Adobe® Illustrator® CS2 Scripting GettingStarted with JavaScript®

Introduction

Scripting offers an easy way to manipulate Illustrator files programmatically. You can write scripts to create documents and objects, and to perform many of the operations that you access interactively in Illustrator, using the palettes and windows. Scripts are particularly useful as a way to automate day-to-day tasks that are complex and repetitive. You simply write the script once, and then run it any time you need to accomplish that task. This saves you both time and the effort of remembering steps and sequences of operations.

Illustrator CS2 supports JavaScript, an easy-to-use, platform-independent scripting language. A JavaScript program, or script, takes the form of a text file with the `.js` extension. You can use any text editor to write your scripts, such as TextEdit, Textpad, or BBEdit.

This overview will help you to become familiar with Illustrator scripting using JavaScript, by walking you through some example scripts that manipulate objects in Illustrator CS2.

This document assumes that:

- You are familiar with the Illustrator user interface, as described in the *Illustrator CS2 User Guide*.
- You are familiar with JavaScript programming language.
- You have installed Illustrator CS2 version 12.0 and an editor to create scripts, such as Textpad or TextEdit.

Executing JavaScript Scripts

To run any script from Illustrator CS2:

1. Choose **File>Scripts>Browse**.
2. Navigate to your script file.
3. Double-click the file or click **Open** to run the script.

To make a script available as a menu item under Illustrator's **File>Scripts** menu, place the script file in the directory `Adobe Illustrator CS2\Presets\Scripts`, then restart Illustrator. Each time you launch Illustrator, every script found in this location is added as a menu item ,Your script will be listed among the prebuilt Automation Scripts that are available in this directory.

Accessing and Referencing Objects

When you write a script, you must first decide what file, or *document*, the script should act on. 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 has selected, or operate on objects that it finds in one of the *object collections*. The following examples illustrate various techniques for accessing, referencing, and manipulating Illustrator objects.

Example 1: Acting on Selected Objects

When the user makes a selection in a displayed 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;
```

Depending on what is selected, the property value can be either an array of art objects or a text string. To correctly handle the selection, you must first test to see if you have artwork or text. To get or manipulate the properties of the selected art items, you must retrieve the individual items in the array. For example, the topmost object is at the zero index in the array:

```
var topObject = app.activeDocument.selection[0]; // for topmost object
```

The following example gets all the selected objects and then displays the type names for theselected items or returns an appropriate message.

1. Open a new blank document in your text editor. For example, in Textpad, choose **File>New**.
2. Save the document as `Example1.js`. For example, in Textpad, choose **File>Save**.
3. Write the initial statements declaring the variables.

   ```
   //Declare and initialize variables
   var msgType = "";
   ```

4. This script has to run on objects in a document, so make sure a document is open.

   ```
   // check if a document is open in Illustrator.
   if( app.documents.length > 0)
   {
   ```

5. If the document is open, get the selected objects.

   ```
      mySelection = app.activeDocument.selection;
   ```

6. Check if there were any objects selected.

   ```
      if (mySelection.length > 0)
      {
   ```

7. If there are selected object, use a loop to get their type names from the `typename` property, and assign an appropriate message to the `msgType` variable.

   ```
         msgType = "Selection items: \n";
         for (var i=0; i<mySelection.length; i++)
         {
          msgType += "\nItem " + i + " typename is: " + mySelection[i].typename;
         }
      }
   ```

   If no objects were selected, assign a message to that effect.

   ```
      else
      {
         msgType = "Nothing is selected.";
      }
   }
   ```

8. If no document is open, set the message to reflect that case.

   ```
   else
   {
      msgType = "No document Open.";
   }
   ```

9. Finally, print the message.

   ```
   alert( msgType );
   ```

10. Save the file.

11. Run the script, as described in Executing JavaScript Scripts above.

    The alert box appears and shows the types of all the selected items.

Example 2: Acting on All Objects in a Document

This next example shows how to use the the `pageItems` class to access all the objects in an Illustrator document, regardless of whether there is a user selection. (For a complete list of objects accessible through the pageItems Class, see the *JavaScript Reference Pg 119*.)

This script navigates through all the `pageItem` objects in the active document and prints their types.

1. Open a new blank document in your text editor, and save it as `Example2.js`.

2. Declare variables.

   ```
   var msg = "";
   ```

3. Check if a document is open. If it is, get any objects in it.

   ```
   // Check if a document is open and get all items in it
   if ( app.documents.length > 0)
   {
      var pgItems = app.activeDocument.pageItems;
   ```

4. If there are any objects in the array, loop through them to get their types and add them to the message.

   ```
   // If PageItems exist
      if ( pgItems.length > 0 )
      {
         var msg = 'PageItems in the active document are - ' ;
         for ( var i = 0 ; i < pgItems.length  ; i++ )
         {
            msg += '\nItem ' + i + ': ' + pgItems[i].typename ;
         }
      }
   }
   ```

   Note that you can use either single or double quotes to write strings. Just make sure you close the string with the same type of quote.

5. If no document is open, set the message to reflect that case.

   ```
   else
   {
     msg = "No document Open.";
   }
   ```

6. Print the msg.

   ```
   alert ( msg );
   ```

7. Save the file.

To run this example:

1. Launch Illustrator.

2. Open a document with some objects in it or create a document and create some objects in it.

3. Run the script, as described in Executing JavaScript Scripts above.

   The alert box appears and shows the types of all the page items.

Example 3: Accessing Objects in Collections

Often you want to access particular objects, or objects of particular types, rather than all items in a page or selected items. For example, you might want to access and manipulate all graph items or all raster items in a page. Different types of objects in a document are collected into collection objects for those types. A collection object contains an array of the objects which you can access by index number or by name.

The following example shows how to access objects from their respective collections

1. Open a new blank document in your text editor, and save it as `Example3.js`.

2. Get all brushes in the brushes collection in the document.

   ```
   var myBrushes = app.activeDocument.brushes;
   ```

3. All array references are zero-based; that is, the first element of the collection has the index [0]. To access the first brush in this collection:

   ```
   var firstBrush = myBrushes[0];
   ```

4. To get the total number of objects, use the `length` property:

   ```
   alert ( "Total Brushes in this document: " + myBrushes.length ) ;
   ```

5. Similarly, let’s access all the graphic styles in a document through the graphicStyles collection object:

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

6. If you know the name of an object, you can access it in the collection using the name. Otherwise, you can use its 0-based index. For example:

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

7. Index of the last style would be myStyles.length-1.

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

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

   ```
   var styleName = lastStyle.name;
   ```

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

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

10. To reference the first pathItem in the active document use the pathItems collection:

    ```
    pathRef = app.activeDocument.pathItems[0];
    ```

11. A `pathItem` is made up of path points. Hence to access the points in the `pathRef` object:

    ```
    pathRefPoints = pathRef.pathPoints;
    ```

    a. To get the type of the point use the `pointType` property:

    ```
    alert ( pathRefPoints[0].pointType );
    ```

12. Save the file.

13. Run the script, as described in Executing JavaScript Scripts above.

    The alert boxes appear with total number of brushes and the pathItem's PointType and and lastStyle is applied to the first pageItem in the document.

You can use this technique to access raster items, groups, items within a group, gradients, colors, textFrames, and other object types in a document and manipulate them needed.

Example 4: Creating New Objects

You can use a script to create new documents, and new objects, rather than accessing any existing documents or objects. This script does not have to check whether a document is open and objects are present, because it creates a new blank document and creates some objects in it.

This simple example creates some basic shapes in the first layer of the frontmost document and applies a random graphic style to it. Each shape is defined by new `pathItem` objects.

1. Open a new blank document in your text editor, and save it as `Example3.js`.

2. Create a new blank Illustrator document.

   ```
    // open a new blank document
   var thisDoc = app.documents.add();
   ```

3. Create a new layer.

   ```
   // add a new layer
   var artLayer = thisDoc.layers.add();
   ```

4. Set the default stroke and fill settings for all the objects using the `app` object (the Illustrator Application object that contains all other Illustrator objects).

   ```
   // default stroke and fill settings
   app.defaultStroked = true;
   app.defaultFilled = true;
   ```

5. Create a rectangle using the `rectangle` method of the `pathItems` object. Pass parameters for position, width, and height of the rectangle. (For details on these parameters see the `pathItems` object in the *JavaScript Reference* .)

```
// Create different pathItem objects
var rect = artLayer.pathItems.rectangle( 762.5, 87.5, 425.0, 75.0 );
```

6. Create a rounded rectangle.

```
var rndRect = artLayer.pathItems.roundedRectangle( 637.5, 87.5, 425.0,
75.0, 20.0, 10.0 );
```

7. Create an ellipse.

```
// Create ellipse, 'reversed' == false, 'inscribed' == true
var ellipse = artLayer.pathItems.ellipse( 512.5, 87.5, 425.0,
                                             75.0, false, true );
```

8. Create an octagon.

```
// Create octagon, an 8-sided polygon
var octagon = artLayer.pathItems.polygon( 300.0, 325.0, 75.0, 8 );
```

9. Create a star with four points.

```
// Create a 4 point star
var star = artLayer.pathItems.star( 300.0, 125.0, 100.0, 20.0, 4 );
```

10. So far, the objects we have created simply use the default stroke and fill we set in the `app` object. Now we will access the new objects using the `pathItems` collection, and apply some random graphic styles to them.

```
// Apply random graphic style to all the pathItem objects in this document
for ( i = 0; i < artLayer.pathItems.length; i++ )
{
   // Obtain a random graphic style number
   styleIndex = Math.round( Math.random() * ( thisDoc.graphicStyles.length -
1 ) );
   // apply the graphic style at the random index position
   thisDoc.graphicStyles[styleIndex].applyTo( artLayer.pathItems[i] );
}
```

11. Save the file.

12. Run the script, as described in Executing JavaScript Scripts above.

    The objects are created with different graphic styles.

Congratulations! You have now written four JavaScripts for Illustrator. There are more scripts available in the `Adobe Technical Info > Scripting > Sample Scripts > JavaScript` folder as part of the Illustrator SDK on the CD.