Scalable Vector Graphics Svg - 2 - Reference

By John Bentley

Static

Overview

(W3C 2016, "Scalable Vector Graphics (SVG) 2", https://www.w3.org/TR/SVG2/), https://www.w3.org/TR/SVG2/ ...

> SVG is a language for describing two-dimensional graphics in XML [XML10]. SVG allows for three types of graphic objects: vector graphic shapes (e.g., paths consisting of straight lines and curves), images and text.

(W3C 2016, "Scalable Vector Graphics (SVG) 2", https://www.w3.org/TR/SVG2/) https://www.w3.org/TR/SVG2/intro.html#AboutSVG

SVG can be used in three contexts:

1. As a standalone document.

```
// \images\svg-standalone.svg
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
  "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg xmlns="http://www.w3.org/2000/svg"
     version="1.1" width="5cm" height="5cm">
  <title>Two groups, each of two rectangles</title>
  <g id="group1" fill="red">
    <rect x="1cm" y="1cm" width="1cm" height="1cm"/>
    <rect x="3cm" y="1cm" width="1cm" height="1cm"/>
  </g>
  <g id="group2" fill="blue">
    <rect x="1cm" y="3cm" width="1cm" height="1cm"/>
    <rect x="3cm" y="3cm" width="1cm" height="1cm"/>
  </g>

  <!-- Show outline of canvas using 'rect' element -->
  <rect x=".01cm" y=".01cm" width="4.98cm" height="4.98cm"
        fill="none" stroke="blue" stroke-width=".02cm"/>
</svg>
```

(W3C 2016, "Scalable Vector Graphics (SVG) 2", https://www.w3.org/TR/SVG2/) https://www.w3.org/TR/SVG2/struct.html C:\Users\John\Documents\Sda\Code\web\Examples\SvgExamples\images\svg-standalone.svg

2. Embedded as an "SVG document fragment" in an XML document (like (HTML)); and

```
<body>
  <h1>Scalable Vector Graphics (SVG) - Embedded Document Fragment</h1>

  <svg xmlns="http://www.w3.org/2000/svg" version="1.1" height="250px">
    <rect x="25" y="25" width="200" height="200" fill="lime" stroke-width="4"
stroke="pink" />
    <circle cx="125" cy="125" r="75" fill="orange" />
    <polyline points="50,150 50,200 200,200 200,100" stroke="red" stroke-width="4"
fill="none" />
    <line x1="50" y1="50" x2="200" y2="200" stroke="blue" stroke-width="4" />
  </svg>

</body>
```

(W3C 2016, "Scalable Vector Graphics (SVG) 2", https://www.w3.org/TR/SVG2/)
https://www.w3.org/TR/SVG2/struct.html
C:\Users\John\Documents\Sda\Code\web\Examples\SvgExamples\svg-embedded-document-fragment.xhtml

3. A standalone document that is referenced from within XHTML, in effect importing it.

```
<body>
  <h1>Scalable Vector Graphics (SVG) - Imported Standalone</h1>

  <img src="images/svg-standalone.svg" alt="A img based svg inclusion" />

</body>
```

(W3C 2016, "Scalable Vector Graphics (SVG) 2", https://www.w3.org/TR/SVG2/)
https://www.w3.org/TR/SVG2/struct.html
C:\Users\John\Documents\Sda\Code\web\Examples\SvgExamples\svg-imported-standalone.xhtml

Harnessing

Descriptive elements

The `desc` and `title` elements provide "descriptive information" about their parent.

```
<body>
  <h1>Scalable Vector Graphics (SVG) - Embedded Document Fragment</h1>

  <svg xmlns="http://www.w3.org/2000/svg" width="600px" height="400px">
    <title>My SVG title</title>
    <desc>A description</desc>
    <rect x="25" y="25" width="550" height="350" fill="white" stroke-width="4"
stroke="pink" />
    ...
  </svg>

</body>
```

C:\Users\John\Documents\Sda\Code\web\Examples\SvgExamples\svg-embedded-document-fragment.xhtml
(W3C 2016, "Scalable Vector Graphics (SVG) 2", https://www.w3.org/TR/SVG2/), "5.8. The 'desc' and 'title' elements", https://www.w3.org/TR/SVG2/struct.html#DescriptionAndTitleElements

The `metadata` element is also a descriptive element.

Element reuse

Reuse can be effected through the following elements: `defs`, `symbol` and `use`.

Elements can be reused by in three ways:

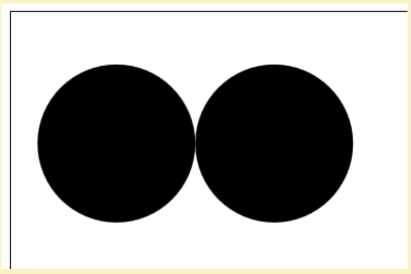
1. For an element that doesn't itself render: having an `id` attribute and being later referenced through a url.

```
<linearGradient id="MyGradient">...</linearGradient>

<rect style="fill:url(#MyGradient)"/>
```

2. For an element that, by default, does render:

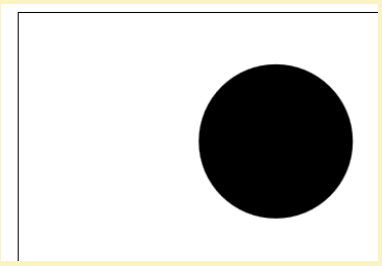
   a. Rendering as normal: having an `id` attribute and being later called by the `use` element.

```
<svg xmlns="http://www.w3.org/2000/svg" width="600px" height="400px">
  <circle id="myCircle" cx="100" cy="125" r="75" fill="black" />
  <use href="#myCircle" x="150" />
</svg>
```



   b. Not rendering: having an id attribute, being a child of `defs`, and being later called by the `use` element.

```
<svg xmlns="http://www.w3.org/2000/svg" width="600px" height="400px">
  <defs>
    <circle id="myCircle" cx="100" cy="125" r="75" fill="black" />
  </defs>
  <use href="#myCircle" x="150" />
</svg>
```



(W3C 2016, "Scalable Vector Graphics (SVG) 2", https://www.w3.org/TR/SVG2/), "5.4. Defining content for reuse, and the 'defs' element", https://www.w3.org/TR/SVG2/struct.html#Head
C:\Users\John\Documents\Sda\Code\web\Examples\SvgExamples\svg-resuse.xhtml

SVG element

In SVG2 the version attribute is not used on the SVG element.

```
// Do this
<svg xmlns="http://www.w3.org/2000/svg">
```

```
  <rect x="25" y="25" width="400px" height="400px" fill="blue"  />
</svg>

// Not this
<svg xmlns="http://www.w3.org/2000/svg" version="2">
  <rect x="25" y="25" width="400px" height="400px" fill="blue"  />
</svg>

// Use in prior SVG standards
<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
  <rect x="25" y="25" width="400px" height="400px" fill="blue"  />
</svg>
```

> The version attribute in the <svg> element is present in the SVG 1.1 and 1.2 specifications, but not in SVG 2.

(W3C 2016, "Scalable Vector Graphics (SVG) 2", https://www.w3.org/TR/SVG2/), "The 'svg' element", https://www.w3.org/TR/SVG2/struct.html#SVGElement
"Re: version attribute of svg element", 2016-04-09, Sairus Patel, https://lists.w3.org/Archives/Public/www-svg/2016Apr/0010.html

Coordinate systems and the viewBox attribute

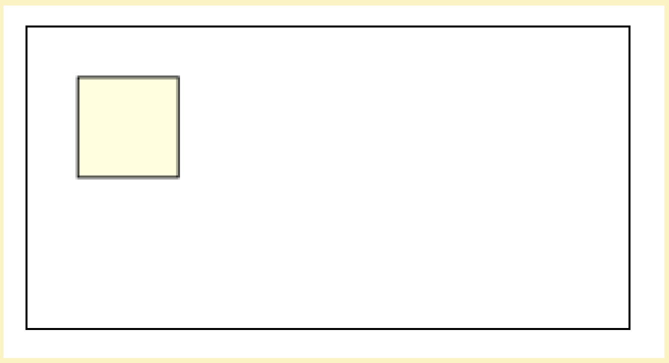
Various SVG contexts setup a "viewport", which establishes two coordinate systems: a **local coordinate system** and a **viewport coordinate system**. The local coordinate system is that which all child element "user units", which don't have an associated unit symbol, are made in reference to. The viewport coordinate system establishes the size of the containing element and transforms (scales) the local coordinate system to match the size of itself (the stipulated width and height of the viewport coordindate system). All child elements are correspondingly transformed (scaled).

The **local coordinate system** is defined by the `viewBox` attribute. The **viewport coordinate system** is defined by the `width` and `height` attributes (if absent, 300px and 150px respectively). Initially the local coordinate system and viewport coordinate system are identical.

(W3C 2016, "Scalable Vector Graphics (SVG) 2", https://www.w3.org/TR/SVG2/), "Chapter 8: Coordinate Systems, Transformations and Units", https://www.w3.org/TR/SVG2/coords.html

Initial example. The local coordinate system and viewport coordinate system are not explicitly defined. Both are identical. By default the viewport is 300px by 150px. In subsequent examples the style block will apply but not shown.

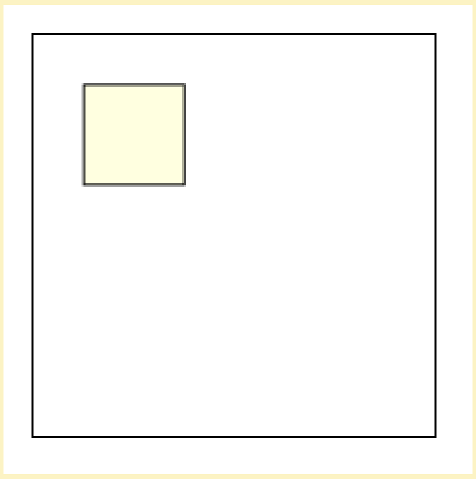
```
<svg xmlns="http://www.w3.org/2000/svg">
  <style>
    /* <![CDATA[ */
    rect {
      fill: lightyellow;
      stroke-width: 1;
      stroke: black;
    }
    /* ]]> */
  </style>
  <rect x="25" y="25" width="50" height="50" />
</svg>
```



C:\Users\John\Documents\Sda\Code\web\Examples\SvgExamples\svg-coordinate-system.xhtml

Alter the viewport dimensions without altering local coordinate system.

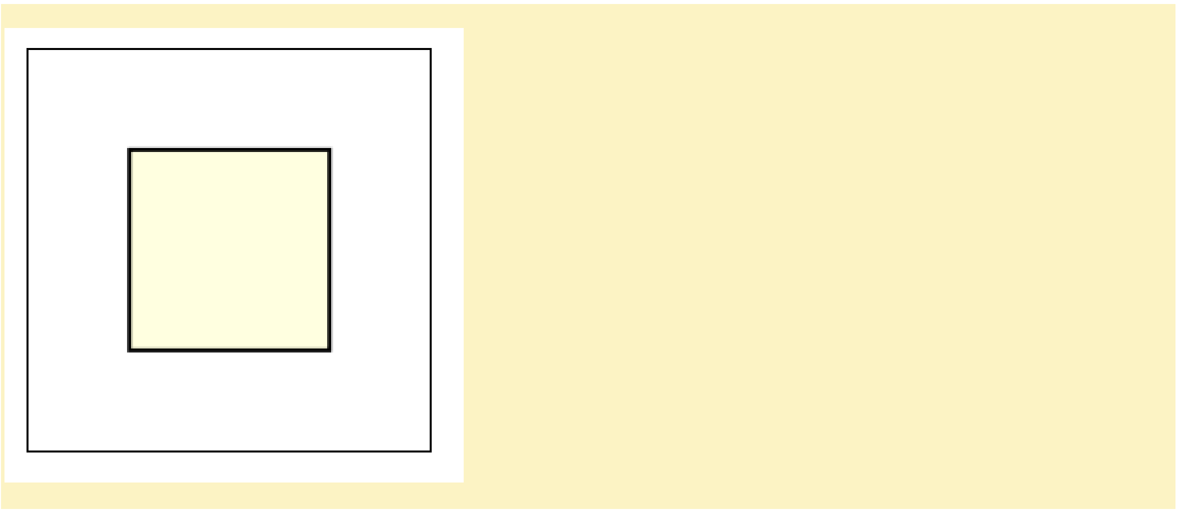
```
<svg xmlns="http://www.w3.org/2000/svg" width="200px" height="200px">
  <rect x="25" y="25" width="50" height="50" />
</svg>
```



C:\Users\John\Documents\Sda\Code\web\Examples\SvgExamples\svg-coordinate-system.xhtml

Scale the child object to fit the viewport by defining a local coordinate system through the `viewBox` attribute.

```
<svg xmlns="http://www.w3.org/2000/svg" width="200px" height="200px" viewBox="0 0 100 100">
  <rect x="25" y="25" width="50" height="50" />
</svg>
```



C:\Users\John\Documents\Sda\Code\web\Examples\SvgExamples\svg-coordinate-system.xhtml

Styling

Intro

Styling in SVG can be achieved either through through presentation attributes or CSS styling.

(W3C 2016, "Scalable Vector Graphics (SVG) 2", https://www.w3.org/TR/SVG2/), "Chapter 6: Styling", https://www.w3.org/TR/SVG2/styling.html

The use of presentational attributes is deprecated.

> In the future, any new properties that apply to SVG content will not gain presentation attributes. Therefore, authors are suggested to use styling properties, either through inline 'style' properties or style sheets, rather than presentation attributes, for styling SVG content.

(W3C 2016, "Scalable Vector Graphics (SVG) 2", https://www.w3.org/TR/SVG2/), "6.6. Presentation attributes", https://www.w3.org/TR/SVG2/styling.html#PresentationAttributes

Presentation attribute styling

Style SVG through presentation attributes as follows ...

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
<head>
  <meta charset="utf-8" />
  <title>Scalable Vector Graphics (SVG) - Styling</title>

  <link rel="stylesheet" href="style/html5-basic.css" />

</head>
<body>
  <h1>Scalable Vector Graphics (SVG) - Styling</h1>
  <svg xmlns="http://www.w3.org/2000/svg" width="600px" height="400px">
    <circle cx="75" cy="100" r="50" stroke="green" fill="rgb(29, 183, 255)" stroke-
width="4" />
  </svg>

</body>
</html>
```

(W3C 2016, "Scalable Vector Graphics (SVG) 2", https://www.w3.org/TR/SVG2/), "Chapter 6: Styling", https://www.w3.org/TR/SVG2/styling.html
C:\Users\John\Documents\Sda\Code\web\Examples\SvgExamples\svg-styling.xhtml

For a list of presentation attributes see ...

(W3C 2016, "Scalable Vector Graphics (SVG) 2", https://www.w3.org/TR/SVG2/), "6.6. Presentation attributes", https://www.w3.org/TR/SVG2/styling.html#PresentationAttributes

CSS styling

Style SVG through CSS in three in-document ways:

1. CSS outside of the SVG element.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
<head>
  <meta charset="utf-8" />
  <title>Scalable Vector Graphics (SVG) - Styling</title>

  <link rel="stylesheet" href="style/html5-basic.css" />

  <style>
    /* <![CDATA[ */
    circle {
      stroke: green;
      fill: rgb(29, 183, 255);
      stroke-width: 4;
    }
    /* ]]> */
  </style>

</head>
<body>
  <h1>Scalable Vector Graphics (SVG) - Styling</h1>

  <svg xmlns="http://www.w3.org/2000/svg" width="600px" height="400px">
    <circle cx="75" cy="100" r="50" />
  </svg>

</body>
</html>
```

C:\Users\John\Documents\Sda\Code\web\Examples\SvgExamples\svg-styling.xhtml

2. With a style block as a child of the SVG element.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
<head>
  <meta charset="utf-8" />
  <title>Scalable Vector Graphics (SVG) - Styling</title>

  <link rel="stylesheet" href="style/html5-basic.css" />
```

```
</head>
<body>
  <h1>Scalable Vector Graphics (SVG) - Styling</h1>

  <svg xmlns="http://www.w3.org/2000/svg" width="600px" height="400px">
    <style>
      /* <![CDATA[ */
      circle {
        stroke: green;
        fill: rgb(29, 183, 255);
        stroke-width: 4;
      }
      /* ]]> */
    </style>
    <circle cx="75" cy="100" r="50" />
  </svg>

</body>
</html>
```

C:\Users\John\Documents\Sda\Code\web\Examples\SvgExamples\svg-styling.xhtml

3. A style attribute on the shape element.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
<head>
  <meta charset="utf-8" />
  <title>Scalable Vector Graphics (SVG) - Styling</title>

  <link rel="stylesheet" href="style/html5-basic.css" />

</head>
<body>
  <h1>Scalable Vector Graphics (SVG) - Styling</h1>

  <svg xmlns="http://www.w3.org/2000/svg" width="600px" height="400px">
    <circle cx="75" cy="100" r="50" style="stroke: green; fill: rgb(29, 183, 255);
stroke-width: 4;" />
  </svg>

</body>
</html>
```

C:\Users\John\Documents\Sda\Code\web\Examples\SvgExamples\svg-styling.xhtml

Style SVG through css in three external document ways:

1. A HTML link outside of the SVG element.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
<head>
  <meta charset="utf-8" />
  <title>Scalable Vector Graphics (SVG) - Styling</title>

  <link rel="stylesheet" href="style/rainbows.css" />

</head>
<body>
  <h1>Scalable Vector Graphics (SVG) - Styling</h1>

  <svg xmlns="http://www.w3.org/2000/svg" width="600px" height="400px">
    <circle cx="75" cy="100" r="50" />
  </svg>

</body>
</html>
```

2. A HTML link inside of the SVG element. Include the xhtml namespace.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
<head>
  <meta charset="utf-8" />
  <title>Scalable Vector Graphics (SVG) - Styling</title>

</head>
<body>
  <h1>Scalable Vector Graphics (SVG) - Styling</h1>

  <svg xmlns="http://www.w3.org/2000/svg" width="600px" height="400px">
    <link rel="stylesheet" href="style/rainbows.css"
xmlns="http://www.w3.org/1999/xhtml" />
    <circle cx="75" cy="100" r="50" />
  </svg>

</body>
</html>
```

(W3C 2016, "Scalable Vector Graphics (SVG) 2", https://www.w3.org/TR/SVG2/), "6.3. External style sheets: the effect of the HTML 'link' element", https://www.w3.org/TR/SVG2/styling.html#LinkElement

3. Through CSS imports.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
<head>
  <meta charset="utf-8" />
  <title>Scalable Vector Graphics (SVG) - Styling</title>

</head>
<body>
  <h1>Scalable Vector Graphics (SVG) - Styling</h1>

  <svg xmlns="http://www.w3.org/2000/svg" width="600px" height="400px">
    <style>
      /* <![CDATA[ */
      @import url(style/rainbows.css);
      /* ]]> */
    </style>
    <circle cx="75" cy="100" r="50" />
  </svg>

</body>
</html>
```

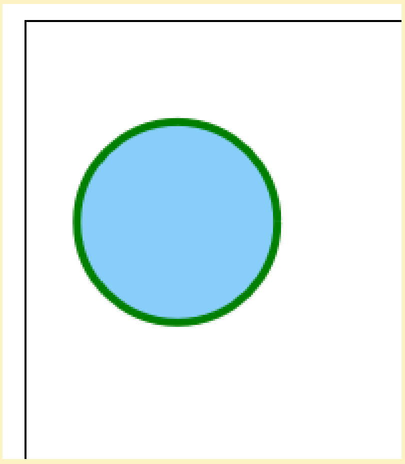
(W3C 2016, "Scalable Vector Graphics (SVG) 2", https://www.w3.org/TR/SVG2/), "6.3. External style sheets: the effect of the HTML 'link' element", https://www.w3.org/TR/SVG2/styling.html#LinkElement

C:\Users\John\Documents\Sda\Code\web\Examples\SvgExamples\svg-styling.xhtml

Precendence

Where both a presentational attribute and a CSS property applies, the CSS takes precedence.

```
<svg xmlns="http://www.w3.org/2000/svg" width="600px" height="400px">
  <style>
    /* <![CDATA[ */
    circle {
      stroke: green;
      fill: lightskyblue;
      stroke-width: 4;
    }
    /* ]]> */
  </style>
  <circle cx="75" cy="100" r="50" fill="white" />
</svg>
```

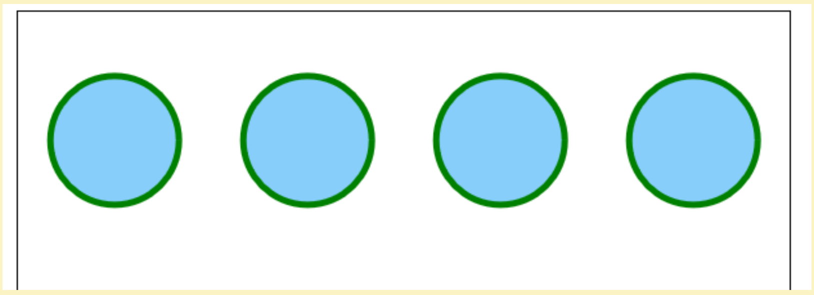


C:\Users\John\Documents\Sda\Code\web\Examples\SvgExamples\svg-styling.xhtml

The g (grouping) element

The grouping element, g, can be used to operate on several child elements at once.

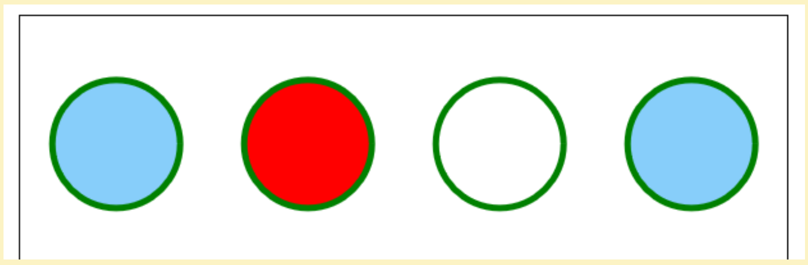
```
<svg xmlns="http://www.w3.org/2000/svg" width="600px" height="400px">
  <!-- Top Row -->
  <g stroke-width="5" fill="lightskyblue" stroke="green">
    <circle cx="075" cy="100" r="50" />
    <circle cx="225" cy="100" r="50" />
    <circle cx="375" cy="100" r="50" />
    <circle cx="525" cy="100" r="50" />
  </g>
</svg>
```



C:\Users\John\Documents\Sda\Code\web\Examples\SvgExamples\svg-grouping-element-g.xhtml
(W3C 2016, "Scalable Vector Graphics (SVG) 2", https://www.w3.org/TR/SVG2/), "5.2. Grouping: the 'g' element", https://www.w3.org/TR/SVG2/struct.html#Groups

For the g, grouping, element any styling done on the children takes precedence.

```
<svg xmlns="http://www.w3.org/2000/svg" width="600px" height="400px">
  <!-- Top Row -->
  <g stroke-width="5" fill="lightskyblue" stroke="green">
    <circle cx="075" cy="100" r="50" />
    <circle cx="225" cy="100" r="50" fill="red" />
    <circle cx="375" cy="100" r="50" fill="white" />
    <circle cx="525" cy="100" r="50" />
  </g>
</svg>
```



C:\Users\John\Documents\Sda\Code\web\Examples\SvgExamples\svg-grouping-element-g.xhtml
(W3C 2016, "Scalable Vector Graphics (SVG) 2", https://www.w3.org/TR/SVG2/), "5.2. Grouping: the 'g' element", *https://www.w3.org/TR/SVG2/struct.html#Groups*

Embedded content

SVG support embedded content via: SVG's `image` and `foreignObject` elements; and HTML's `video`, `audio`, `iframe` and `canvas` elements.

(W3C 2016, "Scalable Vector Graphics (SVG) 2", https://www.w3.org/TR/SVG2/), "Chapter 12: Embedded Content", *https://www.w3.org/TR/SVG2/embedded.html*

> The 'foreignObject' element allows for inclusion of elements in a non-SVG namespace which is rendered within a region of the SVG graphic using other user agent processes. ... A 'foreignObject' may be used in conjunction with the 'switch' element and the 'requiredExtensions' attribute to provide proper checking for user agent support and provide an alternate rendering in case user agent support is not available.

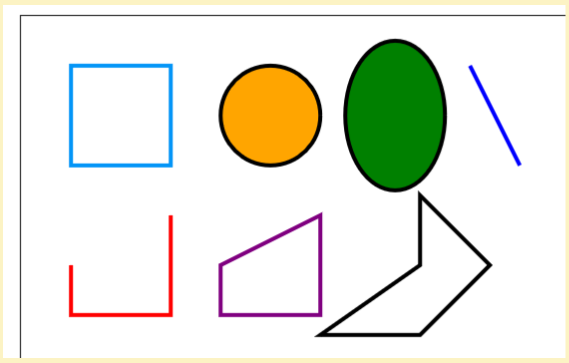
```
<svg xmlns="http://www.w3.org/2000/svg">
  <switch>
    <!-- Process the embedded XHTML if the requiredExtensions attribute
        evaluates to true (i.e., the user agent supports XHTML
        embedded within SVG). -->
    <foreignObject width="600px" height="400px">
      <!-- XHTML content goes here -->
        <p xmlns="http://www.w3.org/1999/xhtml" style="inline-size: 200px">Here is a
paragraph that requires word wrap. Foreign Object.</p>
    </foreignObject>
    <!-- Else, process the following alternate SVG.
        Note that there are no testing attributes on the 'text' element.
        If no testing attributes are provided, it is as if there
        were testing attributes and they evaluated to true.-->
    <text font-size="10" font-family="Verdana">
      <tspan x="10" y="10">Here is a paragraph that</tspan>
      <tspan x="10" y="20">requires word wrap. Fallback</tspan>
    </text>
  </switch>
</svg>
```

C:\Users\John\Documents\Sda\Code\web\Examples\SvgExamples\svg-foreignObject.xhtml
(W3C 2016, "Scalable Vector Graphics (SVG) 2", https://www.w3.org/TR/SVG2/), "The 'foreignObject' element", *https://www.w3.org/TR/SVG2/embedded.html#ForeignObjectElement*

Basic Shapes

Basic shapes.

```
<svg xmlns="http://www.w3.org/2000/svg" width="600px" height="400px" stroke-width="4">
  <!-- Top Row -->
  <rect x="50" y="50" width="100" height="100" fill="white"  stroke="rgb(3, 149, 247)" />
  <circle cx="250" cy="100" r="50" fill="orange" stroke="black"  />
  <ellipse cx="375" cy="100" rx="50" ry="75" fill="green" stroke="black"  />
  <line x1="450" y1="50" x2="500" y2="150" stroke="blue"  />

  <!-- Bottom Row -->
  <polyline points="50,250 50,300 150,300 150,200" stroke="red"  fill="none" />
  <polygon points="200,250 200,300 300,300 300,200" stroke="purple"  fill="none" />

  <!-- Not a "basic shape" as such -->
  <path d="M 400,250 l 0,-70 l 70,70 0,0 -70,70 h-100Z"  stroke="black" fill="none" />
</svg>
```



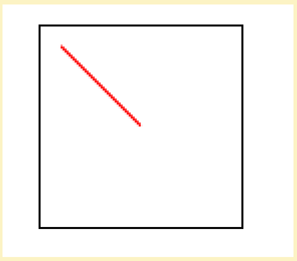
C:\Users\John\Documents\Sda\Code\web\Examples\SvgExamples\svg-basic-shapes.xhtml
(W3C 2016, "Scalable Vector Graphics (SVG) 2", https://www.w3.org/TR/SVG2/), "Chapter 10: Basic Shapes", https://www.w3.org/TR/SVG2/shapes.html

Path

Overview

The SVG path element is a powerful element that serves as a basis for all sorts of animations.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
<head>
  <meta charset="utf-8" />
  <title>Scalable Vector Graphics (SVG) - Path Element</title>

  <link rel="stylesheet" href="style/html5-basic.css" />

  <style>
    /* <![CDATA[ */

    svg {
      /* Don't use a border in production to prevent fallback ugliness */
      border: 1px solid black;
    }

    path {
      fill: none;
      stroke: red;
      stroke-width: 1px;
    }

    /* ]]> */
  </style>

</head>
<body>
  <h1>Scalable Vector Graphics (SVG) - Path Element</h1>

  <svg xmlns="http://www.w3.org/2000/svg" version="1.1" width="100px" height="100px">
    <path d="m 10,10 L 50,50 z" />
  </svg>

</body>
</html>
```



C:\Users\John\Documents\Sda\Code\web\Examples\SvgExamples\svg-embedded-path-element.xhtml

The value of the path's d (path data) attribute is composed of several commands. The syntax rules:

1. Each command is composed of a letter followed be zero, one, or two co-ordinates.

```
M300,70 l 0,-70
```

2. Either a comma or space MUST separate co-ordinates of a command.

```
M 300 70 l 0 -70       // Valid
M 30070 l 0 -70        // Invalid
```

3. Superfluous white space can be eliminated.

```
M300 70 l0 -7          // Note "l" is an ell not a one.
```

4. If the command letter of subsequent commands is the same, it can be eliminated.

```
M 300,70 l 0,-70 l 70,70      // Valid
M 300,70 l 0,-70   70,70      // Second ell eliminated.
```

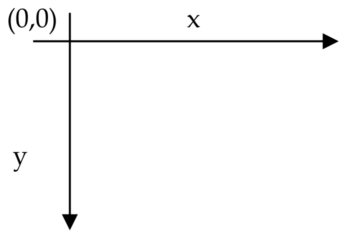
5. A capital letter means the coordinates are absolute, a lower case letter means the coordinates are relative to the previous coordinates.

```
M 300,70 L 0,-70       // 0,-70 are absolute
M 300,70 l 0,-70       // 0,-70 are relative.
```

6. The path value can be split across multiple lines.

```
<path d="M 300,70 l 0,-70 l 70,70
              0,0 -70,70 h-100" />
```

7. The co-ordinate system is as follows:



(W3C 2016, "Scalable Vector Graphics (SVG) 2", https://www.w3.org/TR/SVG2/), "Path Data", https://www.w3.org/TR/SVG2/paths.html#PathData
C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyThirdApp\app\src\main\res\drawable\vector_drawable.xml

Style guide for pathData. Use commas to separate a command's coordinates; except for the first and last command space separate the command letter from coordinates; eliminate repeated command letter.

```
M 300,70 l 0,-70 l 70,70 0,0 -70,70 h-100Z
```

C:\Users\John\Documents\Sda\Code\Android\Examples\Training\MyThirdApp\app\src\main\res\drawable\vector_drawable.xml

The path data commands:

Command	**Name**	**Paramaters**
M (absolute) m (relative)	moveto	(x y)+
Z or z	closepath	(none)
L (absolute) l (relative)	lineto	(x y)+
H (absolute) h (relative)	horizontal lineto	x+
V (absolute) v (relative)	vertical lineto	y+
C (absolute) c (relative)	cubic Bézier curveto	(x1 y1 x2 y2 x y)+
S (absolute) s (relative)	shorthand/smooth cubic Bézier curveto	(x2 y2 x y)+
Q (absolute) q (relative)	quadratic Bézier curveto	(x1 y1 x y)+

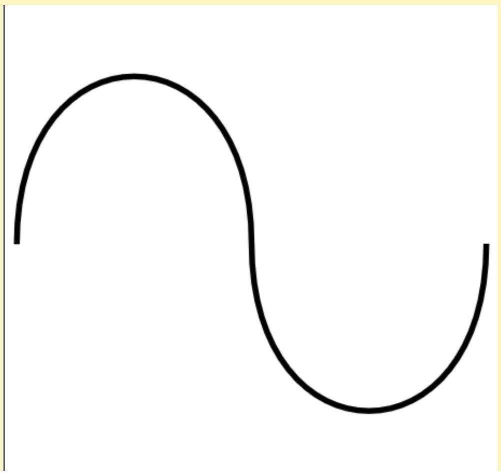
T (absolute) t (relative)	Shorthand/smooth quadratic Bézier curveto	(x y)+
A (absolute) a (relative)	elliptical arc	(rx ry x-axis-rotation large-arc-flag sweep-flag x y)+

(W3C 2016, "Scalable Vector Graphics (SVG) 2", https://www.w3.org/TR/SVG2/), "Path Data", https://www.w3.org/TR/SVG2/paths.html#PathData

Curve command syntax

C (absolute) c (relative), cubic Bézier curveto, (x1 y1 x2 y2 x y)+.

```
<svg width="420" height="420" xmlns="http://www.w3.org/2000/svg"
xmlns:svg="http://www.w3.org/2000/svg">
  <!-- Manual Command  -->
  <path id="manualBezierCurve" d="m10,210 c0,-190 200,-190 200,0 c0,190 200,190 200,0"
    stroke-width="5" stroke="#000000" fill="none" />
</svg>
```



Draws a cubic Bézier curve from the current point to (x,y) using (x1,y1) as the control point at the beginning of the curve and (x2,y2) as the control point at the end of the curve. C (uppercase) indicates that absolute coordinates will follow; c (lowercase) indicates that relative coordinates will follow. Multiple sets of coordinates may be specified to draw a polybézier. At the end of the command, the new current point becomes the final (x,y) coordinate pair used in the polybézier.

(W3C 2016, "Scalable Vector Graphics (SVG) 2", https://www.w3.org/TR/SVG2/), "Path Data", https://www.w3.org/TR/SVG2/paths.html#PathData

Curves, manual creation.

To create a manual beizer curve:

- Create a 420px by 420px Inkscape document. Grid on. Guides on.
- Draw your beizer curve in Inkscape.
- Use the measurement tool to determine start points, control points, end points, etc.
- Apply calculations to an SVG in C:\Users\John\Documents\Sda\Code\web\Examples\SvgExamples\svg-embedded-path-data.xhtml
- Open in a browser.

Text

Intro

> The geometric font characteristics are expressed in a coordinate system based on the EM box.

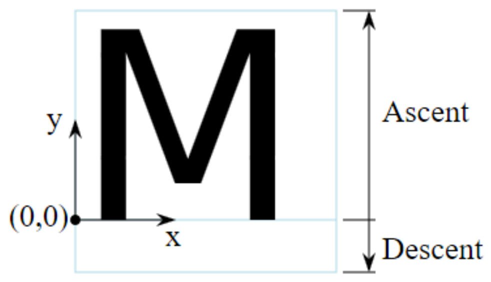
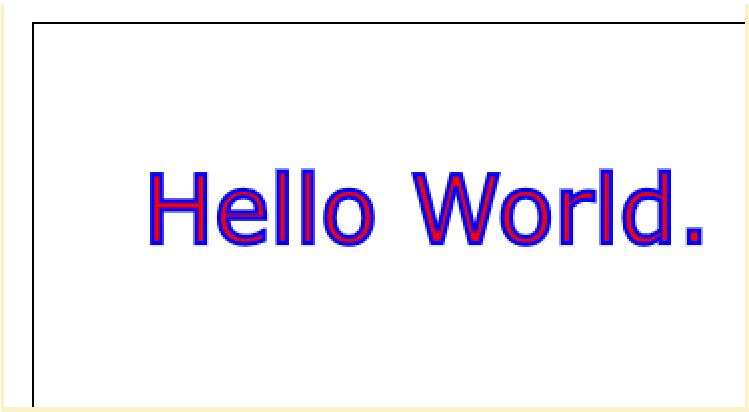


Figure 1 An 'M' inside an Em box (blue square). The 'M' sits on the baseline (blue line). The origin of the coordinate system is shown by the small black circle.

(W3C 2016, "Scalable Vector Graphics (SVG) 2", https://www.w3.org/TR/SVG2/), "11.1.3. Glyph metrics and layout", https://www.w3.org/TR/SVG2/text.html#GlyphsMetrics

Basic example using `text` element. Use x and y attributes for positioning.

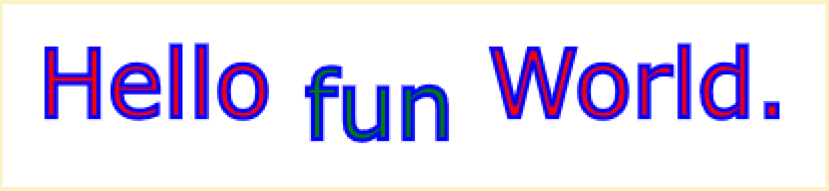
```
<svg xmlns="http://www.w3.org/2000/svg">
  <style>
    /* <![CDATA[ */
    text {
      font-family: Verdana, Geneva, Tahoma, sans-serif;
      font-size: 64pt;
      fill: red;
      stroke: blue;
      stroke-width: 2;
    }
    /* ]]> */
  </style>
  <text x="50" y="100">Hello World.</text>
</svg>
```



C:\Users\John\Documents\Sda\Code\web\Examples\SvgExamples\svg-text.xhtml

The `tspan` element allows you to restyle or reposition a string within a `text` element (or another `tspan` element). Use dx and dy for positioning relative to containing block. Using a relative positioning, changes the positioning for the remaining text, even outside the current element, unless otherwise overridden.

```
<svg xmlns="http://www.w3.org/2000/svg">
  <style>
    /* <![CDATA[ */

    text, tspan {
      font-family: Verdana, Geneva, Tahoma, sans-serif;
      font-size: 32pt;
      fill: red;
      stroke: blue;
      stroke-width: 2;
    }

    #recolor {
      fill: green;
    }

    /* ]]> */
  </style>
  <text x="50" y="200">
    Hello
    <tspan id="recolor" dy="10">fun</tspan>
    <tspan dy="-10">World.</tspan>
  </text>
</svg>
```



C:\Users\John\Documents\Sda\Code\web\Examples\SvgExamples\svg-text.xhtml

You can rotate, but only from an attribute (not CSS).

```
<tspan id="recolor" dy="10" rotate="10">fun</tspan>
```

(W3C 2016, "Scalable Vector Graphics (SVG) 2", https://www.w3.org/TR/SVG2/), " Notes on 'x', 'y', 'dx', 'dy' and 'rotate'', https://www.w3.org/TR/SVG2/text.html#TSpanNotes

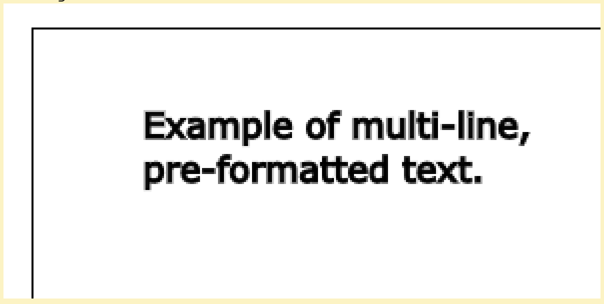
Text Layout (Wrapping, etc.)

You can layout text in three ways: preformatted; auto-wrapped; and text on path.

(W3C 2016, "Scalable Vector Graphics (SVG) 2", https://www.w3.org/TR/SVG2/), "Text layout – Introduction', https://www.w3.org/TR/SVG2/text.html#TextLayout

For preformatted text, don't use the inline-size property, and position your lines as desired.

```
<svg xmlns="http://www.w3.org/2000/svg">
  <style>
    /* <![CDATA[ */

    text, tspan {
      font-family: Verdana, Geneva, Tahoma, sans-serif;
      font-size: 12pt;
      fill: black;
    }

    /* ]]> */
  </style>
  <text x="50" y="50">
      Example of multi-line,
      <tspan x="50" y="70">pre-formatted text.</tspan>
    </text>
</svg>
```



*(W3C 2016, "Scalable Vector Graphics (SVG) 2", https://www.w3.org/TR/SVG2/), "Text layout – Introduction', https://www.w3.org/TR/SVG2/text.html#TextLayout
C:\Users\John\Documents\Sda\Code\web\Examples\SvgExamples\svg-text.xhtml*

Preformatted text using the `"white-space: pre;"` css style property. Doesn't handle special characters (would need to escape).

```
    <svg xmlns="http://www.w3.org/2000/svg">
      <text x="10" y="10" style="white-space: pre;">
Example of multi-line, auto-formatted text in
*text*, using *white-space: pre*.
Lorem, ipsum dolor sit amet consectetur adipisicing elit.
Nesciunt, delectus!.
      </text>
    </svg>
```

*(W3C 2016, "Scalable Vector Graphics (SVG) 2", https://www.w3.org/TR/SVG2/), " White space", https://www.w3.org/TR/SVG2/text.html#WhiteSpace
C:\Users\John\Documents\Sda\Code\web\Examples\SvgExamples\svg-text.xhtml*

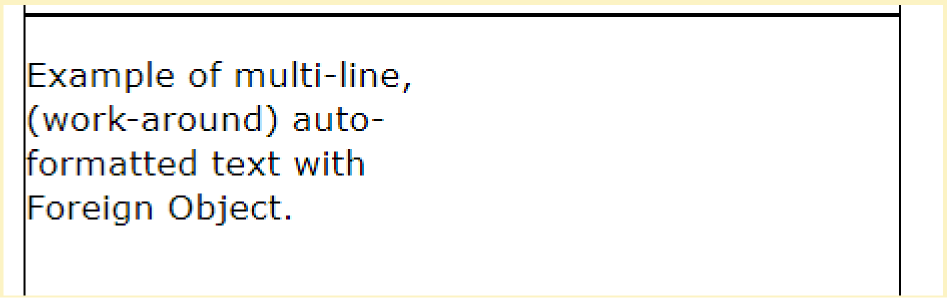
(Not working at 2018-08-13) For auto-wrapped text, set the inline-size property.

```
// According to spec you do this, instead, via the inline-size property. Doesn't work at
2018-08-11
<svg xmlns="http://www.w3.org/2000/svg">
  <text x="20" y="45" style="inline-size: 200px">
    Example of multi-line, auto-formatted text in
    <code>text</code>. Lorem, ipsum dolor sit amet consectetur adipisicing elit. Nesciunt,
delectus!.
  </text>
</svg>
```

(W3C 2016, "Scalable Vector Graphics (SVG) 2", https://www.w3.org/TR/SVG2/), "Text layout – Introduction', https://www.w3.org/TR/SVG2/text.html#TextLayout
C:\Users\John\Documents\Sda\Code\web\Examples\SvgExamples\svg-text.xhtml

(Work around) for auto-wrapped text, use `foreignObject` and regular XHTML.

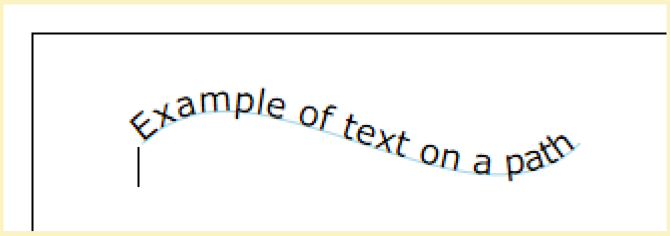
```
<svg xmlns="http://www.w3.org/2000/svg">
  <switch>
    <!-- Process the embedded XHTML if the requiredExtensions attribute
        evaluates to true (i.e., the user agent supports XHTML
        embedded within SVG). -->
    <foreignObject width="600px" height="400px">
      <!-- XHTML content goes here -->
        <p xmlns="http://www.w3.org/1999/xhtml" style="inline-size: 200px">Here is a
paragraph that requires word wrap. Foreign Object.</p>
      </foreignObject>
    <!-- Else, process the following alternate SVG.
        Note that there are no testing attributes on the 'text' element.
        If no testing attributes are provided, it is as if there
        were testing attributes and they evaluated to true.-->
    <text font-size="10" font-family="Verdana">
      <tspan x="10" y="10">Here is a paragraph that</tspan>
      <tspan x="10" y="20">requires word wrap. Fallback</tspan>
    </text>
  </switch>
</svg>
```


Example of multi-line, (work-around) auto-formatted text with Foreign Object.

(W3C 2016, "Scalable Vector Graphics (SVG) 2", https://www.w3.org/TR/SVG2/), "Text layout – Introduction', https://www.w3.org/TR/SVG2/text.html#TextLayout
C:\Users\John\Documents\Sda\Code\web\Examples\SvgExamples\svg-text.xhtml

To position text on a path do the following ...

```
<svg xmlns="http://www.w3.org/2000/svg">
  <path id="myPath" stroke="lightblue" fill="none" d="M 50,50 C 100,0 200,100 250,50" />
  <text>
    <textPath href="#myPath">Example of text on a path.</textPath>
  </text>
</svg>
```



(W3C 2016, "Scalable Vector Graphics (SVG) 2", https://www.w3.org/TR/SVG2/), "Text layout – Introduction', https://www.w3.org/TR/SVG2/text.html#TextLayout
C:\Users\John\Documents\Sda\Code\web\Examples\SvgExamples\svg-text.xhtml

(Workaround) For auto-wrapped text, set the inline-size property, via `foreignObject`.

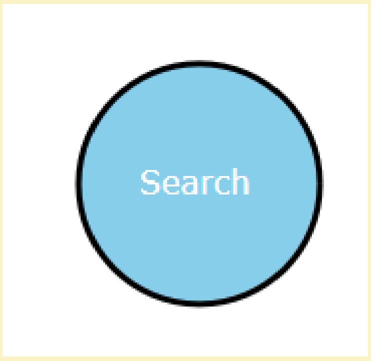
```
<svg xmlns="http://www.w3.org/2000/svg">
  <switch>
    <!-- Process the embedded XHTML if the requiredExtensions attribute
        evaluates to true (i.e., the user agent supports XHTML
        embedded within SVG). -->
    <foreignObject width="600px" height="400px">
      <!-- XHTML content goes here -->
        <p xmlns="http://www.w3.org/1999/xhtml" style="inline-size: 200px">Here is a
paragraph that requires word wrap. Foreign Object.</p>
    </foreignObject>
    <!-- Else, process the following alternate SVG.
        Note that there are no testing attributes on the 'text' element.
        If no testing attributes are provided, it is as if there
        were testing attributes and they evaluated to true.-->
    <text font-size="10" font-family="Verdana">
      <tspan x="10" y="10">Here is a paragraph that</tspan>
      <tspan x="10" y="20">requires word wrap. Fallback</tspan>
    </text>
  </switch>
</svg>
```

C:\Users\John\Documents\Sda\Code\web\Examples\SvgExamples\svg-foreignObject.xhtml

Linking

You can link out of SVG by wrapping SVG objects inside an `a` element.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
<head>
  <meta charset="utf-8" />
  <title>Scalable Vector Graphics (SVG) - Linking</title>

  <!-- Scaffold -->
  <style>
    /* <![CDATA[ */

    body {
      margin-left: 100px;
    }

    svg {
      border: 1px solid black;
      width: 600px;
      height: 400px;
    }

    /* ]]> */
  </style>
  <!-- Instructional code -->
  <style>
    /* <![CDATA[ */

    circle {
      fill: white;
      stroke: #000000;
      stroke-width: 3px;
    }

    a:hover circle {
      fill: skyblue;
    }

    a:hover text {
      fill: white;
    }

    text, tspan {
      font-family: Verdana, Geneva, Tahoma, sans-serif;
      font-size: 12pt;
      fill: black;
    }

    /* ]]> */
  </style>

</head>
<body>
  <h1>Scalable Vector Graphics (SVG) - Linking</h1>

  <p>Mouse over the button.</p>

  <svg xmlns="http://www.w3.org/2000/svg">
    <a href="http://www.google.com">
      <circle cx="250" cy="100" r="60" />
      <text x="220" y="105">Search</text>
    </a>
  </svg>

</body>
</html>
```



(W3C 2016, "Scalable Vector Graphics (SVG) 2", https://www.w3.org/TR/SVG2/), " Links out of SVG content: the 'a' element", https://www.w3.org/TR/SVG2/linking.html#Links
C:\Users\John\Documents\Sda\Code\web\Examples\SvgExamples\svg-linking.xhtml

SVG Transformations

Introduction

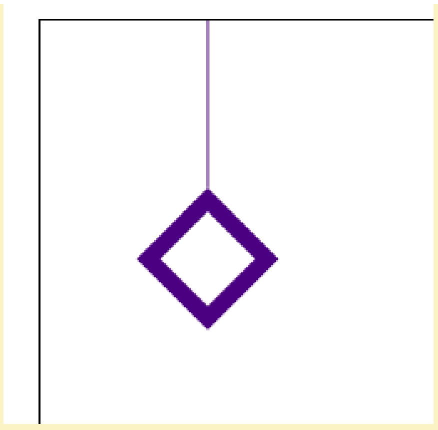
For an introduction to CSS Transformations see ..\..\Css\Reference\CascadingStyleSheets-Css-Reference.docx.

Having established your local (with `viewBox`) and viewport coordinate systems (with `width` and `height`), SVG transformations can be achieved either: with CSS transforms; or the SVG transform presentation attribute.

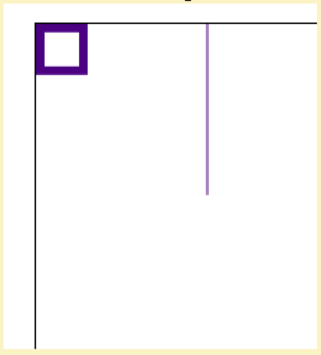
(W3C 2016, "Scalable Vector Graphics (SVG) 2", https://www.w3.org/TR/SVG2/), "The 'transform' property", https://www.w3.org/TR/SVG2/coords.html#TransformProperty ..\..\Css\Reference\CascadingStyleSheets-Css-Reference.docx (CSS Transforms)

Achieve SVG Transformations with the transform presentation attribute as follows.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
<head>
  <meta charset="utf-8" />
  <title>Scalable Vector Graphics (SVG) - Transforms</title>

  <style>
    /* <![CDATA[ */

    body {
      margin-left: 5em;
    }

    svg {
      border: 1px solid black;
    }

    svg * {
      stroke-width: 5px;
      stroke: indigo;
      fill: white;
    }
    /* ]]> */
  </style>

</head>
<body>
  <h1>Scalable Vector Graphics (SVG) - Transforms</h1>
  <svg xmlns="http://www.w3.org/2000/svg" width="600px" height="400px" >
    <rect x="2.5" y="2.5" width="25" height="25" transform="translate(100,100) rotate(45)
scale(2)" />
    <line x1="100" y1="0" x2="100" y2="100" style="stroke-width: 1" />
  </svg>
</body>
```



```
// Without transform
<rect x="2.5" y="2.5" width="25" height="25" />
```



C:\Users\John\Documents\Sda\Code\web\Examples\SvgExamples\svg-transforms.xhtml

Achieve SVG Transformations with CSS transforms as follows.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
<head>
  <meta charset="utf-8" />
  <title>Scalable Vector Graphics (SVG) - Transforms</title>

  <style>
    /* <![CDATA[ */

    body {
      margin-left: 5em;
    }

    svg {
      border: 1px solid black;
    }

    svg * {
      stroke-width: 5px;
      stroke: indigo;
      fill: white;
    }

    rect {
      transform: translate(100px,100px) rotate(45deg) scale(2);
    }
    /* ]]> */
  </style>

</head>
<body>
  <h1>Scalable Vector Graphics (SVG) - Transforms</h1>
  <svg xmlns="http://www.w3.org/2000/svg" width="600px" height="400px">
```

```
    <rect x="2.5" y="2.5" width="25" height="25" />
    <line x1="100" y1="0" x2="100" y2="100" style="stroke-width: 1" />
  </svg>
</body>
</html>
```

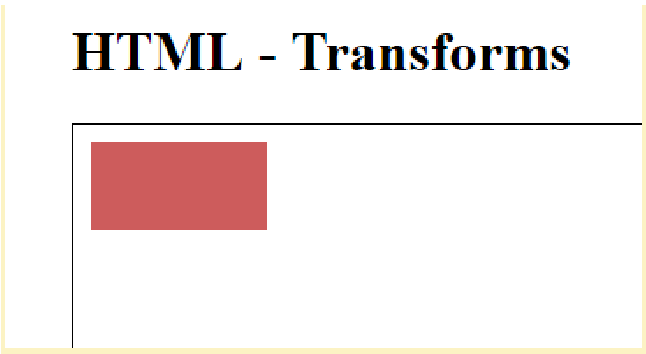
C:\Users\John\Documents\Sda\Code\web\Examples\SvgExamples\svg-transforms.xhtml

For code portability between transforming SVG objets V HTML object (in general) it is probably best to use CSS transforms.

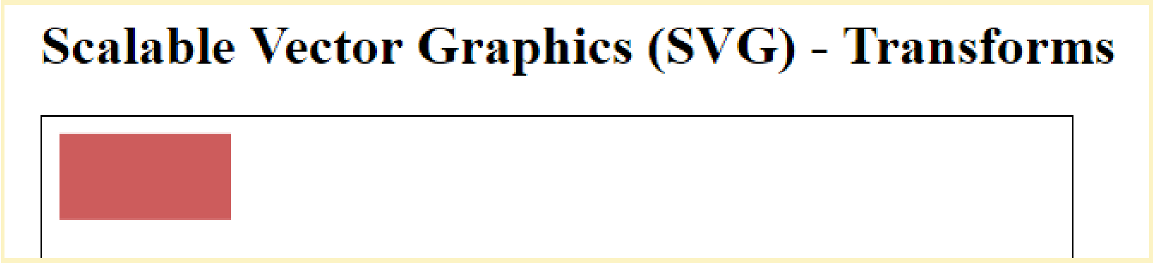
HTML V SVG transformation differences

To compare differences between HTML and SVG we use the following harnesses:

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
<head>
  <meta charset="utf-8" />

  <title>HTML - Transforms</title>

  <style>
    /* <![CDATA[ */

    body {
      margin-left: 100px;
    }

    main {
      position: relative;
      border: 1px solid black;
      width: 600px;
      height: 400px;
    }

    main * {

      background-color: indianred;
      color: black;
    }

    #target {
      /* border: 5px solid indigo; */
      position: absolute;
      width: 100px;
      height: 50px;
      top: 10px;
      left: 10px;
    }

    /* For transforms */
    #target {

      /* transform-origin: top left; */
      /* transform: translate(100px, 100px); */
    }

    /* ]]> */
  </style>

</head>
<body>
  <h1>HTML - Transforms</h1>
  <main>
    <div id="target" />
  </main>
</body>
</html>
```



```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
<head>
  <meta charset="utf-8" />

  <title>Scalable Vector Graphics (SVG) - Transforms</title>

  <style>
    /* <![CDATA[ */

    body {
      margin-left: 100px;
    }

    svg {
      border: 1px solid black;
      width: 600px;
      height: 400px;
    }

    svg * {
      fill: indianred;
    }

    /* For transforms */
    #target {
      /* transform-origin: center center;
      transform: translate(10px, 10px) rotate(358deg); */
    }

    /* ]]> */
  </style>

</head>
<body>
  <h1>Scalable Vector Graphics (SVG) - Transforms</h1>
  <svg xmlns="http://www.w3.org/2000/svg">
    <rect id="target" width="100" height="50" x="10px" y="10px" />
  </svg>
</body>
</html>
```



C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\css2017\transforms-html.xhtml
C:\Users\John\Documents\Sda\Code\web\Examples\SvgExamples\svg-transforms.xhtml

> In the HTML case, the origin of the element's system of coordinates is situated at the element's 50% 50% point ... In the SVG case however, the origin is situated at the 0 0 point of the SVG canvas.

The origin differences therefore have the following impacts:

- For translations: no difference.

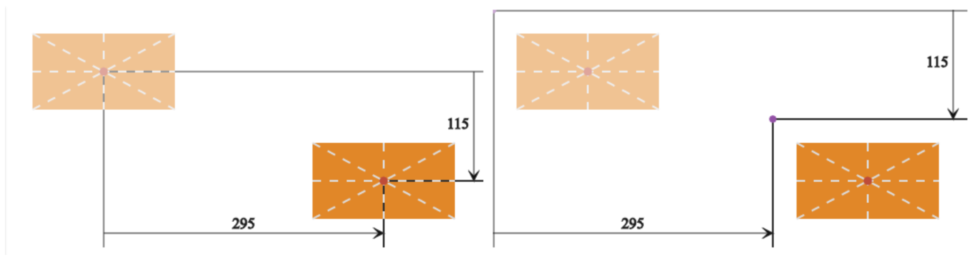


Figure 2 translate transform: HTML elements (left) vs SVG elements (right).

- For rotations: the following ...

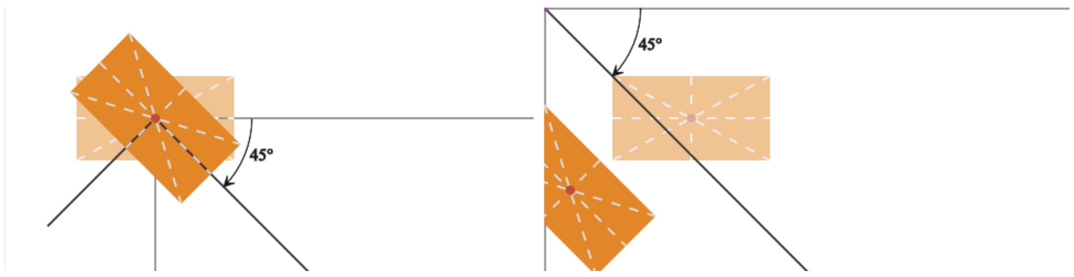
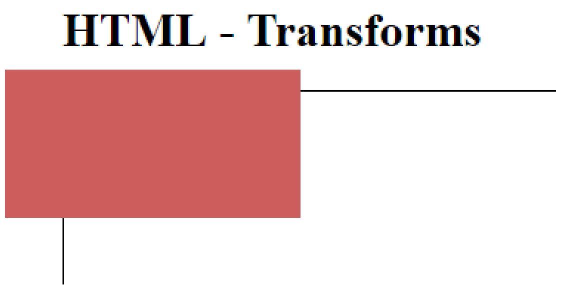
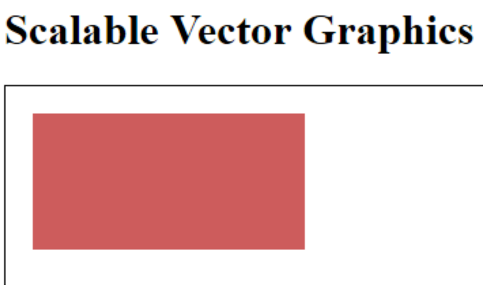


Figure 3 basic rotate transform: HTML elements (left) vs SVG elements (right).

(Tudor 2015, "Transforms on SVG Elements", https://css-tricks.com/transforms-on-svg-elements/)

- For scaleing: the following ...





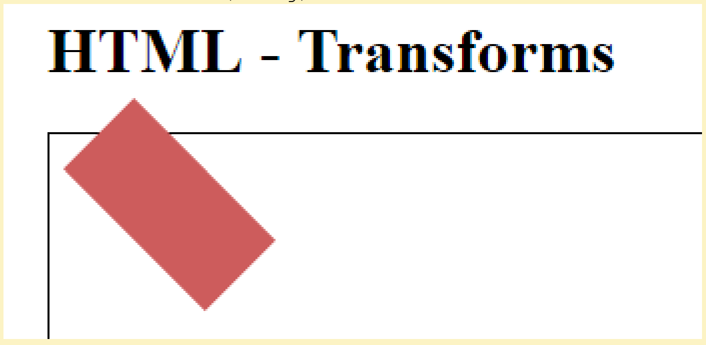
C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\css2017\transforms-html.xhtml
C:\Users\John\Documents\Sda\Code\web\Examples\SvgExamples\svg-transforms.xhtml

Taming SVG transformations

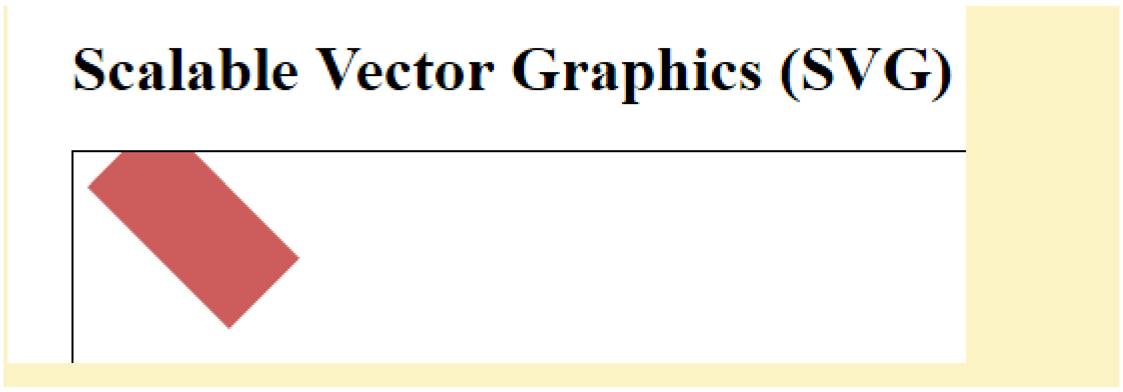
We can make SVG transformations (rotation, scaling, and skewing) behave like transformations applied to an HTML object through the following technique:

- Chain transforms with: the first translation moving the SVG origin to where the centre of the target object currently exists (this requires a manual calculation); perform the desired transformations; undo the translation done in the first step.

```
// HTML case
transform: rotate(45deg);
```



```
// SVG case
transform: translate(60px, 35px) rotate(45deg) translate(-60px, -35px);
```



C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\css2017\transforms-html.xhtml
C:\Users\John\Documents\Sda\Code\web\Examples\SvgExamples\svg-transforms.xhtml
(Tudor 2015, "Transforms on SVG Elements", https://css-tricks.com/transforms-on-svg-elements/)

Painting: Filling and Stroking

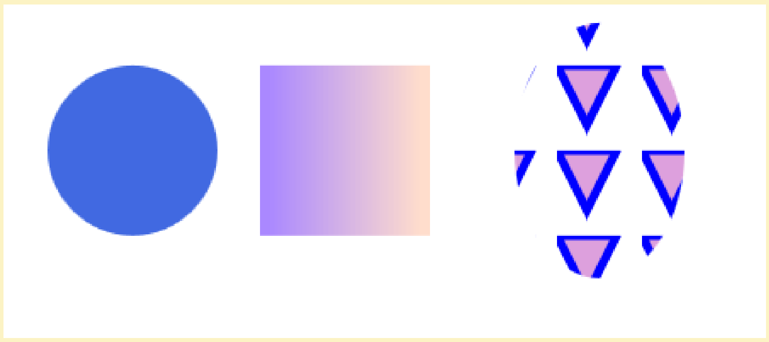
Two paint operations are:

- Filling: setting the interior colour of an object.
- Stroking: setting the outline colour of an object.

(W3C 2016, "Scalable Vector Graphics (SVG) 2", https://www.w3.org/TR/SVG2/), "Chapter 13: Painting: Filling, Stroking and Marker Symbols", https://www.w3.org/TR/SVG2/painting.html#Definitions

Four types of paint: solid colors; gradients (linear or radial); patterns; and images. For example here's how to set types of paint for fill ...

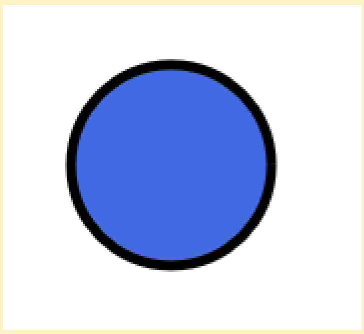
```
<svg xmlns="http://www.w3.org/2000/svg">
  <defs>
    <linearGradient id="myGradient">
      <stop offset="5%" stop-color="#A8F" />
      <stop offset="95%" stop-color="#FDC" />
    </linearGradient>
    <pattern id="trianglePattern" patternUnits="userSpaceOnUse" x="0" y="0" width="50"
height="50" viewBox="0 0 10 10">
      <path d="M 0 0 L 7 0 L 3.5 7 z" fill="plum" stroke="blue" />
    </pattern>
  </defs>

  <!-- Top Row -->

  <!-- Fill, Solid -->
  <circle cx="100" cy="100" r="50" style="fill: royalblue;" />

  <!--Fill, Gradient (linear) -->
  <rect x="175" y="50" width="100" height="100" style="fill:url(#myGradient);" />

  <!-- Fill, Pattern -->
  <ellipse cx="375" cy="100" rx="50" ry="75" style="fill:url(#trianglePattern);" />

  <!--Fill, Image (CSS3)) -->
  <!-- <rect x="75" y="200" width="100" height="100" style="fill: url(images/mount-
everest.jpg);" /> -->
</svg>
```



(W3C 2016, "Scalable Vector Graphics (SVG) 2", https://www.w3.org/TR/SVG2/), "Chapter 13: Painting: Filling, Stroking and Marker Symbols", https://www.w3.org/TR/SVG2/painting.html#Definitions
C:\Users\John\Documents\Sda\Code\web\Examples\SvgExamples\svg-painting-stroke-and-fill.xhtml

Stroke example ...

```
<circle cx="100" cy="250" r="50" style="fill: royalblue; stroke: black; stroke-width: 5px;"
/>
```



(W3C 2016, "Scalable Vector Graphics (SVG) 2", https://www.w3.org/TR/SVG2/), "Chapter 13: Painting: Filling, Stroking and Marker Symbols", https://www.w3.org/TR/SVG2/painting.html#Definitions
C:\Users\John\Documents\Sda\Code\web\Examples\SvgExamples\svg-painting-stroke-and-fill.xhtml

The paint values are:

- none: no paint.
- child: the last child paint server element of the element being painted.
- child(n): the nth child paint server element of the element being painted.
- <url>[none|<color>]?: A url to a paint sever element, optionally followed by a fallback value.
- color: a solid color value.
- context-fill: use the paint value of fill from a context element.
- context-stroke: use the paint value of stroke from a context element.

Paint servers: gradients and patterns

Overview

A paint server is a resource defined outside an object which an object can use to set its fill or stroke.

(W3C 2016, "Scalable Vector Graphics (SVG) 2", https://www.w3.org/TR/SVG2/), "Paint Servers, Introduction", https://www.w3.org/TR/SVG2/pservers.html#Introduction

Gradients

Once a gradient is defined, a graphics element can be filled or stroked with the gradient by setting the fill or stroke properties to reference the gradient.

(W3C 2016, "Scalable Vector Graphics (SVG) 2", https://www.w3.org/TR/SVG2/), "Gradients", https://www.w3.org/TR/SVG2/pservers.html#Gradients

Gradients are linear or circular. Each gradient has two or more colors (one color renders as a solid color), the location of which is determined by color stops. A "normal" to the gradient vector determines where the middle fade in-out occurs.

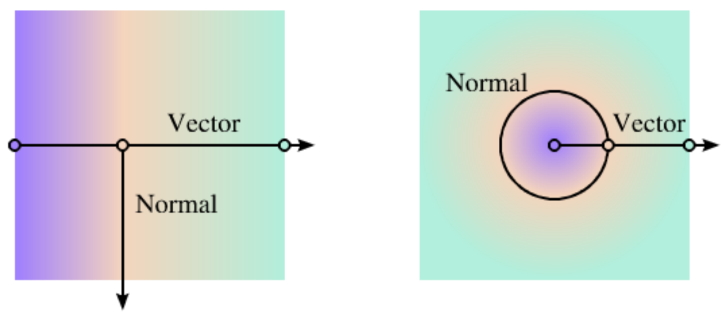
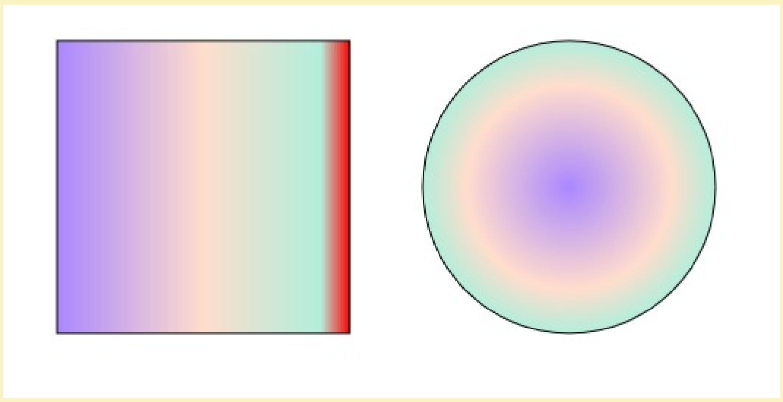


Figure 4 Linear and radial gradients with the gradient vector indicated. The vector consists of three stops shown by small circles. One gradient normal is shown for each gradient.

(W3C 2016, "Scalable Vector Graphics (SVG) 2", https://www.w3.org/TR/SVG2/), "Gradients", https://www.w3.org/TR/SVG2/pservers.html#Gradients

Basic linear and radial gradient example.

```
<svg xmlns="http://www.w3.org/2000/svg">
  <style>
    svg * {
      stroke: #000000;
      stroke-width: 1px;
    }
  </style>

  <defs>
    <linearGradient id="myLinearGradient">
      <stop offset="0%" stop-color="#a8f" />
      <stop offset="50%" stop-color="#fdc" />
      <stop offset="90%" stop-color="#b2eedb" />
      <stop offset="100%" stop-color="red" />
    </linearGradient>
    <radialGradient id="myRadialGradient">
      <stop offset="0%" stop-color="#a8f" />
      <stop offset="70%" stop-color="#fdc" />
      <stop offset="100%" stop-color="#b2eedb" />
    </radialGradient>
  </defs>

  <rect x="50" y="50" width="200" height="200" style="fill: url(#myLinearGradient)" />
  <circle cx="400" cy="150" r="100" style="fill: url(#myRadialGradient)" />
 </svg>
```

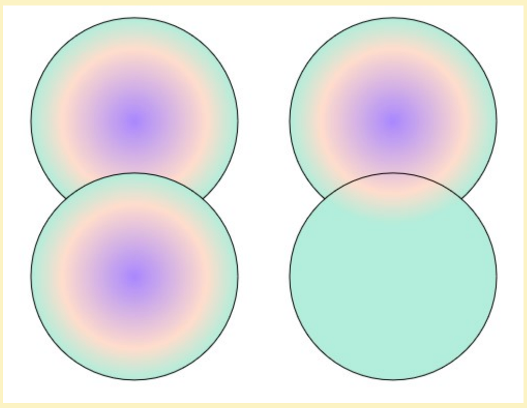


(W3C 2016, "Scalable Vector Graphics (SVG) 2", https://www.w3.org/TR/SVG2/), "Gradients", https://www.w3.org/TR/SVG2/pservers.html#Gradients
C:\Users\John\Documents\Sda\Code\web\Examples\SvgExamples\svg-paint-servers.xhtml

A gradient will have a gradientUnits attribute. The meaning of the values are:

- "objectBoundingBox" (the initial value): make the gradient relative to the using object.
- "userSpaceOnUse": make the gradient relative to the parent user co-ordinate space (e.g. the enclosing svg element).

```
<svg xmlns="http://www.w3.org/2000/svg">
  <defs>
    <radialGradient id="myRadialGradient-objectBoundingBox"
gradientUnits="objectBoundingBox">
      <stop offset="0%" stop-color="#a8f" />
      <stop offset="70%" stop-color="#fdc" />
      <stop offset="100%" stop-color="#b2eedb" />
    </radialGradient>
    <radialGradient id="myRadialGradient-userSpaceOnUse" gradientUnits="userSpaceOnUse"
cx="400" cy="125" r="100">
      <stop offset="0%" stop-color="#a8f" />
      <stop offset="70%" stop-color="#fdc" />
      <stop offset="100%" stop-color="#b2eedb" />
    </radialGradient>
  </defs>

  <circle cx="150" cy="125" r="100" style="fill: url(#myRadialGradient-objectBoundingBox)"
/>
  <circle cx="150" cy="275" r="100" style="fill: url(#myRadialGradient-objectBoundingBox)"
/>
  <circle cx="400" cy="125" r="100" style="fill: url(#myRadialGradient-objectBoundingBox)"
/>
  <circle cx="400" cy="275" r="100" style="fill: url(#myRadialGradient-userSpaceOnUse)" />
  </svg>
```



(W3C 2016, "Scalable Vector Graphics (SVG) 2", https://www.w3.org/TR/SVG2/), "Gradients", https://www.w3.org/TR/SVG2/pservers.html#Gradients
C:\Users\John\Documents\Sda\Code\web\Examples\SvgExamples\svg-paint-servers-gradientUnits.xhtml

Patterns

Pattern example.

```
<svg xmlns="http://www.w3.org/2000/svg">
  <style>
    svg * {
      stroke: #000000;
      stroke-width: 1px;
    }
  </style>

  <defs>
...
    <pattern id="greyDiagonalsPattern" viewBox="0 0 100 100" width="20%" height="20%">
      <path d="M0,0 h40 L100,60 v40 z m 0,60 v40 h40 z" fill="grey" />
    </pattern>
  </defs>

...
  <ellipse cx="625" cy="150" rx="75" ry="100" style="fill:url(#greyDiagonalsPattern);" />
</svg>
```

C:\Users\John\Documents\Sda\Code\web\Examples\SvgExamples\svg-paint-servers.xhtml
(W3C 2016, "Scalable Vector Graphics (SVG) 2", https://www.w3.org/TR/SVG2/), "Patterns", https://www.w3.org/TR/SVG2/pservers.html#Patterns

Painting: Markers

> The ‘marker’ element defines the graphics that are to be used for drawing markers on a shape.

(W3C 2016, "Scalable Vector Graphics (SVG) 2", https://www.w3.org/TR/SVG2/), "The ‘marker’ element", https://www.w3.org/TR/SVG2/painting.html#MarkerElement

> The ‘markerWidth’ and ‘markerHeight’ attributes represent the size of the SVG viewport into which the marker is to be fitted according to the ‘viewBox’ and

> ‘preserveAspectRatio’ attributes. A value of zero for either attribute results in nothing being rendered for the marker.

In practice the trick seems to be just to make this large enough.

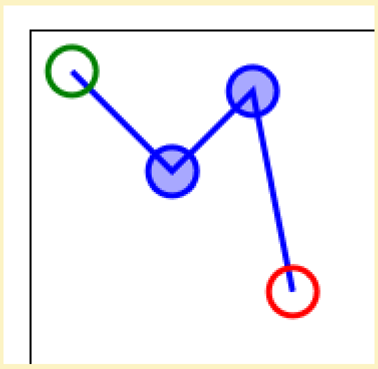
(W3C 2016, "Scalable Vector Graphics (SVG) 2", https://www.w3.org/TR/SVG2/), "The ‘marker’ element", https://www.w3.org/TR/SVG2/painting.html#MarkerElement

> The ‘refX’ and ‘refY’ attributes define the reference point of the marker.

(W3C 2016, "Scalable Vector Graphics (SVG) 2", https://www.w3.org/TR/SVG2/), "The ‘marker’ element", https://www.w3.org/TR/SVG2/painting.html#MarkerElement

Basic marker example.

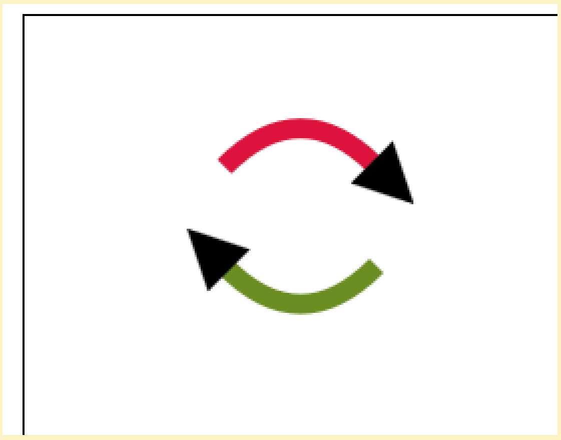
```
<svg xmlns="http://www.w3.org/2000/svg" width="400px" height="400px">
  <style>
    /* <![CDATA[ */
    path {
      fill: none;
      stroke: blue;
      stroke-width: 3px;
    }
    /* ]]> */
  </style>

  <defs>
    <marker id="circleMarkerGreen"
            markerWidth="10" markerHeight="10"
            viewBox="0 0 10 10"
            refX="5" refY="5"
            markerUnits="strokeWidth"
            >
      <circle cx="5" cy="5" r="4" stroke="green" fill="none" />
    </marker>
    <marker id="circleMarkerBlue"
            markerWidth="10" markerHeight="10"
            viewBox="0 0 10 10"
            refX="5" refY="5"
            markerUnits="strokeWidth"
            >
      <circle cx="5" cy="5" r="4" stroke="blue" fill="rgba(0, 0, 255, 0.342)" />
    </marker>
    <marker id="circleMarkerRed"
            markerWidth="10" markerHeight="10"
            viewBox="0 0 10 10"
            refX="5" refY="5"
            markerUnits="strokeWidth"
            >
      <circle cx="5" cy="5" r="4" stroke="red" fill="none" />
    </marker>
  </defs>
  <path d="m 20,20 l 50,50 l 40,-40 l 20,100"
          marker-start="url(#circleMarkerGreen)"
          marker-mid="url(#circleMarkerBlue)"
          marker-end="url(#circleMarkerRed)"
          />
</svg>
```



(W3C 2016, "Scalable Vector Graphics (SVG) 2", https://www.w3.org/TR/SVG2/), "Vertex markers: the 'marker-start', 'marker-mid' and 'marker-end' properties", https://www.w3.org/TR/SVG2/painting.html#VertexMarkerProperties C:\Users\John\Documents\Sda\Code\web\Examples\SvgExamples\svg-markers.xhtml

Example with the orient="auto" property.

```
<svg xmlns="http://www.w3.org/2000/svg">

  <defs>
    <marker id="triangleMarker"
            markerWidth="4" markerHeight="3"
            viewBox="0 0 10 10"
            refX="1" refY="5"
            markerUnits="strokeWidth"
            orient="auto"
            >
      <!-- context-stroke currently unsupported by browsers -->
      <path d="M 0 0 L 10 5 L 0 10 z" fill="context-stroke" />
    </marker>
  </defs>

  <g fill="none" stroke-width="10" marker-end="url(#triangleMarker)">
    <path stroke="crimson" d="M 100,75 C 125,50 150,50 175,75"/>
    <path stroke="olivedrab" d="M 175,125 C 150,150 125,150 100,125" />
  </g>

</svg>
```



(W3C 2016, "Scalable Vector Graphics (SVG) 2", https://www.w3.org/TR/SVG2/), "Vertex markers: the 'marker-start', 'marker-mid' and 'marker-end' properties", https://www.w3.org/TR/SVG2/painting.html#VertexMarkerProperties C:\Users\John\Documents\Sda\Code\web\Examples\SvgExamples\svg-markers-arrows.xhtml

Tools

Overview

SVG creation tools that are good:

- Inkscape (Desktop app). https://inkscape.org/en/
- Google SVG Edit (online or downloadable).
  - Latest version: https://github.com/SVG-Edit/svgedit/releases
  - Online version: https://svg-edit.github.io/svgedit/releases/svg-edit-2.8.1/svg-editor.html
  - Project home: https://github.com/SVG-Edit/svgedit

SVG conversion tools (e.g. From PNG or JPG to SVG) that are good (Generally converts to black and white … you'll have to touch up with editor):

- Inkscape (Desktop app). https://inkscape.org/en/
- https://image.online-convert.com/convert-to-svg
- https://online-converting.com/image/convert-to-svg/

Creation Technique

Google SVG ...

Create an SVG circle as a path using Google SVG edit:

- Go to the latest version, https://github.com/SVG-Edit/svgedit/releases, and choose "Standard version live".
- SVG-Edit > New Image.
- SVG-Edit > Document Properties. Adjust as desired (e.g. 300px X 300px).
- Turn the grid on.
- Click on the Ellipse icon and choose the circle command.
- With mouse in the centre draw mouse.
- Adjust fill and stroke properties as desired.
- [Convert to path] toolbar icon.
- [SVG] toolbar icon.
- Copy <svg>...</svg> element into C:\Users\John\Documents\Sda\Code\web\Examples\SvgExamples\svg-display-harness.xhtml
- Open in a browser.

Inkscape …

Display Inkscape generated SVG as embedded in an XHTML doc:

- Save Inkscape file (Inkscape saves files as SVG natively).
- Open C:\Users\John\Documents\Sda\Code\web\Examples\SvgExamples\svg-embedded-path-data.xhtml and edit <img> to point to svg file.
- Open .xhtml in a browser.

For details about creating paths in Inkscape see:
http://tavmjong.free.fr/INKSCAPE/MANUAL/html/Paths-Creating.html

Prepare the Inkscape document:

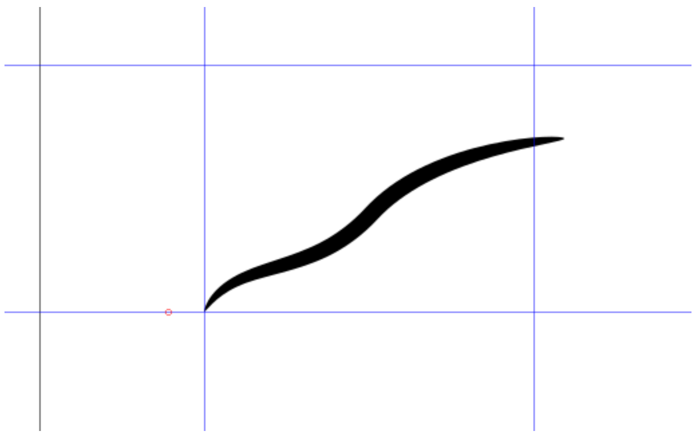
- File > Document Properties.
    - |Page|. Custom size: Units px; width 640; height 480.
    - Close document properties.
- View > Zoom > Page (5).
- Create some guides:
    - Verify [View] > [Guides (|)] enabled.
    - Drag multiple guides from the left and right ruler.
    - Toogle Guide visibility (|).

Create a cubic Bézier curve:

- Icon [Draw Bezier curves and straight lines (Shift + F6)]. Ensure Mode is [Create Regular Bezier Path].
- Click and drag a series of curves.
- Right click to end, or complete the shape.
- Optionally [Edit paths by node (F2)].
- [Object] > [Fill and Stroke]. As desired.
- Do "Display Inkscape generated SVG as embedded in an XHTML doc".

Create a cubic Bézier curve with a cursive shape.

- Select Shape: Elipse.
- Draw a Bézier curve.



Animation

Overview

Interactivity in SVG, responding to events, can be achieved through:

1. User-initiated action that triggers UI Events. E.g. Button presses. This can cause animations or scripts to executed.
2. Links, using the `a` element.
3. The `zoomAndPan` attribute on the `svg` element.

(W3C 2016, "Scalable Vector Graphics (SVG) 2", https://www.w3.org/TR/SVG2/), "Scripting and Interactivity", https://www.w3.org/TR/SVG2/interact.html

SVG supports animations via the following techniques:

1. **CSS transitions** (W3C The latest, "CSS Transitions", https://www.w3.org/TR/css-transitions-1/). https://www.w3.org/TR/css-transitions-1/

   (Bentley 2018, "Cascading Style Sheets - Css -Reference", \\Atlas\Users\John\Documents\Sda\Info\Web\KB\Css\Reference\CascadingStyleSheets-Css-Reference.docx)

2. **CSS animations** (W3C The latest, "CSS Animations Level 1", https://www.w3.org/TR/css-animations-1/). https://www.w3.org/TR/css-animations-1/.

   (Bentley 2018, "Cascading Style Sheets - Css -Reference", \\Atlas\Users\John\Documents\Sda\Info\Web\KB\Css\Reference\CascadingStyleSheets-Css-Reference.docx)

3. **SVG animations** (W3C 2018, "SVG Animations Level 2, Editor's Draft", https://svgwg.org/specs/animations/). https://svgwg.org/specs/animations/
4. Scripting attributes and CSS settings through the **SVG DOM** (W3C 2016, "Scalable Vector Graphics (SVG) 2", https://www.w3.org/TR/SVG2/), "Appendix A: SVG Document Object Model (DOM)", https://www.w3.org/TR/SVG2/svgdom.html. In particular using **animation frames** (W3C n.d., "HTML 5.3, the Latest", Accessed 2018-03-09. https://www.w3.org/TR/html53/), "Animation Frames", https://www.w3.org/TR/html53/webappapis.html#animation-frames
5. **Web Animations** (W3C The latest, "Web Animations", https://drafts.csswg.org/web-animations/). https://drafts.csswg.org/web-animations/

(W3C 2016, "Scalable Vector Graphics (SVG) 2", https://www.w3.org/TR/SVG2/), "Appendix E: Animating SVG Documents", https://www.w3.org/TScR/SVG2/animate.html

SVG animate using CSS transitions

The X and Y properties of an SVG object are not css properties so generally you'd use another animation technique for SVG.

As for using CSS transitions on HTML objects you: set the transition-property; a transition-duration; a start value for that property; and an end value for that property upon some event (e.g. during a css defined hover or javascript event).

```
...
    /* Transitions */
    path  {
      transition-property: stroke-dashoffset;
      /* Transition duration set in script to facilitate reset then go */

      /* Draw 60 ticks at the end of each interval.
        Relies on transition duration being set (in script) to 60s */
      transition-timing-function: steps(60, end);

      /* The start value must be explicitly defined */
      stroke-dashoffset: var(--path-length);
    }  /* ]]> */
  </style>

  <script>
  /* <![CDATA[ */

    function go () {
      // Draws a path over time
      var elements = document.getElementsByTagName("path");
      for (var i = 0; i < elements.length; i++) {
        elements[i].style.transitionDuration = "1s";
        // Setting the end value of the transition property triggers the transition.
        elements[i].style.strokeDashoffset = "0";
      }
    }

    function reset () {
      var elements = document.getElementsByTagName("path");
      for (var i = 0; i < elements.length; i++) {
        elements[i].style.transitionDuration = "0s";
        elements[i].style.strokeDashoffset = pathLength;
      }
    }
```

C:\Users\John\Documents\Sda\Code\web\Examples\SvgExamples\svg-animate-using-css-transitions.xhtml

SVG animate using CSS animations

The X and Y properties of an SVG object are not css properties so generally you'd use another animation technique for SVG.

As for using CSS animations on HTML objects you: define a from and to keyframe (with any intermediate states) and name it; reference the keyframe name; define animation duration, timing function, and other things.

```
    path {
      fill: none;
      stroke: red;
      stroke-width: 10px;
      transform: translate(50px,50px);

      /* Initialize path so it doesn't appear */
      stroke-dasharray: var(--path-length);
      stroke-dashoffset: var(--path-length);
    }

    /* Animations */
    @keyframes my-animation {
      from {
        stroke-dashoffset: var(--path-length);
      }
      to {
        stroke-dashoffset: 0;
      }
    }

    .go  {
      animation-name: my-animation;
      animation-duration: 60s;

      /* Draw 60 ticks at the end of each interval.
        Relies on transition duration being set (in script) to 60s */
      animation-timing-function: steps(60, end);

      /* Preserve the effect of the animation at ending */
      animation-fill-mode: forwards;
    }
  /* ]]> */
  </style>
..
<body>
...

  <svg xmlns="http://www.w3.org/2000/svg" width="600px" height="400px">
    <!-- Created with SVG-edit - https://github.com/SVG-Edit/svgedit-->
    <path id="redCircle" d="m7.82411,149.66667c0,-78.91855 63.92403,-142.84258 142.84258,-142.84258c78.91855,0 142.84258,63.92403 142.84258,142.84258c0,78.91855 -63.92403,142.84258 -142.84258,142.84258c-78.91855,0 -142.84258,-63.92403 -142.84258,-142.84258z" />
  </svg>
```

C:\Users\John\Documents\Sda\Code\web\Examples\SvgExamples\svg-animate-using-css-animations.xhtml

Start an SVG animation using CSS animations by setting the SVG target to a class with `setAttribute`, not `className =` .

```
  <script>
  /* <![CDATA[ */
...
    function go () {
      var targetElement = document.getElementById("redCircle");
      /* targetElement.className = "g0" // doesn't work */
      targetElement.setAttribute("class", "go");
    }

    function reset () {
      var targetElement = document.getElementById("redCircle");
      targetElement.setAttribute("class", "");
    }

...

  /* ]]> */
  </script>
</head>
<body>
  <h1>Scalable Vector Graphics (SVG) - Animations using CSS Animations</h1>

  <svg xmlns="http://www.w3.org/2000/svg" width="600px" height="400px">
    <!-- Created with SVG-edit - https://github.com/SVG-Edit/svgedit-->
    <path id="redCircle" d="m7.82411,149.66667c0,-78.91855 63.92403,-142.84258 142.84258,-142.84258c78.91855,0 142.84258,63.92403 142.84258,142.84258c0,78.91855 -63.92403,142.84258 -142.84258,142.84258c-78.91855,0 -142.84258,-63.92403 -142.84258,-142.84258z" />
  </svg>

  <button onclick="go();">Go</button>
  <button onclick="reset();" type="reset">Reset</button>
  <output></output>

</body>
```

C:\Users\John\Documents\Sda\Code\web\Examples\SvgExamples\svg-animate-using-css-animations.xhtml

SVG animate using SVG animations

SVG animations define the following "animation elements": `animate`, `animateMotion`, `animateTransform`, `set`, and `discard`.

```
<rect x="10" y="10" width="100" height="100">
  <!-- begin="indefinite" allows us to start the animation from script -->
  <animate attributeName="x" begin="indefinite" dur="1s" from="10" to="300" fill="freeze"
/>
</rect>
```

(W3C 2018, "SVG Animations Level 2, Editor's Draft", https://svgwg.org/specs/animations/), "2.3. Relationship to SMIL Animation", https://svgwg.org/specs/animations/#RelationshipToSMILAnimation
C:\Users\John\Documents\Sda\Code\web\Examples\SvgExamples\svg-animate-using-svg-animations.xhtml
C:\Users\John\Documents\Sda\Code\web\Examples\SvgExamples\svg-animate-using-svg-animations-line-animation.xhtml

`animateMotion` may have the following attributes: `path`; `keyPoints`; and `rotate`. `animateMotion` also have an `mpath` child element.

(W3C 2018, "SVG Animations Level 2, Editor's Draft", https://svgwg.org/specs/animations/), "2.3. Relationship to SMIL Animation", https://svgwg.org/specs/animations/#RelationshipToSMILAnimation

SVG animation overview example.

(W3C 2018, "SVG Animations Level 2, Editor's Draft", https://svgwg.org/specs/animations/), "Animation elements example", https://svgwg.org/specs/animations/#AnimationElementsExample

Start an SVG animation:

- Without javascript, using the "event-value" type of the begin attribute="id.event" (without an "on" prefix) on an animation element; or

```
<svg xmlns="http://www.w3.org/2000/svg" width="600px" height="400px">
  <rect x="10" y="10" width="100" height="100">
    <animate attributeName="x" begin="go.click" dur="2s" from="10" to="300"
fill="freeze" />
  </rect>
</svg>

<button id="go">Go</button>
```

(W3C 2018, "SVG Animations Level 2, Editor's Draft", https://svgwg.org/specs/animations/), " Attributes to control the timing of the animation", "begin" attribute, "event-value" value type, https://svgwg.org/specs/animations/#TimingAttributes
C:\Users\John\Documents\Sda\Code\web\Examples\SvgExamples\svg-animate-using-svg-animations.xhtml

- From javascript, by setting an animation element's begin attribute to "indefinite"; and calling beginElement() from script;

```
function go () {
  var elements = document.getElementsByTagName("animate");
  for (var i = 0; i < elements.length; i++) {
    elements[i].beginElement();
  }
}

<svg xmlns="http://www.w3.org/2000/svg" width="600px" height="400px">
  <!-- Top Row -->
  <rect x="10" y="10" width="100" height="100">
    <!-- begin="indefinite" allows us to start the animation from script -->
    <animate attributeName="x" begin="indefinite" dur="2s" from="10" to="300"
fill="freeze" />
  </rect>
```

(W3C 2018, "SVG Animations Level 2, Editor's Draft", https://svgwg.org/specs/animations/), " Attributes to control the timing of the animation", "begin" attribute, "indefinite" value type, https://svgwg.org/specs/animations/#TimingAttributes
C:\Users\John\Documents\Sda\Code\web\Examples\SvgExamples\svg-animate-using-svg-animations.xhtml

Reset an SVG Animation:

```
 ...
  <script>
    /* <![CDATA[ */
    function go() {
      var elements = document.getElementsByTagName("animate");
      for (var i = 0; i < elements.length; i++) {
        elements[i].setAttribute("fill", "freeze");
        elements[i].beginElement();
      }
    }

    function reset() {
      var elements = document.getElementsByTagName("animate");
      for (var i = 0; i < elements.length; i++) {
        elements[i].setAttribute("fill", "remove");
        elements[i].endElement();
      }
    }

    /* ]]> */
  </script>
</head>
<body>
  <h1>Scalable Vector Graphics (SVG) - SVG Animations</h1>

  <svg xmlns="http://www.w3.org/2000/svg" width="600px" height="400px">
    <rect x="10" y="10" width="100" height="100">
      <!-- begin="indefinite" allows us to start the animation from script -->
      <animate attributeName="x" begin="indefinite" dur="1s" from="10" to="300"
fill="freeze" />
    </rect>

  </svg>

  <button onclick="go();" type="button">Go</button>
  <button onclick="reset();" type="reset">Reset</button>

</body>
```

> Unlike other SVG DOM interfaces, the SVG DOM does not specify convenience DOM properties corresponding to the various language attributes on SVG's animation elements. ...The current method for accessing and modifying the attributes on the animation elements is to use the standard getAttribute, setAttribute, getAttributeNS and setAttributeNS defined in DOM4 [DOM4]

(W3C 2018, "SVG Animations Level 2, Editor's Draft", https://svgwg.org/specs/animations/), "Interface SVGAnimationElement", https://svgwg.org/specs/animations/#InterfaceSVGAnimationElement
C:\Users\John\Documents\Sda\Code\web\Examples\SvgExamples\svg-animate-using-svg-animations.xhtml

SVG animate using DOM

Unless you need complete control it will be better to use another animation technique, apart from SVG animate using DOM. Other animation techniques will generally be simpler to code and has the in-built timing functions that you'll typically need.

Bentley 2018.

SVG animate using the DOM and the `window.requestAnimationFrame(animate)` callback function.

```
  <script>
    /* <![CDATA[ */
    // User input
    var startX = 10;
    var endX = 300;
    var totalTime = 2000; // Milliseconds

    // Scafold variables
    var element = null;
    var output = null;
    var distanceX = endX - startX;
    var elaspedTime = 0;
    var stopped = false;
    var requestId = 0;
    var startTime = 0;

    // The callback function
    function animate(time) {
      if (!stopped) {
        elaspedTime = (Date.now() - startTime);
        position = startX + (elaspedTime/totalTime) * distanceX;
        output.textContent = position;
        element.setAttribute("x", position);
        if (position >= endX ) { stopped = true };
        // Recursive callback
        requestId = window.requestAnimationFrame(animate);
      }
    }

    function start() {
      // Get the first rect.
      element = document.getElementsByTagName("rect")[0];
      output = document.getElementsByTagName("output")[0];
      startTime = Date.now();
      stopped = false;
      requestId = window.requestAnimationFrame(animate);
    }

    function stop () {
      if (requestId) {
          window.cancelAnimationFrame(requestId);
      }
      stopped = true;
    }

    function reset () {
      stop();
      position = startX;
      output.textContent = position;
      element.setAttribute("x", position);
    }
  /* ]]> */
  </script>
</head>
<body>
  <h1>Scalable Vector Graphics (SVG) - SVG Animations using SVG DOM and
requestAnimateFrame</h1>

  <svg xmlns="http://www.w3.org/2000/svg" width="600px" height="400px">
    <rect x="10" y="10" width="100" height="100" />
  </svg>

  <button onclick="start();" type="button">Start</button>
  <button onclick="stop();" type="button">Stop</button>
  <button onclick="reset();" type="reset">Reset</button>
  <output></output>

</body>
```

C:\Users\John\Documents\Sda\Code\web\Examples\SvgExamples\svg-animate-using-svg-dom-and-requestAnimateFrame.xhtml
[*https://webplatform.github.io/docs/dom/Window/requestAnimationFrame/*](https://webplatform.github.io/docs/dom/Window/requestAnimationFrame/)

SVG animate using Web animations

Web animations involve using the Web animation programmatic interface chiefly via the `element.animate()` method.

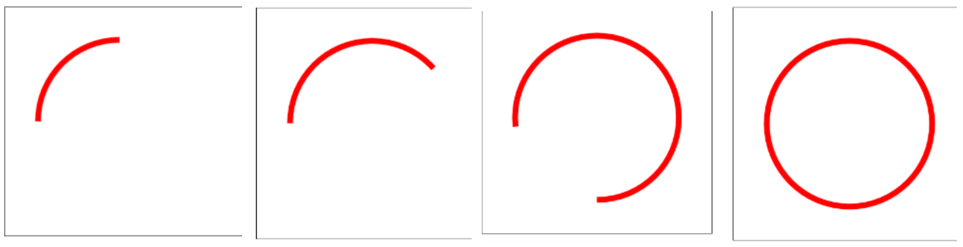
```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
<head>
  <meta charset="utf-8" />
  <title>Scalable Vector Graphics (SVG) - SVG Animations using Web animations</title>

  <!-- Scaffold -->
  <style>
  /* <![CDATA[ */
    body {
      margin-left: 100px;
    }

    svg {
      border: 1px solid black;
      width: 600px;
      height: 400px;
      display: block;
      margin-bottom: 1em;
    }

    rect {
      /* fill: none; */
      stroke: #000000;
      stroke-width: 1px;
      fill: blue;
    }
  /* ]]> */
  </style>

  <script>
    /* <![CDATA[ */
    // User input
    var startX = 10;
    var endX = 300;
    var totalTime = 2000; // Milliseconds

    // Scafold variables
    var element = null;
    var animation = null;

    function start() {
      // Get the first rect.
      element = document.getElementsByTagName("rect")[0];
      animation = element.animate(
        // Key frames
        [
          { transform: 'translateX(' + startX + 'px)' },
          { transform: 'translateX(' + endX + 'px)' }
        ],
        // Effect Timing
        {
          fill: 'forwards',
          duration: totalTime,
          iterations: 1
        });
    }

    function stop () {
      animation.pause();
    }

    function reset () {
      animation.cancel();
      element.setAttribute("x", startX);
    }
  /* ]]> */
  </script>
</head>
<body>
  <h1>Scalable Vector Graphics (SVG) - SVG Animations using Web animations</h1>

  <svg xmlns="http://www.w3.org/2000/svg" width="600px" height="400px">
    <rect x="10" y="10" width="100" height="100" />
  </svg>

  <button onclick="start();" type="button">Start</button>
  <button onclick="stop();" type="button">Stop</button>
  <button onclick="reset();" type="reset">Reset</button>
  <output></output>

</body>
</html>
```

(W3C The latest, "Web Animations", https://drafts.csswg.org/web-animations/) https://drafts.csswg.org/web-animations/
https://developer.mozilla.org/en-US/docs/Web/API/Web_Animations_API/Using_the_Web_Animations_API
C:\Users\John\Documents\Sda\Code\web\Examples\SvgExamples\svg-animate-using-web-animations.xhtml

SVG animation techniques

Line animation

Do line animation by operating on a SVG path using one of: css tranisitions; css animations; svg animations.



(Coyier 2014, "How SVG Line Animation Works", https://css-tricks.com/svg-line-animation-works/), https://css-tricks.com/svg-line-animation-works/
C:\Users\John\Documents\Sda\Code\web\Examples\SvgExamples\svg-animate-using-css-transitions.xhtml
C:\Users\John\Documents\Sda\Code\web\Examples\SvgExamples\svg-animate-using-css-animations.xhtml
C:\Users\John\Documents\Sda\Code\web\Examples\SvgExamples\svg-animate-using-svg-animations-line-animation.xhtml

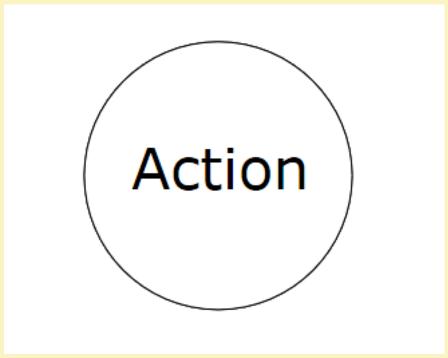
Technique

Outline glow on hover

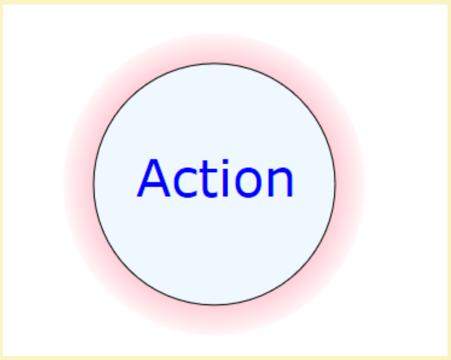
Outline glow on hover.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
<head>
  <meta charset="utf-8" />
  <title>Scalable Vector Graphics (SVG) - Outline glow on hover</title>

  <!-- Scaffold -->
  <style>
    /* <![CDATA[ */

    body {
      margin-left: 100px;
    }

    svg {
      border: 1px solid black;
      width: 600px;
      height: 400px;
    }

    /* ]]> */
  </style>

  <!-- Instructional code -->
  <style>
    /* <![CDATA[ */

    circle {
      fill: white;
      stroke: black;
      stroke-width: 1px;
    }

    text {
      font-family: Verdana, Geneva, Tahoma, sans-serif;
      font-size: 32pt;
      fill: black;
    }


    a:hover circle#target {
      stroke: url(#myRadialGradient);
      stroke-width: 50px;
    }

    a:hover circle#overlay {
      fill: aliceblue;
    }

    a:hover text {
      fill: blue;
    }

    /* ]]> */
  </style>
```

```
</head>
<body>
  <h1>Scalable Vector Graphics (SVG) - Outline glow on hover</h1>

  <svg xmlns="http://www.w3.org/2000/svg">
    <defs>
      <radialGradient id="myRadialGradient" gradientUnits="userSpaceOnUse" cx="300"
cy="200" r="200">
        <stop offset="0%" stop-color="white" />
        <stop offset="45%" stop-color="pink" />
        <stop offset="65%" stop-color="white" />
      </radialGradient>
    </defs>
    <a href="#">
      <circle id="target" cx="300" cy="200" r="100" />
      <circle id="overlay" cx="300" cy="200" r="100"  />
      <text x="235" y="210">Action</text>
    </a>
  </svg>

</body>
</html>

// Default:
```



```
// On hover (hand pointer not shown):
```



C:\Users\John\Documents\Sda\Code\web\Examples\SvgExamples\svg-technique-outline-glow-on-hover.xhtml

References

Bentley, John. 2018. "Cascading Style Sheets - Css -Reference". \\Atlas\Users\John\Documents\Sda\Info\Web\KB\Css\Reference\CascadingStyleSheets-Css-Reference.docx. 2018.

Coyier, Chris. 2014. "How SVG Line Animation Works". CSS-Tricks. https://css-tricks.com/svg-line-animation-works/. 2014-02-18.

Tudor, Ana. 2015. "Transforms on SVG Elements". CSS-Tricks. https://css-tricks.com/transforms-on-svg-elements/. 2015-05-05.

W3C. The latest. "CSS Animations Level 1". https://www.w3.org/TR/css-animations-1/. The latest.

———. The latest. "CSS Transitions". https://www.w3.org/TR/css-transitions-1/. The latest.

———. n.d. "HTML 5.3, the Latest". Accessed 2018-03-09. https://www.w3.org/TR/html53/. n.d.

———. 2016. "Scalable Vector Graphics (SVG) 2". https://www.w3.org/TR/SVG2/. 2016-09-15.

———. 2018. "SVG Animations Level 2, Editor's Draft". https://svgwg.org/specs/animations/. 2018-09-06.

———. The latest. "Web Animations". https://drafts.csswg.org/web-animations/. The latest.

Document Licence