

Using SVG
with
Adobe[®]
Illustrator[®] 9.0

Introduction

1

Starting Out

2

SVG Export Options

2

The Layers Palette and SVG Groups

6

Rendering and Performance

9

Integrating JavaScript with SVG

16

Tips and Techniques

21

Additional Resources

21

Using SVG with Adobe Illustrator 9.0

Authoring Guide

Introduction

1

This paper summarizes capabilities and issues that you may encounter when using Adobe® Illustrator® 9.0 to create Scalable Vector Graphics (SVG) documents. This paper details how to use Adobe Illustrator to:

- Get started creating SVG content.
- Export SVG files, taking account for export options.
- Organize layers that relate to SVG groups.
- Optimize SVG performance.
- Add and associate JavaScript code with SVG.
- Create hyperlinks for SVG elements, so that a user can click on an SVG graphic and be sent to a new URL.

About Scalable Vector Graphics (SVG)

SVG is an XML-based language for describing interactive, animated two-dimensional graphics. As its name implies, SVG supports vector graphics (lines and curves), but it also supports text and raster images. Its feature set includes transformations, transparency, clipping paths, and filter effects.

SVG is an application of XML, so you could write SVG documents with a basic file editor, although that would be impractical for any significant content. Content developers are likely to use Adobe Illustrator along with other familiar production tools to create SVG documents.

About Adobe Illustrator 9.0

Adobe Illustrator 9.0 is used by artists and designers to create graphics for use in print, on the Web, and in dynamic media projects. Illustrator features easy object creation, layered organization, transparency blending, and other powerful visual effects. Adobe Illustrator's work area (artwork window, tool box, and floating palettes) are easy to use and are familiar and reassuring to anyone who has used other Adobe applications.

Audience

This paper is intended for people who want to use Adobe Illustrator 9.0 to create SVG documents. The paper assumes some prior knowledge of both Adobe Illustrator and SVG.

If you want more information about Adobe Illustrator, read the book *Adobe Illustrator 9.0 Classroom in a Book®*, which is part of the official training series for Adobe tools. For more information on SVG, see the list of [Additional Resources](#) at the end of this document.

Starting Out

As you plan to use Adobe Illustrator to create an SVG document, several issues should be handled initially.

First, SVG colors are specified in the sRGB color space, so you should use RGB, not CMYK, for an Adobe Illustrator document.

If you anticipate merging several SVG files at a later time, for the outermost `<svg>` element, you should define a consistent viewport coordinate system as well as `viewBox` attribute values. When the documents are later merged, you will have a consistent frame of reference.

You need to be careful about how Adobe Illustrator determines the size of the SVG. Adobe Illustrator exports SVG with points as the standard unit. For raster images, Adobe Illustrator assumes 1 point (1 pt) to be equal to 1 pixel (1 px). This will only be true, if there are 72 pixels per inch, and many devices have widely different screen resolutions. So the size of an exported SVG image is likely to be different from a file in any image format (such as GIF, JPG, or PNG).

One way to solve this is to change the size declaration of the SVG file. For example, if the SVG file currently has this declaration:

```
<svg width="300.00pt" height="200.00pt" viewBox="0 0 300.00 200.00" xml:space="preserve">
```

edit the SVG file by hand to convert the points to pixels:

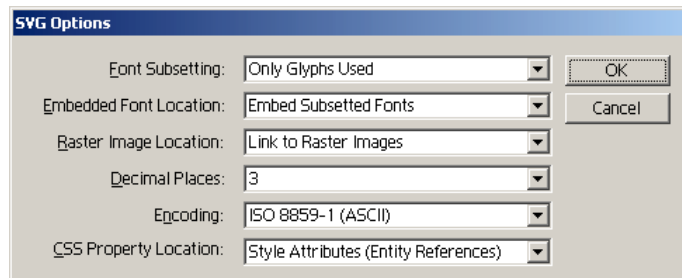
```
<svg width="300.00px" height="200.00px" viewBox="0 0 300.00 200.00" xml:space="preserve">
```

There are a variety of unit identifiers, other than px and pt, including cm, mm, in, (centimeters, millimeters, and inches) and percentages.

SVG Export Options

Adobe Illustrator offers automated translation of graphics data into the SVG format. To produce an SVG file, choose File > Export to open up a file browser. In the file browser, where “Save

as type:" is requested, choose SVG or SVGZ (for compressed SVG). Then you will see the following SVG Options dialog box, describing options for how to save the data in SVG format. Before saving the SVG file, consider the effects of the options in these menus.



SVG Options Dialog Box

Subsetting and Embedding Fonts

The first two drop-down menu choices in the dialog box materially affect how much font data is saved and where it is saved.

For Font Subsetting (the first menu in the SVG Options dialog box), if you select *Only Glyphs Used*, only specific characters that have been used are saved in the file. Saving selected glyphs rather than entire fonts saves file space, but problems arise if the SVG file is later modified and new characters are introduced. If you think your file may require future editing, you may want to choose *Common English* or *Common Roman* (the latter includes European accented characters and other symbols). Use of non-Roman symbols (such as Kanji) or other special characters may require you to save glyphs, either alone or in addition to Western European letters.

If you select *Embed Subsetted Fonts* for Embedded Font Location (the second menu in the SVG Options dialog box), then the fonts used are stored directly in the SVG file. By selecting *Link To Subsetted Fonts*, Adobe Illustrator saves the fonts in CEF (Compact Embedded Font) format in separate, external files. A `<style>` element is added to the SVG file to reference the font in the external CEF file.

If you choose to link to external CEF files, rather than embed the CEF data within the SVG file, you will have more files to manage. However, with external CEF files, several SVG applications could share the same CEF file without embedding the same font, and there would be less to download.

If *None (Use System Fonts)* is chosen from the Font Subsetting menu, font information is not provided (neither externally nor embedded) as CEF data for the exported SVG. If you didn't use any text or if all the text you used was converted into outlines, you may want to choose *None (Use System Fonts)*. If a font referenced in the SVG file does not exist on the system, the text may be rendered in any font the system decides. (If only system fonts are used, embedding and linking fonts becomes a moot issue.)

Raster Image Location

Raster images that are used in SVG graphics may either be embedded within the SVG file or externally linked. In the SVG Options dialog, the menu labeled Raster Image Location allows you to choose which one you want to use. Both methods use the `xlink` attribute of the `<image>` element, but the value of the hyperlink reference is either a short file name (external link) or the entire data for an image.

The advantages of embedding or externally referencing a raster image are similar to those of CEF files. With an embedded raster image, there are fewer files to manage and no chance of misplacing the image. On the other hand, with external links, several SVG applications may share the raster image file, which is more efficient than each SVG file embedding the same image.

When exporting a rasterized image from Adobe Illustrator, the resulting image data is represented in PNG (Portable Network Graphics) format. If you want the image data in a different format (such as GIF or JPEG), you will need to perform an external format conversion with a tool such as Adobe® Photoshop®.

If a GIF or JPEG image is imported into Adobe Illustrator, it remains in its original format and is not converted into PNG. While in Adobe Illustrator, the image may undergo a variety of operations, such as filter effects, but when it's exported, it will be exported in its original format. In addition, if you want the image data to be the same as your originally placed image (retaining its file name and any optimizations), you must export your SVG file to the **same directory** as the originally placed image.

Decimal Places

You may select the decimal precision of real number values (including coordinates, lengths, transformation values) generated by Adobe Illustrator. There's a trade-off between increased precision and the memory needed to represent data. Without enough decimal places, transformations can become noticeably coarse and inaccurate. The scaling transformation, in particular, magnifies these inaccuracies, requiring increased precision.

Encoding

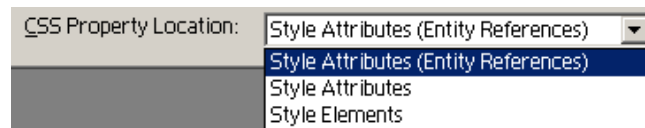
An SVG file contains text that represents markup and character data. SVG viewers must accept and process text that meet industry character encoding standards, such as ASCII or Unicode. ASCII is an older encoding standard, representing each character with 7 bits. Unicode was subsequently developed to provide consistent, unique numerical encoding for many more characters (including mathematical symbols and scripts in Asian and Middle-Eastern languages). SVG supports two Unicode Transformation Format (UTF) encodings: UTF-8, a variable length 8-bit encoding form, and UTF-16, a 16-bit format. The selection from the SVG Options menu

labeled *Encoding* determines which **one** of the following encoding declarations becomes a necessary part of the XML declaration.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<?xml version="1.0" encoding="utf-8"?>
<?xml version="1.0" encoding="utf-16"?>
```

Cascading Style Sheets (CSS) Property Location

Styling properties are designed to separate the presentation style of documents from their content, controlling the appearance of fonts, text, and other visual media. In some scenarios, a separate style sheet is needed so that the presentation style is changeable from outside the content data. From the final menu (CSS Property Location) on the SVG Options dialog box, you can choose how styling properties are generated from Adobe Illustrator.



Options for Representing Styles

With either of the first two choices in the CSS Property Location menu, *style* attributes are created. Style attributes may be favored when SVG presentation and content must be kept together, such as when the SVG file is to be used as an exchange format. With the first menu choice, *Style Attributes (Entity References)*, an Entity Reference, such as *st5* below, is initially created and may be reapplied many times within the same file.

```
<!ENTITY st5 "font-size:18;">
<g style="&st5;">
```

If you anticipate that a style attribute may be globally changed within a single file, the Entity Reference is highly recommended. However, if you anticipate merging SVG files together, the Entity References are unlikely to be the same in both files. With the second menu choice, *Style Attributes*, the style attributes may be specified inside the SVG element. Merging SVG files is much easier without indirect Entity References.

```
<g style="font-size:18;">
```

If you want the complete flexibility afforded by separating presentation style and content data, the third menu choice, *Style Elements*, is your best bet. Style elements are defined in a *<style>* element:

```
<style type="text/css">
<![CDATA[
    .st0{font-family:'Arial-ItalicMT';}
    .st1{fill-rule:nonzero;stroke:#000000;}
]]>
</style>
```

Here's how a style element may be used as the class attribute of a group <g> element:

```
<g class="st1">
```

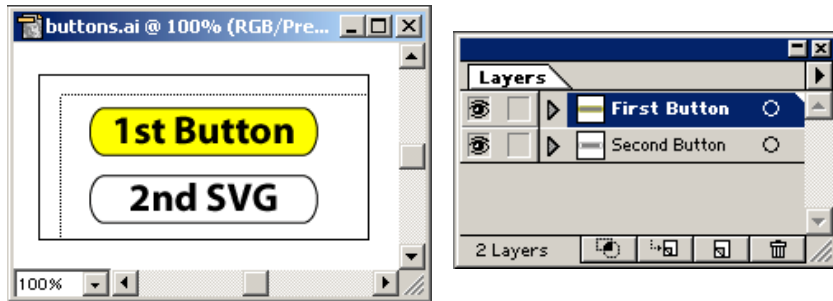
The Layers Palette and SVG Groups

In Adobe Illustrator, the Layers palette controls a variety of operations to organize artwork. Fills, strokes, transparency settings, and effects may be applied to an entire layer, as well as to individual objects. Layers may be rearranged with nested sublayers and in different stacking orders. Layers may be temporarily locked, preventing them from being edited, and layers may also be temporarily hidden.

When the layers are exported from Adobe Illustrator into SVG, they are converted into group <g> elements. Nested layers are preserved as SVG nested groups. Hidden layers are exported using the SVG styling property "display:none". However, locked Adobe Illustrator layers are not accorded special treatment in SVG. Once the Adobe Illustrator artwork has been exported, there isn't a logical way to prevent someone from modifying the SVG file.

Layer and Object Names

In the following Adobe Illustrator session, there are two layers that have been named *First Button* and *Second Button*. Each layer contains the text and path objects for one button. The preview window and layers palette for this example are shown below:



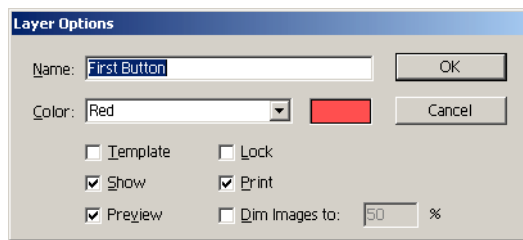
Layers and Artwork in Adobe Illustrator

The Adobe Illustrator names are used for the corresponding elements in the exported SVG files. The previously described layers would be exported into the following SVG groups:

```
<g id="Second_x0020_Button">
  <!-- elements deleted -->
</g>
<g id="First_x0020_Button">
  <!-- elements deleted -->
</g>
```

The object names are not always exported exactly as SVG ids. For example, since spaces are not allowed in SVG element ids (spaces are not allowed in any XML id), an encoded string is substituted for each space. For text objects, in Adobe Illustrator, the default object names are the actual text strings, but these names are not exported as SVG ids. Clipping-path names are also not exported to SVG.

Double-clicking the name of a layer or object displays an Options dialog box (as seen below), where you can change the name of the layer or object.

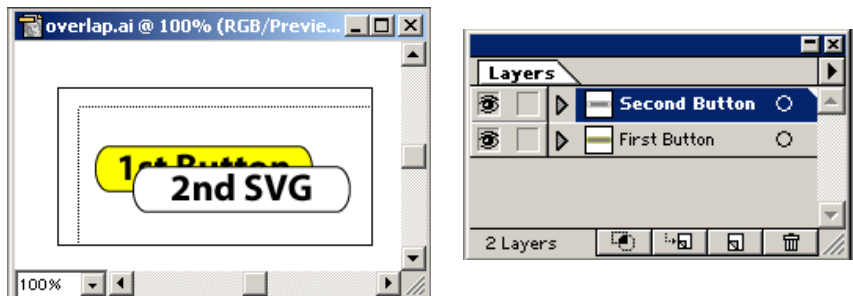


Rename Layers, using the Layer Options Dialog Box

Rendering Order

Both the Adobe Illustrator Layers palette and the SVG groups preserve rendering order. In the Layers palette, the first layer listed is the topmost drawn. Each subsequent layer is rendered underneath the previous one. The rendering order of SVG is the opposite of Adobe Illustrator. In an SVG document, the first group element is rendered first, and subsequent groups are drawn on top of previous groups—the “Painter’s Algorithm.”

To counteract the ordering difference, when Adobe Illustrator layers are exported to SVG, their stacking order is reversed. In the following example, the objects have been moved so that they overlap. Both Adobe Illustrator and SVG result in the same image, with Second Button rendered atop First Button. In this Adobe Illustrator Layers palette, the layers are ordered, Second Button followed by First Button:

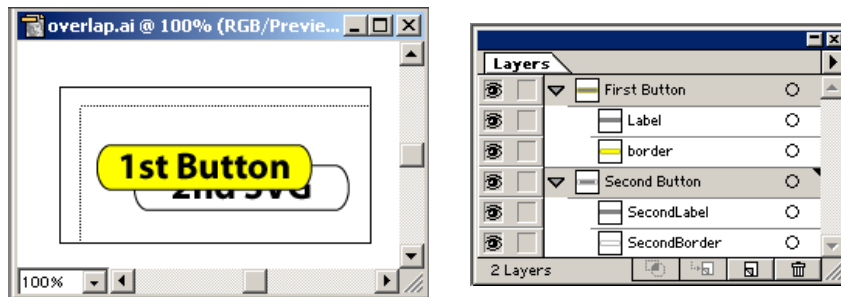


Arranging the Order of Layers affects Rendering Order

In the corresponding SVG, the order of the group elements has been reversed. When exported to SVG, the layer in back becomes the first group element; the layer in front becomes the second group element .

```
<g id="First_x0020_Button">
  <!-- elements deleted -->
</g>
<g id="Second_x0020_Button">
  <!-- elements deleted -->
</g>
```

The rendering order of Adobe Illustrator layers and their corresponding SVG groups extends to individual Adobe Illustrator objects and their corresponding SVG elements. In the following Layers palette, the objects in each layer (here named Label and border) are arranged from front to back.



The Rendering Order of Layers and Objects

When exported to an SVG document, the element in the front is placed after the element in the back.

```
<g id="Second_x0020_Button" style="&st2;">
  <path id="SecondBorder" d="... deleted..."/>
  <text id="SecondLabel1" transform="matrix(1 0 0 1 24.1533
63.0947)">
    <tspan x="0" y="0" style="&st1; &st5; &st6;">2nd SVG</tspan>
  </text>
</g>
<g id="First_x0020_Button" style="&st7;">
  <path id="border" style="&st4;" d="...deleted..."/>
  <text id="Label" transform="matrix(1 0 0 1 13.8457 21.979)">
    <tspan x="0" y="0" style="&st8; &st5; &st6;">1st Button
  </tspan></text>
</g>
```

Rendering and Performance

Adobe Illustrator easily exports artwork into SVG documents, but you should be aware of how the exported data will be rendered. In some cases, structuring your Illustrator artwork correctly will increase the rendering performance of an SVG document. In general, you should:

- expect that vectors (not primitives) will be generated.
- apply transparency to objects or layers, as appropriate.
- reuse SVG elements and reduce the number of vertices in path data (vectors), where possible.
- carefully examine exported code with radial and linear gradients; avoid gradient meshes.
- use path data and avoid rasterization, where possible, or at least anticipate the effects of rasterization.
- be aware of situations when text is converted into vectors or rasters.

Vectors Exported as Path Data

When exporting vectors to SVG, Adobe Illustrator creates only `<path>` elements, not SVG primitives. Even though Adobe Illustrator has an ellipse tool and a rectangle tool, it does not export `<rect>` or `<ellipse>` elements. `<circle>`, `<line>`, `<polyline>`, and `<polygon>` elements are not generated, either.

Therefore, you cannot employ a technique (like a script or an SVG animation element) that manipulates a special characteristic of a primitive. For example, you cannot write JavaScript function to change the radius of a circle, because Adobe Illustrator will not export an object as a `<circle>` element. (Of course, you could always edit an SVG document by hand and replace a `<path>` element with a `<circle>`.)

Transparency/Opacity

Using Adobe Illustrator, you can use the Transparency palette to choose the opacity of a selected object or layer. (By default, all objects and layers have an opacity of 1.0 or 100%.) When exported to SVG, the Transparency palette setting is converted to the value of the `opacity` property of a corresponding `<path>` element, text element, or group. In SVG, group elements have the effect of rendering to a separate canvas onto which child elements are painted, so you get different results, depending upon whether an object or layer is targeted in Adobe Illustrator.

In the following example, in Adobe Illustrator, each rectangle has been assigned an opacity value of 66% (in SVG, each `<path>` element is assigned an opacity of 0.66). The result is that each `<path>` is blended atop the previous one, and overlapping regions appear blended.



When Each Path Object is Transparent

In this contrasting example, each rectangle has been assigned an opacity value of 100% (fully opaque). The layer as a whole has been assigned an opacity value of 66%. The rectangles are rendered into a temporary canvas, and each rectangle is opaque relative to one another. Then the entire layer/canvas is made 66% opaque and is blended atop any previous rendering.



When The Entire Layer is Transparent

Vectors: Reuse and Reduce

SVG preserves both vector and raster data, but vector data is a great source of SVG's capabilities. With vector data, you can zoom in with no loss of visual quality. To represent the same image, vector data is often more compact than raster data. In many cases, you may also further optimize vector data to increase download and rendering performance.

In SVG, the `<defs>` element defines a more complex SVG element, and the `<use>` element may repeatedly reuse the previously defined element. Utilizing `<defs>` and `<use>` increases performance, but Adobe Illustrator does not create them in exported SVG documents. For instance, when cutting and pasting an object in Adobe Illustrator, an entirely new element is created. Reusing elements with `<defs>` and `<use>` requires editing the SVG file by hand. To see an example of that successfully employs hand editing, see the [Adobe Theater Demo](#) where a single chair is defined for each seating section and is then replicated throughout the section.

The same technique is an improvement in a variety of situations. In Adobe Illustrator, when multiple strokes are applied to the same object, the exported SVG does not reuse path data. In the following example, the style called Black Red Dashes has been applied to a path.



A Style Applied to a Path

The SVG generated upon export declares the exact same path twice:

```
<g id="Layer" style="&st2;">
  <g>
    <path style="&st0;" d="M77.9033,55.2529H2V2h75.9033v53.2529z"/>
    <path style="&st1;" d="M77.9033,55.2529H2V2h75.9033v53.2529z"/>
  </g>
</g>
```

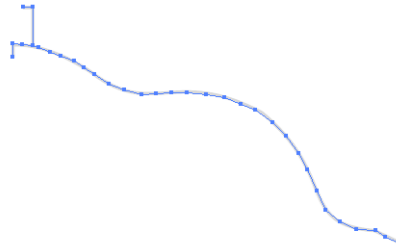
By editing this by hand, the path can be defined once and reused, as shown below:

```
<defs>
  <g id="rect">
    <path d="M77.9033,55.2529H2V2h75.9033v53.2529z"/>
  </g>
</defs>
<g id="Layer" style="&st2;">
  <use id="black" xlink:href="#rect" style="&st0;"/>
  <use id="red" xlink:href="#rect" style="&st1;"/>
</g>
```

Simplify Paths Carefully

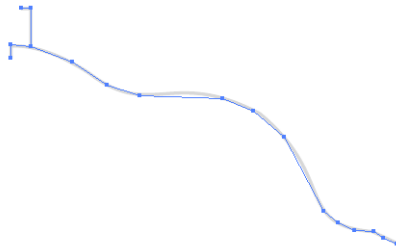
Adobe Illustrator has a path simplification tool which reduces the number of anchor points of a path. However, one must be careful to ensure that the tool doesn't unintentionally increase the number of vertices that result in an exported SVG file.

To bring up this tool, first select the path or paths you want to work upon. Then choose Object > Path > Simplify to open up a dialog box. The following figure shows paths with 34 anchor points, prior to simplification.



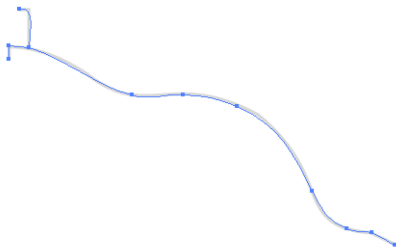
Original Paths with 34 Anchor Points

With the option Straight Lines chosen in the dialog box, the exported SVG will definitely be smaller than the original path, but may be a noticeably ragged approximation. You should experiment with the number of anchor points until you are satisfied with the trade off. In the figure below, the paths are reduced to 18 anchor points and drawn with only straight lines.



Paths Simplified into Straight Lines

If the option Straight Lines is not chosen, Adobe Illustrator will reduce the number of anchor points, but represent the result as curved paths. The exported SVG will convert the curves into straight lines, so the resulting SVG may have a greater number of vectors than the original.



Simplified into Curved Paths; Fewer Anchor Points, but More SVG Vectors

Brushes Produce Vectors

In the image below, several brush strokes are displayed, some using Art Brushes from the Brush palette, including Grape Leaf, 3D Arrow, Grass Blade, Scroll Pen, and Charcoal.



Adobe Illustrator Brushes are Vector Path Data

When exported, Adobe Illustrator brushes are rendered as vector path data. The 3D Arrow brush tends to produce very concise path data. Some brushes, such as the Charcoal, FireAsh, or ScrollPen, may produce a lot of path data, which should be considered before their use.

Gradients

From Adobe Illustrator, radial and linear gradients are exported without rasterization. The following is an example of exported SVG for a radial gradient, generated for a circular path:

```
<g id="Layer_x0020_2" style="&st3;">
  <radialGradient id="aigrd2" cx="180" cy="56" r="35.2881"
    fx="180" fy="56" gradientUnits="userSpaceOnUse">
    <stop offset="0.0056" style="stop-color:#9D30FF"/>
    <stop offset="0.1367" style="stop-color:#9E30FB"/>
    <stop offset="0.2623" style="stop-color:#A030F0"/>
    <stop offset="0.3859" style="stop-color:#A330DC"/>
    <stop offset="0.5081" style="stop-color:#A831C1"/>
    <stop offset="0.6294" style="stop-color:#AF319E"/>
    <stop offset="0.75" style="stop-color:#B73273"/>
    <stop offset="0.8702" style="stop-color:#C03240"/>
    <stop offset="0.9871" style="stop-color:#CB3307"/>
    <stop offset="1" style="stop-color:#CC3300"/>
  </radialGradient>
  <path style="&st2;" d="M149.5,56c0-21.8154,13.6553-39.5,30.5-
39.5s30.5,17.6846,30.5,39.5S196.8447,95.5,180,95.5S149.5,77.815
4,149.5,56z"/>
</g>
```

Adobe Illustrator may export more gradient stops than are needed to achieve the proper visual effect. You should examine any generated code and see if you can adjust and reduce the gradient stops by hand.

Although radial and linear gradients are supported in SVG, gradient meshes are not. Before export from Adobe Illustrator, gradient meshes are rasterized and none of the gradient data is preserved.

SVG Rasterization

In some cases, like gradient meshes, Adobe Illustrator will not maintain vector data and instead rasterizes the image. Once rasterized, you lose several key aspects of SVG, such as scalability—as the viewer zooms closer, a bitmap will appear coarser. Also, a content developer cannot use a text editor to search and edit raster data like she can with other SVG elements.

However, there are several situations that result in rasterization. The Adobe Illustrator user may intentionally choose **Effect > Rasterize** to convert selected objects into bitmaps. Also all the Photoshop effects in the **Effect** drop down menu—**Artistic**, **Blur**, **Brush Strokes**, **Distort**, **Pixelate**, **Sharpen**, **Sketch**, **Stylize**, **Texture**, and **Video**—rasterize their results for SVG export. Similarly, any **Styles** that use these Photoshop effects (such as **Blazing Sun**, **Cement**, and **Flames**) will also generate bitmaps.

Note that some techniques in the **Effects** menu do not rasterize their result. Effects such as **Free Distort**, **Punk & Bloat**, **Roughen**, **Scribble and Tweak**, **Twirl**, **Zig Zag**, and **Round Corners** produce path data (for both original text and vector graphics). The **Transform** effect does not convert the text data into a path; it keeps it as text data. The **Drop Shadow** effect does not change the original text or graphics, but it does create a bitmap shadow that appears “behind” the original objects.

Exported SVG Text

SVG typically maintains text as character data, which allows the text to be easily viewed, edited by hand, and searched for in source code. Multilingual SVG content is enabled, because the text strings are easily replaced for different languages. Also, SVG operations such as transformations, clipping, and masking may be applied to text elements.

With Adobe Illustrator, some character data is not exported as SVG text elements. With the aforementioned filter effects, the text elements are converted into raster images. In the example below, text strings have had the **Gaussian blur** filter effect applied. The text strings are not exported to the SVG file, but a raster representing the blurred letters is exported.

1st Button
2nd SVG

Gaussian Blur Filter Applied; Exported as SVG Raster

Adobe Illustrator may also export character data as a standard `<text>` element or convert it to vector path data. In the Brush palette, the Typography brush does not preserve character data; it converts text into path data. In the following example, the style Cast Shadow has been used on a text objects. After the Cast Shadow style is applied, the initial text element is still available in the exported SVG file. However, the shadow that is beneath the letters is not text, but has been calculated as path data. If you decided to edit this SVG file by hand to change the text string, the upright letters would change, but not the shadow beneath it.

1st Button
2nd SVG

Upon Export, Shadows are Converted to Path Data

Adobe Illustrator can export to an SVG document with more advanced text capabilities. For kerning and placing type on a path, the text strings are broken into multiple `<tspan>` elements. You may be able to optimize the SVG by hand coding the `kerning` property or the `<textPath>` element.

Avoiding Rasterization

Because SVG supports a variety of filter effects, it's possible to write SVG code that maintains vector and text information and performs the filter effect. For example, instead of using Adobe Illustrator's Gaussian Blur effect, which rasterizes, you could hand code use of the SVG `<feGaussianBlur>` element, such as in this filter definition:

```
<defs>
  <filter id="DropShadowFilter" filterUnits="objectBoundingBox"
    x="-10%" y="-10%" width="120%" height="120%">
    <feGaussianBlur in="SourceAlpha" stdDeviation="2"
      result="BlurAlpha"/>
    <feOffset in="BlurAlpha" dx="2" dy="2"
      result="OffsetBlurAlpha"/>
    <feMerge>
      <feMergeNode in="OffsetBlurAlpha"/>
      <feMergeNode in="SourceGraphic"/>
    </feMerge>
  </filter>
</defs>
```

Then you could utilize the filter to display text:

```
<g id="shadow" style="filter:url(#DropShadowFilter);
  &st4; &st5;">
  <text transform="matrix(1 0 0 1 20 150)">Text</text>
</g>
```

or to render vector-based graphics:

```
<g id="square" style="&st2;">
  <path style="filter:url(#DropShadowFilter);"
        d="M114.5,87.5h-84v-63h84v63z"/>
</g>
```

The SVG `DropShadowFilter` is applied to the text string, which is shown below.

1st Button
2nd SVG

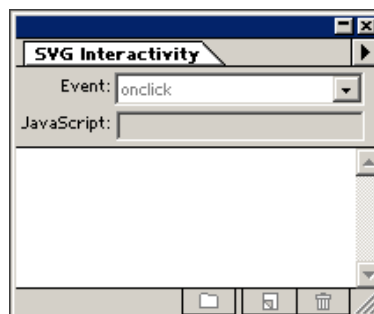
An SVG Filter Effect

Integrating JavaScript with SVG

A scripting language, such as JavaScript, opens unlimited functionality to an SVG document. In the simplest cases, pointer and keyboard movements (for example, if the mouse moves over or is clicked on an SVG element) may be assigned to SVG elements to invoke scripting functions. Scripts may be written that utilize the Document Object Model (DOM) to access and modify the structure of the SVG document, creating or deleting SVG elements, if needed.

The SVG Interactivity Palette

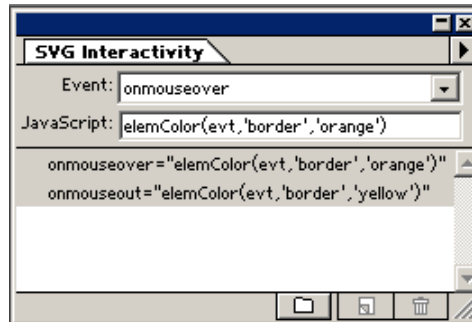
Adobe Illustrator provides the SVG Interactivity Palette to start to associate a JavaScript event to an object or container. To open the SVG Interactivity Palette, go to the menu bar and choose `Window > Show SVG Interactivity`.



SVG Interactivity Palette for JavaScript

Select your target object or container (layer or group). Be careful about attaching the event handler to your target, or you may inadvertently target a layer, when you mean to target a single element, or vice-versa.

Then in the SVG Interactivity Palette, choose an event and type a JavaScript function to associate to it. To finalize your entry, either hit return while focus remains in the JavaScript text field or click on the second button at the bottom of the palette. To add another entry, just choose another event; to delete an entry, click on the trash can at the bottom of the palette. In the palette shown below, mouse movement over the SVG element will cause the JavaScript function `elemColor(evt, 'border', 'orange')` to be invoked.

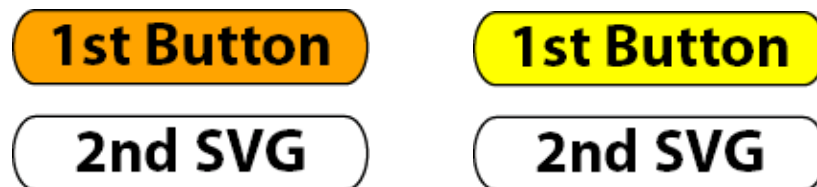


Entering JavaScript functions, using the SVG Interactivity Palette

Upon export, the event attribute `onmouseover` is assigned the value of a JavaScript function, and that attribute is added to the SVG element, resulting in something similar to this:

```
<g id="First_x0020_Button"
  onmouseover="elemColor(evt, 'border', 'orange')"...>
```

The event attribute `onmouseover` takes effect when the mouse pointer is moved over an element; the event attribute `onmouseout`, when the pointer is moved away from an element. In this example, when the mouse pointer is moved over the 1st Button, the color of the background of the button (below left) is turned to orange. When the pointer is removed, the color is restored to yellow (below right).

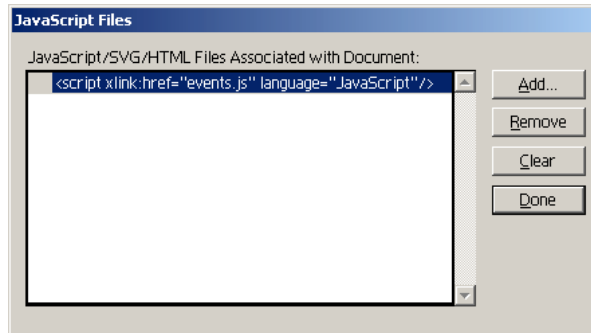


Mouse Actions Scripted to Cause Color Change

The next step is to specify where the source for `elemColor()` and similar JavaScript functions are found. You can manually edit the SVG file and insert `<script>` elements into it, or you can segregate the JavaScript outside the SVG document.

To put the JavaScript code in its own file, click on the first button at the bottom of the SVG Interactivity palette, which brings up a dialog box. If you choose *Add*, you are prompted for the

name of the external file with the JavaScript code. In the example below, the file name `events.js` has been entered:



Link JavaScript Files with the Exported SVG Document

which results in the following code in the SVG document:

```

<defs>
  <script xlink:href="events.js" language="JavaScript" >
  </script>
</defs>

```

Note that the `events.js` and other sample JavaScript files are available on the Adobe Illustrator 9.0 CD and from the [Adobe SVG Zone](#) web site.

Event Attributes

The following table lists and describes event attributes that are supported in SVG and may be attached to SVG elements. These event attributes are available in the drop-down menu in the SVG Interactivity Palette.

EVENT ATTRIBUTE	DESCRIPTION
<code>onload</code>	This event is triggered after the SVG document has been completely read and parsed. Often used to call one-time only initialization functions.
<code>onunload</code>	Occurs when the SVG document is removed from a window or frame
<code>onmouseover</code>	The pointer is moved onto an element.
<code>onmousemove</code>	The pointer is moved while it is over an element.
<code>onmouseout</code>	The pointer is moved away from an element.
<code>onmousedown</code>	The pointer button is pressed over an element.
<code>onmouseup</code>	The pointer button is released over an element.
<code>onclick</code>	When a mousedown and mouseup occur over the same screen location.

EVENT ATTRIBUTE	DESCRIPTION
onactivate	Activation may occur through a mouse click or keypress, depending upon the SVG element.
onfocusin	The element receives focus, such as selection by the pointer.
onfocusout	The element loses focus, often when another element receives focus.
onresize	The document view is resized.
onscroll	The document view is scrolled or panned.
onzoom	The zoom level is changed for the document.
onabort	When page loading is stopped before the element is completely loaded.
onerror	When an element does not load properly or other error occurs.
onbegin, onend, onrepeat	When an animation element starts, stops, or is replayed.

JavaScript, Beyond the Palette

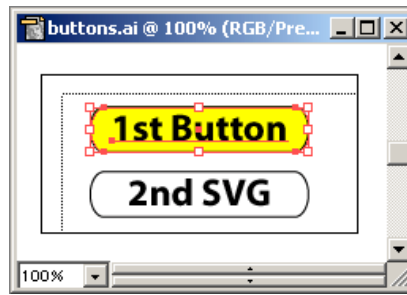
Writing scripting functions opens up SVG documents to a variety of applications, especially since methods of the Document Object Model (DOM) are available to access and manipulate the SVG structure itself. For example, in the [SVG Draw demonstration](#), JavaScript code controls the markup system. A JavaScript function clones SVG elements and inserts them into the SVG document hierarchy. Other JavaScript functions delete SVG elements from the SVG document or change their attribute values.

Like SVG documents, JavaScript code is also easily repurposed and reused. The SVG Draw markup demonstration can become the basis for annotating another SVG document. For instance, for an existing SVG mapping application, a programmer could integrate the SVG Draw JavaScript to add its annotation capabilities. Annotation would enable the user of the mapping application to highlight driving directions.

Linking SVG to URLs

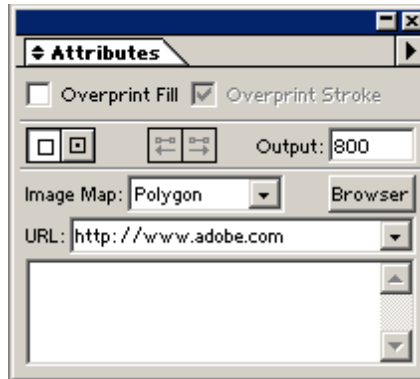
When creating artwork for Web pages, it's common to use image maps to delineate the shapes of regions or objects and to make URL hyperlinks for those shapes. However, with SVG, it's possible to designate SVG elements as links. Using SVG elements provides more flexibility than using an image map, because the linkable area adjusts to transformations of the SVG element—zoom in, and the clickable region scales accordingly.

To use Adobe Illustrator, have your illustration preview available. Open the Attributes palette (from the menu bar, choose Window > Show Attributes). Select the object you wish to link to, as shown below.



In Adobe Illustrator, Select the Object to be Assigned a Link

In the Attributes palette, select either polygon or rectangle for the image map shape. For SVG, there is no difference which shape you choose, because the link is applied to the object, not an image map, and either choice activates the URL field. Enter a URL to specify the link.



Assigning a URL Link with the Attributes Palette

For the exported SVG file, clicking anywhere over the star-shaped element causes the user to be connected to the URL. If the element has been scaled larger or smaller, the selectable region properly adjusts. Inside the SVG document, Adobe Illustrator has created an `<a>` element around the selected object and set the value for the `xlink:href` attribute:

```
<a xlink:href="http://www.adobe.com">
  <g id="First_x0020_Button" style="&st7;">
    ... deleted ...
  </g>
</a>
```

Another way to achieve a similar effect is to use the SVG Interactivity Palette to assign the JavaScript function `open()` to the `onclick` event attribute for a layer or object. Using the

SVG Interactivity Palette opens the URL in a new window; in contrast to using the Attributes palette to create an xlink, which browses to the URL in the same window.



Establishing a URL Link using the SVG Interactivity Palette

Tips and Techniques

Remove Unnecessary Layers and Groups

Every nested layer or group in Adobe Illustrator is exported as a `<g>` element in SVG. So by reducing the amount of nesting, there will be fewer SVG elements to traverse.

Remove Invisible or Off-Screen Objects and Layers

If an object or layer has been marked as invisible, it will still be exported to SVG. Unless there is some scripting that will interactively make the object or layer visible, it's better to remove it completely, or it will be exported to the SVG with the `display` property set to `none`.

Scrollbars

If you want to get scrollbars to appear automatically when the size of an image exceeds the size of the window, one of the easiest ways is to embed the SVG document in an HTML file.

Additional Resources

There is also a companion paper entitled *Using SVG with Adobe® GoLive® 5.0*, which discusses using Adobe GoLive to layout, produce, and manage SVG documents.

For more information on SVG, check out the following web sites:

- The Adobe SVG Zone, created for Web designers who are considering developing sites with SVG. The Adobe SVG Viewer may be downloaded from here.

<http://www.adobe.com/svg>

- A large, well-maintained list of SVG sites

<http://www.adobe.com/svg/community/external.html>

- An introduction to Scalable Vector Graphics

<http://www.xml.com/pub/a/2001/03/21/svg.html>

- A repository of SVG samples that demonstrate animation, interactivity, and other techniques

<http://www.adobe.com/svg/demos/samples.html>

- The official SVG 1.0 Specification

<http://www.w3.org/TR/SVG/>

