SVG 101

SCALABLE VECTOR GRAPHICS (SVG):

- language for describing 2D graphics consisting of paths, images, and text in XML
- graphics are able to be scaled and resized without losing image quality
- several different ways to include SVG, for example: `<img>`, `background-image`, or inline (within HTML)

WRITING SVG INLINE

- editable (vs using `<img>` or `background-image`)
- graphics software helps export SVG to embed into HTML

DOCUMENT ORGANIZATION

`<svg>`

- element that holds all SVG elements
- used to nest a standalone SVG fragment within HTML
- `width`, `height`, `preserveAspectRatio`, `viewBox` attributes establish canvas and coordinate system

<g>

- element for grouping related graphics together (organization!)
- allows for manipulation of the group as a whole (animations!)

`<use>`

- element that allows reuse of already rendered graphics (DRY!)
- `x`, `y`, `width`, `height` attributes define the mapping location details of the graphic
- `xlink:href` attribute calls on the element to be reused

```
<svg>
  <g id="smalltalk">
     <!-- dog paths -->
  </g>

  <use x="50" y="50" xlink:href="#smalltalk">
</svg>
```

`<defs>`

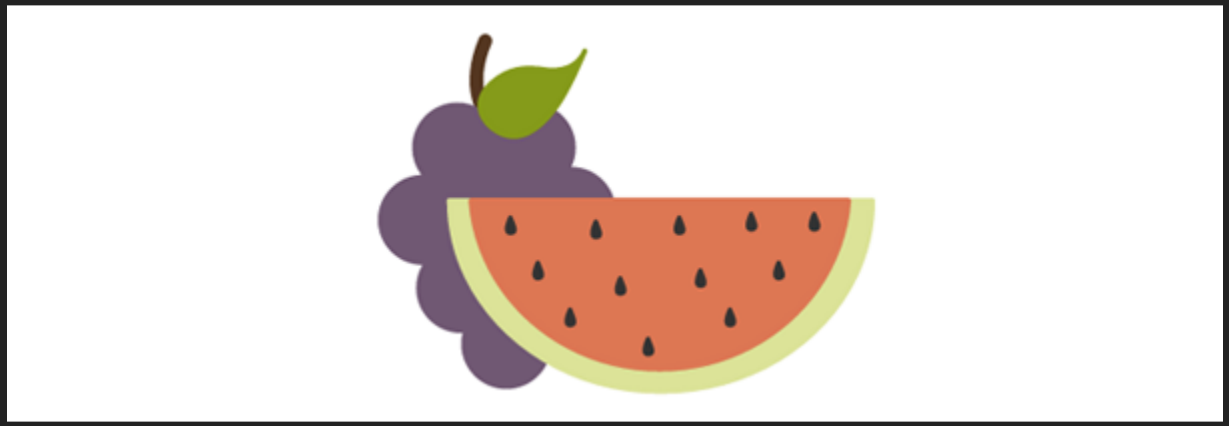
- element that defines graphics that are referenced and rendered through the `xlink:href` attribute
- no visual output until referenced by unique `id`

```
<svg>
  <defs>
    <linearGradient id="funky-cool-stuff">
      <stop offset="0%" stop-color="#BBC42A">
      <stop offset="100%" stop-color="#765373">
    </linearGradient>
  </defs>

  <rect width="50" height="50" fill="url(#funky-cool-stuff)">
</svg>
```

NOTE ON STACKING ORDER

- `z-index` doesn't work here
- the order in which SVG elements are stacked depends on their placement within the SVG fragment

```
<svg>
  <g id="grapes">
    <!-- stem paths -->
    <!-- grape paths -->
    <!-- leaf paths -->
  </g>

  <g id="watermelon">
    <!-- outside paths (green) -->
    <!-- inside paths (red) -->
    <!-- seeds paths -->
  </g>
</svg>
```



BASIC SHAPES & PATHS

SVG SHAPES

- Rectangles
- Circles
- Ellipses
- Straight Lines
- Polylines
- Polygons

RECTANGLE

```
<svg>
  <rect width="200"
        height="100"
        fill="#BBC42A" />
</svg>
```

Example

RECTANGLE

Options:

`width` - width

`height` - height

`fill` - fill color

RECTANGLE

Additional options:

`x` - horizontal position within workspace

`y` - vertical position within workspace

`rx` - horizontal radius (for rounded corners)

`ry` - vertical radius (for rounded corners)

`stroke` - border color

`stroke-width` - border width

CIRCLE

```
<svg>
  <circle cx="75"
          cy="75"
          r="75"
          fill="#ED6E46" />
</svg>
```

Example

CIRCLE

Options:

`cx` - horizontal location of circle center within workspace

`cy` - vertical location of circle center within workspace

`r` - size of circle radius

`fill` - fill color

ELLIPSE

```
<svg>
  <ellipse cx="100"
           cy="100"
           rx="100"
           ry="50"
           fill="#7AA20D" />
</svg>
```

Example

ELLIPSE

Options:

`cx` - horizontal location of shape center within workspace

`cy` - vertical location of shape center within workspace

`rx` - horizontal radius

`ry` - vertical radius

`fill` - fill color

LINE

```
<svg>
  <line x1="5"
        y1="5"
        x2="100"
        y2="100"
        stroke="#765373"
        stroke-width="8"/>
</svg>
```

Example

LINE

Options:

`x1` - horizontal start of line

`y1` - vertical start of line

`x2` - horizontal end of line

`y2` - vertical end of line

`stroke` - line color

`stroke-width` - line width

POLYLINE (OPEN SHAPE)

```
<svg>
  <polyline points="0,40 40,40 40,80 80,80 80,120 120,120 120,160"

fill="white"

stroke="#BBC42A"

stroke-width="6"

stroke-linejoin="round" />
</svg>
```

Example

POLYLINE (OPEN SHAPE)

Options:

`points` - coordinate points along x and y axes

`fill` - fill color

`stroke` - line color

`stroke-width` - line width

`stroke-linejoin` - shape of angle point (rounded or sharp)

POLYGON (CLOSED SHAPE)

```
<svg>
  <polygon points="50,5 100,5 125,30 125,80 100,105 50,105 25,80 25,3
           fill="#ED6E46" />
</svg>
```

Example

POLYGON (CLOSED SHAPE)

Options:

`points` - coordinate points along x and y axes

`fill` - fill color

PATHS

- An SVG `path` element is an outline of a shape
- It follow this general markup pattern, where the `d` attribute includes data about the shape

```
<svg>
  <path d="<path data specifics>" />
</svg>
```

PATHS

- Path element for a green lime:

```
<svg width="258px" height="184px">
  <path
    fill="#7AA20D"
    stroke="#7AA20D"
    stroke-width="9"
    stroke-linejoin="round"
    d="M248.761,92c0,9.801-7.93,17.731-17.71,17.731c-0.319,0-0.617,
       0-0.935-0.021c-10.035,37.291-51.174,65.206-100.414,65.206
       c-49.261,0-90.443-27.979-100.435-65.334c-0.765,0.106-1.531,
       0.149-2.317,0.149c-9.78,0-17.71-7.93-17.71-17.731  c0-9.78,
       7.93-17.71,17.71-17.71c0.787,0,1.552,0.042,2.317,0.149C39.238,
       37.084,80.419,9.083,129.702,9.083    c49.24,0,90.379,27.937,
       100.414,65.228h0.021c0.298-0.021,0.617-0.021,0.914-0.021C240.8
       74.29,248.761,82.22,248.761,92z" />
</svg>
```

BUT DON'T WORRY!

That wasn't written by a human.

SVG EXPORTS

- Vector creation software (like Illustrator or Sketch) can export a vector graphic into SVG format
- This file can be run through an SVG optimizer to eliminate extra space and code, and decrease file size
  - http://petercollingridge.appspot.com/svg-optimiser
  - https://github.com/svg/svgo

THE TEXT ELEMENT

MARKUP

```
<svg width="20" height="20">
  <text
    x="xpos"
    y="ypos"
    fill="#000000"
    font-size="20"
    font-family="monospace">A</text>
</svg>
```

BASIC ATTRIBUTES - POSITIONING

Absolute

```
x - The start of text along the X axis
y - The start of text along the Y axis
```

Relative (when used with `<tspan>`)

```
dx - The start of text along the X axis
dy - The start of text along the Y axis
```

BASIC ATTRIBUTES

- `rotate` - Rotates the entire element

If you use one number, it rotates each letter the same amount

```
EX: rotate="20"
```

If you use a comma separated list of numbers, each corresponding letter will be rotated according to the lists values

```
rotate="20, 56, 34, 12"
```

BASIC ATTRIBUTES - TRANSFORM

Will transform the svg based on the value supplied to the property.

```
transform="rotate(40)"
```

Other possible values are:

- translate(x y)
- scale(x y)
- skewX()
- skewY()

BASIC ATTRIBUTES

TEXTLENGTH & LENGTHADJUST

Extends the length between each letter

```
textLength="10"
```

Ex: http://jsbin.com/bibegon/1/edit?html,output

BASIC ATTRIBUTES

TEXTLENGTH & LENGTHADJUST

lengthAdjust applies to the spacing between the characters

```
lengthAdjust="spacing"
```

Other possible values are:

- spacingAndGlyphs - which changes the width of the letters

Ex: http://jsbin.com/joxepiq/1/edit?html,output

ELEMENTS - <TSPAN>

Allows you to seperate words and give them positioning relative to its parent

```
<svg width="100" height="100">
  <text fill="#000000" x="0" y="20" font-size="20" font-family="'Lech
    Lorem
    <tspan dx="-15" dy="10" fill="#ff0000">Ipsum</tspan>
  </text>
</svg>
```

dx & dy can accept multiple values like rotate.

The letters positioning will reflect each value.

EX: http://jsbin.com/doqonu/2/edit?html,output

BASIC ATTRIBUTES - SPACING

KERNING, LETTER-SPACING & WORD-SPACING

kerning applies to the kerning tables included in the font

```
kerning="auto" (default, doesn't have to be specified)
```

Other can accept a value:

```
kerning="20"
```

BASIC ATTRIBUTES - SPACING

KERNING, LETTER-SPACING & WORD-SPACING

letter-spacing works like kerning but is supplemental and used in conjunction with kerning.

```
letter-spacing="normal|20|inherit"
```

BASIC ATTRIBUTES - SPACING

KERNING, LETTER-SPACING & WORD-SPACING

word-spacing changes the spacing between words.

```
word-spacing="normal|20|inherit"
```

Ex: http://jsbin.com/yexuqoc/1/edit?html,output

BASIC ATTRIBUTES

TEXT-DECORATION

text-decoration works like the css property

```
text-decoration="normal|inherit|underline|line-through"
```

Ex: http://jsbin.com/xadoxo/2/edit?html,output

PATH

Path allows a text to be written along a curved line

```
<svg>
  <defs>
    <path
      id="someID"
      d="[software generated path]" />
  </defs>
  <use xlink:href="#someID" fill="" stroke="#000000" stroke-width="nu
  <text>
    <textPath xlink:href="#someID" otherProps...>Lorem Ipsum</textPat
  </text>
</svg>
```

```
xlink:href must match the path id property
startOffset refers to the offset distance where the text should start
<use /> element allows you to style the path itself, or its invisible
```

EX: http://jsbin.com/setise/1/edit?html,output

SVG Text Editor: https://github.com/alvin-milton/svg-editor

References:

- http://svgpocketguide.com/book/
- http://courseware.codeschool.com/you-me-svg/CodeSchool-YouMeSvg.pdf