description

If `true`, this item is selected.

Type

Boolean

SymbolItem.sliced

`app.activeDocument.symbolItems[index].sliced`

Description

If `true`, the item sliced. Default: `false`

Type

Boolean

SymbolItem.symbol

`app.activeDocument.symbolItems[index].symbol`

Description

The symbol that was used to create this `symbolItem`.

Type

Symbol

SymbolItem.tags

`app.activeDocument.symbolItems[index].tags`

Description

The tags contained in this item.

Type

Tags; read-only.

SymbolItem.top

`app.activeDocument.symbolItems[index].top`

Description

The position of the top of the item (in points, measured from the bottom of the page).

Type

Number (double)

SymbolItem.typename

`app.activeDocument.symbolItems[index].typename`

Description

The class name of the referenced object.

Type

String; read-only.

SymbolItem.uRL

`app.activeDocument.symbolItems[index].uRL`

Description

The value of the Adobe URL tag assigned to this item.

Type

String

SymbolItem.visibilityVariable

`app.activeDocument.symbolItems[index].visibilityVariable`

Description

The visibility variable bound to the item.

Type

Variable

SymbolItem.visibleBounds

`app.activeDocument.symbolItems[index].visibleBounds`

Description

The visible bounds of the item including stroke width.

Type

Array of 4 Numbers; read-only.

SymbolItem.width

`app.activeDocument.symbolItems[index].width`

Description

The width of the item.

Type

Number (double)

SymbolItem.wrapInside

`app.activeDocument.symbolItems[index].wrapInside`

Description

If `true`, the text frame object should be wrapped inside this object.

Type

Boolean

SymbolItem.wrapOffset

`app.activeDocument.symbolItems[index].wrapOffset`

Description

The offset to use when wrapping text around this object.

Type

Number (double)

SymbolItem.wrapped

`app.activeDocument.symbolItems[index].wrapped`

Description

If `true`, wrap text frame objects around this object (text frame must be above the object).

Type

Boolean

SymbolItem.zOrderPosition

`app.activeDocument.symbolItems[index].zOrderPosition`

Description

The position of this item within the stacking order of the group or layer (`parent`) that contains the item.

Type

Number; read-only.

104.5.2 Methods

SymbolItem.duplicate()

```
app.activeDocument.symbolItems[index].duplicate([relativeObject][,
insertionLocation])
```

Description

Creates a duplicate of the selected object.

Parameters

Parameter	Type	Description
`relativeObject`	Object, optional	Object to duplicate to
`insertionLocation`	*ElementPlacement*, optional	Location to insert element

Returns

SymbolItem

SymbolItem.move()

```
app.activeDocument.symbolItems[index].move(relativeObject, insertionLocation)
```

Description

Moves the object.

Parameters

Parameter	Type	Description
`relativeObject`	Object	Object to move element within
`insertionLocation`	*ElementPlacement*, optional	Location to move element to

Returns

SymbolItem

SymbolItem.remove()

```
app.activeDocument.symbolItems[index].remove()
```

Description

Deletes this object.

Returns

Nothing.

SymbolItem.resize()

```
app.activeDocument.symbolItems[index].resize(scaleX, scaleY[,
changePositions][,changeFillPatterns][,changeFillGradients][,
changeStrokePattern][,changeLineWidths][,scaleAbout])
```

Description

Scales the art item where `scaleX` is the horizontal scaling factor and `scaleY` is the vertical scaling factor. 100.0 = 100%.

Parameters

Parameter	Type	Description
`scaleX`	Number (double)	Horizontal scaling factor
`scaleY`	Number (double)	Vertical scaling factor
`changePositions`	Boolean, optional	Whether to effect art object positions and orientations
`changeFillPatterns`	Boolean, optional	Whether to transform fill patterns
`changeFillGradients`	Boolean, optional	Whether to transform fill gradients
`changeStrokePattern`	Boolean, optional	Whether to transform stroke patterns
`changeLineWidths`	Number (double), optional	The amount to scale line widths
`scaleAbout`	*Transformation*, optional	The point to use as anchor, to transform about

Returns

Nothing.

SymbolItem.rotate()

```
app.activeDocument.symbolItems[index].rotate(angle[,changePositions][,
changeFillPatterns][,changeFillGradients][,changeStrokePattern][,rotateAbout])
```

Description

Rotates the art item relative to the current rotation.

The object is rotated counter-clockwise if the `angle` value is positive, clockwise if the value is negative.

Parameters

Parameter	Type	Description
`angle`	Number (double)	The angle amount to rotate the element
`changePositions`	Boolean, optional	Whether to effect art object positions and orientations
`changeFillPatterns`	Boolean, optional	Whether to transform fill patterns
`changeFillGradients`	Boolean, optional	Whether to transform fill gradients
`changeStrokePattern`	Boolean, optional	Whether to transform stroke patterns
`rotateAbout`	*Transformation*, optional	The point to use as anchor, to transform about

Returns

Nothing.

SymbolItem.transform()

```
app.activeDocument.symbolItems[index].transform(transformationMatrix[,
changePositions][, changeFillPatterns][, changeFillGradients][,
changeStrokePattern][, changeLineWidths][, transformAbout])
```

Description

Transforms the art item by applying a transformation matrix.

Parameters

Parameter	Type	Description
`transformationMatrix`	*Matrix*	Transformation matrix to apply
`changePositions`	Boolean, optional	Whether to change Positions
`changeFillPatterns`	Boolean, optional	Whether to change Fill Patterns
`changeFillGradients`	Boolean, optional	Whether to change Fill Gradients
`changeStrokePattern`	Boolean, optional	Whether to change Stroke Pattern
`changeLineWidths`	Number (double), optional	The amount to scale line widths
`transformAbout`	*Transformation*, optional	The point to use as anchor, to transform about

Returns

Nothing.

SymbolItem.translate()

```
app.activeDocument.symbolItems[index].translate([deltaX][, deltaY][,
transformObjects][, transformFillPatterns][, transformFillGradients][,
transformStrokePatterns])
```

Description

Repositions the art item relative to the current position, where `deltaX` is the horizontal offset and `deltaY` is the vertical offset.

Parameters

Parameter	Type	Description
`deltaX`	Number (double), optional	Horizontal offset
`deltaY`	Number (double), optional	Vertical offset
`transformObjects`	Boolean, optional	Whether to transform Objects
`transformFillPatterns`	Boolean, optional	Whether to transform Fill Patterns
`transformFillGradients`	Boolean, optional	Whether to transform Fill Gradients
`transformStrokePatterns`	Boolean, optional	Whether to transform Stroke Patterns

Returns

Nothing.

SymbolItem.zOrder()

```
app.activeDocument.symbolItems[index].zOrder(zOrderCmd)
```

Description

Arranges the art item's position in the stacking order of the group or layer (parent) of this object.

Parameters

Parameter	Type	Description
`zOrderCmd`	*ZOrderMethod*	Stacking order arrangement method

Returns

Nothing.

104.6 TextFrameItem

```
app.activeDocument.textFrames[index
```

Description

The basic art item for displaying text. From the user interface, this is text created with the Text tool. There are three types of text art in Illustrator: point text, path text, and area text. The type is indicated by the text frame's *kind* property.

When you create a text frame, you also create a *Story* object. However, threading text frames combines the frames into a single story object. To thread frames, use the *nextFrame* or *previousFrame* property.

104.6.1 Properties

TextFrameItem.anchor

```
app.activeDocument.textFrames[index].anchor
```

Description

The position of the anchor point, the start of the base line for point text.

Type

Array of 2 numbers

TextFrameItem.antialias

```
app.activeDocument.textFrames[index].antialias
```

Description

The type of anti-aliasing to use in the text.

Type

TextAntialias

TextFrameItem.characters

```
app.activeDocument.textFrames[index].characters
```

Description

All the characters in this text frame.

Type

Characters, read-only.

TextFrameItem.columnCount

```
app.activeDocument.textFrames[index].columnCount
```

Description

The column count in the text frame (area text only).

Type

Number (long)

TextFrameItem.columnGutter

```
app.activeDocument.textFrames[index].columnGutter
```

Description

The column gutter in the text frame (area text only).

Type

Number (double)

TextFrameItem.contents

```
app.activeDocument.textFrames[index].contents
```

Description

The text string.

Type

String

TextFrameItem.contentVariable

`app.activeDocument.textFrames[index].contentVariable`

Description

The content variable bound to this text frame item.

Type

Variable

TextFrameItem.endTValue

`app.activeDocument.textFrames[index].endTValue`

Description

The end position of text along a path, as a value relative to the path's segments (path text only).

Type

Number (double)

TextFrameItem.flowLinksHorizontally

`app.activeDocument.textFrames[index].flowLinksHorizontally`

Description

If `true`, flow text between linked frames horizontally first (area text only).

Type

Boolean

TextFrameItem.insertionPoints

`app.activeDocument.textFrames[index].insertionPoints`

Description

All the insertion points in this text range.

Type

InsertionPoints, read-only.

TextFrameItem.kind

```
app.activeDocument.textFrames[index].kind
```

Description

The type of a text frame item (area, path or point).

Type

TextType, read-only.

TextFrameItem.lines

```
app.activeDocument.textFrames[index].lines
```

Description

All the lines in this text frame.

Type

Lines, read-only.

TextFrameItem.matrix

```
app.activeDocument.textFrames[index].matrix
```

Description

The transformation matrix for this text frame.

Type

Matrix, read-only.

TextFrameItem.nextFrame

```
app.activeDocument.textFrames[index].nextFrame
```

Description

The linked text frame following this one.

Type

TextFrameItem

TextFrameItem.opticalAlignment

```
app.activeDocument.textFrames[index].opticalAlignment
```

Description

If `true`, the optical alignment feature is active.

Type

Boolean

TextFrameItem.orientation

```
app.activeDocument.textFrames[index].orientation
```

Description

The orientation of the text.

Type

TextOrientation

TextFrameItem.paragraphs

```
app.activeDocument.textFrames[index].paragraphs
```

Description

All the paragraphs in this text frame.

Type

Paragraphs, read-only.

TextFrameItem.parent

```
app.activeDocument.textFrames[index].parent
```

Description

The parent of this object.

Type

Layer or *GroupItem*, read-only.

TextFrameItem.previousFrame

```
app.activeDocument.textFrames[index].previousFrame
```

Description

The linked text frame preceding this one.

Type

TextFrameItem

TextFrameItem.rowCount

```
app.activeDocument.textFrames[index].rowCount
```

Description

The row count in the text frame (area text only).

Type

Number (long)

TextFrameItem.rowGutter

```
app.activeDocument.textFrames[index].rowGutter
```

Description

The row gutter in the text frame (area text only).

Type

Number (double)

TextFrameItem.spacing

```
app.activeDocument.textFrames[index].spacing
```

Description

The amount of spacing.

Type

Number (double)

TextFrameItem.startTValue

```
app.activeDocument.textFrames[index].startTValue
```

Description

The start position of text along a path, as a value relative to the path's segments (path text only).

Type

Number (double)

TextFrameItem.story

```
app.activeDocument.textFrames[index].story
```

Description

The story to which the text frame belongs.

Type

Story, read-only.

TextFrameItem.textPath

```
app.activeDocument.textFrames[index].textPath
```

Description

The path item associated with the text frame. Note: Valid only when *kind* is area or path.

Type

TextPath

TextFrameItem.textRange

```
app.activeDocument.textFrames[index].textRange
```

Description

The text range of the text frame.

Type

TextRange, read-only.

TextFrameItem.textRanges

```
app.activeDocument.textFrames[index].textRanges
```

Description

All the text in this text frame.

Type

TextRanges, read-only.

TextFrameItem.textSelection

```
app.activeDocument.textFrames[index].textSelection
```

Description

The selected text range(s) in the text frame.

Type

Array of *TextRange*, read-only.

TextFrameItem.typename

```
app.activeDocument.textFrames[index].typename
```

Description

The class name of the referenced object.

Type

String, read-only.

TextFrameItem.words

```
app.activeDocument.textFrames[index].words
```

Description

All the words in this text frame.

Type

Words, read-only.

104.6.2 Methods

TextFrameItem.convertAreaObjectToPointObject()

```
app.activeDocument.textFrames[index].convertAreaObjectToPointObject()
```

Description

Converts the area-type text frame to a point-type text frame.

Returns

TextFrameItem

TextFrameItem.convertPointObjectToAreaObject()

```
app.activeDocument.textFrames[index].convertPointObjectToAreaObject()
```

Description

Converts the point-type text frame to an area-type text frame.

Returns

TextFrameItem

TextFrameItem.createOutline()

```
app.activeDocument.textFrames[index].createOutline()
```

Description

Converts the text in the text frame to outlines.

Returns

GroupItem

TextFrameItem.duplicate()

```
app.activeDocument.textFrames[index].duplicate([relativeObject] [,
insertionLocation])
```

Description

Creates a duplicate of the selected object.

Parameters

Parameter	Type	Description
`relativeObject`	Object, optional	Object to duplicate to
`insertionLocation`	*ElementPlacement*, optional	Location to insert element

Returns

TextRange

TextFrameItem.move()

```
app.activeDocument.textFrames[index].move(relativeObject, insertionLocation)
```

Description

Moves the object.

Parameters

Parameter	Type	Description
`relativeObject`	Object	Object to move element within
`insertionLocation`	*ElementPlacement*, optional	Location to move element to

Returns

TextRange

TextFrameItem.remove()

```
app.activeDocument.textFrames[index].remove()
```

Description

Deletes this object.

Returns

Nothing.

TextFrameItem.resize()

```
app.activeDocument.textFrames[index].resize(scaleX, scaleY[,changePositions][,
changeFillPatterns][,changeFillGradients][,changeStrokePattern][,
changeLineWidths][,scaleAbout])
```

Description

Scales the art item where `scaleX` is the horizontal scaling factor and `scaleY` is the vertical scaling factor. 100.0 = 100%.

Parameters

Parameter	Type	Description
`scaleX`	Number (double)	Horizontal scaling factor
`scaleY`	Number (double)	Vertical scaling factor
`changePositions`	Boolean, optional	Whether to effect art object positions and orientations
`changeFillPatterns`	Boolean, optional	Whether to transform fill patterns
`changeFillGradients`	Boolean, optional	Whether to transform fill gradients
`changeStrokePattern`	Boolean, optional	Whether to transform stroke patterns
`changeLineWidths`	Number (double), optional	The amount to scale line widths
`scaleAbout`	*Transformation*, optional	The point to use as anchor, to transform about

Returns

Nothing.

TextFrameItem.rotate()

```
app.activeDocument.textFrames[index].rotate(angle[,changePositions][,
changeFillPatterns][,changeFillGradients][,changeStrokePattern][,rotateAbout])
```

Description

Rotates the art item relative to the current rotation. The object is rotated counter-clockwise if the `angle` value is positive, clockwise if the value is negative.

Parameters

Parameter	Type	Description
`angle`	Number (double)	The angle amount to rotate the element
`changePositions`	Boolean, optional	Whether to effect art object positions and orientations
`changeFillPatterns`	Boolean, optional	Whether to transform fill patterns
`changeFillGradients`	Boolean, optional	Whether to transform fill gradients
`changeStrokePattern`	Boolean, optional	Whether to transform stroke patterns
`rotateAbout`	*Transformation*, optional	The point to use as anchor, to transform about

Returns

Nothing.

TextFrameItem.transform()

```
app.activeDocument.textFrames[index].transform(transformationMatrix[,
changePositions][, changeFillPatterns][, changeFillGradients][,
changeStrokePattern][, changeLineWidths][, transformAbout])
```

Description

Transforms the art item by applying a transformation matrix.

Parameters

Parameter	Type	Description
`transformationMatrix`	*Matrix*	Transformation matrix to apply
`changePositions`	Boolean, optional	Whether to change Positions
`changeFillPatterns`	Boolean, optional	Whether to change Fill Patterns
`changeFillGradients`	Boolean, optional	Whether to change Fill Gradients
`changeStrokePattern`	Boolean, optional	Whether to change Stroke Pattern
`changeLineWidths`	Number (double), optional	The amount to scale line widths
`transformAbout`	*Transformation*, optional	The point to use as anchor, to transform about

Returns

Nothing.

TextFrameItem.translate()

```
app.activeDocument.textFrames[index].translate([deltaX][, deltaY][,
transformObjects][, transformFillPatterns][, transformFillGradients][,
transformStrokePatterns])
```

Description

Repositions the art item relative to the current position, where `deltaX` is the horizontal offset and `deltaY` is the vertical offset.

Parameters

Parameter	Type	Description
`deltaX`	Number (double), optional	Horizontal offset
`deltaY`	Number (double), optional	Vertical offset
`transformObjects`	Boolean, optional	Whether to transform Objects
`transformFillPatterns`	Boolean, optional	Whether to transform Fill Patterns
`transformFillGradients`	Boolean, optional	Whether to transform Fill Gradients
`transformStrokePatterns`	Boolean, optional	Whether to transform Stroke Patterns

Returns

Nothing.

TextFrameItem.zOrder()

```
app.activeDocument.textFrames[index].zOrder(zOrderCmd)
```

Description

Arranges the art item's position in the stacking order of the group or layer (parent) of this object.

Parameters

Parameter	Type	Description
`zOrderCmd`	*ZOrderMethod*	Stacking order arrangement method

Returns

Nothing.

104.6.3 Example

Rotate a text art item

```
// Duplicates and rotates the selected text art item 5 times
if ( app.documents.length > 0 ) {
  selectedItems = app.activeDocument.selection;

  // make sure something is selected.
  if ( selectedItems.length > 0 ) {

    // The selection must be a text art item
    if ( selectedItems[0].typename == "TextFrame" ) {

      // Get the parent of the text art so new text art items
      // can be inserted in the same group or layer
      dupSrc = selectedItems[0];
      textContainer = dupSrc.parent;

      // Create 5 new versions of the text art each rotated a bit
      for ( i = 1; i <= 5; i++ ) {
        dupText = dupSrc.duplicate( textContainer, ElementPlacement.PLACEATEND );
        dupText.rotate(180 * i/6);
      }
    }
  }
}
```

You can reference page items through the *PageItems* property in a *Document*, *Layer*, or *GroupItem*.

When you access an individual item in one of these collections, the reference is a page item of one of a particular type. For example, if you use *PageItems* to reference a graph item, the typename value of that object is *GraphItem*.

104.7 Properties

104.7.1 PageItems.length

`app.activeDocument.pageItems.length`

Description

The number of objects in the collection.

Type

Number, read-only.

104.7.2 PageItems.parent

`app.activeDocument.pageItems.parent`

Description

The parent of this object.

Type

Object, read-only.

104.7.3 PageItems.typename

`app.activeDocument.pageItems.typename`

Description

The class name of the referenced object.

Type

String, read-only.

104.8 Methods

104.8.1 PageItems.getByName()

`app.activeDocument.pageItems.getByName(name)`

Description

Gets the first element in the collection with the specified name.

Parameters

Parameter	Type	Description
`name`	String	Name of element to get

Returns

PageItem

104.8.2 PageItems.index()

`app.activeDocument.pageItems.index(itemKey)`

Description

Gets an element from the collection.

Parameters

Parameter	Type	Description
`itemKey`	String, Number	String or number key

Returns

PageItem

104.8.3 PageItems.removeAll()

`app.activeDocument.pageItems.removeAll()`

Description

Deletes all elements in this collection.

Returns

Nothing.

104.9 Example

104.9.1 Getting references to external files in page items

```
// Gets all file-references in the current document using the pageItems object,
// then displays them in a new document

if (app.documents.length > 0) {
  var fileReferences = new Array();
  var sourceDoc = app.activeDocument;

  for (var i = 0; i < sourceDoc.pageItems.length; i++) {
    var artItem = sourceDoc.pageItems[i];
    switch (artItem.typename) {
      case "PlacedItem":
        fileReferences.push(artItem.file.fsName);
        break;
      case "RasterItem":
        if (!artItem.embedded) {
          fileReferences.push(artItem.file.fsName);
        }
        break;
```

(continues on next page)

(continued from previous page)

```
    }
  }

  // Write the file references to a new document
  var reportDoc = documents.add();
  var areaTextPath = reportDoc.pathItems.rectangle(reportDoc.height, 0, reportDoc.
↪width, reportDoc.height);
  var fileNameText = reportDoc.textFrames.areaText(areaTextPath);
  fileNameText.textRange.size = 24;
  var paragraphCount = 3;
  var sourceName = sourceDoc.name;
  var text = "File references in \'" + sourceName + "\':\r\r";
  for (i = 0; i < fileReferences.length; i++) {
    text += (fileReferences[i] + "\r");
    paragraphCount++;
  }
  fileNameText.contents = text;
}
```

CHAPTER 105

Paper

`app.class`

Description

Associates paper information with a paper name. `Paper` objects are used by *Printer* objects.

105.1 Properties

105.1.1 Paper.name

`paper.name`

Description

The paper name.

Type

String.

105.1.2 Paper.paperInfo

`paper.paperInfo`

Description

The paper information.

Type

PaperInfo

105.1.3 Paper.typename

`paper.typename`

Description

The class name of the object.

Type

String, read-only.

CHAPTER 106

PaperInfo

`printerList[printerIndex].printerInfo.paperSizes[paperSizeIndex].paperInfo`

Description

Paper information for use in printing documents.

106.1 Properties

106.1.1 PaperInfo.customPaper

`printerList[printerIndex].printerInfo.paperSizes[paperSizeIndex].paperInfo.customPaper`

Description

If `true`, it is a custom paper.

Type

Boolean.

106.1.2 PaperInfo.height

`printerList[printerIndex].printerInfo.paperSizes[paperSizeIndex].paperInfo.height`

Description

The paper's height in points.

Type

Number (double).

106.1.3 PaperInfo.imageableArea

`printerList[printerIndex].printerInfo.paperSizes[paperSizeIndex].paperInfo.imageableArea`

Description

The imageable area.

Type

Array of 4 numbers.

106.1.4 PaperInfo.typename

`printerList[printerIndex].printerInfo.paperSizes[paperSizeIndex].paperInfo.typename`

Description

The class name of the object.

Type

String, read-only.

106.1.5 PaperInfo.width

`printerList[printerIndex].printerInfo.paperSizes[paperSizeIndex].paperInfo.width`

Description

The paper's width in points.

Type

Number (double).

106.2 Example

106.2.1 Finding paper information

```
// Displays the papers and paper sizes available for the 2nd printer in a text frame

var docRef = documents.add();
var itemRef = docRef.pathItems.rectangle(600, 300, 200, 100);
var textRef = docRef.textFrames.add();
textRef.top = 600;
textRef.left = 50;

// get paper objects for 2nd printer
var printerRef = printerList[1];
textRef.contents = printerRef.name;
textRef.contents += " paper list:\r";
var paragraphCount = 2;

// get details of each paper
var iCount = printerRef.printerInfo.paperSizes.length;
for (var i = 0; i < iCount; i++) {
  var paperRef = printerRef.printerInfo.paperSizes[i];
  var paperInfoRef = paperRef.paperInfo;
  textRef.contents += paperRef.name;
  textRef.contents += "\t";
  textRef.contents += paperInfoRef.height;
  textRef.contents += " x ";
  textRef.contents += paperInfoRef.width;
  textRef.contents += "\r";
  paragraphCount++;
}
redraw();
```

CHAPTER 107

ParagraphAttributes

`app.activeDocument.textFrames[index].paragraphs[index].paragraphAttributes`

Description

Specifies the properties and attributes of a paragraph contained in a text frame.

Note: Paragraph attributes do not have default values, and are `undefined` untile xplicitly set.

107.1 Properties

107.1.1 ParagraphAttributes.autoLeadingAmount

`app.activeDocument.textFrames[index].paragraphs[index].paragraphAttributes.autoLeadingAmount`

Description

Auto leading amount expressed as a percentage.

Type

Number (double).

107.1.2 ParagraphAttributes.bunriKinshi

`app.activeDocument.textFrames[index].paragraphs[index].paragraphAttributes.bunriKinshi`

Description

If `true`, BunriKinshi is enabled.

Type

Boolean.

107.1.3 ParagraphAttributes.burasagariType

`app.activeDocument.textFrames[index].paragraphs[index].paragraphAttributes.burasagariType`

Description

The Burasagari type.

Type

BurasagariTypeEnum

107.1.4 ParagraphAttributes.desiredGlyphScaling

`app.activeDocument.textFrames[index].paragraphs[index].paragraphAttributes.desiredGlyphScaling`

Description

Desired glyph scaling, expressed as a percentage of the default character width. Range: 50.0 to 200.0. At 100.0, the width of characters is not changed.

Type

Number (double).

107.1.5 ParagraphAttributes.desiredLetterSpacing

`app.activeDocument.textFrames[index].paragraphs[index].paragraphAttributes.desiredLetterSpacing`

Description

Desired letter, spacing expressed as a percentage of the default kerning or tracking Range: -100.0 to 500.0. At 0, no space is added between letters. At 100.0, an entire space width is added between letters.

Type

Number (double).

107.1.6 ParagraphAttributes.desiredWordSpacing

`app.activeDocument.textFrames[index].paragraphs[index].paragraphAttributes.desiredWordSpacing`

Description

Desired word spacing, expressed as a percentage of the default space for the font. Range: 0.0 to 1000.0; at 100.00. No space is added between words.

Type

Number (double).

107.1.7 ParagraphAttributes.everyLineComposer

`app.activeDocument.textFrames[index].paragraphs[index].paragraphAttributes.everyLineComposer`

Description

If `true`, the Every-line Composer is enabled. If `false`, the Single-line Composer is enabled.

Type

Boolean.

107.1.8 ParagraphAttributes.firstLineIndent

`app.activeDocument.textFrames[index].paragraphs[index].paragraphAttributes.firstLineIndent`

Description

First line left indent in points.

Type

Number (double).

107.1.9 ParagraphAttributes.hyphenateCapitalizedWords

`app.activeDocument.textFrames[index].paragraphs[index].paragraphAttributes.hyphenateCapitalizedWords`

Description

If `true`, hyphenation is enabled for capitalized words.

Type

Boolean.

107.1.10 ParagraphAttributes.hyphenation

`app.activeDocument.textFrames[index].paragraphs[index].paragraphAttributes.hyphenation`

Description

If `true`, hyphenation is enabled for the paragraph.

Type

Boolean.

107.1.11 ParagraphAttributes.hyphenationPreference

`app.activeDocument.textFrames[index].paragraphs[index].paragraphAttributes.hyphenationPreference`

Description

Hyphenation preference scale for better spacing (0) or fewer hyphens (1). Range: 0.0 to 1.0.

Type

Number (double).

107.1.12 ParagraphAttributes.hyphenationZone

`app.activeDocument.textFrames[index].paragraphs[index].paragraphAttributes.hyphenationZone`

Description

The distance (in points) from the right edge of the paragraph that marks the part of the line where hyphenation is not allowed.

Note: 0 allows all hyphenation. Valid only when *ParagraphAttributes.everyLineComposer* is `false`.

Type

Number (double).

107.1.13 ParagraphAttributes.justification

`app.activeDocument.textFrames[index].paragraphs[index].paragraphAttributes.justification`

Description

Paragraph justification.

Type

Justification

107.1.14 ParagraphAttributes.kinsoku

`app.activeDocument.textFrames[index].paragraphs[index].paragraphAttributes.kinsoku`

Description

The Kinsoku Shori name.

Type

String.

107.1.15 ParagraphAttributes.kinsokuOrder

`app.activeDocument.textFrames[index].paragraphs[index].paragraphAttributes.kinsokuOrder`

Description

The preferred Kinsoku order.

Type

KinsokuOrderEnum

107.1.16 ParagraphAttributes.kurikaeshiMojiShori

`app.activeDocument.textFrames[index].paragraphs[index].paragraphAttributes.kurikaeshiMojiShori`

Description

If `true`, KurikaeshiMojiShori is enabled.

Type

Boolean.

107.1.17 ParagraphAttributes.leadingType

`app.activeDocument.textFrames[index].paragraphs[index].paragraphAttributes.leadingType`

Description

Auto leading type.

Type

AutoLeadingType

107.1.18 ParagraphAttributes.leftIndent

```
app.activeDocument.textFrames[index].paragraphs[index].paragraphAttributes.
leftIndent
```

Description

The left indent of margin in points.

Type

Number (double).

107.1.19 ParagraphAttributes.maximumConsecutiveHyphens

```
app.activeDocument.textFrames[index].paragraphs[index].paragraphAttributes.
maximumConsecutiveHyphens
```

Description

Maximum number of consecutive hyphenated lines.

Type

Number (long).

107.1.20 ParagraphAttributes.maximumGlyphScaling

```
app.activeDocument.textFrames[index].paragraphs[index].paragraphAttributes.
maximumGlyphScaling
```

Description

Maximum glyph scaling, expressed as a percentage of the default character width. Range: 50.0 to 200.0; at 100.0. The width of characters is not changed.

Note: Valid only for justified paragraphs.

Type

Number (double).

107.1.21 ParagraphAttributes.maximumLetterSpacing

```
app.activeDocument.textFrames[index].paragraphs[index].paragraphAttributes.
maximumLetterSpacing
```

Description

Maximum letter spacing, expressed as a percentage of the default kerning or tracking Range: -100.0 to 500.0; at 0. No space is added between letters. At 100.0, an entire space width is added between letters.

Note: Valid only for justified paragraphs.

Type

Number (double).

107.1.22 ParagraphAttributes.maximumWordSpacing

`app.activeDocument.textFrames[index].paragraphs[index].paragraphAttributes.maximumWordSpacing`

Description

Maximum word spacing, expressed as a percentage of the default space for the font. Range: 0.0 to 1000.0; at 100.00. No space is added between words.

Note: Valid only for justified paragraphs.

Type

Number (double).

107.1.23 ParagraphAttributes.minimumAfterHyphen

`app.activeDocument.textFrames[index].paragraphs[index].paragraphAttributes.minimumAfterHyphen`

Description

Minimum number of characters after a hyphen.

Type

Number (long).

107.1.24 ParagraphAttributes.minimumBeforeHyphen

`app.activeDocument.textFrames[index].paragraphs[index].paragraphAttributes.minimumBeforeHyphen`

Description

Minimum number of characters before a hyphen.

Type

Number (long).

107.1.25 ParagraphAttributes.minimumGlyphScaling

```
app.activeDocument.textFrames[index].paragraphs[index].paragraphAttributes.
minimumGlyphScaling
```

Description

Minimum glyph scaling, expressed as a percentage of the default character width. Range: 50.0 to 200.0. At 100.0, the width of characters is not changed.

Note: Valid only for justified paragraphs.

Type

Number (double).

107.1.26 ParagraphAttributes.minimumHyphenatedWordSize

```
app.activeDocument.textFrames[index].paragraphs[index].paragraphAttributes.
minimumHyphenatedWordSize
```

Description

Minimum number of characters for a word to be hyphenated.

Type

Number (long).

107.1.27 ParagraphAttributes.minimumLetterSpacing

```
app.activeDocument.textFrames[index].paragraphs[index].paragraphAttributes.
minimumLetterSpacing
```

Description

Minimum letter spacing, expressed as a percentage of the default kerning or tracking Range: -100.0 to 500.0; at 0. No space is added between letters. At 100.0, an entire space width is added between letters.

Note: Valid only for justified paragraphs.

Type

Number (double).

107.1.28 ParagraphAttributes.minimumWordSpacing

```
app.activeDocument.textFrames[index].paragraphs[index].paragraphAttributes.
minimumWordSpacing
```

Description

Minimum word spacing, expressed as a percentage of the default space for the font. Range: 0.0 to 1000.0; at 100.00. No space is added between words.

Note: Valid only for justified paragraphs.

Type

Number (double).

107.1.29 ParagraphAttributes.mojikumi

```
app.activeDocument.textFrames[index].paragraphs[index].paragraphAttributes.
mojikumi
```

Description

The Mojikumi name.

Type

String.

107.1.30 ParagraphAttributes.parent

```
app.activeDocument.textFrames[index].paragraphs[index].paragraphAttributes.
parent
```

Description

The object's container.

Type

Object, read-only.

107.1.31 ParagraphAttributes.rightIndent

```
app.activeDocument.textFrames[index].paragraphs[index].paragraphAttributes.
rightIndent
```

Description

Right indent of margin in points.

Type

Number (double).

107.1.32 ParagraphAttributes.romanHanging

`app.activeDocument.textFrames[index].paragraphs[index].paragraphAttributes.romanHanging`

Description

If `true`, Roman hanging punctuation is enabled.

Type

Boolean.

107.1.33 ParagraphAttributes.singleWordJustification

`app.activeDocument.textFrames[index].paragraphs[index].paragraphAttributes.singleWordJustification`

Description

Single word justification.

Type

Justification

107.1.34 ParagraphAttributes.spaceAfter

`app.activeDocument.textFrames[index].paragraphs[index].paragraphAttributes.spaceAfter`

Description

Spacing after paragraph in points.

Type

Number (double).

107.1.35 ParagraphAttributes.spaceBefore

`app.activeDocument.textFrames[index].paragraphs[index].paragraphAttributes.spaceBefore`

Description

Spacing before paragraph in points.

Type

Number (double).

107.1.36 ParagraphAttributes.tabStops

`app.activeDocument.textFrames[index].paragraphs[index].paragraphAttributes.tabStops`

Description

Tab stop settings.

Type

TabStopInfo

107.1.37 ParagraphAttributes.typename

`app.activeDocument.textFrames[index].paragraphs[index].paragraphAttributes.typename`

Description

The class name of the object.

Type

String, read-only.

107.2 Example

107.2.1 Changing justification in paragraphs

```
// Creates a new document with 1 text frame and 3 paragraphs
// then gives each paragraph a different justification

var docRef = documents.add();
var pathRef = docRef.pathItems.rectangle(600, 200, 200, 400);
var textRef = docRef.textFrames.areaText(pathRef);
textRef.paragraphs.add("Left justified paragraph.");
textRef.paragraphs.add("Center justified paragraph.");
textRef.paragraphs.add("Right justified paragraph.");
textRef.textRange.characterAttributes.size = 28;

// change the justification of each paragraph
// using the paragraph attributes object
var paraAttr_0 = textRef.paragraphs[0].paragraphAttributes;
paraAttr_0.justification = Justification.RIGHT;

var paraAttr_1 = textRef.paragraphs[1].paragraphAttributes;
paraAttr_1.justification = Justification.CENTER;

var paraAttr_2 = textRef.paragraphs[2].paragraphAttributes;
paraAttr_2.justification = Justification.LEFT;
```

CHAPTER 108

Paragraphs

`app.activeDocument.textFrames[index].paragraphs`

Description

A collection of *TextRange* objects, with each `TextRange` representing a paragraph. The elements are not named; you must access them by index.

108.1 Properties

108.1.1 Paragraphs.length

`app.activeDocument.textFrames[index].paragraphs.length`

Description

The number of objects in the collection.

Type

Number, read-only.

108.1.2 Paragraphs.parent

`app.activeDocument.textFrames[index].paragraphs.parent`

Description

The parent of this object.

Type

Object, read-only.

108.1.3 Paragraphs.typename

`app.activeDocument.textFrames[index].paragraphs.typename`

Description

The class name of the referenced object.

Type

String, read-only.

108.2 Methods

108.2.1 Paragraphs.add()

`app.activeDocument.textFrames[index].paragraphs.add(contents [,relativeObject] [,insertionLocation])`

Description

Adds a new paragraph with specified text contents at the specified location in the current document. If location is not specified, adds the new paragraph to the containing text frame after the current text selection or insertion point.

Parameters

Parameter	Type	Description
`contents`	String	Text contents to add
`relativeObject`	*TextFrameItem*, optional	Object to add item to
`insertionLocation`	*ElementPlacement*, optional	Location to place text

Returns

TextRange

108.2.2 Paragraphs.addBefore()

`app.activeDocument.textFrames[index].paragraphs.addBefore(contents)`

Description

Adds a new paragraph with specified text contents before the current text selection or insertion point.

Parameters

Parameter	Type	Description
`contents`	String	Text contents to add

Returns

TextRange

108.2.3 Paragraphs.index()

`app.activeDocument.textFrames[index].paragraphs.index(itemKey)`

Description

Gets an element from the collection.

Parameters

Parameter	Type	Description
`itemKey`	String, Number	String or number key

Returns

TextRange

108.2.4 Paragraphs.removeAll()

`app.activeDocument.textFrames[index].paragraphs.removeAll()`

Description

Deletes all elements in this collection.

Returns

Nothing.

108.3 Example

108.3.1 Counting paragraphs

```
// Counts all paragraphs in current doc and stores result in paragraphCount
if (app.documents.length > 0) {
  var doc = app.activeDocument;
  var paragraphCount = 0;
  for (var i = 0; i < doc.textFrames.length; i++) {
    paragraphCount += doc.textFrames[i].paragraphs.length;
  }
}
```

CHAPTER 109

ParagraphStyle

`app.activeDocument.paragraphStyles[index`

Description

Associates character and paragraph attributes with a style name. The style object can be used to apply those attributes to the text in a TextFrame object. See *Creating and applying a paragraph style* example.

109.1 Properties

109.1.1 ParagraphStyle.characterAttributes

`app.activeDocument.paragraphStyles[index.characterAttributes`

Description

The character properties for the text range.

Type

CharacterAttributes, read-only.

109.1.2 ParagraphStyle.name

`app.activeDocument.paragraphStyles[index.name`

Description

The paragraph style's name.

Type

String.

109.1.3 ParagraphStyle.paragraphAttributes

```
app.activeDocument.paragraphStyles[index.paragraphAttributes
```

Description

The paragraph properties for the text range.

Type

CharacterAttributes, read-only.

109.1.4 ParagraphStyle.parent

```
app.activeDocument.paragraphStyles[index.parent
```

Description

The object's container.

Type

Object, read-only.

109.1.5 ParagraphStyle.typename

```
app.activeDocument.paragraphStyles[index.typename
```

Description

The class name of the object.

Type

String, read-only.

109.2 Methods

109.2.1 ParagraphStyle.applyTo()

```
app.activeDocument.paragraphStyles[index.applyTo(textItem [,
clearingOverrides])
```

Description

Applies this paragraph style to the specified text item.

Parameters

Parameter	Type	Description
`textItem`	Object	Paragraph item to apply style to
`clearingOverrides`	Boolean, optional	Whether to clear overrides

Returns

Nothing.

109.2.2 ParagraphStyle.remove()

```
app.activeDocument.paragraphStyles[index.remove()
```

Description

Deletes the object.

Returns

Nothing.

CHAPTER 110

ParagraphStyles

`app.activeDocument.paragraphStyles`

Description

A collection of *ParagraphStyle* objects.

110.1 Properties

110.1.1 ParagraphStyles.length

`app.activeDocument.paragraphStyles.length`

Description

Number of elements in the collection.

Type

Number, read-only.

110.1.2 ParagraphStyles.parent

`app.activeDocument.paragraphStyles.parent`

Description

The object's container.

Type

Object, read-only.

110.1.3 ParagraphStyles.typename

```
app.activeDocument.paragraphStyles.typename
```

Description

The class name of the object.

Type

String, read-only.

110.2 Methods

110.2.1 ParagraphStyles.add()

```
app.activeDocument.paragraphStyles.add(name)
```

Description

Creates a named paragraph style.

Parameters

Parameter	Type	Description
`name`	String	Name of element to get

Returns

CharacterAttributes

110.2.2 ParagraphStyles.getByName()

```
app.activeDocument.paragraphStyles.getByName(name)
```

Description

Get the first element in the collection with the provided name.

Parameters

Parameter	Type	Description
`name`	String	Name of element to get

Returns

CharacterAttributes

110.2.3 ParagraphStyles.index()

`app.activeDocument.paragraphStyles.index(itemKey)`

Description

Gets an element from the collection.

Parameters

Parameter	Type	Description
`itemKey`	String, Number	String or number key

Returns

CharacterAttributes

110.2.4 ParagraphStyles.removeAll()

`app.activeDocument.paragraphStyles.removeAll()`

Description

Deletes all elements in the collection.

Returns

Nothing.

110.3 Example

110.3.1 Creating and applying a paragraph style

```
// Creates a new document with 1 text frame and 3 paragraphs
// gives each paragraph a different justification, then creates
// a paragraph style and applies it to all paragraphs

var docRef = documents.add();
var pathRef = docRef.pathItems.rectangle(600, 200, 200, 400);
var textRef = docRef.textFrames.areaText(pathRef);
textRef.paragraphs.add("Left justified paragraph.");
textRef.paragraphs.add("Center justified paragraph.");
textRef.paragraphs.add("Right justified paragraph.");
textRef.textRange.characterAttributes.size = 28;

// change the justification of each paragraph
// using the paragraph attributes object
var paraAttr_0 = textRef.paragraphs[0].paragraphAttributes;
paraAttr_0.justification = Justification.RIGHT;

var paraAttr_1 = textRef.paragraphs[1].paragraphAttributes;
paraAttr_1.justification = Justification.CENTER;
```

(continues on next page)

(continued from previous page)

```
var paraAttr_2 = textRef.paragraphs[2].paragraphAttributes;
paraAttr_2.justification = Justification.LEFT;

// create a new paragraph style
var paraStyle = docRef.paragraphStyles.add("LeftIndent");

// add some paragraph attributes
var paraAttr = paraStyle.paragraphAttributes;
paraAttr.justification = Justification.LEFT;
paraAttr.firstLineIndent = 10;

// apply the style to each item in the document
var iCount = textRef.paragraphs.length;
for (var i = 0; i < iCount; i++) {
  paraStyle.applyTo(textRef.paragraphs[i], true);
}
redraw();
```

CHAPTER 111

PathItems

`app.activeDocument.pathItems`

Description

A collection of *PathItem* objects.

The methods `ellipse`, `polygon`, `rectangle`, `roundedRectangle`, and `star` allow you to create complex path items using straightforward parameters.

If you do not provide any parameters when calling these methods, default values are used.

111.1 Properties

111.1.1 PathItems.length

`app.activeDocument.pathItems.length`

Description

Number of elements in the collection.

Type

Number, read-only.

111.1.2 PathItems.parent

`app.activeDocument.pathItems.parent`

Description

The object's container.

Type

Object, read-only.

111.1.3 PathItems.typename

`app.activeDocument.pathItems.typename`

Description

The class name of the object.

Type

String, read-only.

111.2 Methods

111.2.1 PathItems.add()

`app.activeDocument.pathItems.add()`

Description

Creates a new object.

Returns

PathItem

111.2.2 PathItems.ellipse()

`app.activeDocument.pathItems.ellipse([top][, left][, width][, height][, reversed][, inscribed])`

Description

Creates a new pathItem in the shape of an ellipse using the supplied parameters.

Defaults

Parameter	Value
`top`	100 pt.
`left`	100 pt.
`width`	50 pt.
`height`	100 pt.
`reversed`	false

Parameters

Parameter	Type	Description
`top`	Number (double), optional	Top of path
`left`	Number (double), optional	Left of path
`width`	Number (double), optional	Width of path
`height`	Number (double), optional	Height of path
`reversed`	Boolean, optional	Whether path is reversed
`inscribed`	Boolean, optional	Whether path is inscribed

Returns

PathItem

111.2.3 PathItems.getByName()

```
app.activeDocument.pathItems.getByName(name)
```

Description

Gets the first element in the collection with the specified name.

Parameters

Parameter	Type	Description
`name`	String	Name of element to get

Returns

PathItem

111.2.4 PathItems.index()

```
app.activeDocument.pathItems.index(itemKey)
```

Description

Gets an element from the collection.

Parameters

Parameter	Type	Description
`itemKey`	String, Number	String or number key

Returns

PathItem

111.2.5 PathItems.polygon()

```
app.activeDocument.pathItems.polygon([centerX][, centerY][, radius][, sides][,
reversed])
```

Description

Creates a new `pathItem` in the shape of an polygon using the supplied parameters.

Defaults

Parameter	Value
`centerX`	200 pt.
`centerY`	300 pt.
`radius`	50 pt.
`sides`	8
`reversed`	false

Parameters

Parameter	Type	Description
`centerX`	Number (double), optional	CenterX of path
`centerY`	Number (double), optional	CenterY of path
`radius`	Number (double), optional	Radius of path
`sides`	Number (long), optional	Number of sides
`reversed`	Boolean, optional	Whether path is reversed

Returns

PathItem

111.2.6 PathItems.rectangle()

```
app.activeDocument.pathItems.rectangle(top, left, width, height[,reversed])
```

Description

Creates a new `pathItem` in the shape of an polygon using the supplied parameters.

Parameters

Parameter	Type	Description
`top`	Number (double)	Top of path
`left`	Number (double)	Left of path
`width`	Number (double)	Width of path
`height`	Number (double)	Height of path
`reversed`	Boolean, optional	Whether path is reversed

Returns

PathItem

111.2.7 PathItems.removeAll()

```
app.activeDocument.pathItems.removeAll()
```

Description

Deletes all elements in this collection.

Returns

Nothing

111.2.8 PathItems.roundedRectangle()

```
app.activeDocument.pathItems.roundedRectangle(top, left, width, height[,
horizontalRadius][, verticalRadius][, reversed])
```

Description

Creates a new pathItem in the shape of a rectangle with rounded corners using the supplied parameters.

Defaults

Parameter	Value
`horizontalRadius`	15 pt.
`verticalRadius`	20 pt.
`reversed`	false

Parameters

Parameter	Type	Description
`top`	Number (double)	Top of path
`left`	Number (double)	Left of path
`width`	Number (double)	Width of path
`height`	Number (double)	Height of path
`horizontalRadius`	Number (double), optional	Horizontal radius of rounded corner
`verticalRadius`	Number (double), optional	Vertical radius of rounded corner
`reversed`	Boolean, optional	Whether path is reversed

Returns

PathItem

111.2.9 PathItems.star()

```
app.activeDocument.pathItems.star([centerX][, centerY][, radius][,
innerRadius][, points][, reversed])
```

Description

Creates a new path item in the shape of a star using the supplied parameters.

Defaults

Parameter	Value
centerX	200 pt.
centerY	300 pt.
radius	50 pt.
innerRadius	20 pt.
points	5
reversed	false

Parameters

Parameter	Type	Description
centerX	Number (double), optional	CenterX of path
centerY	Number (double), optional	CenterY of path
radius	Number (double), optional	Radius of path
innerRadius	Number (double), optional	Inner radius of path
points	Number (long), optional	Number of points
reversed	Boolean, optional	Whether path is reversed

Returns

PathItem

111.3 Example

111.3.1 Creating shapes

```
// Creates 5 shapes in layer 1 of document 1
// and applies a random graphic style to each
var doc = app.documents.add();
var artLayer = doc.layers[0];
app.defaultStroked = true;
app.defaultFilled = true;

var rect = artLayer.pathItems.rectangle(762.5, 87.5, 425.0, 75.0);
var rndRect = artLayer.pathItems.roundedRectangle(637.5, 87.5, 425.0, 75.0, 20.0, 10.
↪0);

// Create ellipse, 'reversed' is false, 'inscribed' is true
var ellipse = artLayer.pathItems.ellipse(512.5, 87.5, 425.0, 75.0, false, true);

// Create octagon, and 8-sided polygon
var octagon = artLayer.pathItems.polygon(300.0, 325.0, 75.0, 8);

// Create a 4 pointed star
var star = artLayer.pathItems.star(300.0, 125.0, 100.0, 20.0, 4);

for (i = 0; i < artLayer.pathItems.length; i++) {
  var styleIndex = Math.round(Math.random() * (doc.graphicStyles.length - 1));
  doc.graphicStyles[styleIndex].applyTo(artLayer.pathItems[i]);
}
```

CHAPTER 112

PathPoint

`app.activeDocument.pathItems[index].pathPoints[index`

Description

A point on a specific path.

Each path point is made up of an anchor point (`anchor`) and a pair of handles (`leftDirection` and `rightDirection`).

112.1 Properties

112.1.1 PathPoint.anchor

`app.activeDocument.pathItems[index].pathPoints[index].anchor`

Description

The position of this point's anchor point.

Type

Array of 2 numbers

112.1.2 PathPoint.leftDirection

`app.activeDocument.pathItems[index].pathPoints[index].leftDirection`

Description

The position of this path point's in control point.

Type

Array of 2 numbers

112.1.3 PathPoint.parent

`app.activeDocument.pathItems[index].pathPoints[index].parent`

Description

The path item that contains this path point.

Type

PathItem; read-only.

112.1.4 PathPoint.pointType

`app.activeDocument.pathItems[index].pathPoints[index].pointType`

Description

The type of path point, either a curve or a corner. Any point can considered a corner point.

Setting the type to a corner forces the left and right direction points to be on a straight line when the user attempts to modify them in the user interface.

Type

PointType

112.1.5 PathPoint.rightDirection

`app.activeDocument.pathItems[index].pathPoints[index].rightDirection`

Description

The position of this path point's out control point.

Type

Array of 2 numbers

112.1.6 PathPoint.selected

`app.activeDocument.pathItems[index].pathPoints[index].selected`

Description

Are points of this path point selected, and if so, which ones.

Type

PathPointSelection

112.1.7 PathPoint.typename

```
app.activeDocument.pathItems[index].pathPoints[index].typename
```

Description

The class name of the referenced object.

Type

String; read-only.

112.2 Methods

112.2.1 PathPoint.remove()

```
app.activeDocument.pathItems[index].pathPoints[index].remove()
```

Description

Removes the referenced point from the path.

Returns

Nothing.

CHAPTER 113

PathPoints

`app.activeDocument.pathItems[index].pathPoints`

Description

A collection of *PathPoint* objects in a specific path.

The elements are not named; you must access them by index.

113.1 Properties

113.1.1 PathPoints.length

`app.activeDocument.pathItems[index].pathPoints.length`

Description

Number of elements in the collection.

Type

Number, read-only.

113.1.2 PathPoints.parent

`app.activeDocument.pathItems[index].pathPoints.parent`

Description

The object's container.

Type

Object, read-only.

113.1.3 PathPoints.typename

`app.activeDocument.pathItems[index].pathPoints.typename`

Description

The class name of the object.

Type

String, read-only.

113.2 Methods

113.2.1 PathPoints.add()

`app.activeDocument.pathItems[index].pathPoints.add()`

Description

Creates a new object.

Returns

PathPoint

113.2.2 PathPoints.index()

`app.activeDocument.pathItems[index].pathPoints.index(itemKey)`

Description

Gets an element from the collection.

Parameters

Parameter	Type	Description
`itemKey`	String, Number	String or number key

Returns

PathPoint

113.2.3 PathPoints.removeAll()

`app.activeDocument.pathItems[index].pathPoints.removeAll()`

Description

Deletes all elements in the collection.

Returns

Nothing.

113.3 Example

113.3.1 Adding a path point to a path

```
// Appends a new PathPoint to an existing path
// and initializes its anchor and handle points.
if (app.documents.length > 0) {
  var doc = app.activeDocument;

  var line = doc.pathItems.add();
  line.stroked = true;
  line.setEntirePath(Array(Array(220, 475), Array(375, 300)));

  // Append another point to the line
  var newPoint = doc.pathItems[0].pathPoints.add();
  newPoint.anchor = Array(220, 300);
  newPoint.leftDirection = newPoint.anchor;
  newPoint.rightDirection = newPoint.anchor;
  newPoint.pointType = PointType.CORNER;
}
```

CHAPTER 114

Pattern

`app.activeDocument.patterns[index`

Description

An Illustrator pattern definition contained in a document.

Patterns are shown in the Swatches palette.

Each pattern is referenced by a *PatternColor* object, which defines the pattern's appearance.

114.1 Properties

114.1.1 Pattern.name

`app.activeDocument.patterns[index].name`

Description

The pattern name.

Type

String

114.1.2 Pattern.parent

`app.activeDocument.patterns[index].parent`

Description

The document that contains this pattern.

Type

Document, read-only.

114.1.3 Pattern.typename

```
app.activeDocument.patterns[index].typename
```

Description

The class name of the object.

Type

String, read-only.

114.2 Methods

114.2.1 Pattern.remove()

```
app.activeDocument.patterns[index].remove()
```

Description

Removes the referenced pattern from the document.

Returns

Nothing.

114.2.2 Pattern.toString()

```
app.activeDocument.patterns[index].toString()
```

Description

Returns the object type of a referenced object. If the object has a name, also returns the name.

Returns

String

CHAPTER 115

Patterns

`app.activeDocument.patterns`

Description

A collection of *Pattern* objects in a document.

115.1 Properties

115.1.1 Patterns.length

`app.activeDocument.patterns.length`

Description

Number of elements in the collection.

Type

Number, read-only.

115.1.2 Patterns.parent

`app.activeDocument.patterns.parent`

Description

The object's container.

Type

Object, read-only.

115.1.3 Patterns.typename

`app.activeDocument.patterns.typename`

Description

The class name of the object.

Type

String, read-only.

115.2 Methods

115.2.1 Patterns.add()

`app.activeDocument.patterns.add()`

Description

Creates a new object.

Returns

Pattern

115.2.2 Patterns.getByName()

`app.activeDocument.patterns.getByName(name)`

Description

Get the first element in the collection with the provided name.

Parameters

Parameter	Type	Description
`name`	String	Name of element to get

Returns

Pattern

115.2.3 Patterns.index()

`app.activeDocument.patterns.index(itemKey)`

Description

Gets an element from the collection.

Parameters

Parameter	Type	Description
`itemKey`	String, Number	String or number key

Returns

Pattern

115.2.4 Patterns.removeAll()

`app.activeDocument.patterns.removeAll()`

Description

Deletes all elements in the collection.

Returns

Nothing.

115.3 Example

115.3.1 Removing a pattern

```
// Deletes the last pattern from the current document.
if (app.documents.length > 0) {
  var lastIndex = app.activeDocument.patterns.length - 1;

  var patternToRemove = app.activeDocument.patterns[lastIndex];
  var patternName = patternToRemove.name;
  patternToRemove.remove();

  // Note after removing Illustrator objects, set the variable that
  // referenced the removed object to null, since it is now invalid.
  patternToRemove = null;
}
```

CHAPTER 116

PDFFileOptions

`app.preferences.PDFFileOptions`

Description

Options for opening a PDF file, used with the *Application.open()* method.

All properties are optional.

116.1 Properties

116.1.1 PDFFileOptions.pageToOpen

`app.preferences.PDFFileOptions.pageToOpen`

Description

What page should be used when opening a multipage document.

Default: 1

Type

Number (long)

116.1.2 PDFFileOptions.parent

`app.preferences.PDFFileOptions.parent`

Description

The object's container.

Type

Object; read-only.

116.1.3 PDFFileOptions.pDFCropToBox

`app.preferences.PDFFileOptions.pDFCropToBox`

Description

Which box should be used when placing a multipage document.

Default: `PDFBoxType.PDFMediaBox`

Type

PDFBoxType

116.1.4 PDFFileOptions.typename

`app.preferences.PDFFileOptions.typename`

Description

The class name of the object.

Type

String; read-only.

116.2 Example

116.2.1 Opening a PDF with options

```
// Opens a PDF file with specified options
var pdfOptions = app.preferences.PDFFileOptions;
pdfOptions.pDFCropToBox = PDFBoxType.PDFBOUNDINGBOX;
pdfOptions.pageToOpen = 2;

// Open a file using these preferences
var fileRef = filePath;

if (fileRef != null) {
  var docRef = open(fileRef, DocumentColorSpace.RGB);
}
```

CHAPTER 117

PDFSaveOptions

`new PDFSaveOptions()`

Description

Options for saving a document as an Adobe PDF file, used with the *Document.saveAs()* method.

All properties are optional.

117.1 Properties

117.1.1 PDFSaveOptions.acrobatLayers

`pDFSaveOptions.acrobatLayers`

Description

Optional. Create Acrobat® layers from top-level layers. Acrobat 6 only.

Default: `false`

Type

Boolean

117.1.2 PDFSaveOptions.artboardRange

`pDFSaveOptions.artboardRange`

Description

Optional. This is considered for multi-asset extraction, which specifies the artboard range. An empty string extracts all the artboards.

Default: empty string

Type

String

117.1.3 PDFSaveOptions.bleedLink

`pDFSaveOptions.bleedLink`

Description

Optional. Link 4 bleed values.

Default: `true`

Type

Boolean

117.1.4 PDFSaveOptions.bleedOffsetRect

`pDFSaveOptions.bleedOffsetRect`

Description

The bleed offset rectangle.

Type

Array of 4 numbers

117.1.5 PDFSaveOptions.colorBars

`pDFSaveOptions.colorBars`

Description

Optional. Draw color bars.

Default: `false`

Type

Boolean

117.1.6 PDFSaveOptions.colorCompression

`pDFSaveOptions.colorCompression`

Description

Optional. The type of color bitmap compression used.

Default: `CompressionQuality.None`

Type

CompressionQuality

117.1.7 PDFSaveOptions.colorConversionID

`pDFSaveOptions.colorConversionID`

Description

Optional. The PDF color conversion policy.

Default: `ColorConversion.None`

Type

ColorConversion

117.1.8 PDFSaveOptions.colorDestinationID

`pDFSaveOptions.colorDestinationID`

Description

Optional. The conversion target for color conversion.

Default: `ColorDestination.None`

Type

ColorDestination

117.1.9 PDFSaveOptions.colorDownsampling

`pDFSaveOptions.colorDownsampling`

Description

Optional. The color downsampling resolution in dots per inch (dpi). If 0, no downsampling is performed.

Default: 150.0

Type

Number (double)

117.1.10 PDFSaveOptions.colorDownsamplingImageThreshold

`pDFSaveOptions.colorDownsamplingImageThreshold`

Description

Optional. Downsample if the image's resolution is above this value.

Default: 225.0

Type

Number (double)

117.1.11 PDFSaveOptions.colorDownsamplingMethod

`pDFSaveOptions.colorDownsamplingMethod`

Description

Optional. How color bitmap images should be resampled.

Default: `DownsampleMethod.NODOWNSAMPLE`

Type

DownsampleMethod

117.1.12 PDFSaveOptions.colorProfileID

`pDFSaveOptions.colorProfileID`

Description

Optional. The color profile to include.

Default: `ColorProfile.None`

Type

ColorProfile

117.1.13 PDFSaveOptions.colorTileSize

`pDFSaveOptions.colorTileSize`

Description

Optional. Tile size when compressing with JPEG2000.

Default: 256

Type

Number (long)

117.1.14 PDFSaveOptions.compatibility

`pDFSaveOptions.compatibility`

Description

Optional. The version of the Acrobat file format to create.

Default: `PDFCompatibility.Acrobat5`

Type

PDFCompatibility

117.1.15 PDFSaveOptions.compressArt

`pDFSaveOptions.compressArt`

Description

Optional. If `true`, the line art and text should be compressed.

Default: `true`

Type

Boolean

117.1.16 PDFSaveOptions.documentPassword

`pDFSaveOptions.documentPassword`

Description

Optional. A password string to open the document.

Default: no string

Type

String

117.1.17 PDFSaveOptions.enableAccess

`pDFSaveOptions.enableAccess`

Description

Optional. If `true`, enable accessing 128-bit.

Default: `true`

Type

Boolean

117.1.18 PDFSaveOptions.enableCopy

`pDFSaveOptions.enableCopy`

Description

Optional. If `true`, enable copying of text 128-bit.

Default: `true`

Type

Boolean

117.1.19 PDFSaveOptions.enableCopyAccess

`pDFSaveOptions.enableCopyAccess`

Description

Optional. If `true`, enable copying and accessing 40-bit.

Default: `true`

Type

Boolean

117.1.20 PDFSaveOptions.enablePlainText

`pDFSaveOptions.enablePlainText`

Description

Optional. If `true`, enable plaintext metadata 128-bit. Available only for Acrobat 6.

Default: `false`

Type

Boolean

117.1.21 PDFSaveOptions.flattenerOptions

`pDFSaveOptions.flattenerOptions`

Description

Optional. The printing flattener options.

Type

PrintFlattenerOptions

117.1.22 PDFSaveOptions.flattenerPreset

`pDFSaveOptions.flattenerPreset`

Description

Optional. The transparency flattener preset name.

Type

String.

117.1.23 PDFSaveOptions.fontSubsetThreshold

`pDFSaveOptions.fontSubsetThreshold`

Description

Optional. Include a subset of fonts when less than this percentage of characters is used in the document. Valid for Illustrator 9 file format. Range: 0.0 to 100.0.

Default: 100.0

Type

Number (double)

117.1.24 PDFSaveOptions.generateThumbnails

`pDFSaveOptions.generateThumbnails`

Description

Optional. If `true`, thumbnail images are generated with the saved file.

Default: `true`

Type

Boolean

117.1.25 PDFSaveOptions.grayscaleCompression

`pDFSaveOptions.grayscaleCompression`

Description

Optional. Quality of grayscale bitmap compression.

Default: `CompressionQuality.None`

Type

CompressionQuality

117.1.26 PDFSaveOptions.grayscaleDownsampling

`pDFSaveOptions.grayscaleDownsampling`

Description

Optional. Downsampling resolution in dots per inch (dpi). If 0, no downsampling is performed.

Default: 150.0

Type

Number (double)

117.1.27 PDFSaveOptions.grayscaleDownsamplingImageThreshold

`pDFSaveOptions.grayscaleDownsamplingImageThreshold`

Description

Optional. Downsample if the image's resolution is above this value.

Default: 225.0

Type

Number (double)

117.1.28 PDFSaveOptions.grayscaleDownsamplingMethod

`pDFSaveOptions.grayscaleDownsamplingMethod`

Description

Optional. How grayscale bitmap images should be resampled

Default: `DownSampleMethod.NODOWNSAMPLE`

Type

DownsampleMethod

117.1.29 PDFSaveOptions.grayscaleTileSize

`pDFSaveOptions.grayscaleTileSize`

Description

Optional. Tile size when compressing with JPEG2000.

Default: 256

Type

Number (long)

117.1.30 PDFSaveOptions.monochromeCompression

`pDFSaveOptions.monochromeCompression`

Description

Optional. Type of monochrome bitmap compression used.

Default: `MonochromeCompression.None`

Type

MonochromeCompression

117.1.31 PDFSaveOptions.monochromeDownsampling

`pDFSaveOptions.monochromeDownsampling`

Description

Optional. Downsampling resolution in dots per inch (dpi). If 0, no downsampling is performed.

Default: 300

Type

Number (double)

117.1.32 PDFSaveOptions.monochromeDownsamplingImageThreshold

`pDFSaveOptions.monochromeDownsamplingImageThreshold`

Description

Optional. Downsample if the image's resolution is above this value.

Default: 450.0

Type

Number (double)

117.1.33 PDFSaveOptions.monochromeDownsamplingMethod

`pDFSaveOptions.monochromeDownsamplingMethod`

Description

Optional. How monochrome bitmap images should be resampled.

Default: `DownSampleMethod.NODOWNSAMPLE`

Type

DownsampleMethod

117.1.34 PDFSaveOptions.offset

`pDFSaveOptions.offset`

Description

Optional. Custom offset in points for using the custom paper.

Default: 0.0

Type

Number (double)

117.1.35 PDFSaveOptions.optimization

`pDFSaveOptions.optimization`

Description

Optional. If `true`, the PDF document should be optimized for fast web viewing.

Default: `false`

Type

Boolean

117.1.36 PDFSaveOptions.outputCondition

`pDFSaveOptions.outputCondition`

Description

Optional. An optional comment to add to the PDF file, describing the intended printing condition.

Default: not included

Type

String

117.1.37 PDFSaveOptions.outputConditionID

`pDFSaveOptions.outputConditionID`

Description

Optional. The name of a registered printing condition.

Default: not included

Type

String

117.1.38 PDFSaveOptions.pageInformation

`pDFSaveOptions.pageInformation`

Description

Optional. If `true`, raw page information.

Default: `false`

Type

Boolean

117.1.39 PDFSaveOptions.pageMarksType

`pDFSaveOptions.pageMarksType`

Description

Optional. The page marks style.

Default: PageMarksType.Roman

Type

PageMarksTypes

117.1.40 PDFSaveOptions.pDFAllowPrinting

`pDFSaveOptions.pDFAllowPrinting`

Description

Optional. PDF security printing permission.

Default: `PDFPrintAllowedEnum.PRINT128HIGHRESOLUTION`

Type

PDFPrintAllowedEnum

117.1.41 PDFSaveOptions.pDFChangesAllowed

`pDFSaveOptions.pDFChangesAllowed`

Description

Optional. Security changes allowed.

Default: `PDFChangeAllowedEnum.CHANGE128ANYCHANGES`

Type

PDFChangesAllowedEnum

117.1.42 PDFSaveOptions.pDFPreset

`pDFSaveOptions.pDFPreset`

Description

Optional. Name of PDF preset to use.

Type

String

117.1.43 PDFSaveOptions.pDFXStandard

`pDFSaveOptions.pDFXStandard`

Description

Optional. The PDF standard with which this document complies.

Default: `PDFXStandard.PDFXNONE`

Type

PDFXStandard

117.1.44 PDFSaveOptions.pDFXStandardDescription

`pDFSaveOptions.pDFXStandardDescription`

Description

Optional. A description of the PDF standard from the selected preset.

Type

String

117.1.45 PDFSaveOptions.permissionPassword

`pDFSaveOptions.permissionPassword`

Description

Optional. A password string to restrict editing security settings.

Default: no string

Type

String

117.1.46 PDFSaveOptions.preserveEditability

`pDFSaveOptions.preserveEditability`

Description

Optional. If `true`, Illustrator editing capabilities should be preserved when saving the document.

Default: `true`

Type

Boolean

117.1.47 PDFSaveOptions.printerResolution

`pDFSaveOptions.printerResolution`

Description

Optional. Flattening printer resolution.

Default: 800.0

Type

Number (double)

117.1.48 PDFSaveOptions.registrationMarks

`pDFSaveOptions.registrationMarks`

Description

Optional. If `true`, draw registration marks.

Default: `false`

Type

Boolean

117.1.49 PDFSaveOptions.requireDocumentPassword

`pDFSaveOptions.requireDocumentPassword`

Description

Optional. Require a password to open the document.

Default: `false`

Type

Boolean

117.1.50 PDFSaveOptions.requirePermissionPassword

`pDFSaveOptions.requirePermissionPassword`

Description

Optional. Use a password to restrict editing security settings.

Default: `false`

Type

Boolean

117.1.51 PDFSaveOptions.trapped

`pDFSaveOptions.trapped`

Description

Optional. If `true`, manual trapping has been prepared for the document.

Default: `false`

Type

Boolean

117.1.52 PDFSaveOptions.trimMarks

`pDFSaveOptions.trimMarks`

Description

Optional. Draw trim marks.

Default: `false`

Type

Boolean

117.1.53 PDFSaveOptions.trimMarkWeight

`pDFSaveOptions.trimMarkWeight`

Description

Optional. The trim mark weight.

Default: `PDFTrimMarkWeight.TRIMMARKWEIGHT0125`

Type

PDFTrimMarkWeight

117.1.54 PDFSaveOptions.typename

`pDFSaveOptions.typename`

Description

Optional. Read-only. The class name of the referenced object.

Type

String

117.1.55 PDFSaveOptions.viewAfterSaving

`pDFSaveOptions.viewAfterSaving`

Description

Optional. View PDF after saving.

Default: `false`

Type

Boolean

117.2 Example

117.2.1 Saving to PDF format

```
// Saves the current document as PDF to dest with specified options
// dest contains the full path and file name to save to
function saveFileToPDF(dest) {
  var doc = app.activeDocument;

  if (app.documents.length > 0) {
    var saveName = new File(dest);
    saveOpts = new PDFSaveOptions();

    saveOpts.compatibility = PDFCompatibility.ACROBAT5;
    saveOpts.generateThumbnails = true;
    saveOpts.preserveEditability = true;

    doc.saveAs(saveName, saveOpts);
  }
}
```

CHAPTER 118

PhotoshopFileOptions

`preferences.photoshopFileOptions`

Description

Options for opening a Photoshop file, used with the *Application.open()* method. All properties are optional.

118.1 Properties

118.1.1 PhotoshopFileOptions.parent

`preferences.photoshopFileOptions.parent`

Description

The parent of this object.

Type

Object; read-only.

118.1.2 PhotoshopFileOptions.pixelAspectRatioCorrection

`preferences.photoshopFileOptions.pixelAspectRatioCorrection`

Description

If `true`, imported images that have a non-square pixel aspect ratio should be adjusted.

Type

Boolean

118.1.3 PhotoshopFileOptions.preserveImageMaps

`preferences.photoshopFileOptions.preserveImageMaps`

Description

If `true`, image maps should be preserved when document is converted.

Default: `true`

Type

Boolean

118.1.4 PhotoshopFileOptions.preserveLayers

`preferences.photoshopFileOptions.preserveLayers`

Description

If `true`, layers should be preserved when document is converted.

Default: `true`

Type

Boolean

118.1.5 PhotoshopFileOptions.preserveSlices

`preferences.photoshopFileOptions.preserveSlices`

Description

If `true`, slices should be preserved when document is converted.

Default: `true`

Type

Boolean

118.1.6 PhotoshopFileOptions.typename

`preferences.photoshopFileOptions.typename`

Description

The class name of the referenced object.

Type

String; read-only.

118.2 Example

118.2.1 Opening a Photoshop file

```
// Opens a Photoshop file containing layers with
// preferences set to preserve layers
var psdOptions = preferences.photoshopFileOptions;
psdOptions.preserveLayers = true;
psdOptions.pixelAspectRatioCorrection = false;

// open a file using these prefs
var fileRef = File(psdFilePath);
if (fileRef != null) {
  var docRef = open(fileRef, DocumentColorSpace.RGB);
}
```

CHAPTER 119

PlacedItems

`app.activeDocument.placedItems`

Description

A collection of *PlacedItem* objects in a document.

119.1 Properties

119.1.1 PlacedItems.length

`app.activeDocument.placedItems.length`

Description

Number of elements in the collection.

Type

Number, read-only.

119.1.2 PlacedItems.parent

`app.activeDocument.placedItems.parent`

Description

The object's container.

Type

Object, read-only.

119.1.3 PlacedItems.typename

```
app.activeDocument.placedItems.typename
```

Description

The class name of the object.

Type

String, read-only.

119.2 Methods

119.2.1 PlacedItems.add()

```
app.activeDocument.placedItems.add()
```

Description

Creates a new object.

Use to place new art in a document. Use the `file` property of the resulting `placedItem` object to link the file containing the artwork. See *PlacedItem*.

Returns

PlacedItem

119.2.2 PlacedItems.getByName()

```
app.activeDocument.placedItems.getByName(name)
```

Description

Get the first element in the collection with the provided name.

Parameters

Parameter	Type	Description
`name`	String	Name of element to get

Returns

PlacedItem

119.2.3 PlacedItems.index()

```
app.activeDocument.placedItems.index(itemKey)
```

Description

Gets an element from the collection.

Parameters

Parameter	Type	Description
`itemKey`	String, Number	String or number key

Returns

PlacedItem

119.2.4 PlacedItems.removeAll()

```
app.activeDocument.placedItems.removeAll()
```

Description

Deletes all elements in the collection.

Returns

Nothing.

CHAPTER 120

PluginItems

`app.activeDocument.pluginItems`

Description

A collection of *PluginItem* objects in a document.

See *Copying a plug-in item*.

120.1 Properties

120.1.1 PluginItems.length

`app.activeDocument.pluginItems.length`

Description

Number of elements in the collection.

Type

Number, read-only.

120.1.2 PluginItems.parent

`app.activeDocument.pluginItems.parent`

Description

The object's container.

Type

Object, read-only.

120.1.3 PluginItems.typename

`app.activeDocument.pluginItems.typename`

Description

The class name of the object.

Type

String, read-only.

120.2 Methods

120.2.1 PluginItems.getByName()

`app.activeDocument.pluginItems.getByName(name)`

Description

Get the first element in the collection with the provided name.

Parameters

Parameter	Type	Description
`name`	String	Name of element to get

Returns

PluginItem

120.2.2 PluginItems.index()

`app.activeDocument.pluginItems.index(itemKey)`

Description

Gets an element from the collection.

Parameters

Parameter	Type	Description
`itemKey`	String, Number	String or number key

Returns

PluginItem

120.2.3 PluginItems.removeAll()

```
app.activeDocument.pluginItems.removeAll()
```

Description

Deletes all elements in the collection.

Returns

Nothing.

CHAPTER 121

PPDFile

`app.PPDFileList[index`

Description

Associates file information with a PostScript Printer Description (PPD) file.

121.1 Properties

121.1.1 PPDFile.name

`app.PPDFileList[index].name`

Description

The PPD model name.

Type

String

121.1.2 PPDFile.PPDInfo

`app.PPDFileList[index].PPDInfo`

Description

The PPD file information.

Type

PPDFileInfo

121.1.3 PPDFile.typename

```
app.PPDFileList[index].typename
```

Description

The class name of the object.

Type

String; read-only.

CHAPTER 122

PPDFileInfo

`app.PPDFileList[index].PPDInfo`

Description

Information about a PostScript Printer Description (PPD) file.

122.1 Properties

122.1.1 PPDFileInfo.languageLevel

`app.PPDFileList[index].PPDInfo.languageLevel`

Description

The PostScript language level.

Type

String

122.1.2 PPDFileInfo.PPDFilePath

`app.PPDFileList[index].PPDInfo.PPDFilePath`

Description

Path specification for the PPD file.

Type

File

122.1.3 PPDFileInfo.screenList

`app.PPDFileList[index].PPDInfo.screenList`

Description

List of color separation screens.

Type

Array of *Screen*

122.1.4 PPDFileInfo.screenSpotFunctionList

`app.PPDFileList[index].PPDInfo.screenSpotFunctionList`

Description

List of color separation screen spot functions.

Type

Array of *ScreenSpotFunction*

122.2 Example

122.2.1 Displaying PPD file properties

```
// Displays postscript level and path for each PPD file found in a new text frame
var sPPD = "";
var docRef = documents.add();

var x = 30;
var y = (docRef.height - 30);

var iLength = PPDFileList.length;
if (iLength > 20)
  iLength = 20;

for (var i = 0; i < iLength; i++) {
  var ppdRef = PPDFileList[i];
  sPPD = ppdRef.name;
  sPPD += "\r\tPS Level ";

  var ppdInfoRef = ppdRef.PPDInfo;
  sPPD += ppdInfoRef.languageLevel;
  sPPD += "\r\tPath: ";
  sPPD += ppdInfoRef.PPDFilePath;

  var textRef = docRef.textFrames.add();
  textRef.textRange.characterAttributes.size = 8;
  textRef.contents = sPPD;
  textRef.top = (y);
```

(continues on next page)

(continued from previous page)

```
  textRef.left = x;

  redraw();

  if ((y -= (textRef.height)) <= 30) {
    y = (docRef.height - 30);
    x += 150;
  }
}
```

122.2.2 PPDFileInfo and related screen information

```
// Displays in a new text frame, the postscript level, file paths, screens, and
// screen spot information for first 10 PPD files found

var sPPD = "";
var docRef = documents.add();

var x = 30;
var y = (docRef.height - 30);

var iLength = PPDFileList.length;

if (iLength > 10)
  iLength = 10;

for (var i = 0; i < iLength; i++) {
  var ppdRef = PPDFileList[i];
  sPPD = ppdRef.name;
  sPPD += "\r\tPS Level ";

  var ppdInfoRef = ppdRef.PPDInfo;
  sPPD += ppdInfoRef.languageLevel;
  sPPD += "\r\tPath: ";
  sPPD += ppdInfoRef.PPDFilePath;
  sPPD += "\r\tScreens:\r";

  var iScreens = ppdInfoRef.screenList.length;
  for (var c = 0; c < iScreens; c++) {

    var screenRef = ppdInfoRef.screenList[c];
    sPPD += "\t\t";
    sPPD += screenRef.name;

    var screenInfoRef = screenRef.screenInfo;
    sPPD += ", Angle = ";
    sPPD += screenInfoRef.angle;
    sPPD += ", Frequency = ";
    sPPD += screenInfoRef.frequency;
    sPPD += "\r";
  }

  sPPD += "\r\tScreenSpots:\r";
```

(continues on next page)

(continued from previous page)

```
  var iScreenSpots = ppdInfoRef.screenSpotFunctionList.length;
  for (var n = 0; n < iScreenSpots; n++) {
    var screenSpotRef = ppdInfoRef.screenSpotFunctionList[n];
    sPPD += "\t\t";
    sPPD += screenSpotRef.name;
    sPPD += ", spotFunction: ";
    sPPD += screenSpotRef.spotFunction;
    sPPD += "\r";
  }

  var textRef = docRef.textFrames.add();
  textRef.textRange.characterAttributes.size = 8;
  textRef.contents = sPPD;
  textRef.top = (y);
  textRef.left = x;

  redraw();

  y -= (textRef.height);
}
```

CHAPTER 123

Preferences

`app.Preferences`

Description

Specifies the preferred options for AutoCAD, FreeHand, PDF, and Photoshop files.

123.1 Properties

123.1.1 Preferences.AutoCADFileOptions

`app.preferences.AutoCADFileOptions`

Description

Options to use when opening or placing an AutoCAD file.

Type

OpenOptionsAutoCAD; read-only.

123.1.2 Preferences.FreeHandFileOptions

`app.preferences.FreeHandFileOptions`

Description

Options to use when opening or placing a FreeHand file.

Type

OpenOptionsFreeHand; read-only.

123.1.3 Preferences.parent

`app.preferences.parent`

Description

The parent of this object.

Type

object; read-only.

123.1.4 Preferences.PDFFileOptions

`app.preferences.PDFFileOptions`

Description

Options to use when opening or placing a PDF file.

Type

PDFFileOptions; read-only.

123.1.5 Preferences.PhotoshopFileOptions

`app.preferences.PhotoshopFileOptions`

Description

Options to use when opening or placing a Photoshop file.

Type

PhotoshopFileOptions; read-only.

123.1.6 Preferences.typename

`app.preferences.typename`

Description

The class name of the referenced object.

Type

string; read-only.

123.2 Methods

123.2.1 Preferences.getBooleanPreference

`app.preferences.getBooleanPreference(key)`

Description

Gets the boolean value of a given application preference.

Parameters

Parameter	Type	Description
`key`	String	Pref key of value to get

Returns

Boolean

123.2.2 Preferences.getIntegerPreference

`app.preferences.getIntegerPreference(key)`

Description

Gets the integer value of a given application preference.

Parameters

Parameter	Type	Description
`key`	String	Pref key of value to get

Returns

Integer

123.2.3 Preferences.getRealPreference

`app.preferences.getRealPreference(key)`

Description

Gets the real-number value of a given application preference.

Parameters

Parameter	Type	Description
`key`	String	Pref key of value to get

Returns

Real

123.2.4 Preferences.getStringPreference

```
app.preferences.getStringPreference(key)
```

Description

Gets the string value of a given application preference.

Parameters

Parameter	Type	Description
key	String	Pref key of value to get

Returns

String

123.2.5 Preferences.removePreference

```
app.preferences.removePreference(key)
```

Description

Deletes a given application preference.

Parameters

Parameter	Type	Description
key	String	Pref key of value to get

Returns

Nothing.

123.2.6 Preferences.setBooleanPreference

```
app.preferences.setBooleanPreference(key, value)
```

Description

Sets the boolean value of a given application preference.

Parameters

Parameter	Type	Description
key	String	Pref key of value to get
value	Boolean	Value to set

Returns

Nothing.

123.2.7 Preferences.setIntegerPreference

```
app.preferences.setIntegerPreference(key, value)
```

Description

Sets the integer value of a given application preference.

Parameters

Parameter	Type	Description
key	String	Pref key of value to get
value	Integer	Value to set

Returns

Nothing.

123.2.8 Preferences.setRealPreference

```
app.preferences.setRealPreference(key, value)
```

Description

Sets the real-number value of a given application preference.

Parameters

Parameter	Type	Description
key	String	Pref key of value to get
value	Double	Value to set

Returns

Nothing.

123.2.9 Preferences.setStringPreference

```
app.preferences.setStringPreference(key, value)
```

Description

Sets the string value of a given application preference.

Parameters

Parameter	Type	Description
key	String	Pref key of value to get
value	String	Value to set

Returns

Nothing.

CHAPTER 124

PrintColorManagementOptions

`new PrintColorManagementOptions()`

Description

Information used for color management of the document.

124.1 Properties

124.1.1 PrintColorManagementOptions.colorProfileMode

`printColorManagementOptions.colorProfileMode`

Description

The color management profile mode. Default: `PrintColorProfile.SOURCEPROFILE`

Type

PrintColorProfile

124.1.2 PrintColorManagementOptions.intent

`printColorManagementOptions.intent`

Description

The color management intent type. Default: `PrintColorIntent.RELATIVECOLORIMETRIC`

Type

PrintColorIntent

124.1.3 PrintColorManagementOptions.name

`printColorManagementOptions.name`

Description

The color management profile name.

Type

String

124.1.4 PrintColorManagementOptions.typename

`printColorManagementOptions.typename`

Description

The class name of the object.

Type

String; read-only.

124.2 Example

124.2.1 Managing colors for printing

```
// Creates a new document, adds symbols, then creates a
// PrintColorManagementOptions object and assigns it
// to a PrintOptions object, then prints with each color intent

// Add some symbol items to a new document
var docRef = documents.add();
var y = docRef.height - 30;

for (var i = 0; i < (docRef.symbols.length); i++) {
  symbolRef = docRef.symbols[i];

  symbolItemRef1 = docRef.symbolItems.add(symbolRef);
  symbolItemRef1.top = y;
  symbolItemRef1.left = 100;

  y -= (symbolItemRef1.height + 10);
}

redraw();

var colorOptions = new PrintColorManagementOptions();
var options = new PrintOptions();
options.colorManagementOptions = colorOptions;
colorOptions.name = "ColorMatch RGB";
```

(continues on next page)

(continued from previous page)

```
// Print the current document once for each color intent.
colorOptions.intent = PrintColorIntent.ABSOLUTECOLORIMETRIC;
docRef.print(options);

colorOptions.intent = PrintColorIntent.PERCEPTUALINTENT;
docRef.print(options);

colorOptions.intent = PrintColorIntent.RELATIVECOLORIMETRIC;
docRef.print(options);

colorOptions.intent = PrintColorIntent.SATURATIONINTENT;
docRef.print(options);
```

CHAPTER 125

PrintColorSeparationOptions

`new PrintColorSeparationOptions()`

Description

Information about the color separations to be used in printing the document.

125.1 Properties

125.1.1 PrintColorSeparationOptions.colorSeparationMode

`printColorSeparationOptions.colorSeparationMode`

Description

The color separation type.

Default: `PrintColorSeparationMode.COMPOSITE`

Type

PrintColorSeparationMode

125.1.2 PrintColorSeparationOptions.convertSpotColors

`printColorSeparationOptions.convertSpotColors`

Description

If `true`, all spot colors should be converted to process colors.

Default: `false`

Type

Boolean

125.1.3 PrintColorSeparationOptions.inkList

`printColorSeparationOptions.inkList`

Description

The list of inks for color separation.

Type

Array of *Ink*

125.1.4 PrintColorSeparationOptions.overPrintBlack

`printColorSeparationOptions.overPrintBlack`

Description

If `true`, overprint in black.

Default: `false`

Type

Boolean

125.1.5 PrintColorSeparationOptions.typename

`printColorSeparationOptions.typename`

Description

Read-only. The class name of the object.

Type

String

125.2 Example

125.2.1 Managing color separations for printing

```
// Creates a new document with symbol items
// and prints document with each separation option

// Add some symbol items to a new document
var docRef = documents.add();
var y = docRef.height - 30;

for (var i = 0; i < (docRef.symbols.length); i++) {
  symbolRef = docRef.symbols[i];

  symbolItemRef1 = docRef.symbolItems.add(symbolRef);
  symbolItemRef1.top = y;
  symbolItemRef1.left = 100;

  y -= (symbolItemRef1.height + 10);
}

// Print with various separation options
var sepOptions = new PrintColorSeparationOptions();
var options = new PrintOptions();
options.colorSeparationOptions = sepOptions;

sepOptions.convertSpotColors = true;
sepOptions.overPrintBlack = true;
sepOptions.colorSeparationMode = PrintColorSeparationMode.COMPOSITE;
docRef.print(options);

sepOptions.colorSeparationMode = PrintColorSeparationMode.INRIPSEPARATION;
docRef.print(options);

sepOptions.convertSpotColors = false;
sepOptions.overPrintBlack = false;
sepOptions.colorSeparationMode = PrintColorSeparationMode.HOSTBASEDSEPARATION;
docRef.print(options);
```

CHAPTER 126

PrintCoordinateOptions

`new PrintCoordinateOptions()`

Description

Information about the media and associated printing parameters.

126.1 Properties

126.1.1 PrintCoordinateOptions.emulsion

`printCoordinateOptions.emulsion`

Description

If `true`, flip artwork horizontally.

Default: `false`

Type

Boolean

126.1.2 PrintCoordinateOptions.fitToPage

`printCoordinateOptions.fitToPage`

Description

If `true`, proportionally scale the artwork to fit on media.

Default: `false`

Type

Boolean

126.1.3 PrintCoordinateOptions.horizontalScale

`printCoordinateOptions.horizontalScale`

Description

The horizontal scaling factor expressed as a percentage (100 = 100%).

Range: 1.0 to 10000.0.

Default: 100.0

Type

Number (double)

126.1.4 PrintCoordinateOptions.orientation

`printCoordinateOptions.orientation`

Description

The artwork orientation.

Default: `PrintOrientation.PORTRAIT`

Type

PrintOrientation

126.1.5 PrintCoordinateOptions.position

`printCoordinateOptions.position`

Description

The artwork position on media.

Default: `PrintPosition.TRANSLATECENTER`

Type

PrintPosition

126.1.6 PrintCoordinateOptions.tiling

`printCoordinateOptions.tiling`

Description

The page tiling mode.

Default: `PrintTiling.TILESINGLEFULLPAGE`

Type

PrintTiling

126.1.7 PrintCoordinateOptions.typename

`printCoordinateOptions.typename`

Description

The class name of the object.

Type

String; read-only.

126.1.8 PrintCoordinateOptions.verticalScale

`printCoordinateOptions.verticalScale`

Description

The vertical scaling factor expressed as a percentage (100 = 100%)

Range: 1.0 to 10000.0.

Default: 100.0

Type

Number (double)

126.2 Example

126.2.1 Managing print coordinates

```
// Creates a new document with symbol items that extend
// off the page then print with each print orientation
var docRef = documents.add();
var y = 500;
var x = -70;
```

(continues on next page)

(continued from previous page)

```
if (docRef.symbols.length > 0) {

  for (var i = 0; i < 5; i++) {
    symbolRef = docRef.symbols[0];

    symbolItemRef1 = docRef.symbolItems.add(symbolRef);
    symbolItemRef1.top = y;
    symbolItemRef1.left = x;

    x += 30;
  }

  redraw();

  // Print it with various Coordinate Options
  var coordinateOptions = new PrintCoordinateOptions();
  var options = new PrintOptions();
  options.coordinateOptions = coordinateOptions;

  coordinateOptions.emulsion = true; // reverse from right to left
  coordinateOptions.fitToPage = true; // fit artwork to page size
  coordinateOptions.orientation = PrintOrientation.LANDSCAPE;
  docRef.print(options);

  coordinateOptions.emulsion = false;
  coordinateOptions.fitToPage = false;
  coordinateOptions.orientation = PrintOrientation.PORTRAIT;
  coordinateOptions.horizontalScale = 50;
  coordinateOptions.verticalScale = 50;
  docRef.print(options);
}
```

CHAPTER 127

Printer

`app.PrinterList[index`

Description

Associates an available printer with printer information.

To request a list of printers, you must first have a document open or an error is returned.

127.1 Properties

127.1.1 Printer.name

`app.printerList[index].name`

Description

The printer name.

Type

String

127.1.2 Printer.printerInfo

`app.printerList[index].printerInfo`

Description

The printer information.

Type

PrinterInfo

127.1.3 Printer.typename

`app.printerList[index].typename`

Description

The class name of the object.

Type

String; read-only.

CHAPTER 128

PrinterInfo

`printerInfo`

Description

Configuration information about a printer.

128.1 Properties

128.1.1 PrinterInfo.binaryPrintingSupport

`printerInfo.binaryPrintingSupport`

Description

If `true`, the printer supports binary printing.

Type

Boolean

128.1.2 PrinterInfo.colorSupport

`printerInfo.colorSupport`

Description

The printer color capability.

Type

PrinterColorMode

128.1.3 PrinterInfo.customPaperSupport

`printerInfo.customPaperSupport`

Description

If `true`, the printer supports custom paper size.

Type

Boolean

128.1.4 PrinterInfo.customPaperTransverseSupport

`printerInfo.customPaperTransverseSupport`

Description

If `true`, the printer supports custom paper transverse.

Type

Boolean

128.1.5 PrinterInfo.deviceResolution

`printerInfo.deviceResolution`

Description

The printer default resolution.

Type

Number (double)

128.1.6 PrinterInfo.inRIPSeparationSupport

`printerInfo.inRIPSeparationSupport`

Description

If `true`, the printer supports InRIP color separation.

Type

Boolean

128.1.7 PrinterInfo.maxDeviceResolution

`printerInfo.maxDeviceResolution`

Description

The printer maximum device resolution.

Type

Number (double)

128.1.8 PrinterInfo.maxPaperHeight

`printerInfo.maxPaperHeight`

Description

Custom paper's maximum height.

Type

Number (double)

128.1.9 PrinterInfo.maxPaperHeightOffset

`printerInfo.maxPaperHeightOffset`

Description

Custom paper's maximum height offset.

Type

Number (double)

128.1.10 PrinterInfo.maxPaperWidth

`printerInfo.maxPaperWidth`

Description

Custom paper's maximum width.

Type

Number (double)

128.1.11 PrinterInfo.maxPaperWidthOffset

`printerInfo.maxPaperWidthOffset`

Description

Custom paper's maximum width offset.

Type

Number (double)

128.1.12 PrinterInfo.minPaperHeight

`printerInfo.minPaperHeight`

Description

Custom paper's minimum height.

Type

Number (double)

128.1.13 PrinterInfo.minPaperHeightOffset

`printerInfo.minPaperHeightOffset`

Description

Custom paper's minimum height offset.

Type

Number (double)

128.1.14 PrinterInfo.minPaperWidth

`printerInfo.minPaperWidth`

Description

Custom paper's minimum width.

Type

Number (double)

128.1.15 PrinterInfo.minPaperWidthOffset

`printerInfo.minPaperWidthOffset`

Description

Custom paper's minimum width offset.

Type

Number (double)

128.1.16 PrinterInfo.paperSizes

`printerInfo.paperSizes`

Description

The list of supported paper sizes.

Type

Array of *Paper*

128.1.17 PrinterInfo.postScriptLevel

`printerInfo.postScriptLevel`

Description

The PostScript Language level.

Type

PrinterPostScriptLevelEnum

128.1.18 PrinterInfo.printerType

`printerInfo.printerType`

Description

The printer type.

Type

PrinterTypeEnum

128.1.19 PrinterInfo.typename

`printerInfo.typename`

Description

The class name of the object.

Type

String; read-only.

128.2 Example

128.2.1 Finding available printers

```
// Displays a list of available printers in a new text frame
var docRef = documents.add();
var iCount = printerList.length;

var textRef = docRef.textFrames.add();
textRef.contents += "Printers...\r";

for (var i = 0; i < iCount; i++) {
  textRef.contents += printerList[i].name;
  textRef.contents += "\r\t";
}

textRef.top = 600;
textRef.left = 200;

redraw();
```

CHAPTER 129

PrintFlattenerOptions

`new PrintFlattenerOptions()`

Description

Contains flattening options for use when Illustrator outputs artwork that contains transparency into a non-native format.

129.1 Properties

129.1.1 PrintFlattenerOptions.clipComplexRegions

`printFlattenerOptions.clipComplexRegions`

Description

If `true`, complex regions should be clipped.

Default: `false`

Type

Boolean

129.1.2 PrintFlattenerOptions.convertStrokesToOutlines

`printFlattenerOptions.convertStrokesToOutlines`

Description

If `true`, convert all strokes to outlines.

Default: `false`

Type

Boolean

129.1.3 PrintFlattenerOptions.convertTextToOutlines

`printFlattenerOptions.convertTextToOutlines`

Description

If `true`, all text is converted to vector paths; preserves the visual appearance of type.

Default: `false`

Type

Boolean

129.1.4 PrintFlattenerOptions.flatteningBalance

`printFlattenerOptions.flatteningBalance`

Description

The flattening balance.

Range: 0.0 to 100.0.

Default: 100.0

Type

Number (long)

129.1.5 PrintFlattenerOptions.gradientResolution

`printFlattenerOptions.gradientResolution`

Description

The gradient resolution in dots per inch (dpi).

Range: 1.0 to 9600.0.

Default: 300.0

Type

Number (double)

129.1.6 PrintFlattenerOptions.overprint

`printFlattenerOptions.overprint`

Description

Whether to preserve, discard, or simulate overprinting.

Default: `PDFOverprint.PRESERVEPDFOVERPRINT`

Type

PDFOverprint

129.1.7 PrintFlattenerOptions.rasterizationResolution

`printFlattenerOptions.rasterizationResolution`

Description

The rasterization resolution in dots per inch (dpi). Range: 1.0 to 9600.0.

Default: 300.0

Type

Number (double)

129.1.8 PrintFlattenerOptions.typename

`printFlattenerOptions.typename`

Description

The class name of the object.

Type

String; read-only.

129.2 Example

129.2.1 Setting print flattening

```
// Creates a new document, adds symbols to the document
// then prints with a range of flattener balance settings
var docRef = documents.add();
var y = docRef.height - 30;

for (var i = 0; i < (docRef.symbols.length); i++) {
  symbolRef = docRef.symbols[i];
```

(continues on next page)

(continued from previous page)

```
  symbolItemRef1 = docRef.symbolItems.add(symbolRef);
  symbolItemRef1.top = y;
  symbolItemRef1.left = 100;

  y -= (symbolItemRef1.height + 10);
}

redraw();

// Create PrintFlattenerOptions object and assign to a PrintOptions object
var flatOpts = new PrintFlattenerOptions();
var printOpts = new PrintOptions();
printOpts.flattenerOptions = flatOpts;

// Set other print options
printOpts.ClipComplexRegions = true;
printOpts.GradientResoultion = 360;
printOpts.RasterizatonResotion = 360;

// Print the current document with flattening balance increments of 20
for (var i = 0; i <= 100; i += 20) {
  flatOpts.flatteningBalance = i;
  activeDocument.print(printOpts);
}
```

CHAPTER 130

PrintFontOptions

```
new PrintFontOptions()
```

Description

Contains information about font downloading and substitution for the fonts used for printing the document.

130.1 Properties

130.1.1 PrintFontOptions.downloadFonts

```
printFontOptions.downloadFonts
```

Description

The font download mode.

Default: `PrintFontDownloadMode.DOWNLOADSUBSET`

Type

PrintFontDownloadMode

130.1.2 PrintFontOptions.fontSubstitution

```
printFontOptions.fontSubstitution
```

Description

The font substitution policy.

Default: `FontSubstitutionPolicy.SUBSTITUTEOBLIQUE`

Type

FontSubstitutionPolicy

130.1.3 PrintFontOptions.typename

`printFontOptions.typename`

Description

The class name of the object.

Type

String, read-only.

130.2 Example

130.2.1 Printing with font options

```
// Creates a new document, adds text then prints with specified font options.
var docRef = documents.add();

var pathRef = docRef.pathItems.rectangle(500, 300, 400, 400);
var textRef = docRef.textFrames.areaText(pathRef);
textRef.contents = "Text example";

//Create PrintFontOptions object and assign to a PrintOptions object
var fontOpts = new PrintFontOptions();
var printOpts = new PrintOptions();
printOpts.fontOptions = fontOpts;

// Set some font options
fontOpts.downloadFonts = PrintFontDownloadMode.DOWNLOADNONE;
fontOpts.fontSubstitution = FontSubstitutionPolicy.SUBSTITUTEDEVICE;

// print it
activeDocument.print(printOpts);
```

CHAPTER 131

PrintJobOptions

```
new PrintJobOptions()
```

Description

Contains information about how the job is to be printed.

131.1 Properties

131.1.1 PrintJobOptions.artboardRange

```
printJobOptions.artboardRange
```

Description

The artboard range to be printed if printAllArtboards is false.

Default: 1-

Type

String

131.1.2 PrintJobOptions.bitmapResolution

```
printJobOptions.bitmapResolution
```

Description

The bitmap resolution. Minimum: 0.0.

Default: 0.0

Type

Number (double)

131.1.3 PrintJobOptions.collate

`printJobOptions.collate`

Description

If `true`, collate print pages.

Default: `false`

Type

Boolean

131.1.4 PrintJobOptions.copies

`printJobOptions.copies`

Description

The number of copies to print. Minimum: 1.

Default: 1

Type

Number (long)

131.1.5 PrintJobOptions.designation

`printJobOptions.designation`

Description

The layers/objects to be printed.

Default: `PrintArtworkDesignation.VISIBLEPRINTABLELAYERS`

Type

PrintArtworkDesignation

131.1.6 PrintJobOptions.file

`printJobOptions.file`

Description

The file to which to print.

Type

File

131.1.7 PrintJobOptions.name

`printJobOptions.name`

Description

The print job name.

Type

String

131.1.8 PrintJobOptions.printAllArtboards

`printJobOptions.printAllArtboards`

Description

Indicates whether to print all artboards.

Default: true

Type

Boolean

131.1.9 PrintJobOptions.printArea

`printJobOptions.printArea`

Description

The printing bounds.

Default: `PrintingBounds.ARTBOARDBOUNDS`

Type

PrintingBounds

131.1.10 PrintJobOptions.printAsBitmap

`printJobOptions.printAsBitmap`

Description

If `true`, print as bitmap.

Default: `false`

Type

Boolean

131.1.11 PrintJobOptions.reversePages

`printJobOptions.reversePages`

Description

If `true`, print pages in reverse order.

Default: `false`

Type

Boolean

131.1.12 PrintJobOptions.typename

`printJobOptions.typename`

Description

Read-only. The class name of the object.

Type

String

131.2 Example

131.2.1 Printing with job options

```
// Creates a new document with layers containing visible, printable,
// non visible and non printable items then prints with each designation
// to view effects of using different job options

var docRef = documents.add();
var textRef_0 = docRef.layers[0].textFrames.add();
textRef_0.contents = "Visible and Printable";
textRef_0.top = 600;
```

(continues on next page)

(continued from previous page)

```
textRef_0.left = 200;

var layerRef_1 = docRef.layers.add();
var textRef_1 = layerRef_1.textFrames.add();
textRef_1.contents = "Visible and Non-Printable";
textRef_1.top = 500;
textRef_1.left = 250;
layerRef_1.printable = false;

var layerRef_2 = docRef.layers.add();
var textRef_2 = layerRef_2.textFrames.add();
textRef_2.contents = "Non-Visible";
textRef_2.top = 400;
textRef_2.left = 300;
layerRef_2.visible = false;
redraw();

// Print with various job options
var printJobOptions = new PrintJobOptions();
var options = new PrintOptions();
options.jobOptions = printJobOptions;

printJobOptions.designation = PrintArtworkDesignation.ALLLAYERS;
printJobOptions.reverse = true;
docRef.print(options);

printJobOptions.collate = false;
printJobOptions.designation = PrintArtworkDesignation.VISIBLELAYERS;
printJobOptions.reverse = false;
docRef.print(options);

printJobOptions.designation = PrintArtworkDesignation.VISIBLEPRINTABLELAYERS;
var docPath = new File("~/printJobTest1.ps");
printJobOptions.file = docPath;
docRef.print(options);
```

CHAPTER 132

PrintOptions

`new PrintOptions()`

Description

Contains information about all printing options including flattening, color management, coordinates, fonts, and paper.

132.1 Properties

132.1.1 PrintOptions.colorManagementOptions

`printOptions.colorManagementOptions`

Description

The printing color management options.

Type

PrintColorManagementOptions

132.1.2 PrintOptions.colorSeparationOptions

`printOptions.colorSeparationOptions`

Description

The printing color separation options.

Type

PrintColorSeparationOptions

132.1.3 PrintOptions.coordinateOptions

`printOptions.coordinateOptions`

Description

The printing coordinate options.

Type

PrintCoordinateOptions

132.1.4 PrintOptions.flattenerOptions

`printOptions.flattenerOptions`

Description

The printing flattener options.

Type

PrintFlattenerOptions

132.1.5 PrintOptions.flattenerPreset

`printOptions.flattenerPreset`

Description

The transparency flattener preset name.

Type

String

132.1.6 PrintOptions.fontOptions

`printOptions.fontOptions`

Description

The printing font options.

Type

PrintFontOptions

132.1.7 PrintOptions.jobOptions

`printOptions.jobOptions`

Description

The printing job options.

Type

PrintJobOptions

132.1.8 PrintOptions.pageMarksOptions

`printOptions.pageMarksOptions`

Description

The printing page marks options.

Type

PrintPageMarksOptions

132.1.9 PrintOptions.paperOptions

`printOptions.paperOptions`

Description

The paper options.

Type

PrintPaperOptions

132.1.10 PrintOptions.postScriptOptions

`printOptions.postScriptOptions`

Description

The printing PostScript options.

Type

PrintPostScriptOptions

132.1.11 PrintOptions.PPDName

`printOptions.PPDName`

Description

The PPD name.

Type

String

132.1.12 PrintOptions.printerName

`printOptions.printerName`

Description

The printer name.

Type

String

132.1.13 PrintOptions.printPreset

`printOptions.printPreset`

Description

The print style.

Type

String

132.2 Example

132.2.1 Setting print options

```
// Creates a new document, adds symbols, specifies a variety of print options,
// assigns each print option to a PrintOptions object,
// then prints with those options
// Create a new document and add some symbol items
var docRef = documents.add();
var y = docRef.height - 30;

for (var i = 0; i < (docRef.symbols.length); i++) {
  symbolRef = docRef.symbols[i];

  symbolItemRef1 = docRef.symbolItems.add(symbolRef);
```

(continues on next page)

(continued from previous page)

```
  symbolItemRef1.top = y;

  symbolItemRef1.left = 100;

  y -= (symbolItemRef1.height + 10);
}

redraw();

// Create multiple options and assign to PrintOptions
var options = new PrintOptions();

var colorOptions = new PrintColorManagementOptions();
colorOptions.name = "ColorMatch RGB";
colorOptions.intent = PrintColorIntent.SATURATIONINTENT;
options.colorManagementOptions = colorOptions;

var printJobOptions = new PrintJobOptions();
printJobOptions.designation = PrintArtworkDesignation.ALLLAYERS;
printJobOptions.reverse = true;

options.jobOptions = printJobOptions;

var coordinateOptions = new PrintCoordinateOptions();
coordinateOptions.fitToPage = true;
options.coordinateOptions = coordinateOptions;

var flatOpts = new PrintFlattenerOptions();
flatOpts.ClipComplexRegions = true;
flatOpts.GradientResoultion = 60;

flatOpts.RasterizatonResotion = 60;
options.flattenerOptions = flatOpts;

// Print with options
docRef.print(options);
```

CHAPTER 133

PrintPageMarksOptions

`new PrintPageMarksOptions()`

Description

The options for printing page marks.

133.1 Properties

133.1.1 PrintPageMarksOptions.bleedOffsetRect

`printPageMarksOptions.bleedOffsetRect`

Description

The bleed offset rectangle.

Type

Array of 4 numbers

133.1.2 PrintPageMarksOptions.colorBars

`printPageMarksOptions.colorBars`

Description

If `true`, enable printing of color bars.

Default: `false`

Type

Boolean

133.1.3 PrintPageMarksOptions.marksOffsetRect

`printPageMarksOptions.marksOffsetRect`

Description

The page marks offset rectangle.

Type

Array of 4 numbers

133.1.4 PrintPageMarksOptions.pageInfoMarks

`printPageMarksOptions.pageInfoMarks`

Description

If `true`, page info marks printing is enabled.

Default: `false`

Type

Boolean

133.1.5 PrintPageMarksOptions.pageMarksType

`printPageMarksOptions.pageMarksType`

Description

The page marks style.

Default: `PageMarksType.Roman`

Type

PageMarksTypes

133.1.6 PrintPageMarksOptions.registrationMarks

`printPageMarksOptions.registrationMarks`

Description

If `true`, registration marks should be printed.

Default: `false`

Type

Boolean

133.1.7 PrintPageMarksOptions.trimMarks

`printPageMarksOptions.trimMarks`

Description

If `true`, trim marks should be printed.

Default: `false`

Type

Boolean

133.1.8 PrintPageMarksOptions.trimMarksWeight

`printPageMarksOptions.trimMarksWeight`

Description

Stroke weight of trim marks. Minimum: 0.0.

Default: 0.125

Type

Number (double)

133.1.9 PrintPageMarksOptions.typename

`printPageMarksOptions.typename`

Description

The class name of the object.

Type

String; read-only.

133.2 Example

133.2.1 Setting page mark printing options

```
// Creates a PrintPageMarksOptions object, assigns it
// to a PrintOptions object, then prints the current document.
var docRef = activeDocument;
var pageMarkOptions= new PrintPageMarksOptions();

var options = new PrintOptions();
options.pageMarksOptions = pageMarkOptions;

pageMarkOptions.colorBars = true;
pageMarkOptions.pageInfoMarks = true;
pageMarkOptions.registrationMarks = true;
pageMarkOptions.trimMarks = true;

docRef.print(options);
```

CHAPTER 134

PrintPaperOptions

```
new PrintPaperOptions()
```

Description

Information about the paper to be used in the print job.

134.1 Properties

134.1.1 PrintPaperOptions.height

```
printPaperOptions.height
```

Description

The custom height (in points) for using the custom paper.

Default: 0.0

Type

Number (double)

134.1.2 PrintPaperOptions.name

```
printPaperOptions.name
```

Description

The paper's name.

Type

String

134.1.3 PrintPaperOptions.offset

`printPaperOptions.offset`

Description

Custom offset (in points) for using the custom paper.

Default: 0.0

Type

Number (double)

134.1.4 PrintPaperOptions.transverse

`printPaperOptions.transverse`

Description

If `true`, transverse the artwork (rotate 90 degrees) on the custom paper.

Default: `false`

Type

Boolean

134.1.5 PrintPaperOptions.typename

`printPaperOptions.typename`

Description

The class name of the object.

Type

String; read-only.

134.1.6 PrintPaperOptions.width

`printPaperOptions.width`

Description

The custom width (in points) for using the custom paper.

Default: 0.0

Type

Number (double)

134.2 Example

134.2.1 Setting print paper options

```
// Creates a new document, adds a path item, applies a graphic style
// then prints with specified paper options
var docRef = documents.add();
var pathRef = docRef.pathItems.rectangle(600, 200, 200, 200);
docRef.graphicStyles[1].applyTo(pathRef);

var paperOpts = new PrintPaperOptions;
var printOpts = new PrintOptions;
printOpts.paperOptions = paperOpts;

var printerCount = printerList.length;
if (printerCount > 0) {

  // Print with the 1st paper from the 1st printer
  for (var i = 0; i < printerList.length; i++) {

    if (printerList[i].printerInfo.paperSizes.length > 0) {
      var printerRef = printerList[i];
    }

    var paperRef = printerRef.printerInfo.paperSizes[0];
    if (printerRef.printerInfo.paperSizes.length > 0){
      paperOpts.name = paperRef.name;
      printOpts.printerName = printerRef.name;
      docRef.print(printOpts);
    }
  }
}
```

CHAPTER 135

PrintPostScriptOptions

`new PrintPostScriptOptions()`

Description

Options for printing to a PostScript printer.

135.1 Properties

135.1.1 PrintPostScriptOptions.binaryPrinting

`printPostScriptOptions.binaryPrinting`

Description

If `true`, printing should be in binary mode.

Default: `false`

Type

Boolean

135.1.2 PrintPostScriptOptions.compatibleShading

`printPostScriptOptions.compatibleShading`

Description

If `true`, use PostScript Level 1-compatible gradient and gradient mesh printing.

Default: `false`

Type

Boolean

135.1.3 PrintPostScriptOptions.forceContinuousTone

`printPostScriptOptions.forceContinuousTone`

Description

If `true`, force continuous tone.

Default: `false`

Type

Boolean

135.1.4 PrintPostScriptOptions.imageCompression

`printPostScriptOptions.imageCompression`

Description

The image compression type.

Default: `PostScriptImageCompressionType.IMAGECOMPRESSIONNONE`

Type

PostScriptImageCompressionType

135.1.5 PrintPostScriptOptions.negativePrinting

`printPostScriptOptions.negativePrinting`

Description

If `true`, print in negative mode.

Default: `false`

Type

Boolean

135.1.6 PrintPostScriptOptions.postScriptLevel

`printPostScriptOptions.postScriptLevel`

Description

The PostScript language level.

Default: `PrinterPostScriptLevelEnum.LEVEL2`

Type

PrinterPostScriptLevelEnum

135.1.7 PrintPostScriptOptions.shadingResolution

`printPostScriptOptions.shadingResolution`

Description

The shading resolution. Range: 1.0 to 9600.0

Default: 300.0

Type

Number (double)

135.1.8 PrintPostScriptOptions.typename

`printPostScriptOptions.typename`

Description

Read-only. The class name of the object.

Type

String

135.2 Example

135.2.1 Setting PostScript printing options

```
// Prints current document with various postscript levels
// Create new postscript options object, assign to print options
var psOpts = new PrintPostScriptOptions();

var printOpts = new PrintOptions();
printOpts.postScriptOptions = psOpts;

// Assign PS level, print
```

(continues on next page)

(continued from previous page)

```
psOpts.postScriptLevel = PrinterPostScriptLevelEnum.PSLEVEL2;
activeDocument.print(printOpts);

psOpts.postScriptLevel = PrinterPostScriptLevelEnum.PSLEVEL3;
activeDocument.print(printOpts);
```

CHAPTER 136

RasterEffectOptions

`RasterEffectOptions`

Description

Specifies raster effects settings for the document. All properties are optional.

136.1 Properties

136.1.1 RasterEffectOptions.antiAliasing

`rasterEffectOptions.antiAliasing`

Description

If `true`, the image should be antialiased.

Default: `false`

Type

Boolean

136.1.2 RasterEffectOptions.clippingMask

`rasterEffectOptions.clippingMask`

Description

If `true`, a clipping mask is created for the image.

Default: `false`

Type

Boolean

136.1.3 RasterEffectOptions.colorModel

`rasterEffectOptions.colorModel`

Description

The color model for the rasterization.

Default: `RasterizationColorModel.DEFAULTCOLORMODEL`

Type

RasterizationColorModel

136.1.4 RasterEffectOptions.convertSpotColors

`rasterEffectOptions.convertSpotColors`

Description

If `true`, all spot colors are converted to process colors for the image.

Default: `false`

Type

Boolean

136.1.5 RasterEffectOptions.padding

`rasterEffectOptions.padding`

Description

The amount of white space (in points) to be added around the object during rasterization.

Default: .0

Type

Number (double)

136.1.6 RasterEffectOptions.resolution

`rasterEffectOptions.resolution`

Description

The rasterization resolution in dots per inch (dpi). Range: 72.0 to 2400.0.

Default: 300.0

Type

Number (double)

136.1.7 RasterEffectOptions.transparency

`rasterEffectOptions.transparency`

Description

If `true`, the image should use transparency.

Default: `false`

Type

Boolean

CHAPTER 137

RasterItems

`app.activeDocument.rasterItems`

Description

A collection of *RasterItem* objects.

137.1 Properties

137.1.1 RasterItems.length

`app.activeDocument.rasterItems.length`

Description

Number of elements in the collection.

Type

Number, read-only.

137.1.2 RasterItems.parent

`app.activeDocument.rasterItems.parent`

Description

The object's container.

Type

Object, read-only.

137.1.3 RasterItems.typename

```
app.activeDocument.rasterItems.typename
```

Description

The class name of the object.

Type

String, read-only.

137.2 Methods

137.2.1 RasterItems.getByName()

```
app.activeDocument.rasterItems.getByName(name)
```

Description

Get the first element in the collection with the provided name.

Parameters

Parameter	Type	Description
`name`	String	Name of element to get

Returns

SymbolItem

137.2.2 RasterItems.index()

```
app.activeDocument.rasterItems.index(itemKey)
```

Description

Gets an element from the collection.

Parameters

Parameter	Type	Description
`itemKey`	String, Number	String or number key

Returns

SymbolItem

137.2.3 RasterItems.removeAll()

`app.activeDocument.rasterItems.removeAll()`

Description

Deletes all elements in the collection.

Returns

Nothing.

137.3 Example

137.3.1 Creating a raster item

```
// Creates a new raster item in a new document from a raster file
// jpgFilePath contains the full path and file name of a jpg file
function createRasterItem(jpgFilePath) {
  var rasterFile = File(jpgFilePath);
  var myDoc = app.documents.add();

  var myPlacedItem = myDoc.placedItems.add();
  myPlacedItem.file = rasterFile;
  myPlacedItem.position = Array(0, myDoc.height);
  myPlacedItem.embed();
}
```

137.3.2 Finding and examining a raster item

```
// Examines the color space of the first raster item in the document and displays
// result in ESTK console
if (app.documents.length > 0 && app.activeDocument.rasterItems.length > 0) {
  var rasterArt = app.activeDocument.rasterItems[0];

  switch (rasterArt.imageColorSpace) {
    case ImageColorSpace.CMYK:
      $.writeln("The color space of the first raster item is CMYK");
      break;

    case ImageColorSpace.RGB:
      $.writeln("The color space of the first raster item is RGB");
      break;

    case ImageColorSpace.GRAYSCALE:
      $.writeln("The color space of the first raster item is GRAYSCALE");
      break;
  }
}
```

CHAPTER 138

RasterizeOptions

`rasterizeOptions`

Description

Specifies options that may be supplied when rasterizing artwork.

All properties are optional.

138.1 Properties

138.1.1 RasterizeOptions.antiAliasingMethod

`rasterizeOptions.antiAliasingMethod`

Description

The type of antialiasing method.

Default: `AntiAliasingMethod.ARTOPTIMIZED`

Type

AntiAliasingMethod

138.1.2 RasterizeOptions.backgroundBlack

`rasterizeOptions.backgroundBlack`

Description

If `true`, the rasterization is done against a black background (instead of white).

Default: `false`

Type

Boolean

138.1.3 RasterizeOptions.clippingMask

`rasterizeOptions.clippingMask`

Description

If `true`, a clipping mask should be created for the image.

Default: `false`

Type

Boolean

138.1.4 RasterizeOptions.colorModel

`rasterizeOptions.colorModel`

Description

The color model for the rasterization.

Default: `RasterizationColorModel.DEFAULTCOLORMODEL`

Type

RasterizationColorModel

138.1.5 RasterizeOptions.convertSpotColors

`rasterizeOptions.convertSpotColors`

Description

If `true`, spot colors should be converted to process colors for the image.

Default: `false`

Type

Boolean

138.1.6 RasterizeOptions.convertTextToOutlines

`rasterizeOptions.convertTextToOutlines`

Description

If `true`, all text is converted to outlines before rasterization.

Default: `false`

Type

Boolean

138.1.7 RasterizeOptions.includeLayers

`rasterizeOptions.includeLayers`

Description

If `true`, the resulting image incorporates layer attributes (like opacity and blend mode).

Default: `false`

Type

Boolean

138.1.8 RasterizeOptions.padding

`rasterizeOptions.padding`

Description

The amount of white space (in points) to be added around the object during rasterization.

Default: .0

Type

Number (double)

138.1.9 RasterizeOptions.resolution

`rasterizeOptions.resolution`

Description

The rasterization resolution in dots per inch (dpi). Range: 72.0 to 2400.0.

Default: 300.0

Type

Number (double)

138.1.10 RasterizeOptions.transparency

`rasterizeOptions.transparency`

Description

If `true`, the image should use transparency.

Default: `false`

Type

Boolean

CHAPTER 139

Screen

`PPDFileList[index].PPDInfo.screenList[index`

Description

Associates a color separation screen with information to be used for printing.

139.1 Properties

139.1.1 Screen.name

`PPDFileList[index].PPDInfo.screenList[index].name`

Description

The color separation screen name.

Type

String

139.1.2 Screen.screenInfo

`PPDFileList[index].PPDInfo.screenList[index].screenInfo`

Description

The color separation screen information.

Type

ScreenInfo

139.1.3 Screen.typename

`PPDFileList[index].PPDInfo.screenList[index].typename`

Description

The class name of the referenced object.

Type

String; read-only.

CHAPTER 140

ScreenInfo

`PPDFileList[index].PPDInfo.screenList[index].screenInfo`

Description

Contains information about the angle and frequency of the color separation screen to be used for printing.

140.1 Properties

140.1.1 ScreenInfo.angle

`PPDFileList[index].PPDInfo.screenList[index].screenInfo.angle`

Description

The screen's angle in degrees.

Type

Number (double).

140.1.2 ScreenInfo.defaultScreen

`PPDFileList[index].PPDInfo.screenList[index].screenInfo.defaultScreen`

Description

If *true*, it is the default screen.

Type

Boolean.

140.1.3 ScreenInfo.frequency

`PPDFileList[index].PPDInfo.screenList[index].screenInfo.frequency`

Description

The screen's frequency.

Type

Number (double).

140.1.4 ScreenInfo.typename

`PPDFileList[index].PPDInfo.screenList[index].screenInfo.typename`

Description

The class name of the referenced object.

Type

String; read-only.

140.2 Example

140.2.1 Getting screen information

```
// Displays in a new text frame, the name, angle and frequency
// of each screen list item
var sInfo = "";
var docRef = documents.add();
if (PPDFileList.length == 0) {
  sInfo = "\r\t\tEmpty PPDFileList";
} else {
  var ppdRef = PPDFileList[0];
  var ppdInfoRef = ppdRef.PPDInfo;
  sInfo += "\r\t\tScreen Objects for 1st PPD File:\r";
  sInfo += "\t\t" + ppdRef.name;

  var iScreens = ppdInfoRef.screenList.length;
  if (iScreens > 0) {
    for (var c = 0; c < iScreens; c++) {

      var screenRef = ppdInfoRef.screenList[c];
      sInfo += "\r\t\t";
      sInfo += screenRef.name;

      var screenInfoRef = screenRef.screenInfo;

      sInfo += ", Angle = ";
      sInfo += screenInfoRef.angle;
```

(continues on next page)

(continued from previous page)

```
      sInfo += ", Frequency = ";
      sInfo += screenInfoRef.frequency;
      sInfo += "\r";
    }
  } else {
    sInfo += "\r\t\tEmpty ScreenList";
  }
}

var textRef = docRef.textFrames.add();
textRef.textRange.characterAttributes.size = 12;
textRef.contents = sInfo;
textRef.top = 600;
textRef.left = 30;

redraw();
```

CHAPTER 141

ScreenSpotFunction

`PPDFileList[index].PPDInfo.screenSpotFunctionList[index`

Description

Contains information about a color separation screen spot function, including its definition in PostScript language code.

141.1 Properties

141.1.1 ScreenSpotFunction.name

`PPDFileList[index].PPDInfo.screenSpotFunctionList[index].name`

Description

The color separation screen spot function name.

Type

String

141.1.2 ScreenSpotFunction.spotFunction

`PPDFileList[index].PPDInfo.screenSpotFunctionList[index].spotFunction`

Description

The spot function expressed in PostScript commands.

Type

String

141.1.3 ScreenSpotFunction.typename

`PPDFileList[index].PPDInfo.screenSpotFunctionList[index].typename`

Description

The class name of the referenced object.

Type

String; read-only.

141.2 Example

141.2.1 Finding screen spot functions

```
// Displays in a new text frame, the screen spot functions for the 1st PPD file.
var docRef = documents.add();
var sInfo = "";
if (PPDFileList.length == 0) {
  sInfo = "\r\t\tEmpty PPDFileList"
} else {
  var ppdRef = PPDFileList[0];
  var ppdInfoRef = ppdRef.PPDInfo;

  var sInfo = "\r\t\tScreenSpotFunctions for 1st PPD File:\r";
  sInfo += "\t\t" + ppdRef.name + "\r";

  var iScreenSpots = ppdInfoRef.screenSpotFunctionList.length;
  if (iScreenSpots > 0) {
    for (var n = 0; n < iScreenSpots; n++) {
      var screenSpotRef = ppdInfoRef.screenSpotFunctionList[n];
      sInfo += "\t\t";

      sInfo += screenSpotRef.name;
      sInfo += ", spotFunction: ";

      sInfo += screenSpotRef.spotFunction;
      sInfo += "\r";
    }
  } else {
    sInfo += "\t\tEmpty ScreenSpotFunctionList";
  }
}

var textRef = docRef.textFrames.add();
textRef.textRange.characterAttributes.size = 12;
textRef.contents = sInfo;
textRef.top = 600;
```

(continues on next page)

(continued from previous page)

```
textRef.left = 30;

redraw();
```

CHAPTER 142

Spot

`app.activeDocument.spots[index`

Description

A custom color definition contained in a *SpotColor* object.

If no properties are specified when creating a spot, default values are provided.

However, if specifying the color, you must use the same color space as the document, either CMYK or RGB. Otherwise, an error results.

The new spot is added to the end of the swatches list in the Swatches palette.

142.1 Properties

142.1.1 Spot.color

`app.activeDocument.spots[index].color`

Description

The color information for this spot color.

Type

Color

142.1.2 Spot.colorType

`app.activeDocument.spots[index].colorType`

Description

The color model for this custom color.

Type

ColorModel

142.1.3 Spot.name

`app.activeDocument.spots[index].name`

Description

The spot color's name.

Type

String

142.1.4 Spot.parent

`app.activeDocument.spots[index].parent`

Description

The document that contains this spot color.

Type

Document; read-only.

142.1.5 Spot.spotKind

`app.activeDocument.spots[index].spotKind`

Description

The kind of spot color (RGB, CMYK or LAB). This is the name of the color kind contained in the spot object.

Type

SpotColorKind; read-only.

142.1.6 Spot.typename

`app.activeDocument.spots[index].typename`

Description

The class name of the referenced object.

Type

String; read-only.

142.2 Methods

142.2.1 Spot.getInternalColor()

`app.activeDocument.spots[index].getInternalColor()`

Description

Gets the internal color of a spot.

Returns

Color components.

142.2.2 Spot.remove()

`app.activeDocument.spots[index].remove()`

Description

Deletes this object.

Returns

Nothing.

142.3 Example

142.3.1 Creating a new spot color

```
// Creates a new spot color in the current document, then applies an 80% tint to the␣
↪color
if ( app.documents.length > 0 ) {
  var doc = app.activeDocument;

  // Create the new spot
  var newSpot = doc.spots.add();
```

(continues on next page)

(continued from previous page)

```
  // Define the new color value
  var newColor = new CMYKColor();
  newColor.cyan = 35;
  newColor.magenta = 0;
  newColor.yellow = 50;
  newColor.black = 0;

  // Define a new SpotColor with an 80% tint
  // of the new Spot's color. The spot color can then
  // be applied to an art item like any other color.
  newSpot.name = "Pea-Green";
  newSpot.colorType = ColorModel.SPOT;
  newSpot.color = newColor;

  var newSpotColor = new SpotColor();
  newSpotColor.spot = newSpot;
  newSpotColor.tint = 80;
}
```

CHAPTER 143

Spots

`app.activeDocument.spots`

Description

A collection of *SpotColor* objects in a document.

143.1 Properties

143.1.1 Spots.length

`app.activeDocument.spots.length`

Description

Number of elements in the collection.

Type

Number, read-only.

143.1.2 Spots.parent

`app.activeDocument.spots.parent`

Description

The object's container.

Type

Object, read-only.

143.1.3 Spots.typename

`app.activeDocument.spots.typename`

Description

The class name of the object.

Type

String, read-only.

143.2 Methods

143.2.1 Spots.add()

`app.activeDocument.spots.add()`

Description

Creates a new object.

Returns

Spot

143.2.2 Spots.getByName()

`app.activeDocument.spots.getByName(name)`

Description

Get the first element in the collection with the provided name.

Parameters

Parameter	Type	Description
`name`	String	Name of element to get

Returns

Spot

143.2.3 Spots.index()

`app.activeDocument.spots.index(itemKey)`

Description

Gets an element from the collection.

Parameters

Parameter	Type	Description
`itemKey`	String, Number	Key of element to get

Returns

Spot

143.2.4 Spots.removeAll()

`app.activeDocument.spots.removeAll()`

Description

Deletes all elements in the collection.

Returns

Nothing.

143.3 Example

143.3.1 Removing spot colors

```
// Deletes all spots colors from the current document
if ( app.documents.length > 0 ) {
  var spotCount = app.activeDocument.spots.length;

  if (spotCount > 0) {
    app.activeDocument.spots.removeAll();
  }
}
```

143.3.2 Creating and applying spot colors

```
// Defines and applies a new spot color in the current document,
// then applies the color to the first path item
if (app.documents.length > 0 && app.activeDocument.pathItems.length > 0) {
  // Define the new color value
  var newRGBColor = new RGBColor();
  newRGBColor.red = 255;
  newRGBColor.green = 0;
  newRGBColor.blue = 0;

  // Create the new spot
  var newSpot = app.activeDocument.spots.add();

  // Define the new SpotColor as 80% of the RGB color
  newSpot.name = "Scripted Red spot";
  newSpot.tint = 80;
  newSpot.color = newRGBColor;

  // Apply a 50% tint of the new spot color to the frontmost path item.
  // Create a spotcolor object, set the tint value,
  var newSpotColor = new SpotColor();
  newSpotColor.spot = newSpot;
  newSpotColor.tint = 50;

  // Use the spot color to set the fill color
  var frontPath = app.activeDocument.pathItems[0];
  frontPath.filled = true;
  frontPath.fillColor = newSpotColor;
}
```

CHAPTER 144

Story

`story`

Description

A contiguous block of text as specified by a text range. A story can contain one or more text frames; if there is more than one, the multiple text frames are linked together to form a single story.

144.1 Properties

144.1.1 Story.characters

`story.characters`

Description

All the characters in this story.

Type

Characters; read-only.

144.1.2 Story.insertionPoints

`story.insertionPoints`

Description

All the insertion points in this story.

Type

InsertionPoints; read-only.

144.1.3 Story.length

`story.length`

Description

The number of characters in the story.

Type

Number (long); read-only.

144.1.4 Story.lines

`story.lines`

Description

All the lines in this story.

Type

Lines; read-only.

144.1.5 Story.paragraphs

`story.paragraphs`

Description

All the paragraphs in this story.

Type

Paragraphs; read-only.

144.1.6 Story.parent

`story.parent`

Description

The object's container.

Type

Object; read-only.

144.1.7 Story.textFrames

`story.textFrames`

Description

The text frame items in this story.

Type

TextFrameItems; read-only.

144.1.8 Story.textRange

`story.textRange`

Description

The text range of the story.

Type

TextRange; read-only.

144.1.9 Story.textRanges

`story.textRanges`

Description

All the text ranges in the story.

Type

TextRanges; read-only.

144.1.10 Story.textSelection

`story.textSelection`

Description

The selected text ranges in the story.

Type

Array of *TextRange*; read-only.

144.1.11 Story.typename

`story.typename`

Description

The class name of the object.

Type

String; read-only.

144.1.12 Story.words

`story.words`

Description

All the words in the story.

Type

Words; read-only.

144.2 Example

144.2.1 Threading text frames into stories

```
// Creates 1 story that flows through 2 text frames and another story that
// is displayed in a 3rd text frame
// Create a new document and add 2 area TextFrames
var docRef = documents.add();
var itemRef1 = docRef.pathItems.rectangle(600, 200, 50, 30);
var textRef1 = docRef.textFrames.areaText(itemRef1);
textRef1.selected = true;

// create 2nd text frame and link it the first
var itemRef2 = docRef.pathItems.rectangle(550, 300, 50, 200);
var textRef2 = docRef.textFrames.areaText(itemRef2, TextOrientation.HORIZONTAL,␣
↪textRef1);
textRef2.selected = true;

// Add enough text to the 1st TextFrame to
// cause it to flow to the 2nd TextFrame.
textRef1.contents = "This is two text frames linked together as one story";
redraw();

// Create a 3rd text frame and count the stories
var textRef3 = docRef.textFrames.add();
textRef3.contents = "Each unlinked textFrame adds a new story."
textRef3.top = 650;
textRef3.left = 200;

redraw();
```

CHAPTER 145

Stories

`stories`

Description

A collection of *Story* objects in a document.

145.1 Properties

145.1.1 Stories.length

`stories.length`

Description

The number of elements in the collection.

Type

Number; read-only.

145.1.2 Stories.parent

`stories.parent`

Description

The object's container.

Type

Object; read-only.

145.1.3 Stories.typename

`stories.typename`

Description

The class name of the referenced object.

Type

String; read-only.

145.2 Methods

145.2.1 Stories.index()

`stories.index(itemKey)`

Description

Gets an element from the collection.

Parameters

Parameter	Type	Description
`itemKey`	String, Number	String or number key

Returns

Story

CHAPTER 146

Swatch

`app.activeDocument.swatches[index`

Description

A color swatch definition contained in a document. The swatches correspond to the swatch palette in the Illustrator user interface.

A script can create a new swatch.

The swatch can hold all types of color data, such as pattern, gradient, CMYK, RGB, gray, and spot.

146.1 Properties

146.1.1 Swatch.color

`app.activeDocument.swatches[index].color`

Description

The color information for this swatch.

Type

Color

146.1.2 Swatch.name

`app.activeDocument.swatches[index].name`

Description

The swatch's name.

Type

String.

146.1.3 Swatch.parent

`app.activeDocument.swatches[index].parent`

Description

The object that contains this swatch.

Type

Document, read-only.

146.1.4 Swatch.typename

`app.activeDocument.swatches[index].typename`

Description

The class name of the object.

Type

String, read-only.

146.2 Methods

146.2.1 Swatch.remove()

`app.activeDocument.swatches[index].remove()`

Description

Deletes this object.

Returns

Nothing.

146.3 Example

146.3.1 Modifying a swatch

```
// Changes the name of the last swatch
if ( app.documents.length > 0 && app.activeDocument.swatches.length > 0 ) {
  var lastIndex = app.activeDocument.swatches.length - 1;
  var lastSwatch = app.activeDocument.swatches[lastIndex];
  lastSwatch.name = "TheLastSwatch";
}
```

CHAPTER 147

Swatches

`app.activeDocument.swatches`

Description

The collection of *Swatch* objects in the document.

147.1 Properties

147.1.1 Swatches.length

`app.activeDocument.swatches.length`

Description

Number of elements in the collection.

Type

Number, read-only.

147.1.2 Swatches.parent

`app.activeDocument.swatches.parent`

Description

The object's container.

Type

Object, read-only.

147.1.3 Swatches.typename

`app.activeDocument.swatches.typename`

Description

The class name of the object.

Type

String, read-only.

147.2 Methods

147.2.1 Swatches.add()

`app.activeDocument.swatches.add()`

Description

Creates a new *Swatch* object.

Returns

Swatch

147.2.2 Swatches.getByName()

`app.activeDocument.swatches.getByName(name)`

Description

Get the first element in the collection with the provided name.

Parameters

Parameter	Type	Description
`name`	String	Name of element to get

Returns

Swatch

147.2.3 Swatches.getSelected()

`app.activeDocument.swatches.getSelected()`

Description

Gets selected swatches in the document.

Returns

List of *Swatch*

147.2.4 Swatches.index()

```
app.activeDocument.swatches.index(itemKey)
```

Description

Gets an element from the collection.

Parameters

Parameter	Type	Description
`itemKey`	String, Number	Key of element to get

Returns

Swatch

147.2.5 Swatches.removeAll()

```
app.activeDocument.swatches.removeAll()
```

Description

Deletes all elements in the collection.

Returns

Nothing.

147.3 Example

147.3.1 Finding and deleting a swatch

```
// Deletes swatch 4 from the current document
if ( app.documents.length > 0 ) {
  if (app.activeDocument.swatches.length > 4) {
    var swatchToDelete = app.activeDocument.swatches[3];
    swatchToDelete.remove();
  }
}
```

CHAPTER 148

SwatchGroup

`swatchGroup`

Description

A group of *Swatch* objects.

148.1 Properties

148.1.1 SwatchGroup.name

`swatchGroup.name`

Description

The name of the swatch group.

Type

string

148.1.2 SwatchGroup.parent

`swatchGroup.parent`

Description

The object that contains the swatch group.

Type

Object, read-only.

148.1.3 SwatchGroup.typename

`swatchGroup.typename`

Description

The class name of the referenced object.

Type

String, read-only

148.2 Methods

148.2.1 SwatchGroup.addSpot()

`swatchGroup.addSpot(spot)`

Description

Adds a spot swatch to the swatch group.

Parameters

Parameter	Type	Description
`spot`	*Spot*	Spot to add

Returns

Nothing.

148.2.2 SwatchGroup.addSwatch()

`swatchGroup.addSwatch(swatch)`

Description

Adds a swatch to the swatch group.

Parameters

Parameter	Type	Description
`swatch`	*Swatch*	Swatch to add

Returns

Nothing.

148.2.3 SwatchGroup.getAllSwatches()

```
swatchGroup.getAllSwatches()
```

Description

Gets a list of all swatches in the swatch group.

Returns

List of *Swatch*

148.2.4 SwatchGroup.remove()

```
swatchGroup.remove()
```

Description

Deletes this object.

Returns

Nothing.

148.2.5 SwatchGroup.removeAll()

```
swatchGroup.removeAll()
```

Description

Deletes all elements in the collection.

Returns

Nothing.

CHAPTER 149

SwatchGroups

`swatchGroups`

Description

A collection of *SwatchGroup* objects.

149.1 Properties

149.1.1 SwatchGroups.length

`swatchGroups.length`

Description

The number of objects in the collection.

Type

Number, read-only.

149.1.2 SwatchGroups.parent

`swatchGroups.parent`

Description

The object's container.

Type

Object, read-only.

149.1.3 SwatchGroups.typename

`swatchGroups.typename`

Description

The class name of the object.

Type

String, read-only.

149.2 Methods

149.2.1 SwatchGroups.add()

`swatchGroups.add()`

Description

Creates a swatch group.

Returns

SwatchGroup

149.2.2 SwatchGroups.getByName()

`swatchGroups.getByName(name)`

Description

Get the first element in the collection with the provided name.

Parameters

Parameter	Type	Description
`name`	String	Name of element to get

Returns

SwatchGroup

149.2.3 SwatchGroups.removeAll()

`swatchGroups.removeAll()`

Description

Deletes all elements in the collection.

Returns

Nothing.

CHAPTER 150

SymbolItems

`app.activeDocument.symbolItems`

Description

The collection of *SymbolItem* objects in the document.

150.1 Properties

150.1.1 SymbolItems.length

`app.activeDocument.symbolItems.length`

Description

Number of elements in the collection.

Type

Number, read-only.

150.1.2 SymbolItems.parent

`app.activeDocument.symbolItems.parent`

Description

The object's container.

Type

Object, read-only.

150.1.3 SymbolItems.typename

```
app.activeDocument.symbolItems.typename
```

Description

The class name of the object.

Type

String, read-only.

150.2 Methods

150.2.1 SymbolItems.add()

```
app.activeDocument.symbolItems.add(symbol)
```

Description

Creates an instance of the specified symbol.

Parameters

Parameter	Type	Description
`symbol`	*Symbol*	Symbol to instance

Returns

SymbolItem

150.2.2 SymbolItems.getByName()

```
app.activeDocument.symbolItems.getByName(name)
```

Description

Get the first element in the collection with the provided name.

Parameters

Parameter	Type	Description
`name`	String	Name of element to get

Returns

SymbolItem

150.2.3 SymbolItems.index()

`app.activeDocument.symbolItems.index(itemKey)`

Description

Gets an element from the collection.

Parameters

Parameter	Type	Description
`itemKey`	String, Number	String or number key

Returns

SymbolItem

150.2.4 SymbolItems.removeAll()

`app.activeDocument.symbolItems.removeAll()`

Description

Deletes all elements in the collection.

Returns

Nothing.

150.3 Example

150.3.1 Creating a symbol

```
// Creates a path item from each graphic style
// then adds each item as a new symbol

var docRef = documents.add();
var y = 750;
var x = 25;

var iCount = docRef.graphicStyles.length;

for (var i=0; i<iCount; i++) {

  var pathRef = docRef.pathItems.rectangle( y, x, 20, 20 );
  docRef.graphicStyles[i].applyTo(pathRef);

  // are we at bottom?
  if ( (y-=60) <= 60 ) {
    y = 750; // go back to the top.
    x+= 200
```

(continues on next page)

(continued from previous page)

```
  }

  redraw();
  docRef.symbolItems.add(pathRef);
}
```

CHAPTER 151

Symbol

`app.activeDocument.symbols[index`

Description

An art item that is stored in the Symbols palette, and can be reused one or more times in the document without duplicating the art data. Symbols are contained in documents.

Instances of `Symbol` in a document are associated with *SymbolItem* objects, which store the art object properties.

151.1 Properties

151.1.1 Symbol.name

`app.activeDocument.symbols[index].name`

Description

The symbol's name

Type

String.

151.1.2 Symbol.parent

`app.activeDocument.symbols[index].parent`

Description

The object that contains the symbol object.

Type

Object, read-only.

151.1.3 Symbol.typename

```
app.activeDocument.symbols[index].typename
```

Description

The class name of the object.

Type

String, read-only.

151.2 Methods

151.2.1 Symbol.duplicate()

```
app.activeDocument.symbols[index].duplicate()
```

Description

Creates a duplicate of this object.

Returns

Symbol

151.2.2 Symbol.remove()

```
app.activeDocument.symbols[index].remove()
```

Description

Deletes this object.

Returns

Nothing.

CHAPTER 152

Symbols

`app.activeDocument.symbols`

Description

The collection of *Symbol* objects in the document.

152.1 Properties

152.1.1 Symbols.length

`app.activeDocument.symbols.length`

Description

Number of elements in the collection.

Type

Number, read-only.

152.1.2 Symbols.parent

`app.activeDocument.symbols.parent`

Description

The object's container.

Type

Object, read-only.

152.1.3 Symbols.typename

```
app.activeDocument.symbols.typename
```

Description

The class name of the object.

Type

String, read-only.

152.2 Methods

152.2.1 Symbols.add()

```
app.activeDocument.symbols.add(sourceArt[, registrationPoint])
```

Description

Returns a symbol object created from the source art item, any of the following:

- *CompoundPathItems*
- *GroupItems*
- *MeshItems*
- *NonNativeItems*
- *PageItems*
- *PathItems*
- *RasterItems*
- *SymbolItems*
- *TextFrameItems*

The default registration point is `SymbolRegistrationPoint.SYMBOLCENTERPOINT`.

Parameters

Parameter	Type	Description
`sourceArt`	*PageItem*	Source art to create symbol from
`registrationPoint`	*SymbolRegistrationPoint*, optional	Registration point to use

Returns

Symbol

152.2.2 Symbols.getByName()

```
app.activeDocument.symbols.getByName(name)
```

Description

Get the first element in the collection with the provided name.

Parameters

Parameter	Type	Description
`name`	String	Name of element to get

Returns

Symbol

152.2.3 Symbols.index()

```
app.activeDocument.symbols.index(itemKey)
```

Description

Gets an element from the collection.

Parameters

Parameter	Type	Description
`itemKey`	String, Number	String or number key

Returns

Symbol

152.2.4 Symbols.removeAll()

```
app.activeDocument.symbols.removeAll()
```

Description

Deletes all elements in the collection.

Returns

Nothing.

152.3 Example

152.3.1 Creating a symbol

```
// Creates a path item from each graphic style
// then adds each item as a new symbol

var docRef = documents.add();
var y = 750;
var x = 25;

var iCount = docRef.graphicStyles.length;

for (var i=0; i<iCount; i++) {

  var pathRef = docRef.pathItems.rectangle( y, x, 20, 20 );
  docRef.graphicStyles[i].applyTo(pathRef);

  // are we at bottom?
  if ( (y-=60) <= 60 ) {
    y = 750; // go back to the top.
    x+= 200
  }

  redraw();
  docRef.symbols.add(pathRef);
}
```

CHAPTER 153

TabStopInfo

`app.activeDocument.textFrames[index].paragraphs[index].paragraphAttributes.tabStops`

Description

Information about the alignment, position, and other details for a tab stop in a *ParagraphAttributes* object.

153.1 Properties

153.1.1 TabStopInfo.alignment

`app.activeDocument.textFrames[index].paragraphs[index].paragraphAttributes.tabStops.alignment`

Description

The alignment of the tab stop.

Default: `Left`

Type

TabStopAlignment

153.1.2 TabStopInfo.decimalCharacter

`app.activeDocument.textFrames[index].paragraphs[index].paragraphAttributes.tabStops.decimalCharacter`

Description

The character used for decimal tab stops.

Default: .

Type

String

153.1.3 TabStopInfo.leader

```
app.activeDocument.textFrames[index].paragraphs[index].paragraphAttributes.
tabStops.leader
```

Description

The leader dot character.

Type

String

153.1.4 TabStopInfo.position

```
app.activeDocument.textFrames[index].paragraphs[index].paragraphAttributes.
tabStops.position
```

Description

The position of the tab stop expressed in points.

Default: 0.0

Type

Number (double)

153.1.5 TabStopInfo.typename

```
app.activeDocument.textFrames[index].paragraphs[index].paragraphAttributes.
tabStops.typename
```

Description

The class name of the object.

Type

String, read-only.

153.2 Example

153.2.1 Displaying tab stop information

```
// Displays tab stop information found in each text frame
// of current document, if any.

var docRef = app.activeDocument;
var sData = "Tab Stops Found \\rTabStop Leader\t\tTabStop Position\r";
var textRef = docRef.textFrames;

for( var i=0 ; i < textRef.length; i++ ) {
  // Get all paragraphs in the textFrames
  var paraRef = textRef[i].paragraphs;

  for ( p=0 ; p < paraRef.length ; p++ ) {
    // Get para attributes for all textRanges in paragraph
    var attrRef = paraRef[p].paragraphAttributes;
    var tabRef = attrRef.tabStops;

    if ( tabRef.length > 0 ) {
      for(var t=0; t<tabRef.length; t++){
        sData += "\t" + tabRef[t].leader + "\t\t";
        sData += "\t\t" + tabRef[t].position + "\r";
      } // end for
    } // end if
  } // end for
} // end for

var newTF = docRef.textFrames.add();
newTF.contents = sData;
newTF.top = 400;
newTF.left = 100; redraw();
```

CHAPTER 154

Tag

`app.activeDocument.selection[index].tags[index`

Description

A label associated with a specific piece of artwork.

Tags allows you to assign an unlimited number of key-value pairs to any page item in a document.

154.1 Properties

154.1.1 Tag.name

`app.activeDocument.selection[index].tags[index].name`

Description

The tag's name.

Type

String, read-only.

154.1.2 Tag.parent

`app.activeDocument.selection[index].tags[index].parent`

Description

The object that contains this tag.

Type

Object, read-only.

154.1.3 Tag.typename

`app.activeDocument.selection[index].tags[index].typename`

Description

The class name of the object.

Type

String, read-only.

154.1.4 Tag.value

`app.activeDocument.selection[index].tags[index].value`

Description

The data stored in this tag.

Type

String, read-only.

154.2 Methods

154.2.1 Tag.remove()

`app.activeDocument.selection[index].tags[index].remove()`

Description

Deletes this object.

Returns

Nothing.

154.3 Example

154.3.1 Using tags

```
// Finds the tags associated with the selected art item,
// show names and values in a separate document

if ( app.documents.length > 0 ) {
  doc = app.activeDocument;

  if ( doc.selection.length > 0 ) {
    for ( i = 0; i < selection.length; i++ ) {
      selectedArt = selection[0];
```

(continues on next page)

(continued from previous page)

```
      tagList = selectedArt.tags;

      if (tagList.length == 0) {
        var tempTag = tagList.add();
        tempTag.name = "OneWord";
        tempTag.value = "anything you want";
      }

      // Create a document and add a line of text per tag
      reportDocument = app.documents.add();
      top_offset = 400;

      for ( i = 0; i < tagList.length; i++ ) {
        tagText = tagList[i].value;
        newItem = reportDocument.textFrames.add();
        newItem.contents = "Tag: (" + tagList[i].name + " , " + tagText + ")";
        newItem.position = Array(100, top_offset);
        newItem.textRange.size = 24;
        top_offset = top_offset - 20;
      }
    }
  }
}
```

CHAPTER 155

Tags

`app.activeDocument.selection[index].tags`

Description

A collection of *Tag* objects.

155.1 Properties

155.1.1 Tags.length

`app.activeDocument.selection[index].tags.length`

Description

The number of elements in the collection.

Type

Number; read-only.

155.1.2 Tags.parent

`app.activeDocument.selection[index].tags.parent`

Description

The object's container.

Type

Object; read-only.

155.1.3 Tags.typename

`app.activeDocument.selection[index].tags.typename`

Description

The class name of the referenced object.

Type

String; read-only.

155.2 Methods

155.2.1 Tags.add()

`app.activeDocument.selection[index].tags.add()`

Description

Creates a new *Tag* object.

Returns

Tag

155.2.2 Tags.getByName()

`app.activeDocument.selection[index].tags.getByName(name)`

Description

Get the first element in the collection with the provided name.

Parameters

Parameter	Type	Description
`name`	String	Name of element to get

Returns

Tag

155.2.3 Tags.index()

`app.activeDocument.selection[index].tags.index(itemKey)`

Description

Gets an element from the collection.

Parameters

Parameter	Type	Description
`itemKey`	String, Number	String or number key

Returns

Tag

155.2.4 Tags.removeAll()

`app.activeDocument.selection[index].tags.removeAll()`

Description

Deletes all elements in this collection.

Returns

Nothing.

155.3 Example

155.3.1 Setting tag values

```
// Adds tags to all RasterItems and PlacedItems in the current document
if ( app.documents.length > 0 ) {
  var doc = app.activeDocument;

  if ( doc.placedItems.length + doc.rasterItems.length > 0 ) {
    for ( i = 0; i < doc.pageItems.length; i++ ) {
      var imageArt = doc.pageItems[i];

      if ( imageArt.typename == "PlacedItem" || imageArt.typename == "RasterItem") {
        // Create a new Tag with the name AdobeURL and the
        // value of the www link

        var urlTAG = imageArt.tags.add();
        urlTAG.name = "AdobeWebSite";
        urlTAG.value = "http://www.adobe.com/";
      }
    }
  } else {
```

(continues on next page)

(continued from previous page)

```
    alert( "No placed or raster items in the document" );
  }
}
```

CHAPTER 156

TextFont

`app.textFonts[index`

Description

Information about a font in the document, found in a *CharacterAttributes* object.

156.1 Properties

156.1.1 TextFont.family

`app.textFonts[index].family`

Description

The font's family name.

Type

String, read-only.

156.1.2 TextFont.name

`app.textFonts[index].name`

Description

The font's full name.

Type

String, read-only.

156.1.3 TextFont.parent

`app.textFonts[index].parent`

Description

The object's container.

Type

Object, read-only.

156.1.4 TextFont.style

`app.textFonts[index].style`

Description

The font's style name.

Type

String, read-only.

156.1.5 TextFont.typename

`app.textFonts[index].typename`

Description

The class name of the object.

Type

String, read-only.

156.2 Example

156.2.1 Setting the font of text

```
// Sets the font of all the text in the document to the first font
if ( app.documents.length > 0 ) {

// Iterate through all text art and apply font 0
  for ( i = 0; i< app.activeDocument.textFrames.length; i++) {
    textArtRange = app.activeDocument.textFrames[i].textRange;
    textArtRange.characterAttributes.textFont = app.textFonts[0];
  }
}
```

CHAPTER 157

TextFonts

`app.textFonts`

Description

A collection of *TextFont* objects.

157.1 Properties

157.1.1 TextFonts.length

`app.textFonts.length`

Description

The number of elements in the collection.

Type

Number; read-only.

157.1.2 TextFonts.parent

`app.textFonts.parent`

Description

The object's container.

Type

Object; read-only.

157.1.3 TextFonts.typename

`app.textFonts.typename`

Description

The class name of the referenced object.

Type

String; read-only.

157.2 Methods

157.2.1 TextFonts.getByName()

`app.textFonts.getByName(name)`

Description

Get the first element in the collection with the provided name.

Parameters

Parameter	Type	Description
`name`	String	Name of element to get

Returns

TextFont

157.2.2 TextFonts.index()

`app.textFonts.index(itemKey)`

Description

Gets an element from the collection.

Parameters

Parameter	Type	Description
`itemKey`	String, Number	String or number key

Returns

TextFont

157.3 Example

157.3.1 Finding fonts

```
// Creates a new A3 sized document and display a list of available fonts until the document is full.

var edgeSpacing = 10;
var columnSpacing = 230;
var docPreset = new DocumentPreset;
docPreset.width = 1191.0;
docPreset.height = 842.0

var docRef = documents.addDocument(DocumentColorSpace.CMYK, docPreset);
var sFontNames = "";
var x = edgeSpacing;
var y = (docRef.height - edgeSpacing);

var iCount = textFonts.length;

for (var i=0; i<iCount; i++) {
 sFontName = textFonts[i].name;
 sFontName += " ";
 sFontNames = sFontName + textFonts[i].style;

 var textRef = docRef.textFrames.add();
 textRef.textRange.characterAttributes.size = 10;
 textRef.contents = sFontNames;
 textRef.top = y;
 textRef.left = x;

 // check wether the text frame will go off the edge of the document
 if ((x + textRef.width)> docRef.width) {
   textRef.remove();
   iCount = i;
   break;
 } else {
   // display text frame
   textRef.textRange.characterAttributes.textFont = textFonts.getByName(textFonts[i].name);
   redraw();

   if ((y-=(textRef.height)) <= 20) {
     y = (docRef.height - edgeSpacing);
     x += columnSpacing;
   }
 }
}
```

CHAPTER 158

TextFrameItems

`app.activeDocument.textFrames`

Description

The collection of *TextFrameItem* objects in the document.

158.1 Properties

158.1.1 TextFrameItems.length

`app.activeDocument.textFrames.length`

Description

The number of elements in the collection.

Type

Number; read-only.

158.1.2 TextFrameItems.parent

`app.activeDocument.textFrames.parent`

Description

The object's container.

Type

Object; read-only.

158.1.3 TextFrameItems.typename

```
app.activeDocument.textFrames.typename
```

Description

The class name of the referenced object.

Type

String; read-only.

158.2 Methods

158.2.1 TextFrameItems.add()

```
app.activeDocument.textFrames.add()
```

Description

Creates a point text frame item.

Returns

TextFrameItem

158.2.2 TextFrameItems.areaText()

```
app.activeDocument.textFrames.areaText(textPath[, orientation][, baseFrame][,
postFix])
```

Description

Creates an area text frame item.

Parameters

Parameter	Type	Description
`textPath`	*PathItem*	Path item to use
`orientation`	*TextOrientation*, optional	Orientation of text
`baseFrame`	*TextFrameItem*, optional	Text frame to use
`postFix`	Boolean, optional	Whether to prefix or postfix the text frame

Returns

TextFrameItem

158.2.3 TextFrameItems.getByName()

```
app.activeDocument.textFrames.getByName(name)
```

Description

Get the first element in the collection with the provided name.

Parameters

Parameter	Type	Description
`name`	String	Name of element to get

Returns

TextFrameItem

158.2.4 TextFrameItems.index()

```
app.activeDocument.textFrames.index(itemKey)
```

Description

Gets an element from the collection.

Parameters

Parameter	Type	Description
`itemKey`	String, Number	String or number key

Returns

TextFrameItem

158.2.5 TextFrameItems.pathText()

```
app.activeDocument.textFrames.pathText(textPath[,startTValue][,endTValue][,
orientation][, baseFrame][, postFix])
```

Description

Creates an on-path text frame item.

Parameters

Parameter	Type	Description
`textPath`	*PathItem*	Path item to use
`startTValue`	Number (double)	Start position of text along the path
`endTValue`	Number (double)	End position of text along the path
`orientation`	*TextOrientation*, optional	Orientation of text
`baseFrame`	*TextFrameItem*, optional	Text frame to use
`postFix`	Boolean, optional	Whether to prefix or postfix the text frame

Returns

TextFrameItem

158.2.6 TextFrameItems.pointText()

```
app.activeDocument.textFrames.pointText(anchor[, orientation])
```

Description

Creates a point text frame item.

Parameters

Parameter	Type	Description
`anchor`	Array of 2 numbers	Point text anchor
`orientation`	*TextOrientation*, optional	Orientation of text

Returns

TextFrameItem

158.2.7 TextFrameItems.removeAll()

```
app.activeDocument.textFrames.removeAll()
```

Description

Deletes all elements in this collection.

Returns

Nothing.

158.3 Example

158.3.1 Creating and modifying text frames

```
// Creates a document with text frames displaying path, area and point
// text, changes the content of each frame then deletes the 2nd frame

// create a new document
var docRef = documents.add();

// create 3 new textFrames (area, line, point)
// Area Text
var rectRef = docRef.pathItems.rectangle(700, 50, 100, 100); var areaTextRef = docRef.
↪textFrames.areaText(rectRef); areaTextRef.contents = "TextFrame #1";
areaTextRef.selected = true;
```

(continues on next page)

(continued from previous page)

```
// Line Text
var lineRef = docRef.pathItems.add();
lineRef.setEntirePath( Array(Array(200, 700), Array(300, 550) ) ); var pathTextRef =␣
↪docRef.textFrames.pathText(lineRef); pathTextRef.contents = "TextFrame #2";
pathTextRef.selected = true;

// Point Text
var pointTextRef = docRef.textFrames.add(); pointTextRef.contents = "TextFrame #3";␣
↪pointTextRef.top = 700;
pointTextRef.left = 400; pointTextRef.selected = true; redraw();

// count the TextFrames
var iCount = docRef.textFrames.length;
var sText = "There are " + iCount + " TextFrames.\r" sText += "Changing contents of␣
↪each TextFrame.";

// change the content of each docRef.textFrames[0].contents = "Area TextFrame.";␣
↪docRef.textFrames[1].contents = "Path TextFrame."; docRef.textFrames[2].contents =
↪"Point TextFrame."; redraw();
docRef.textFrames[1].remove(); redraw();

// count again

var iCount = docRef.textFrames.length;
```

CHAPTER 159

TextPath

`textPath`

Description

A path or list of paths for area or path text. A path consists of path points that define its geometry.

159.1 Properties

159.1.1 TextPath.area

`textPath.area`

Description

The area of this path in square points. If the area is negative, the path is wound counterclockwise.

Self-intersecting paths can contain sub-areas that cancel each other out, which makes this value zero even though the path has apparent area.

Type

Number (double), read-only.

159.1.2 TextPath.blendingMode

`textPath.blendingMode`

Description

The blend mode used when compositing an object.

Type

BlendModes

159.1.3 TextPath.clipping

`textPath.clipping`

Description

If true, this path should be used as a clipping path.

Type

Boolean

159.1.4 TextPath.closed

`textPath.closed`

Description

If true, this path is closed.

Type

Boolean

159.1.5 TextPath.editable

`textPath.editable`

Description

Read-only. If true, this item is editable.

Type

Boolean

159.1.6 TextPath.evenodd

`textPath.evenodd`

Description

If true, the even-odd rule should be used to determine insideness.

Type

Boolean

159.1.7 TextPath.fillColor

`textPath.fillColor`

Description

The fill color of the path.

Type

Color

159.1.8 TextPath.filled

`textPath.filled`

Description

If true, the path be filled.

Type

Boolean

159.1.9 TextPath.fillOverprint

`textPath.fillOverprint`

Description

If true, the art beneath a filled object should be overprinted.

Type

Boolean

159.1.10 TextPath.guides

`textPath.guides`

Description

If true, this path is a guide object.

Type

Boolean

159.1.11 TextPath.height

`textPath.height`

Description

The height of the group item.

Type

Number (double)

159.1.12 TextPath.left

`textPath.left`

Description

The position of the left side of the item (in points, measured from the left side of the page).

Type

Number (double)

159.1.13 TextPath.note

`textPath.note`

Description

The note text assigned to the path.

Type

String

159.1.14 TextPath.opacity

`textPath.opacity`

Description

The opacity of the object. Range: 0.0 to 100.0

Type

Number (double)

159.1.15 TextPath.parent

`textPath.parent`

Description

Read-only. The parent of this object.

Type

Layer or *GroupItem*

159.1.16 TextPath.pathPoints

`textPath.pathPoints`

Description

Read-only. The path points contained in this path item.

Type

PathPoints

159.1.17 TextPath.polarity

`textPath.polarity`

Description

The polarity of the path.

Type

PolarityValues

159.1.18 TextPath.position

`textPath.position`

Description

The position (in points) of the top left corner of the textPathItem object in the format [x, y]. Does not include stroke weight.

Type

Array of 2 numbers

159.1.19 TextPath.resolution

`textPath.resolution`

Description

The resolution of the path in dots per inch (dpi).

Type

Number (double)

159.1.20 TextPath.selectedPathPoints

`textPath.selectedPathPoints`

Description

Read-only. All of the selected path points in the path.

Type

PathPoints

159.1.21 TextPath.strokeCap

`textPath.strokeCap`

Description

The type of line capping.

Type

StrokeCap

159.1.22 TextPath.strokeColor

`textPath.strokeColor`

Description

The stroke color for the path.

Type

Color

159.1.23 TextPath.stroked

`textPath.stroked`

Description

If true, the path should be stroked.

Type

Boolean

159.1.24 TextPath.strokeDashes

`textPath.strokeDashes`

Description

Dash lengths. Set to an empty object, {}, for a solid line.

Type

Object

159.1.25 TextPath.strokeDashOffset

`textPath.strokeDashOffset`

Description

The default distance into the dash pattern at which the pattern should be started.

Type

Number (double)

159.1.26 TextPath.strokeJoin

`textPath.strokeJoin`

Description

Type of joints for the path.

Type

StrokeJoin

159.1.27 TextPath.strokeMiterLimit

`textPath.strokeMiterLimit`

Description

When a default stroke join is set to mitered, this property specifies when the join will be converted to beveled (squared-off) by default. The default miter limit of 4 means that when the length of the point reaches four times the stroke weight, the join switches from a miter join to a bevel join. A value of 1 specifies a bevel join. Range: 1 to 500. Default: 4

Type

Number (double)

159.1.28 TextPath.strokeOverprint

`textPath.strokeOverprint`

Description

If true, the art beneath a stroked object should be overprinted.

Type

Boolean

159.1.29 TextPath.strokeWidth

`textPath.strokeWidth`

Description

Width of the stroke.

Type

Number (double)

159.1.30 TextPath.top

`textPath.top`

Description

The position of the top of the item (in points, measured from the bottom of the page).

Type

Number (double)

159.1.31 TextPath.typename

`textPath.typename`

Description

Read-only. The class name of the referenced object.

Type

String

159.1.32 TextPath.width

`textPath.width`

Description

The width of the item.

Type

Number (double)

159.2 Methods

159.2.1 TextPath.setEntirePath()

`textPath.setEntirePath(pathPoints)`

Description

Sets the path using the array of points specified as [x, y] coordinate pairs.

Parameters

Parameter	Type	Description
`pathPoints`	Array of [x, y] coordinate pairs	Path points to set path as

Returns

Nothing.

159.3 Example

CHAPTER 160

TextRange

`app.activeDocument.textFrames[index].textRange`

Description

A range of text in a specific text art item. TextRange gives you access to the text contained in text art items.

160.1 Properties

160.1.1 TextRange.characterAttributes

`app.activeDocument.textFrames[index].textRange.characterAttributes`

Description

The character properties for the text range.

Type

CharacterAttributes, read-only.

160.1.2 TextRange.characterOffset

`app.activeDocument.textFrames[index].textRange.characterOffset`

Description

Offset of the first character.

Type

Number (long)

160.1.3 TextRange.characters

`app.activeDocument.textFrames[index].textRange.characters`

Description

All the characters in this text range.

Type

Characters, read-only.

160.1.4 TextRange.characterStyles

`app.activeDocument.textFrames[index].textRange.characterStyles`

Description

All referenced character styles in the text range.

Type

CharacterStyles, read-only.

160.1.5 TextRange.contents

`app.activeDocument.textFrames[index].textRange.contents`

Description

The text string.

Type

String

160.1.6 TextRange.end

`app.activeDocument.textFrames[index].textRange.end`

Description

End index of the text range.

Type

Int32

160.1.7 TextRange.insertionPoints

`app.activeDocument.textFrames[index].textRange.insertionPoints`

Description

All the insertion points in this text range.

Type

InsertionPoints, read-only.

160.1.8 TextRange.kerning

`app.activeDocument.textFrames[index].textRange.kerning`

Description

Controls the spacing between two characters, in thousandths of an em. An integer.

Type

Number (long)

160.1.9 TextRange.length

`app.activeDocument.textFrames[index].textRange.length`

Description

The length (in characters). Minimum: 0

Type

Number (long)

160.1.10 TextRange.lines

`app.activeDocument.textFrames[index].textRange.lines`

Description

All the lines in this text range.

Type

Lines, read-only.

160.1.11 TextRange.paragraphAttributes

```
app.activeDocument.textFrames[index].textRange.paragraphAttributes
```

Description

The paragraph properties for the text range.

Type

ParagraphAttributes, read-only.

160.1.12 TextRange.paragraphs

```
app.activeDocument.textFrames[index].textRange.paragraphs
```

Description

All the paragraphs in this text range.

Type

Paragraphs, read-only.

160.1.13 TextRange.paragraphStyles

```
app.activeDocument.textFrames[index].textRange.paragraphStyles
```

Description

All referenced paragraph styles in the text range.

Type

ParagraphStyles, read-only.

160.1.14 TextRange.parent

```
app.activeDocument.textFrames[index].textRange.parent
```

Description

The object's container.

Type

TextRange, read-only.

160.1.15 TextRange.start

`app.activeDocument.textFrames[index].textRange.start`

Description

Start index of the text range.

Type

Int32

160.1.16 TextRange.story

`app.activeDocument.textFrames[index].textRange.story`

Description

The story to which the text range belongs.

Type

Story, read-only.

160.1.17 TextRange.textRanges

`app.activeDocument.textFrames[index].textRange.textRanges`

Description

All of the text in this text range.

Type

TextRanges, read-only.

160.1.18 TextRange.textSelection

`app.activeDocument.textFrames[index].textRange.textSelection`

Description

The selected text ranges in the text range.

Type

Array of *TextRange*, read-only.

160.1.19 TextRange.typename

```
app.activeDocument.textFrames[index].textRange.typename
```

Description

The class name of the object.

Type

String, read-only.

160.1.20 TextRange.words

```
app.activeDocument.textFrames[index].textRange.words
```

Description

All the words contained in this text range.

Type

Words, read-only.

160.2 Methods

160.2.1 TextRange.changeCaseTo()

```
app.activeDocument.textFrames[index].textRange.changeCaseTo(type)
```

Description

Changes the capitalization of text

Parameters

Parameter	Type	Description
`type`	*CaseChangeType*	Capitalization case to change to

Returns

Nothing

160.2.2 TextRange.deSelect()

```
app.activeDocument.textFrames[index].textRange.deSelect()
```

Description

Deselects the text range.

Returns

Nothing.

160.2.3 TextRange.duplicate()

```
app.activeDocument.textFrames[index].textRange.duplicate([relativeObject][,
insertionLocation])
```

Description

Creates a duplicate of this object.

Parameters

Parameter	Type	Description
`relativeObject`	Object, optional	Object to duplicate to
`insertionLocation`	*ElementPlacement*, optional	Location to insert element

Returns

TextRange

160.2.4 TextRange.getLocalCharOverridesJSON()

```
app.activeDocument.textFrames[index].textRange.getLocalCharOverridesJSON()
```

Description

Gets json representation of character overrides.

Returns

String

160.2.5 TextRange.getLocalParaOverridesJSON()

```
app.activeDocument.textFrames[index].textRange.getLocalParaOverridesJSON()
```

Description

Gets json representation of paragraph overrides.

Returns

String

160.2.6 TextRange.getParagraphLength()

```
app.activeDocument.textFrames[index].textRange.getParagraphLength()
```

Description

Gets the length of the first paragraph of the text range.

Returns

Int32

160.2.7 TextRange.getTextRunLength()

```
app.activeDocument.textFrames[index].textRange.getTextRunLength()
```

Description

Gets the length of the first text run of the text range.

Returns

Int32

160.2.8 TextRange.move()

```
app.activeDocument.textFrames[index].textRange.move(relativeObject,
insertionLocation)
```

Description

Moves the object.

Parameters

Parameter	Type	Description
`relativeObject`	Object	Object to move element within
`insertionLocation`	*ElementPlacement*, optional	Location to move element to

Returns

TextRange

160.2.9 TextRange.remove()

```
app.activeDocument.textFrames[index].textRange.remove()
```

Description

Deletes the object.

Returns

Nothing

160.2.10 TextRange.select()

`app.activeDocument.textFrames[index].textRange.select([addToDocument])`

Description

Selects the text range.

Parameters

Parameter	Type	Description
`addToDocument`	Boolean, optional	Whether to add or replace current selection

Returns

Nothing

160.3 Example

160.3.1 Manipulating Text

```
// Changes size of the first character of each word in the
// current document by changing the size attribute of each character

if ( app.documents.length > 0 ) {
  for ( i = 0; i < app.activeDocument.textFrames.length; i++ ) {
    var text = app.activeDocument.textFrames[i].textRange;
    for ( j = 0 ; j < text.words.length; j++ ) {
      //each word is a textRange object
      var textWord = text.words[j];

      // Characters are textRanges too.
      // Get the first character of each word and increase it's size.

      var firstChars = textWord.characters[0];
      firstChars.size = firstChars.size * 1.5;
    }
  }
}
```

CHAPTER 161

TextRanges

`textRanges`

Description

A collection of *TextRange* objects.

161.1 Properties

161.1.1 TextRanges.length

`textRanges.length`

Description

The number of elements in the collection.

Type

Number; read-only.

161.1.2 TextRanges.parent

`textRanges.parent`

Description

The object's container.

Type

Object; read-only.

161.1.3 TextRanges.typename

`textRanges.typename`

Description

The class name of the referenced object.

Type

String; read-only.

161.2 Methods

161.2.1 TextRanges.index()

`textRanges.index(itemKey)`

Description

Gets an element from the collection.

Parameters

Parameter	Type	Description
`itemKey`	String, Number	String or number key

Returns

Variable

161.2.2 TextRanges.removeAll()

`textRanges.removeAll()`

Description

Deletes all elements in this collection.

Returns

Nothing.

CHAPTER 162

TracingObject

`TracingObject`

Description

A tracing object, which associates source raster art item with a vector-art plug-in group created by tracing. Scripts can initiate tracing using *PlacedItem.trace* or *RasterItem.trace()*.

The resulting PluginItem object represents the vector art group, and has this object in its tracing property.

A script can force the tracing operation by calling *Application.redraw()*. The operation is asynchronous, so a script should call `redraw` after creating the tracing object, but before accessing its properties or expanding the tracing to convert it to an art item group.

The read-only properties that describe the tracing result have valid values only after the first tracing operation completes. A value of 0 indicates that the operation has not yet been completed.

162.1 Properties

162.1.1 TracingObject.anchorCount

`tracingObject.anchorCount`

Description

The number of anchors in the tracing result.

Type

Number (long); read-only.

162.1.2 TracingObject.areaCount

`tracingObject.areaCount`

Description

The number of areas in the tracing result.

Type

Number (long); read-only.

162.1.3 TracingObject.imageResolution

`tracingObject.imageResolution`

Description

The resolution of the source image in pixels per inch.

Type

Number (real); read-only.

162.1.4 TracingObject.parent

`tracingObject.parent`

Description

The object's container.

Type

Object; read-only.

162.1.5 TracingObject.pathCount

`tracingObject.pathCount`

Description

The number of paths in the tracing result.

Type

Number (long); read-only.

162.1.6 TracingObject.sourceArt

`tracingObject.sourceArt`

Description

The raster art used to create the associated vector art plug-in group.

Type

PlacedItem or *RasterItem*

162.1.7 TracingObject.tracingOptions

`tracingObject.tracingOptions`

Description

The options used to convert the raster artwork to vector art.

Type

TracingOptions

162.1.8 TracingObject.typename

`tracingObject.typename`

Description

The class name of the object.

Type

String; read-only.

162.1.9 TracingObject.usedColorCount

`tracingObject.usedColorCount`

Description

The number of colors used in the tracing result.

Type

Number (long); read-only.

162.2 Methods

162.2.1 TracingObject.expandTracing()

```
tracingObject.expandTracing([viewed])
```

Description

Converts the vector art into a new group item. The new GroupItem object replaces the PluginItem object in the document.

By default, `viewed` is `false`, and the new group contains only the tracing result (the filled or stroked paths).

If `viewed` is `true`, the new group retains additional information that was specified for the viewing mode, such as outlines and overlays.

Deletes this object and its associated *PluginItem* object. Any group-level attributes that were applied to the plug-in item are applied to the top level of the new group item.

Parameters

Parameter	Type	Description
`viewed`	Boolean, optional	todo

Returns

GroupItem

162.2.2 TracingObject.releaseTracing()

```
tracingObject.releaseTracing()
```

Description

Reverts the artwork in the document to the original source raster art and removes the traced vector art. Returns the original object used to create the tracing, and deletes this object and its associated PluginItem object.

Parameters

Returns

PlacedItem or *RasterItem*

CHAPTER 163

TracingOptions

`image.tracing.tracingOptions`

Description

A set of options used in converting raster art to vector art by tracing.

163.1 Properties

163.1.1 TracingOptions.cornerAngle

`image.tracing.tracingOptions.cornerAngle`

Description

The sharpness, in degrees of a turn in the original image that is considered a corner in the tracing result path.

Range: 0 to 180

Type

Number (double)

163.1.2 TracingOptions.fills

`image.tracing.tracingOptions.fills`

Description

If `true`, trace with fills. At least one of `fills` or `strokes` must be true.

Type

Boolean

163.1.3 TracingOptions.ignoreWhite

`image.tracing.tracingOptions.ignoreWhite`

Description

If `true`, ignores white fill color.

Type

Boolean

163.1.4 TracingOptions.livePaintOutput

`image.tracing.tracingOptions.livePaintOutput`

Description

If `true`, result is LivePaint art. If `false`, it is classic art.

Note: A script should only set this value in preparation for a subsequent expand operation. Leaving a tracing on the artboard when this property is `true` can lead to unexpected application behavior.

Type

Boolean

163.1.5 TracingOptions.maxColors

`image.tracing.tracingOptions.maxColors`

Description

The maximum number of colors allowed for automatic palette generation.

Used only if `tracingMode` is `TracingModeType.TRACINGMODECOLOR` or `TracingModeType.TRACINGMODEGRAY`.

Range: 2 to 256

Type

Number (long)

163.1.6 TracingOptions.maxStrokeWeight

`image.tracing.tracingOptions.maxStrokeWeight`

Description

The maximum stroke weight, when `strokes` is true.

Range: 0.01 to 100.0

Type

Number (double)

163.1.7 TracingOptions.minArea

`image.tracing.tracingOptions.minArea`

Description

The smallest feature, in square pixels, that is traced.

For example, if it is 4, a feature of 2 pixels wide by 2 pixels high is traced.

Type

Number (long)

163.1.8 TracingOptions.minStrokeLength

`image.tracing.tracingOptions.minStrokeLength`

Description

The minimum length in pixels of features in the original image that can be stroked, when `strokes` is `true`.

Smaller features are omitted. Range: 0.0 to 200.0. Default: 20.0

Type

Number (double)

163.1.9 TracingOptions.outputToSwatches

`image.tracing.tracingOptions.outputToSwatches`

Description

If `true`, named colors (swatches) are generated for each new color created by the tracing result.

Used only if `tracingMode` is `TracingModeType.TRACINGMODECOLOR` or `TracingModeType.TRACINGMODEGRAY`.

Type

Boolean

163.1.10 TracingOptions.palette

`image.tracing.tracingOptions.palette`

Description

The name of a color palette to use for tracing. If the empty string, use the automatic palette.

Used only if `tracingMode` is `TracingModeType.TRACINGMODECOLOR` or `TracingModeType.TRACINGMODEGRAY`.

Type

String

163.1.11 TracingOptions.parent

`image.tracing.tracingOptions.parent`

Description

The object's container.

Type

Object, read-only.

163.1.12 TracingOptions.pathFitting

`image.tracing.tracingOptions.pathFitting`

Description

The distance between the traced shape and the original pixel shape. Lower values create a tighter path fitting.

Higher values create a looser path fitting. Range: 0.0 to 10.0

Type

Number (double)

163.1.13 TracingOptions.preprocessBlur

`image.tracing.tracingOptions.preprocessBlur`

Description

The amount of blur used during preprocessing, in pixels. Blurring helps reduce small artifacts and smooth jagged edges in the tracing result. Range: 0.0 to 2.0

Type

Number (double)

163.1.14 TracingOptions.preset

`image.tracing.tracingOptions.preset`

Description

The name of a preset file containing these options.

Type

String, read-only.

163.1.15 TracingOptions.resample

`image.tracing.tracingOptions.resample`

Description

If `true`, resample when tracing. (This setting is not captured in a preset file.)

Always `true` when the raster source art is placed or linked.

Type

Boolean

163.1.16 TracingOptions.resampleResolution

`image.tracing.tracingOptions.resampleResolution`

Description

The resolution to use when resampling in pixels per inch (ppi).

Lower resolution increases the speed of the tracing operation. (This setting is not captured in a preset file.)

Type

Number (double)

163.1.17 TracingOptions.strokes

`image.tracing.tracingOptions.strokes`

Description

If `true`, trace with strokes. At least one of fills or strokes must be true.

Used only if `tracingMode` is `TracingModeType.TRACINGMODEBLACKANDWHITE`.

Type

Boolean

163.1.18 TracingOptions.threshold

`image.tracing.tracingOptions.threshold`

Description

The threshold value of black-and-white tracing. All pixels with a grayscale value greater than this are converted to black.

Used only if `tracingMode` is `TracingModeType.TRACINGMODEBLACKANDWHITE`.

Range: 0 to 255

Type

Number (long)

163.1.19 TracingOptions.tracingMode

`image.tracing.tracingOptions.tracingMode`

Description

The color mode for tracing.

Type

TracingModeType

163.1.20 TracingOptions.typename

`image.tracing.tracingOptions.typename`

Description

Read-only. The class name of the object.

Type

String

163.1.21 TracingOptions.viewRaster

`image.tracing.tracingOptions.viewRaster`

Description

The view for previews of the raster image. (This setting is not captured in a preset file.)

Type

ViewRasterType

163.1.22 TracingOptions.viewVector

`image.tracing.tracingOptions.viewVector`

Description

The view for previews of the vector result. (This setting is not captured in a preset file.)

Type

ViewVectorType

163.2 Methods

163.2.1 TracingOptions.loadFromPreset()

`image.tracing.tracingOptions.loadFromPreset(parameter)`

Description

Loads a set of options from the specified preset, as found in the `Application.tracingPresetList` array.

Parameters

Parameter	Type	Description
`presetName`	String	Preset name to load

Returns

Boolean

163.2.2 TracingOptions.storeToPreset()

`image.tracing.tracingOptions.storeToPreset(parameter)`

Description

Saves this set of options in the specified preset.

Use a name found in the `Application.tracingPresetList` array, or a new name to create a new preset.

For an existing preset, overwrites an unlocked preset and returns `true`.

Returns `false` if the preset is locked.

Parameters

Parameter	Type	Description
`presetName`	String	Preset name to save as

Returns

Boolean

CHAPTER 164

Variable

`app.activeDocument.variables[index`

Description

A document-level variable that can be imported or exported.

A variable is a dynamic object used to create data-driven graphics.

For an example, see *Dataset*.

Variables are accessed in Illustrator through the Variables palette.

164.1 Properties

164.1.1 Variable.kind

`app.activeDocument.variables[index].kind`

Description

The variable's type.

Type

VariableKind

164.1.2 Variable.name

`app.activeDocument.variables[index].name`

Description

The name of the variable.

Type

string

164.1.3 Variable.pageItems

`app.activeDocument.variables[index].pageItems`

Description

All of the artwork in the variable.

Type

PageItems, read-only

164.1.4 Variable.parent

`app.activeDocument.variables[index].parent`

Description

Read-only. The object that contains the variable.

Type

Object

164.1.5 Variable.typename

`app.activeDocument.variables[index].typename`

Description

The class name of the referenced object.

Type

String, read-only

164.2 Methods

164.2.1 Variable.remove()

```
app.activeDocument.variables[index].remove()
```

Description

Removes the variable from the collection of variables.

Returns

Nothing.

CHAPTER 165

Variables

`app.activeDocument.variables`

Description

The collection of *Variable* objects in the document.

For an example of how to create variables, see *Using variables and datasets*.

165.1 Properties

165.1.1 Variables.length

`app.activeDocument.variables.length`

Description

The number of variables in the document

Type

Number; read-only.

165.1.2 Variables.parent

`app.activeDocument.variables.parent`

Description

The object that contains the collection of variables

Type

Object; read-only.

165.1.3 Variables.typename

`app.activeDocument.variables.typename`

Description

The class name of the referenced object.

Type

String; read-only.

165.2 Methods

165.2.1 Variables.add()

`app.activeDocument.variables.add()`

Description

Adds a new variable to the collection.

Returns

Variable

165.2.2 Variables.getByName()

`app.activeDocument.variables.getByName(name)`

Description

Get the first element in the collection with the provided name.

Parameters

Parameter	Type	Description
`name`	String	Name of element to get

Returns

Variable

165.2.3 Variables.index()

```
app.activeDocument.variables.index(itemKey)
```

Description

Gets an element from the collection.

Parameters

Parameter	Type	Description
`itemKey`	String, Number	String or number key

Returns

Variable

165.2.4 Variables.removeAll()

```
app.activeDocument.variables.removeAll()
```

Description

Deletes all elements in this collection.

Returns

Nothing.

CHAPTER 166

View

`app.activeDocument.views[index`

Description

A document view in an Illustrator document, which represents a window view onto a document.

Scripts cannot create new views, but can modify some properties of existing views, including the center point, screen mode, and zoom.

166.1 Properties

166.1.1 View.bounds

`app.activeDocument.views[index].bounds`

Description

Read-only. The bounding rectangle of this view relative to the current document's bounds.

Type

Array of 4 Numbers

166.1.2 View.centerPoint

`app.activeDocument.views[index].centerPoint`

Description

The center point of this view relative to the current document's bounds.

Type

Array of 2 Numbers

166.1.3 View.parent

`app.activeDocument.views[index].parent`

Description

Read-only. The document that contains this view.

Type

Document

166.1.4 View.screenMode

`app.activeDocument.views[index].screenMode`

Description

The mode of display for this view.

Type

ScreenMode

166.1.5 View.typename

`app.activeDocument.views[index].typename`

Description

Read-only. The class name of the referenced object.

Type

String

166.1.6 View.zoom

`app.activeDocument.views[index].zoom`

Description

The zoom factor of this view, where 100.0 is 100%.

Type

Number (double)

CHAPTER 167

Views

`app.activeDocument.views`

Description

A collection of *View* objects in a document.

167.1 Properties

167.1.1 Views.length

`app.activeDocument.views.length`

Description

The number of objects in the collection

Type

Number; read-only.

167.1.2 Views.parent

`app.activeDocument.views.parent`

Description

The parent of this object.

Type

Object; read-only.

167.1.3 Views.typename

```
app.activeDocument.views.typename
```

Description

The class name of the referenced object.

Type

String; read-only.

167.2 Methods

167.2.1 Views.index()

```
app.activeDocument.views.index(itemKey)
```

Description

Gets an element from the collection.

Parameters

Parameter	Type	Description
`itemKey`	String, Number	String or number key

Returns

View

CHAPTER 168

Words

`app.activeDocument.textFrames[index].words`

Description

A collection of words in a text item, where each word is a *TextRange* object.

The elements are not named; you must access them by index.

168.1 Properties

168.1.1 Words.length

`app.activeDocument.textFrames[index].words.length`

Description

The number of objects in the collection

Type

Number; read-only.

168.1.2 Words.parent

`app.activeDocument.textFrames[index].words.parent`

Description

The parent of this object.

Type

Object; read-only.

168.1.3 Words.typename

```
app.activeDocument.textFrames[index].words.typename
```

Description

The class name of the referenced object.

Type

String; read-only.

168.2 Methods

168.2.1 Words.add()

```
app.activeDocument.textFrames[index].words.add(contents[, relativeObject][,
inseertLocation])
```

Description

Adds a word to the current document at the specified location.

If no location is specified, adds it to the containing text frame after the current word selection or insertion point.

Parameters

Parameter	Type	Description
`contents`	String	Word to add
`relativeObject`	*TextFrameItem*, optional	Object to add item to
`insertionLocation`	*ElementPlacement*, optional	Location to insert text

Returns

TextRange

168.2.2 Words.addBefore()

```
app.activeDocument.textFrames[index].words.addBefore(contents)
```

Description

Adds a word before the current word selection or insertion point.

Parameters

Parameter	Type	Description
`contents`	String	Word to add

Returns

TextRange

168.2.3 Words.index()

`app.activeDocument.textFrames[index].words.index(itemKey)`

Description

Gets an element from the collection.

Parameters

Parameter	Type	Description
`itemKey`	String, Number	String or number key

Returns

TextRange

168.2.4 Words.removeAll()

`app.activeDocument.textFrames[index].words.removeAll()`

Description

Deletes all elements in this collection.

Returns

Nothing.

168.3 Example

168.3.1 Counting words

```
// Counts all words in current document and stores total in numWords
if ( app.documents.length > 0 ) {
  var numWords = 0;

  for ( i = 0; i < app.activeDocument.textFrames.length; i++) {
    numWords += app.activeDocument.textFrames[i].words.length;
  }
}
```

168.3.2 Applying attributes to words

```
// Creates a new magenta color and applies the color to all words meeting a specific
↪criteria
if (app.documents.length > 0 && app.activeDocument.textFrames.length > 0) {
  // Create the color to apply to the words
  var wordColor = new RGBColor();
  wordColor.red = 255;
  wordColor.green = 0;
  wordColor.blue = 255;

  // Set the value of the word to look for searchWord1 = "the";
  var searchWord2 = "The";
  var searchWord3 = "THE";

  // Iterate through all words in the document
  // and color the words that match searchWord

  for (var i = 0; i < app.activeDocument.textFrames.length; i++) {
    var textArt = activeDocument.textFrames[i];

    for (var j = 0; j < textArt.words.length; j++) {
      var word = textArt.words[j];

      if (word.contents == searchWord1 || word.contents == searchWord2 || word.
↪contents == searchWord3) {
        word.filled = true;
        word.fillColor = wordColor;
      }
    }
  }
}
```