You, Me & SVG!

You, Me & SVG!

This course will answer these questions:

- What are SVGs?
- When you should use SVGs over raster images?
- How do you build SVGs?
- How do you use SVG elements together to build icons or other images?

Prerequisites for this course include a working knowledge of HTML and CSS.

Level 1

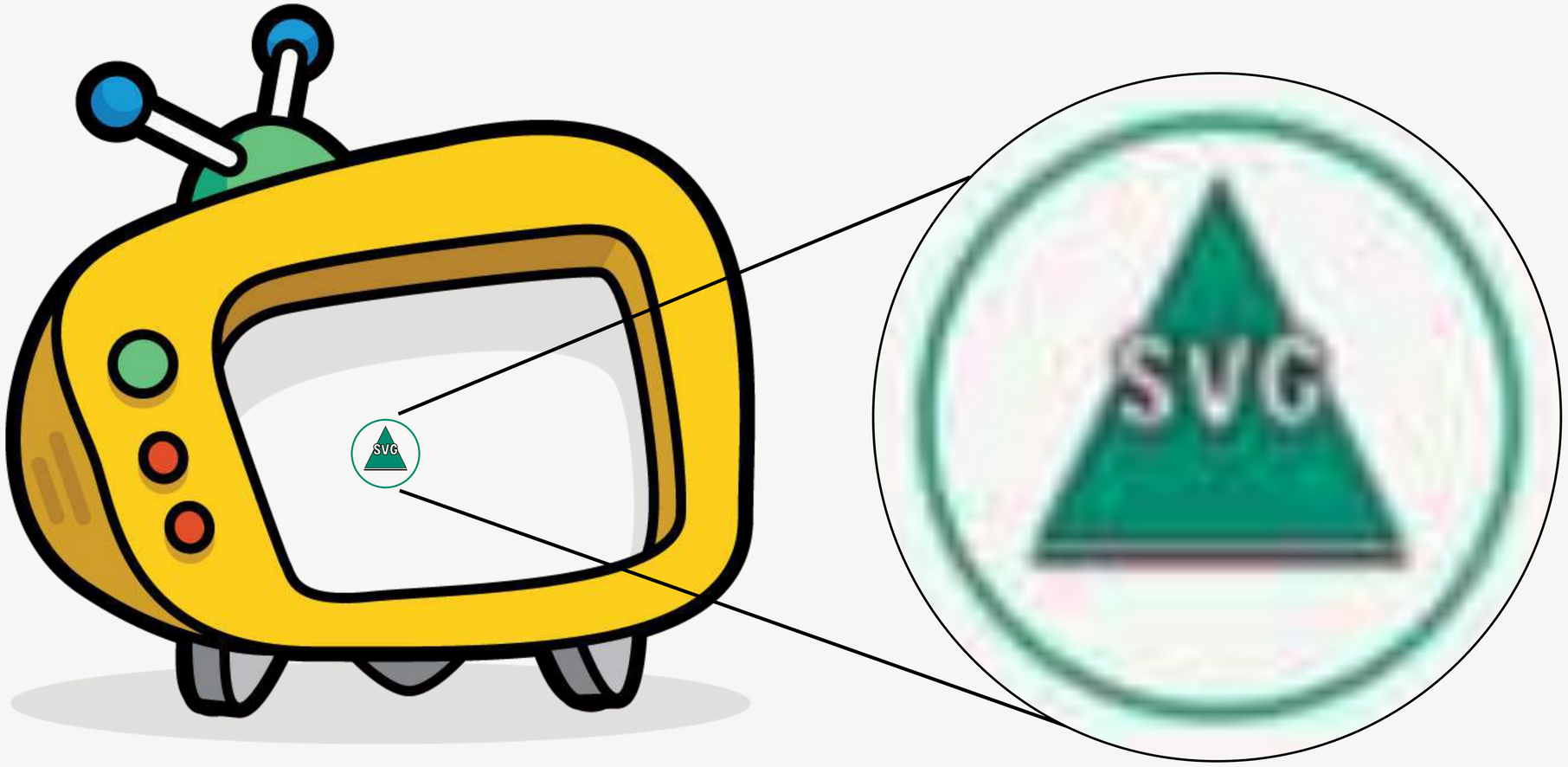
Oh, the Shapes You Can Make

Section 1 – SVG Fun

Welcome to Schmuffle Land!

All-New X59 Retina Screen

How does the badge look on the X59 Retina screen?



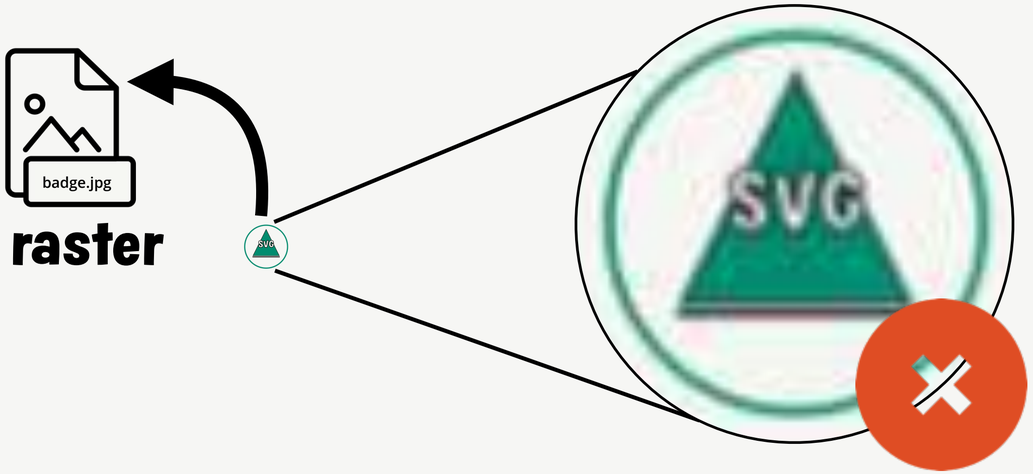
90,000px x 60,000px

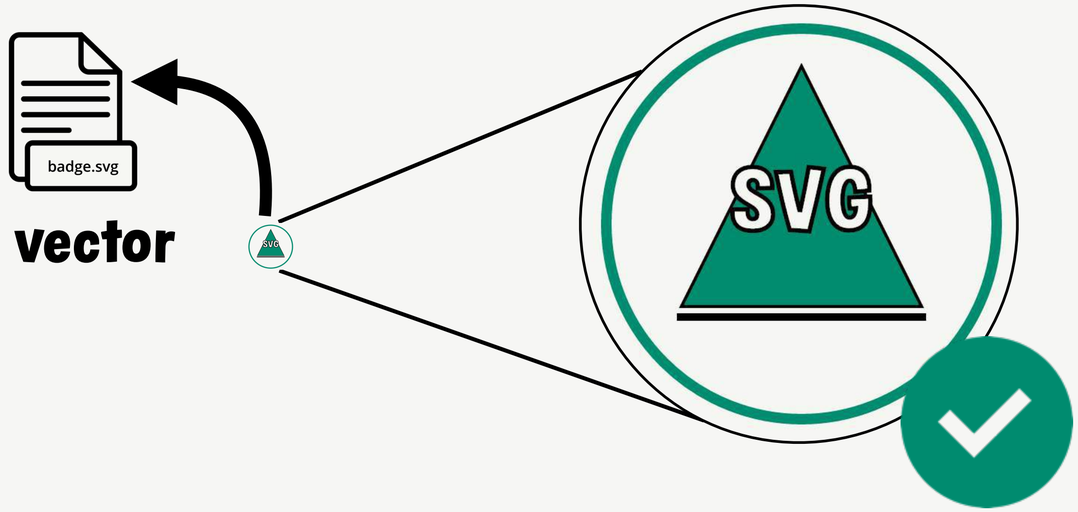
ICK, it's blurry!

You, Me & SVG!

Scalable Vector Graphics FTW

Raster images don’t work for every size screen. Future-proof your assets by using SVG!





Our SVG will also work for any future size screens — even ones the size of Schmuffle Land itself!

Including SVG as <img> Source

A common way to use SVGs is to treat them like any other file type and include them with an **img** tag.

index.html

```
<!DOCTYPE html>
<html>
  <head>…</head>
  <body>
    <h1>My First SVG</h1>
    <img src='our_first.svg'>
  </body>
</html>
```

How do we actually create an SVG, though?

Creating Our First SVG Element

Let’s jump right into an SVG file and create our own.

our_first.svg

```
<svg>

</svg>
```

The first step is to use the SVG element.

This looks similar to HTML tags because both HTML and SVG are types of XML.

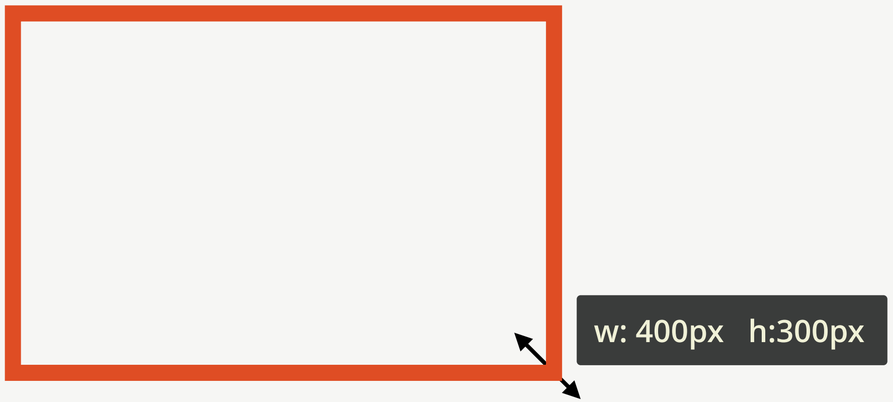
Setting the SVG's Viewport Size

We are going to set the window through which the SVG will be visible. This frame or canvas we draw our SVG on is called the viewport.

our_first.svg

```
<svg height="300" width="400" >

</svg>
```

Setting width and height of the viewport



Specifying SVG Namespace and Version

We need to tell the browser that we are going to be using a different version of XML, with non-HTML tags, and *what* version of SVG we are going to use.

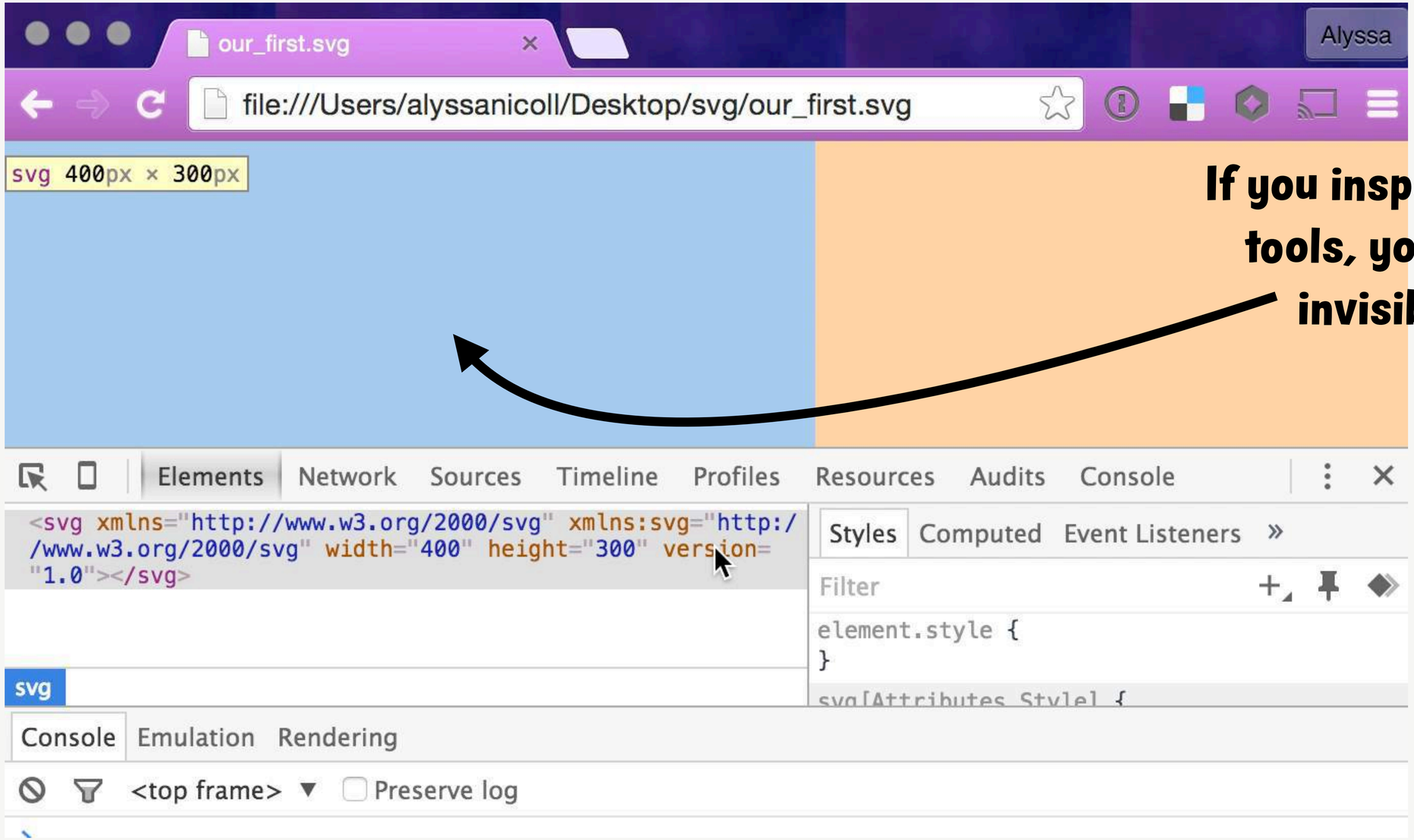
our_first.svg

```
<svg height="300"
     width="400"
     xmlns="http://www.w3.org/2000/svg"
     version="1.1">
</svg>
```

These long scary lines just tell the browser:
Hey, we are about to use some SVG tags here,
so get ready to draw!

Loading Our SVG in the Browser

Our canvas is now there, but we haven't drawn anything on it.



Ready to Start Building an Icon Now!

Let’s build out the X59 icon in SVG elements for the Schmuffle folk.

=

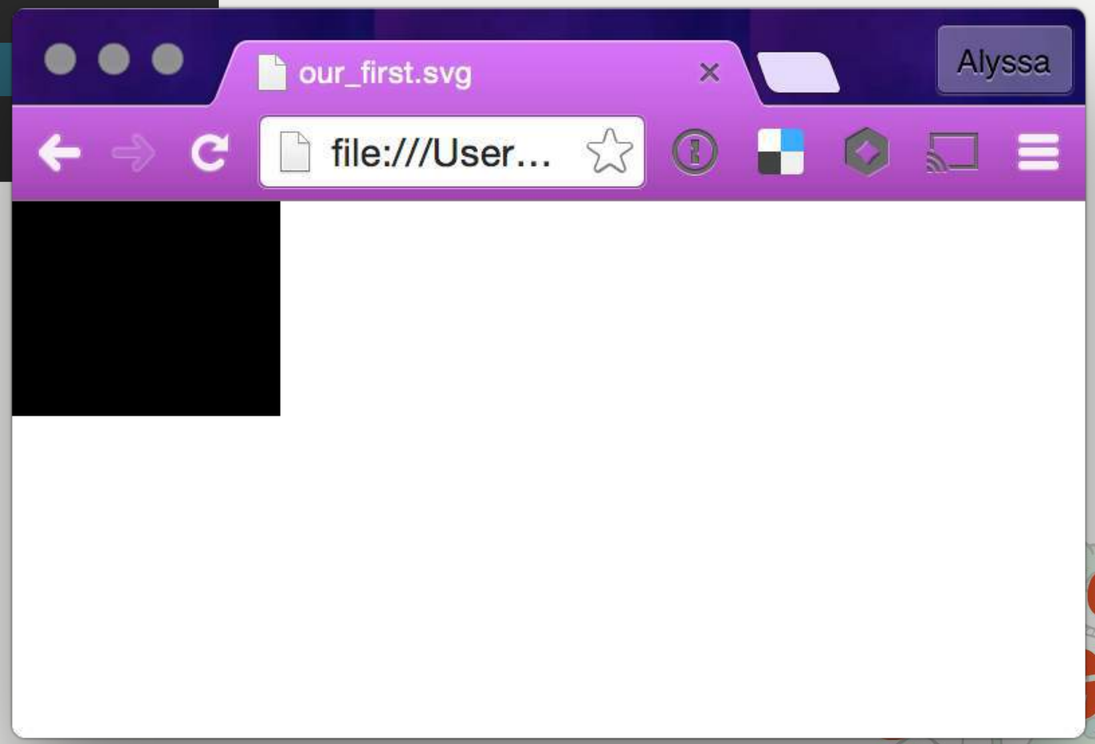
This is the simplified version of the X59 Retina screen!

Drawing Our First Rectangle

The **<rect> tag** allows us to draw rectangles on our SVG canvas.

our_first.svg

```
<svg height="300"
     width="400"
     xmlns="http://www.w3.org/2000/svg"
     version="1.1">
  <rect height="80" width="100"/>
</svg>
```



Adding a Second Rectangle

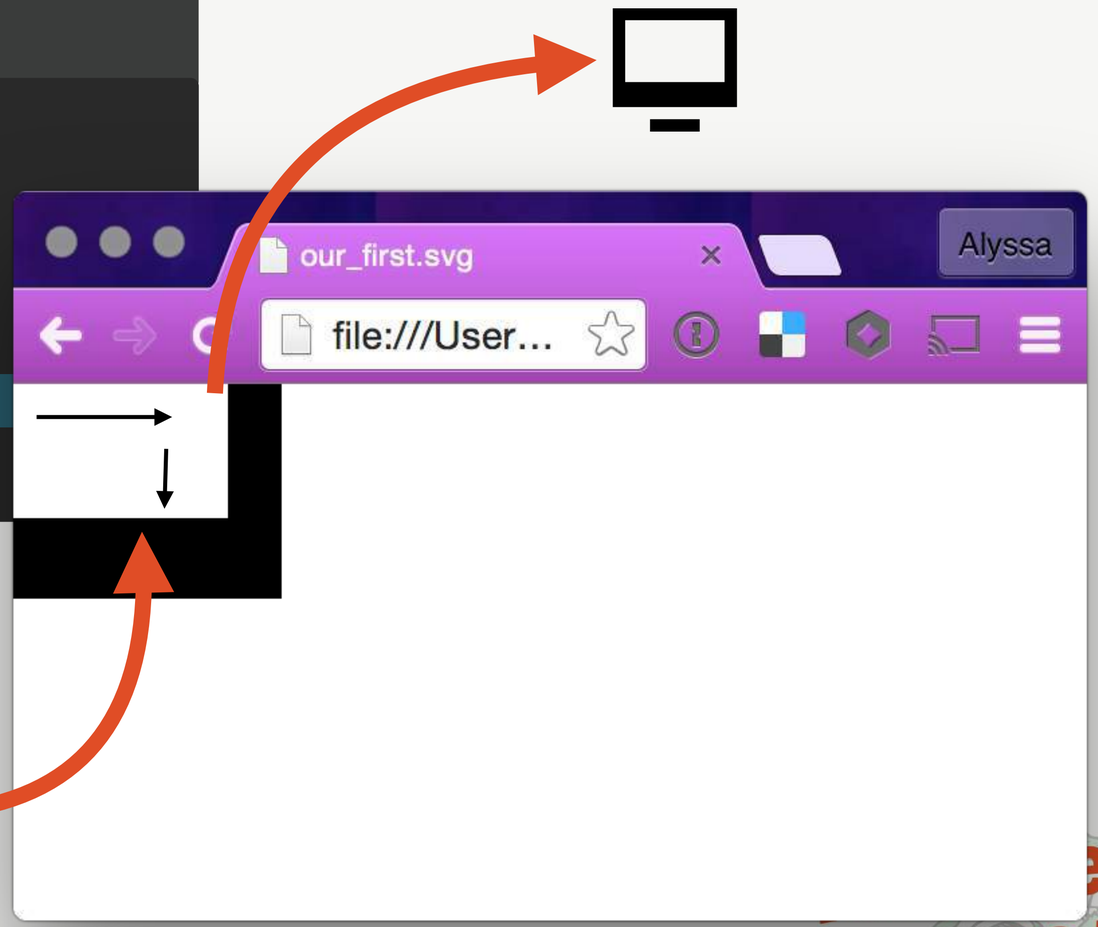
We will create a second **rect**, but this one will have a white background.

our_first.svg

```
<svg height="300"
     width="400"
     xmlns="http://www.w3.org/2000/svg"
     version="1.1">
  <rect height="80" width="100"/>
  <rect height="50" width="80" fill="white"/>
</svg>
```

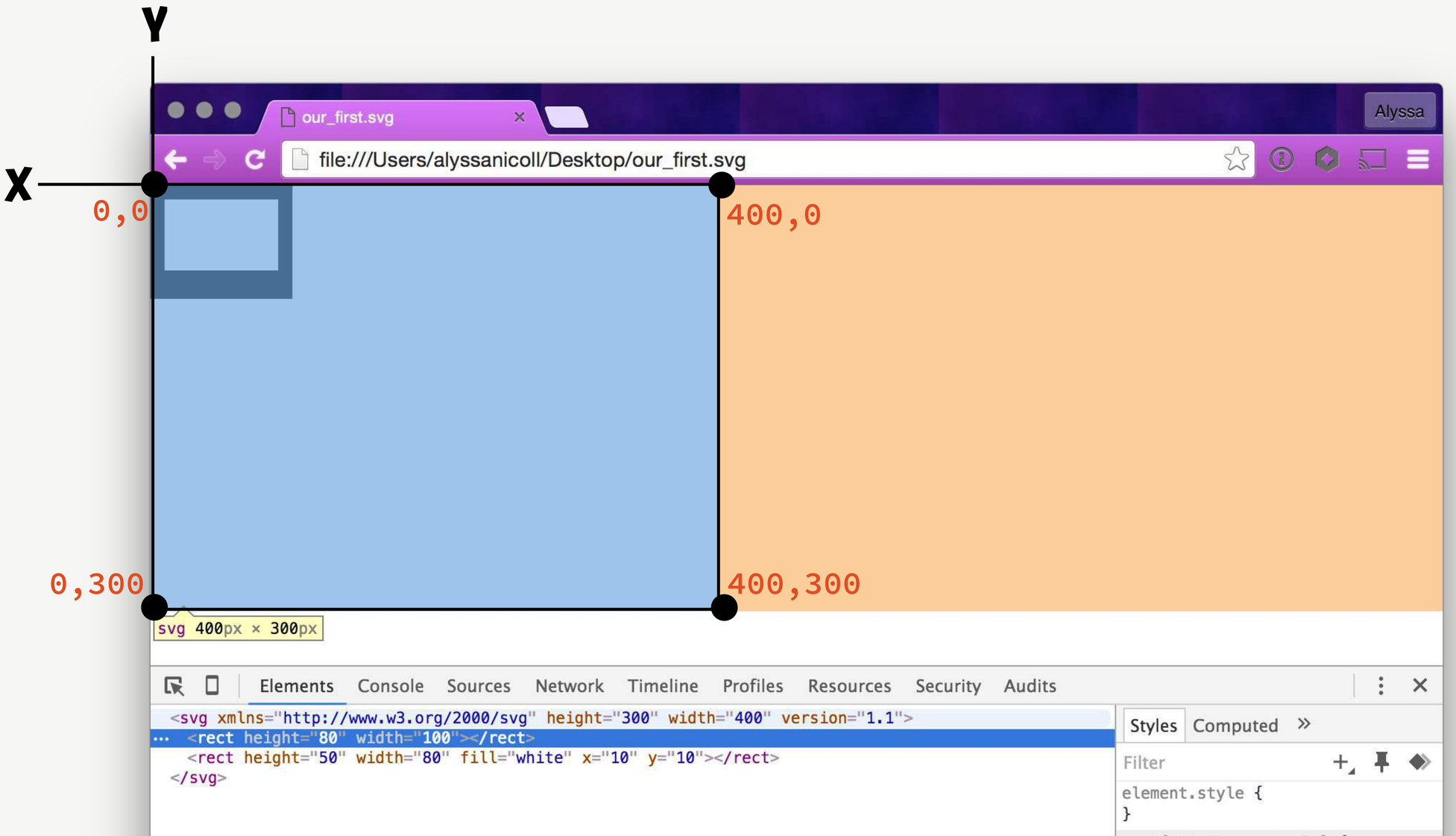
Fill is used to give background color.

How do we move this top white rect, though?



Understanding Viewport Coordinate System

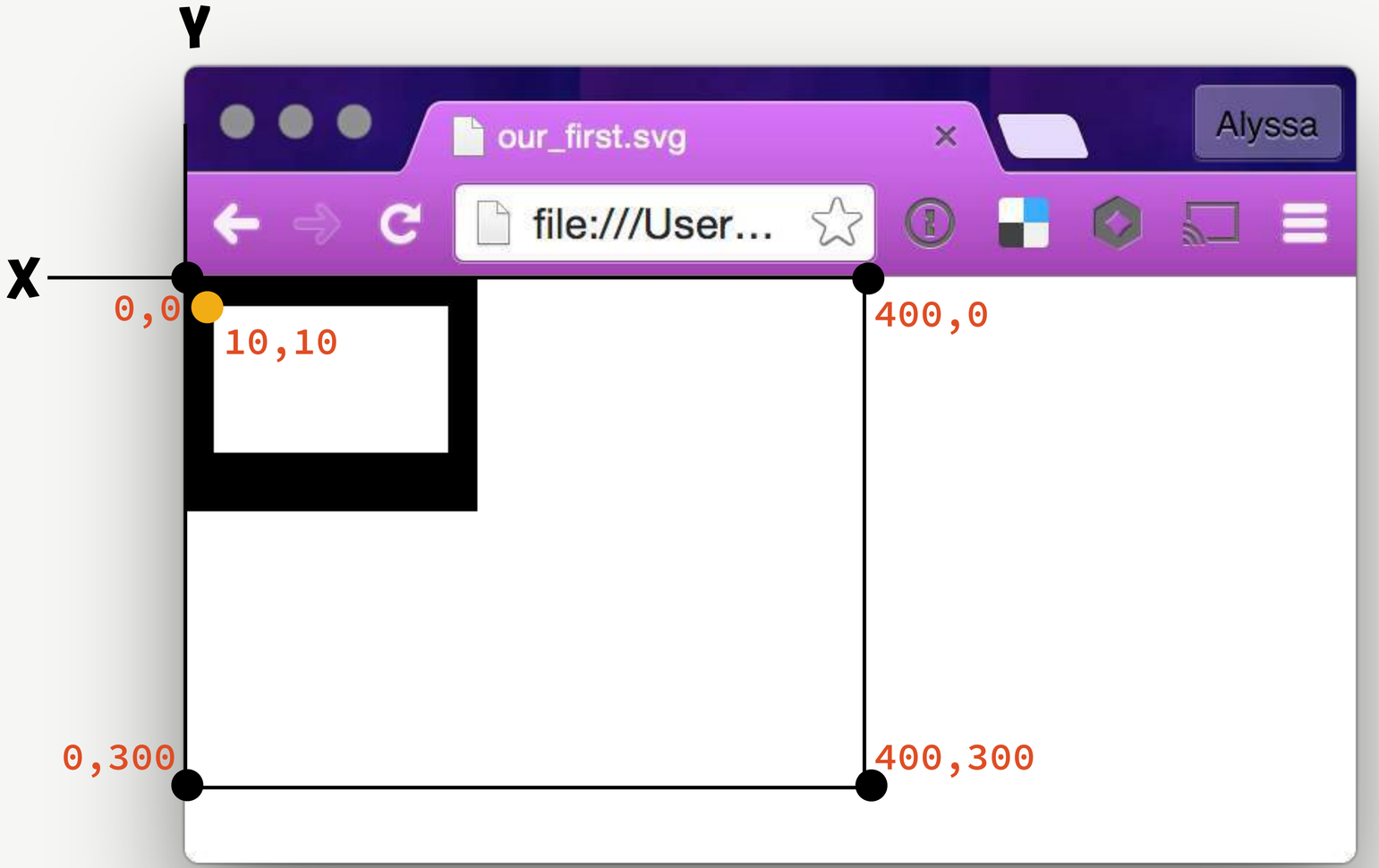
Right now, our rectangles are being drawn at 0,0. They are anchored to their top left point.



Moving the White Rectangle

We need to specify a new anchor point for our white rectangle.

```
<rect height="50" width="80" fill="white" x="10" y="10"/>
```

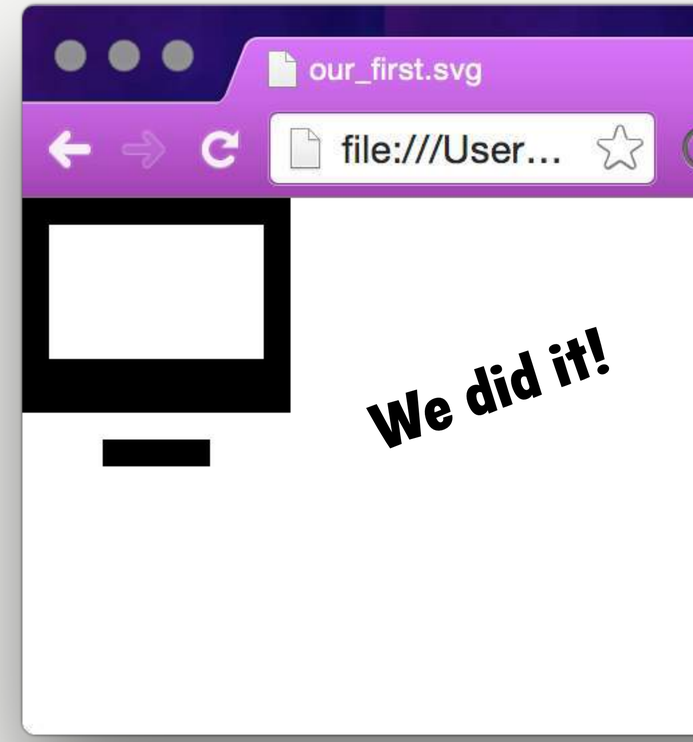


Creating One Last Rect to Finish the Icon

We will create one more rectangle below the other two. This will be the base of the X59 screen icon.

our_first.svg

```
<svg height="300"
     width="400"
     xmlns="http://www.w3.org/2000/svg"
     version="1.1">
  <rect height="80" width="100"/>
  <rect height="50" width="80" fill="white" x="10" y="10"/>
  <rect height="10" width="40" x="30" y="90"/>
</svg>
```

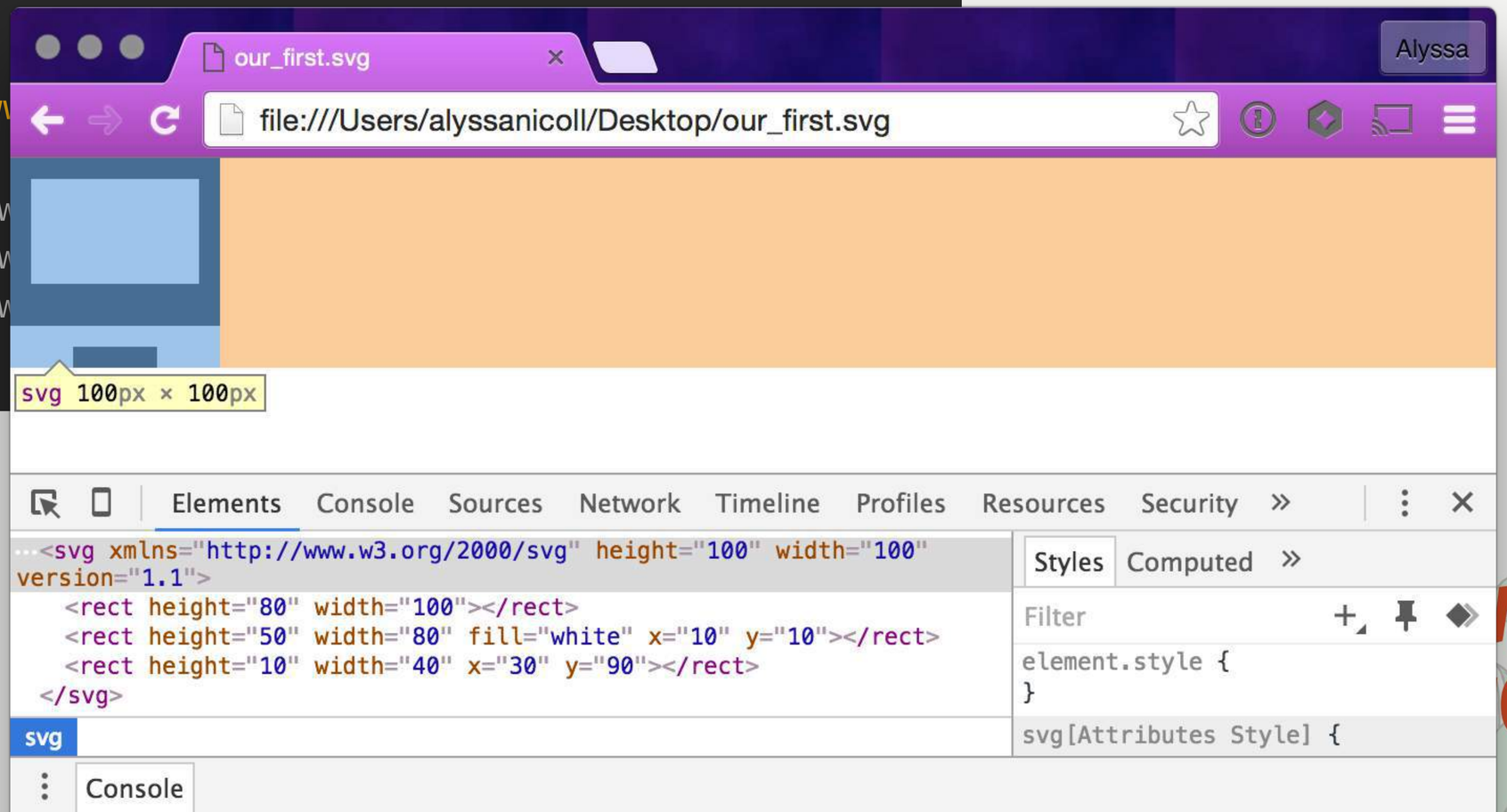


Adjusting Our Viewport

We want our icon to be 100px by 100px. Let's modify our viewport to this size!

our_first.svg

```
<svg height="100"
     width="100"
     xmlns="http://ww
     version="1.1">
  <rect height="80" w
  <rect height="50" w
  <rect height="10" w
</svg>
```



Level 2

Would You, Could You With a Badge?

Section 1 – Circles by the Ton

Drawing a New Icon

Let’s create the icon for our Schmuffle screen!

=

We will start by including the file...

index.html

```
<!DOCTYPE html>
<html>
  <head>...</head>
  <body>
    <h1>Schmuffle-Screen Icon</h1>
    <img src="phone_icon.svg"/>
  </body>
</html>
```

phone_icon.svg

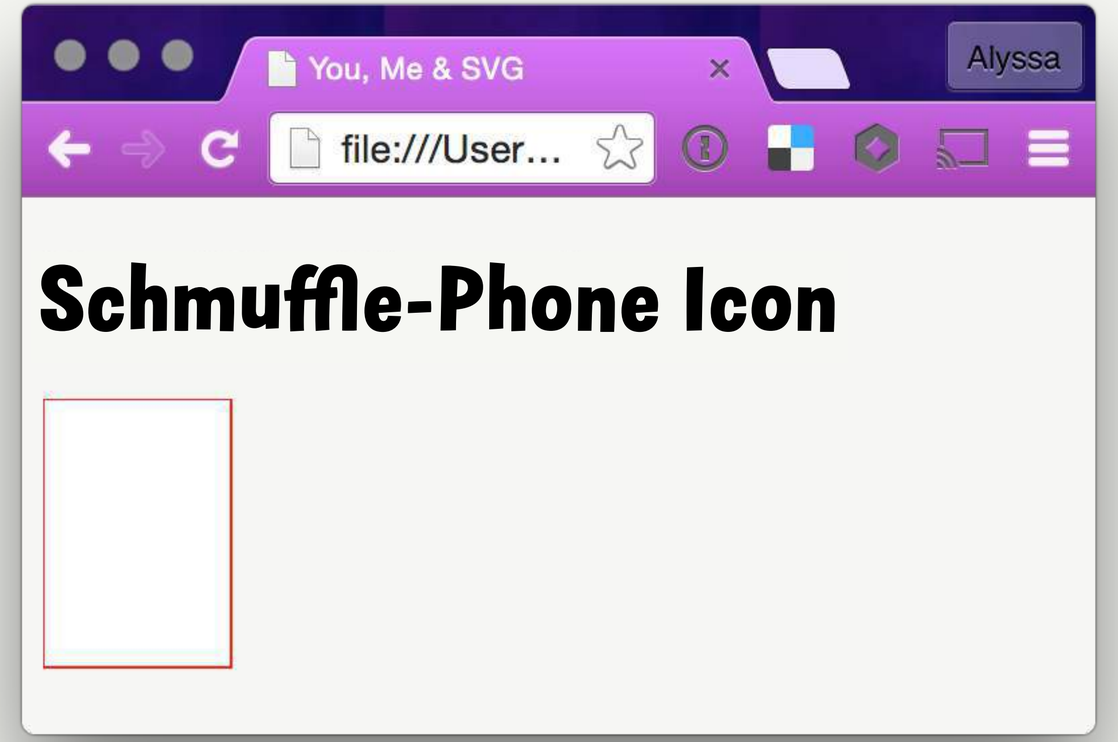
```
<svg...>
 <rect/>
</svg>
```

... and creating a rectangle.

Adding a Rectangle Stroke

We add a stroke color and we get a 1px outline.

```
<rect height="100" width="70" fill="white" stroke="#FF2626"/>
```

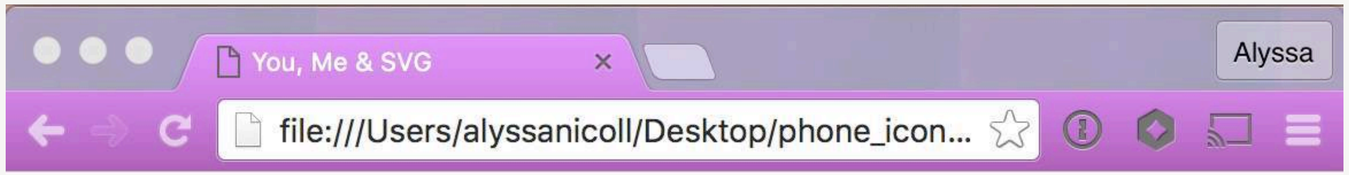


Creating a Thicker Outline

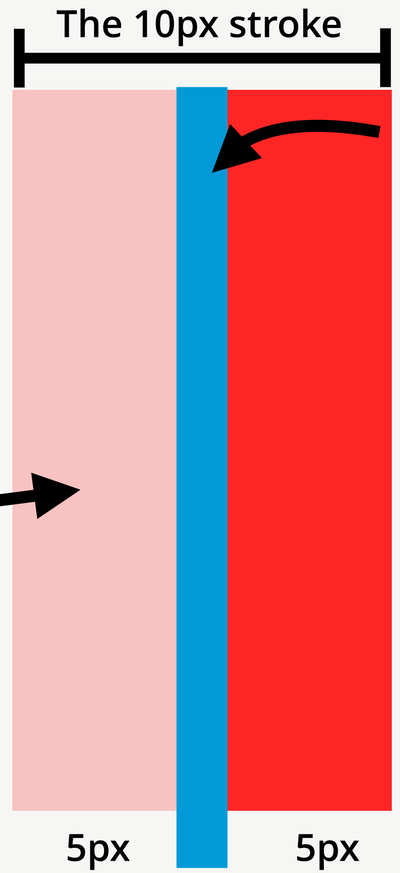
We can create a thicker line with the stroke-width attribute.

```
<rect height="100" width="70" fill="white" stroke="#FF2626" stroke-width="10"/>
```

Investigating the Stroke Cut



Our icons default position starts in the top left corner 0,0.



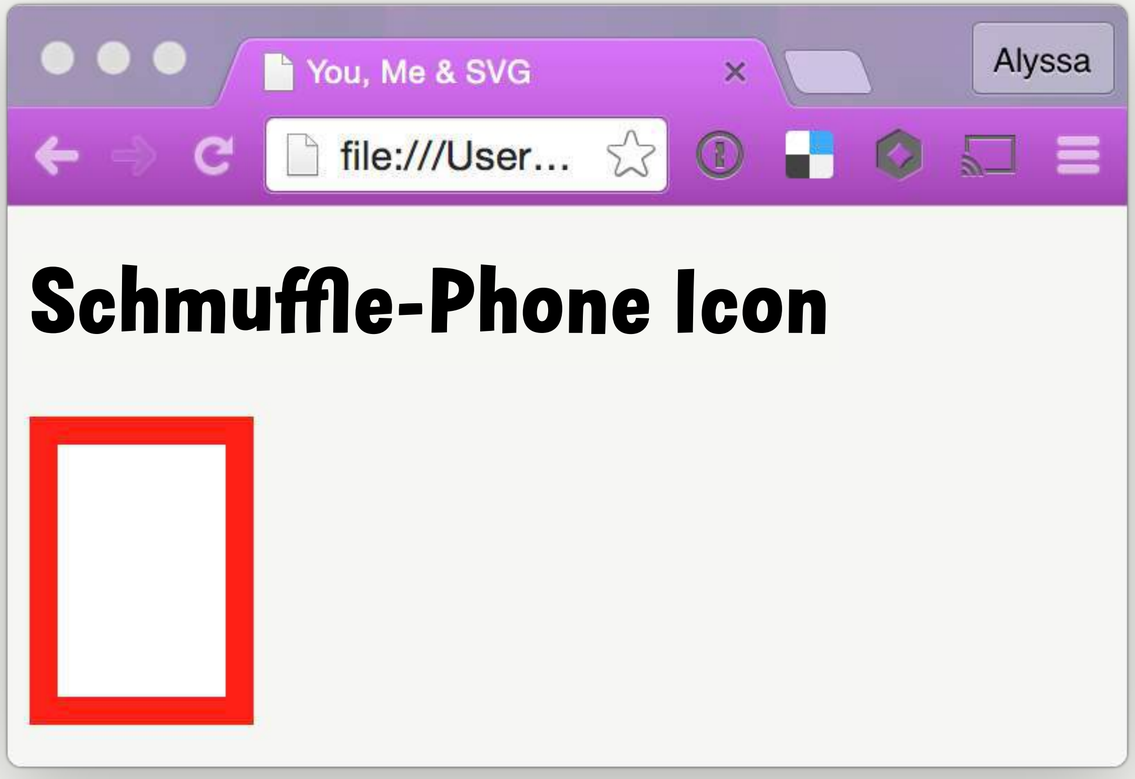
The outline stroke is centered along the rectangle's border.

You, Me & SVG!

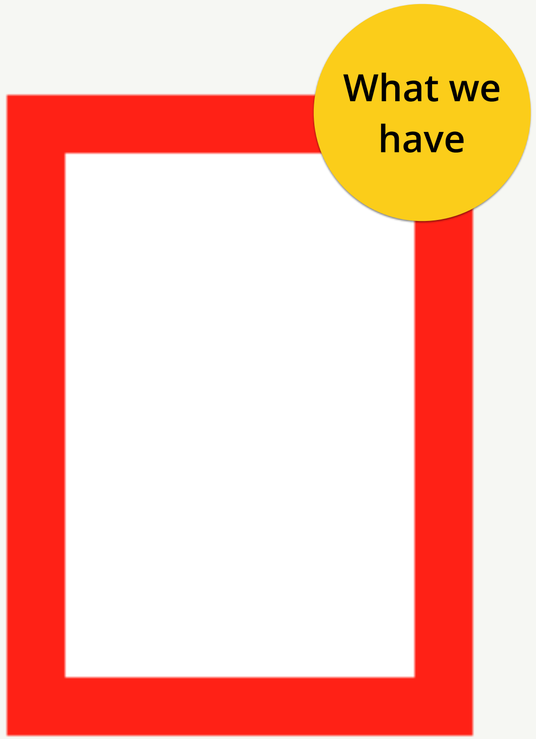
Positioning the Rect

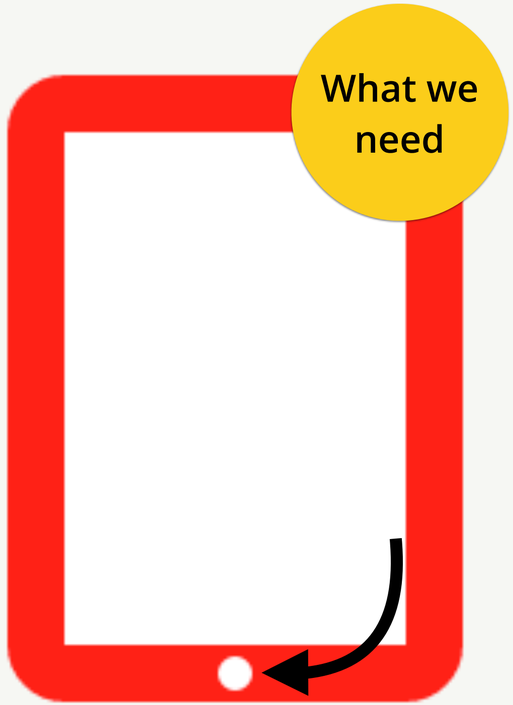
We can move the rectangle origin using x and y.

```
<rect height="100" width="70" fill="white" stroke="#FF2626" stroke-width="10"
      x="5" y="5"/>
```



What’s Left?

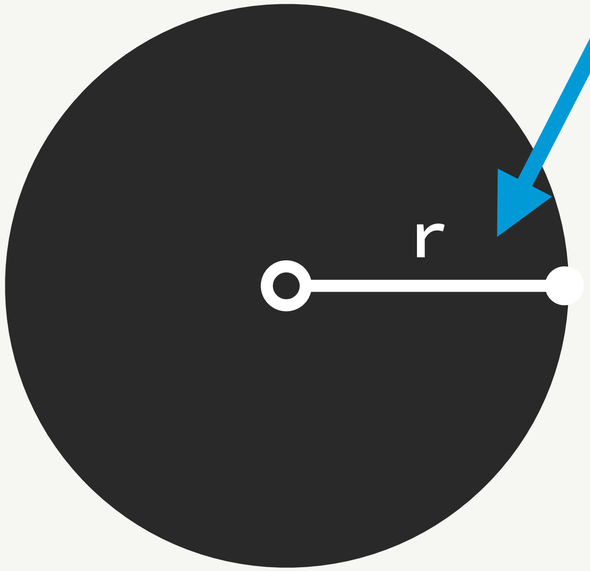




Drawing an SVG Circle

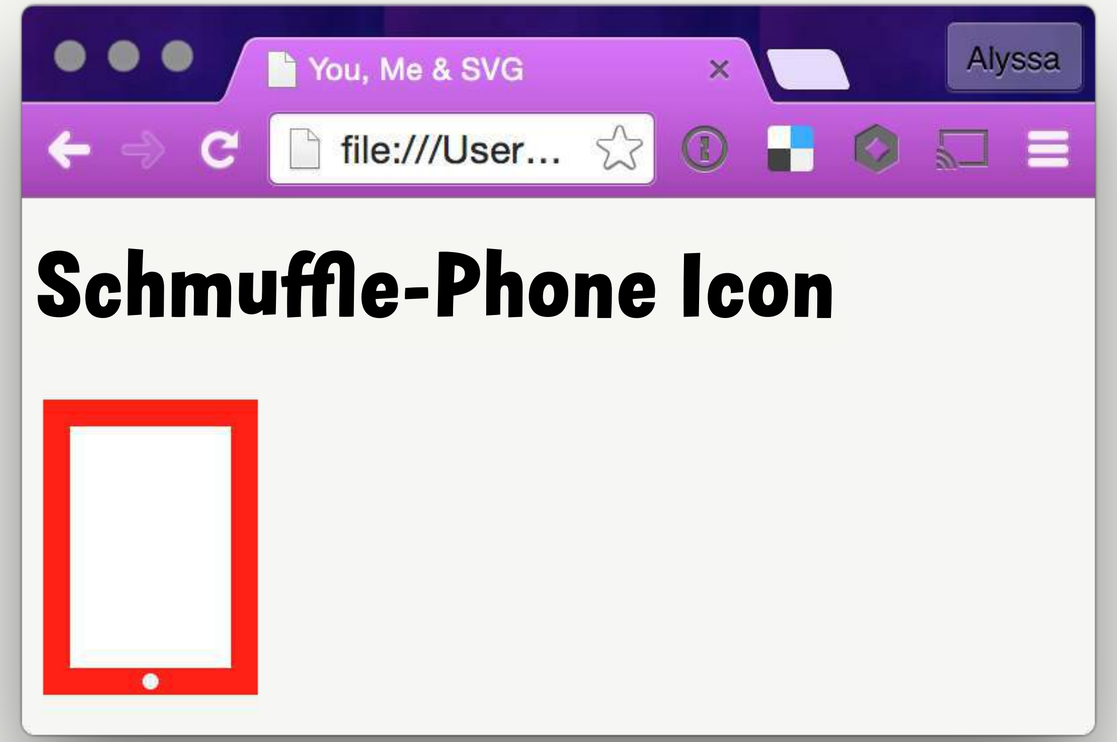
First, we specify the center points cx, cy and specify a radius.

```
<rect height="100" width="70" fill="white" stroke="#FF2626"
    stroke-width="10" x="5" y="5"/>
<circle cx="40" cy="105" r="3"/>
```



Filling the Circle White

```
<circle cx="40" cy="105" r="3" fill="white"/>
```



Now we just need rounded corners on our rect!

But First, We Need to Know About Ellipses

The center points and the fill are the same as the circle element.

```
<ellipse cx="50" cy="50" fill="blue" rx="10" ry="25"/>
```

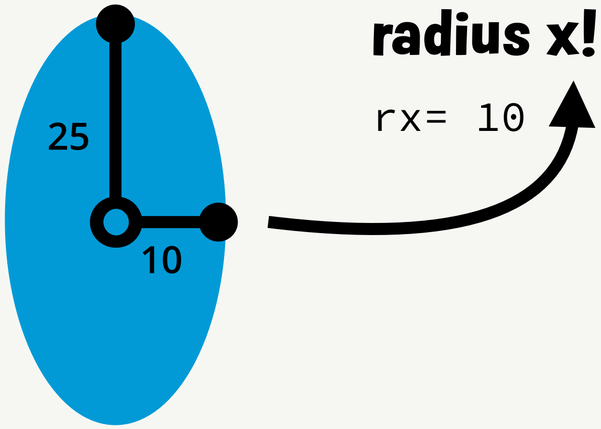
We specify both axes.

radius y!

`ry= 25`

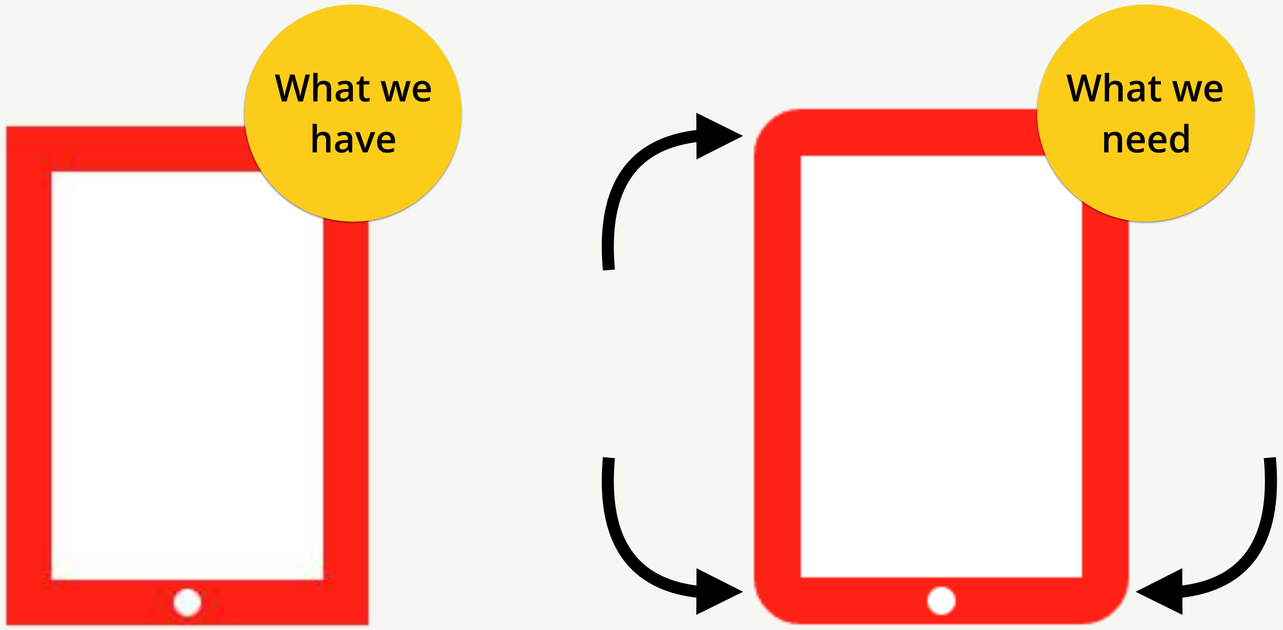
radius x!

`rx= 10`



Unlike a circle, which has the same radius for its x and y axes, ellipse has two different values for x and y axes.

We Still Need to Round the Corners



Rounding Rectangle Corners

To round the corners, we use rx and ry, which set the radiuses on an invisible ellipse.

```
<rect height="100" width="70" fill="white" stroke="#FF2626" stroke-width="10"
x="5" y="5" rx="10" ry="25"/>
```

rx: x-axis radius of the ellipse

ry: y-axis radius of the ellipse

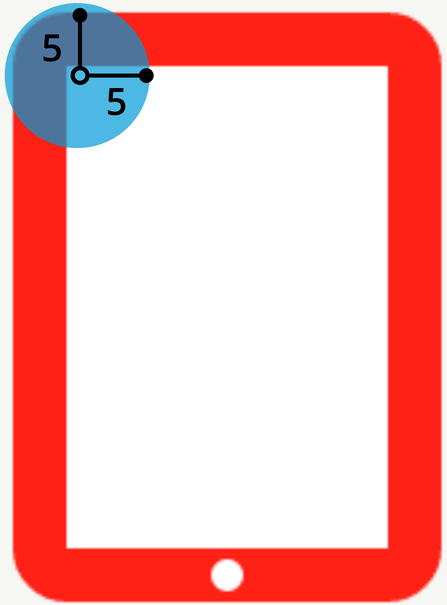
Drawing Our Icon's Rounded Corners

We want our rounded corners to be symmetrical, so we will use the same value for both rx and ry.

```
<rect height="100" width="70" fill="white" stroke="#FF2626"
stroke-width="10" x="5" y="5" rx="5" ry="5"/>
```

same as

```
<rect height="100" width="70" fill="white" stroke="#FF2626"
stroke-width="10" x="5" y="5" rx="5"/>
```



If you only specify rx, the browser will assume the same value for ry!

So Far We Have Used the <img> Tag to Import SVG

index.html

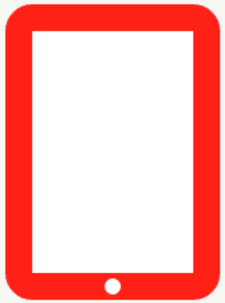
```
<!DOCTYPE html>
<html>
  <head>...</head>
  <body>
    <h1>Schmuffle-Screen Icon</h1>
    <img src="phone_icon.svg"/>
  </body>
</html>
```

SVG in an img tag

phone_icon.svg

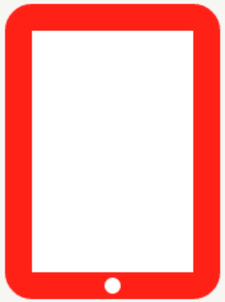
```
<svg height="110" width="80" xmlns="http…" ...>
  <rect height="100" width="70" fill="white" stroke="#FF2626"
        stroke-width="10" x="5" y="5" rx="5"/>
  <circle cx="40" cy="105" r="3" fill="white"/>
</svg>
```

With the image tag, you are able to animate the SVG as a whole...



rotate

scale



animate (transition) on/off screen

Changing SVG's Background?

What if we wanted to scale our phone icon's button? Could we do this with CSS?

Nope! You cannot select inner elements of the SVG when you're including it with an <img>...

But what if we wanted to change the color of our SVG's background?

```
<svg height="110" width="80" xmlns="http…" ... fill="color">
```

Again, nope. We would need to do this through CSS because the SVG tag has no "fill" attribute.

SO HOW CAN WE DO THESE STYLING AND ANIMATING THINGS?!

Another Way to Include Your SVG

There is another, more powerful way to include your SVG! Inline will allow us to control the individual parts of the SVG element.

index.html

```
<!DOCTYPE html>
<html>
  <head>...</head>
  <body>
    <h1>Schmuffle-Phone Icon</h1>

  </body>
</html>
```

SVG INLINE

```
<svg height="110" width="80" xmlns="http…" ...>
  <rect height="100"
        width="70"
        fill="white"
        stroke="#FF2626"
        stroke-width="10"
        x="5"
        y="5"
        rx="5"/>
  <circle cx="40" cy="105" r="3" fill="white"/>
</svg>
```

SVG Inline HTML

Inline gives us access to the inner elements to iterate on the style or animate with CSS!

index.html

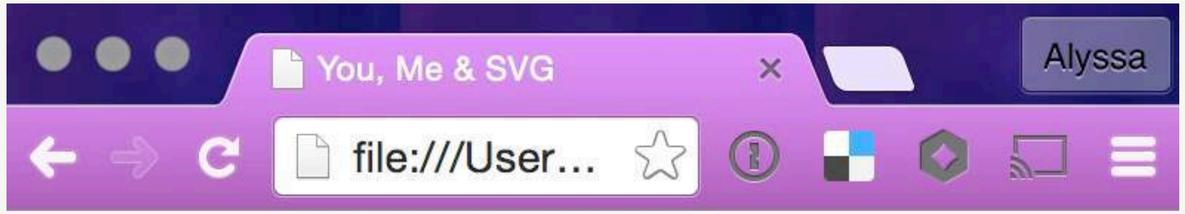
```
<!DOCTYPE html>
<html>
  <head>...</head>
  <body>
    <h1>Schmuffle-Phone Icon</h1>
      <svg height="110" width="80" xmlns="http…" ...>
        <rect height="100"
              width="70"
              fill="white"
              stroke="#FF2626"
              stroke-width="10"
              x="5"
              y="5"
              rx="5"/>
        <circle cx="40" cy="105" r="3" fill="white"/>
      </svg>
  </body>
</html>
```

Because the svg is inline ...

SVG Inline With Animation

style.css

```css
circle {
  animation: grow 2s infinite;
  transform-origin: center;
}

@keyframes grow {
  0%    {transform: scale(1);}
  50%   {transform: scale(0.5);}
  100% {transform: scale(1);}
}
```



Schmuffle-Phone Icon

We have access to animate/style individual pieces of the SVG in our CSS!

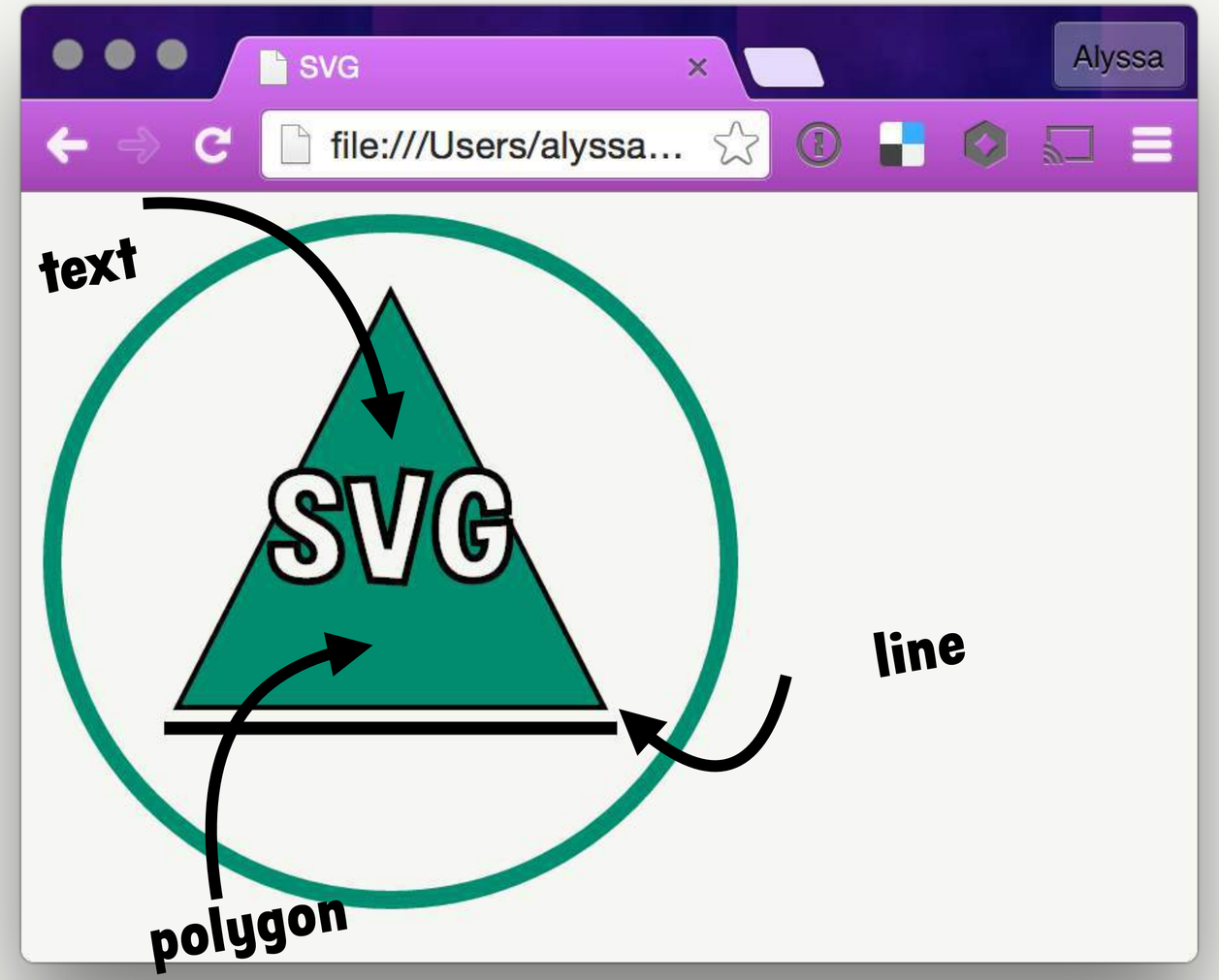
Level 2

Would You, Could You With a Badge?

Section 2 – Shapes for You

Drawing a Fancy Schmancy Badge

We'll need to learn a few more shapes to build this SVG, like text, polygon, and line.



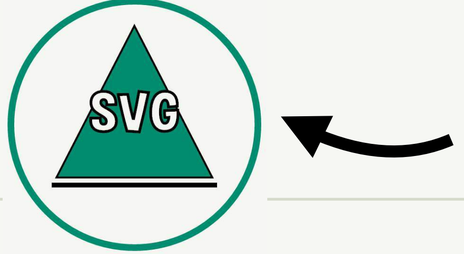
Creating a New SVG Tag

Let’s set the viewport size, version, and namespace attributes.

index.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>SVG</title>
  </head>
  <body>
      <svg  height="268"
            width="268"
            version="1.1"
            xmlns="http://www.w3.org/2000/svg">
      </svg>
   </body>
 </html>
```

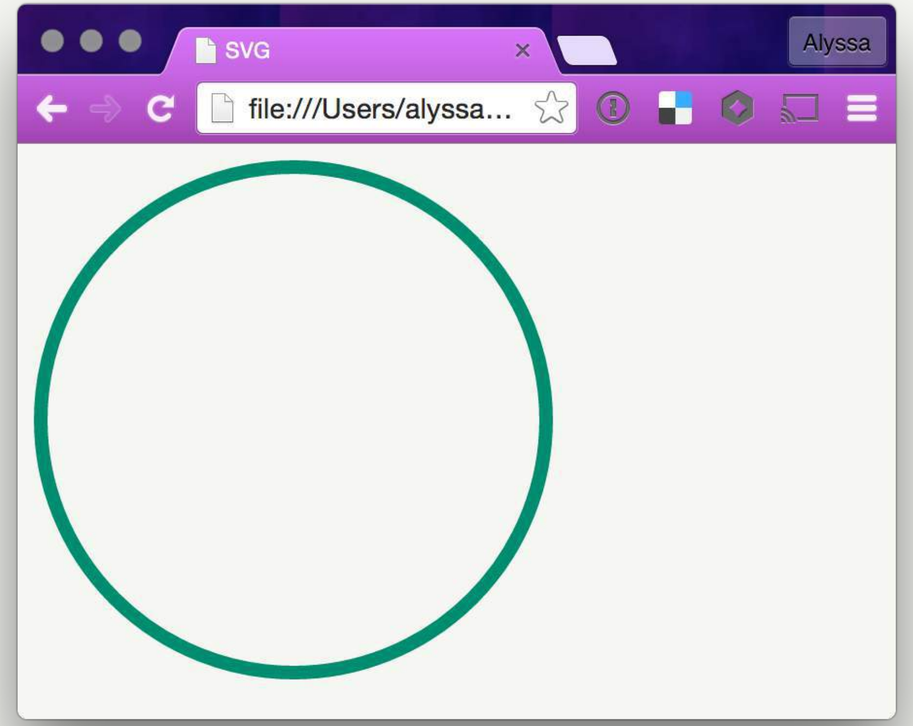
Starting Off With a Circle



Our circle should have a 130 radius, 7px green border, no fill color, and be centered at 134x134.

index.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>SVG</title>
  </head>
  <body>
    <svg ...>
      <circle r="130" cx="134" cy="134" fill="none"
    stroke="#008B6F" stroke-width="7"/>
    </svg>
  </body>
</html>
```



Many of these styles can be separated into a stylesheet.

Using a style.css File

This cleans up our HTML and puts the styles where they belong.

index.html

```
<!DOCTYPE html>
<html>
  <head>...
    <link rel="stylesheet" href="style.css"/>
  </head>
  <body>
    <svg ...>
      <circle r="130" cx="134" cy="134"/>
    </svg>
  </body>
</html>
```

Anything that sets coordinates, though, must be inline!

style.css

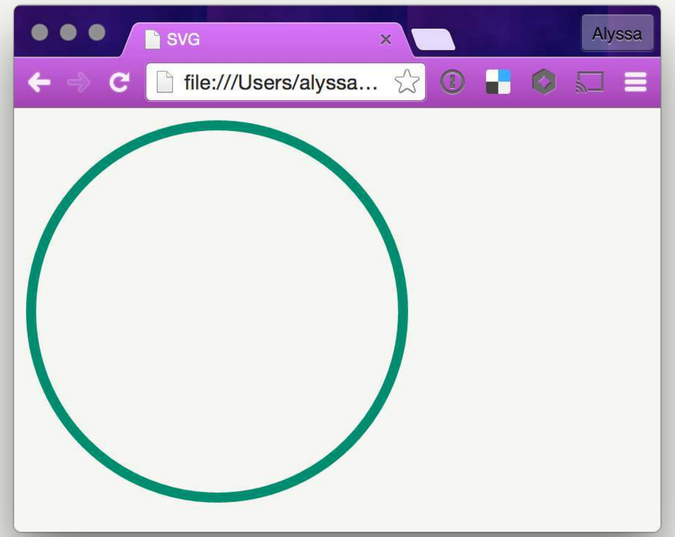
```
circle {
  fill: none;
  stroke: #008B6F;
  stroke-width: 7px;

}
```

Notice a unit identifier (px) is required in the CSS file!

Continuing to Build Our Badge

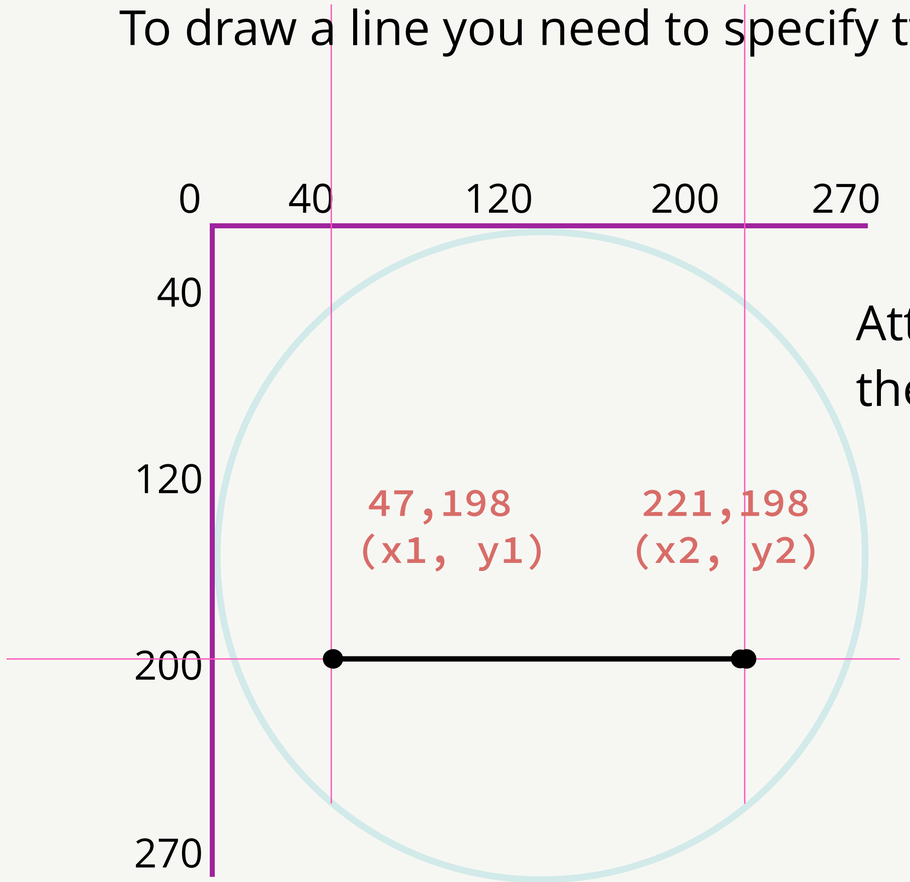
index.html

```
<!DOCTYPE html>
<html>
  <head>...
    <link rel="stylesheet" href="style.css"/>
  </head>
  <body>
    <svg ...>
      <circle r="130" cx="134" cy="134"/>
    </svg>
  </body>
</html>
```



Positioning the Line

To draw a line you need to specify two x,y points.



Attributes x1,y1 tell the line where to start.

Attributes x2,y2 tell the line where to end.

index.html

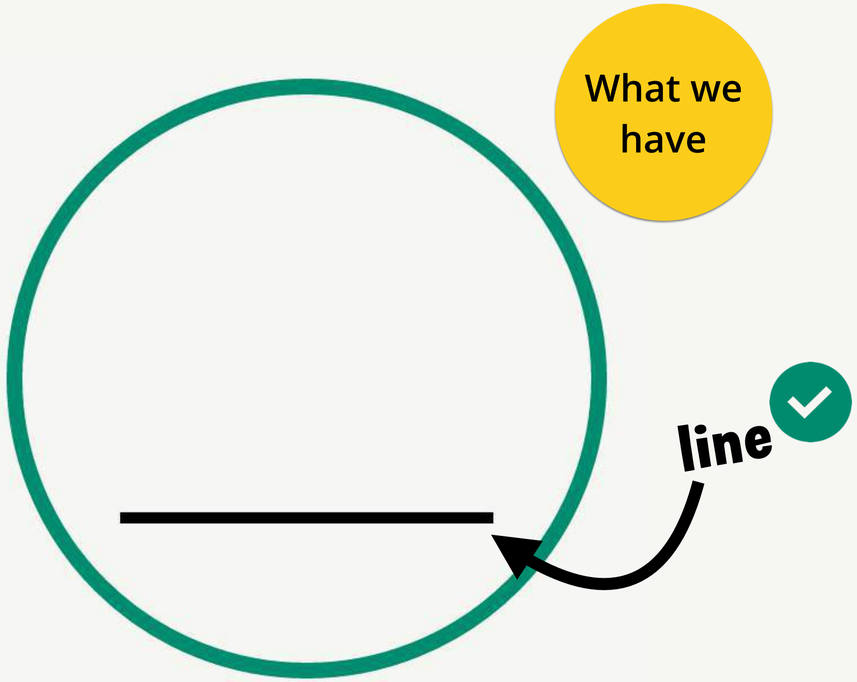
```
...
<line x1="47" y1="198" x2="221" y2="198" />
```

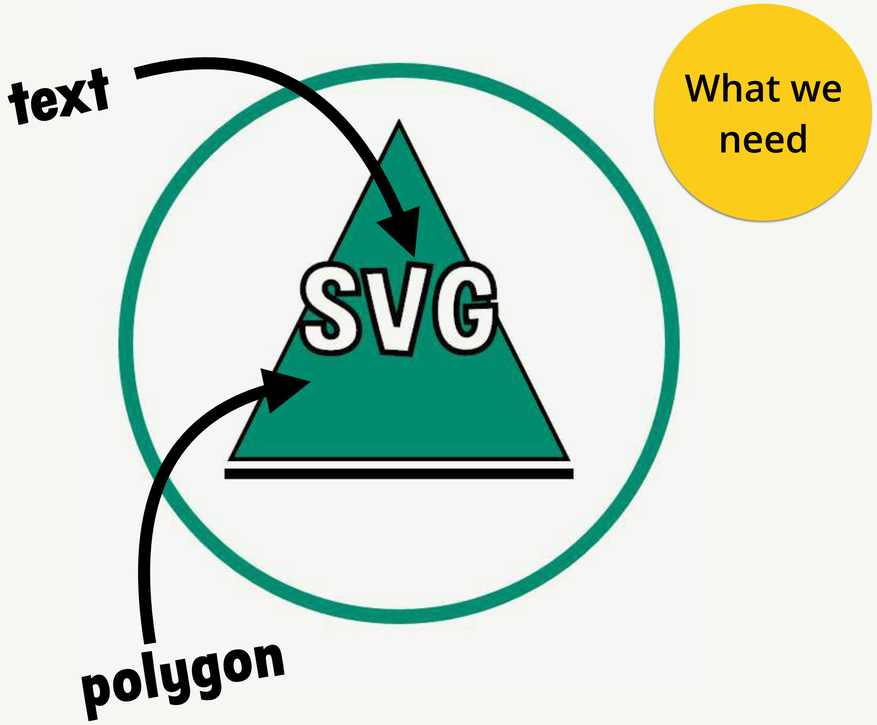
style.css

```
...
line {
  stroke: black;
  stroke-width: 5px;
}
```

SVG Text Element

We have the line of the badge — now we need the text!





Using the SVG Text Element

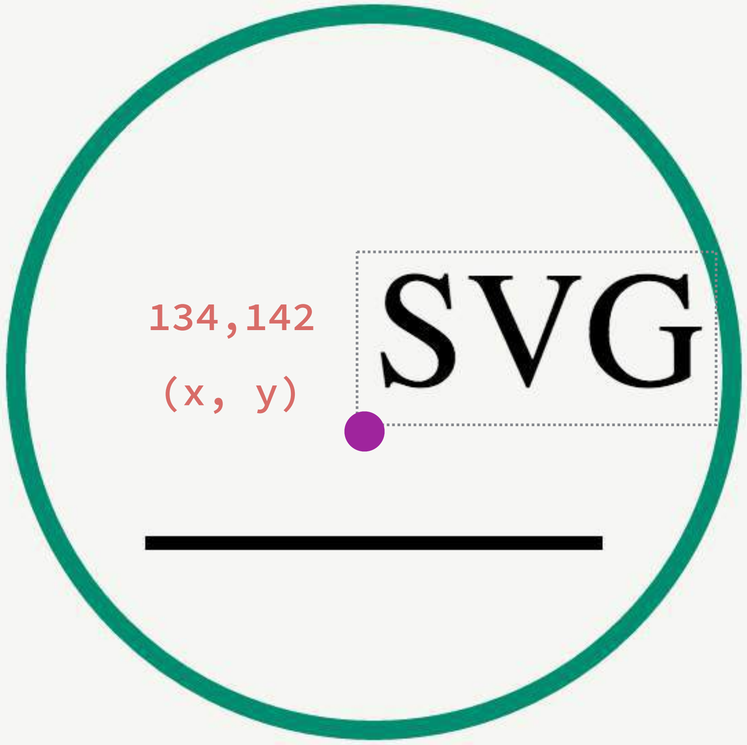
To get text to appear, we need to specify the anchor points and font size.

index.html

```
...
<text x="134" y="142">SVG</text>
```

style.css

```
text {
  font-size: 60px;
}
```



The default anchor point is bottom left of the text box.

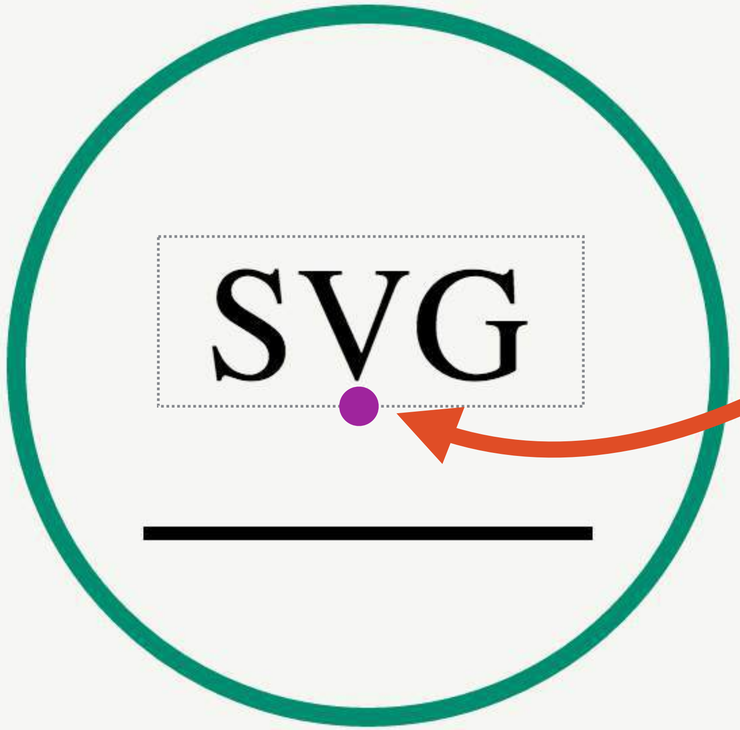
Changing the Default Text Anchor

index.html

```
...
<text x="134" y="142">SVG</text>
```

style.css

```
text {
  font-size: 60px;
  text-anchor: middle;
}
```



Now our text is centered.

Styling Our Text

index.html

```
...
<text x="134" y="142">SVG</text>
```

style.css

```
text {
  font-size: 60px;
  text-anchor: middle;
  font-family : 'FilmotypeMajor';
  stroke: #000;
  stroke-width: 3px;
  fill: #F6F7F3;
}
```

setting the font family

Setting stroke width and color

Setting the color of the font

Understanding What Must Be Inline

index.html

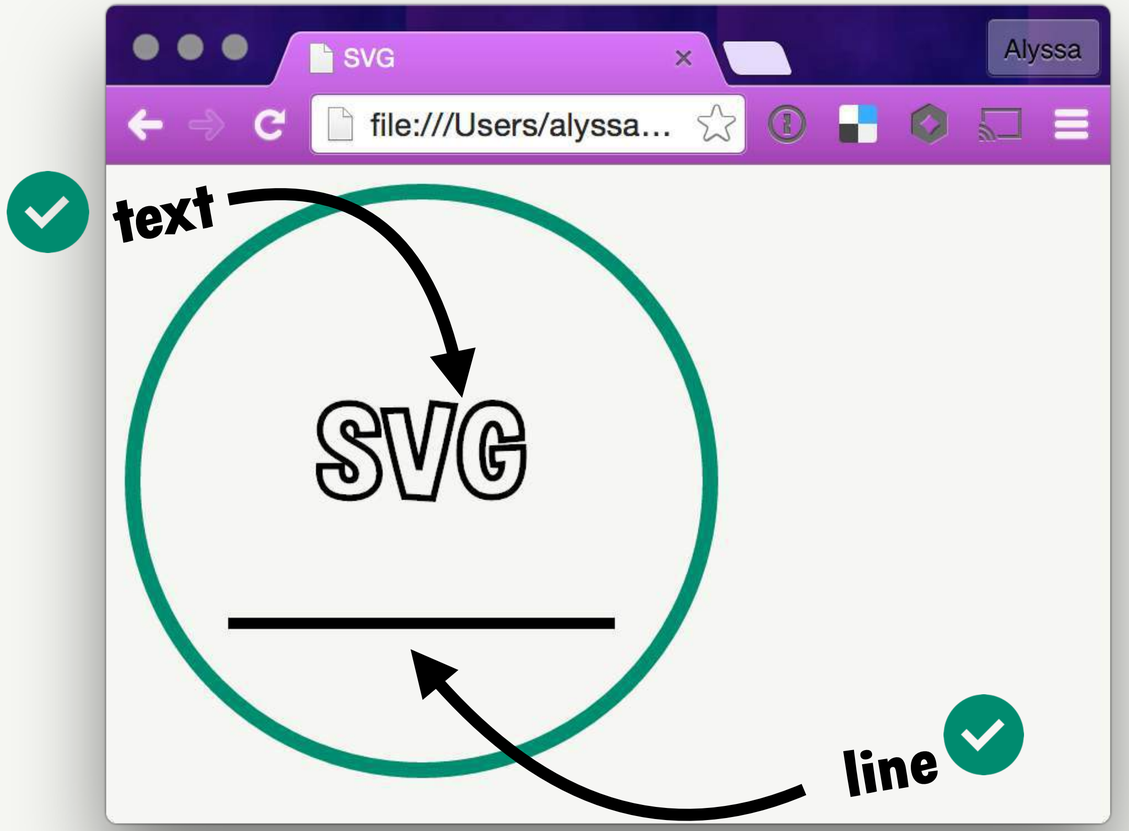
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>SVG</title>
    <link rel="stylesheet" href="style.css">
  </head>
  <body>
    <svg ...>

      <circle r="130" cx="134" cy="134"/>
      <line x1="47" y1="198" x2="221" y2="198"/>
      <text x="134" y="142">SVG</text>

    </svg>
  </body>
</html>
```

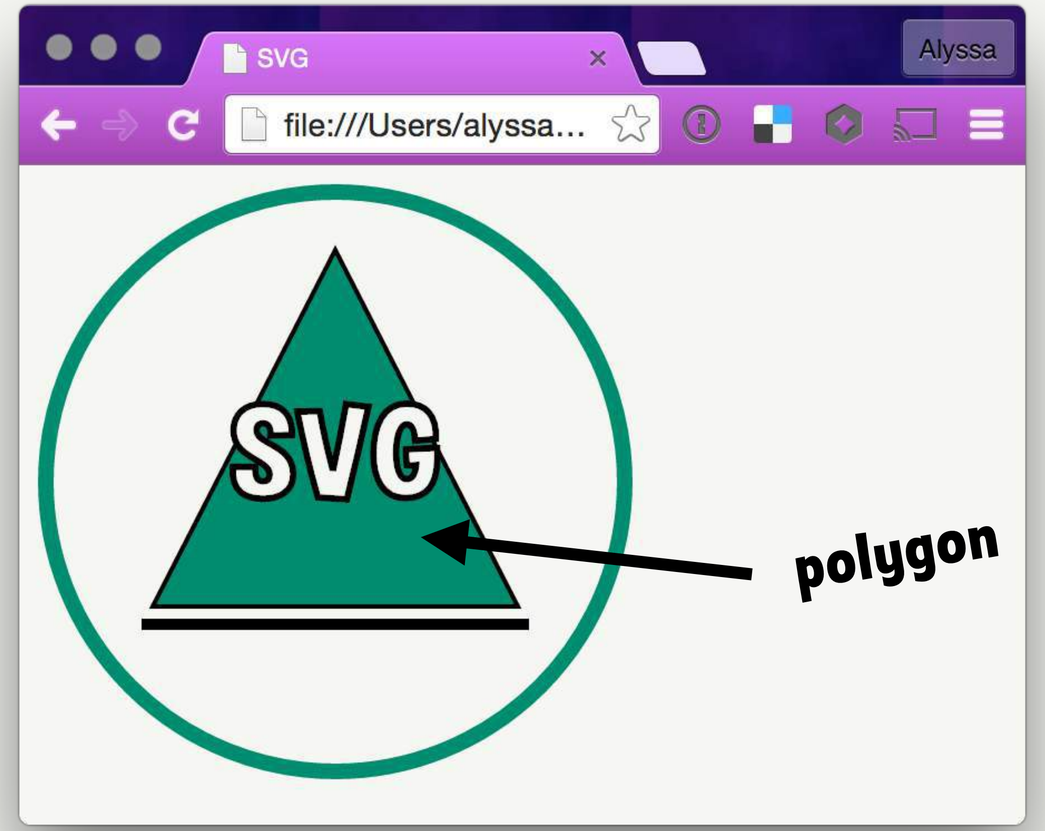
Must be inline — won't work in CSS

Any attribute to do with coordinates stays inline!

Drawing One Last Shape

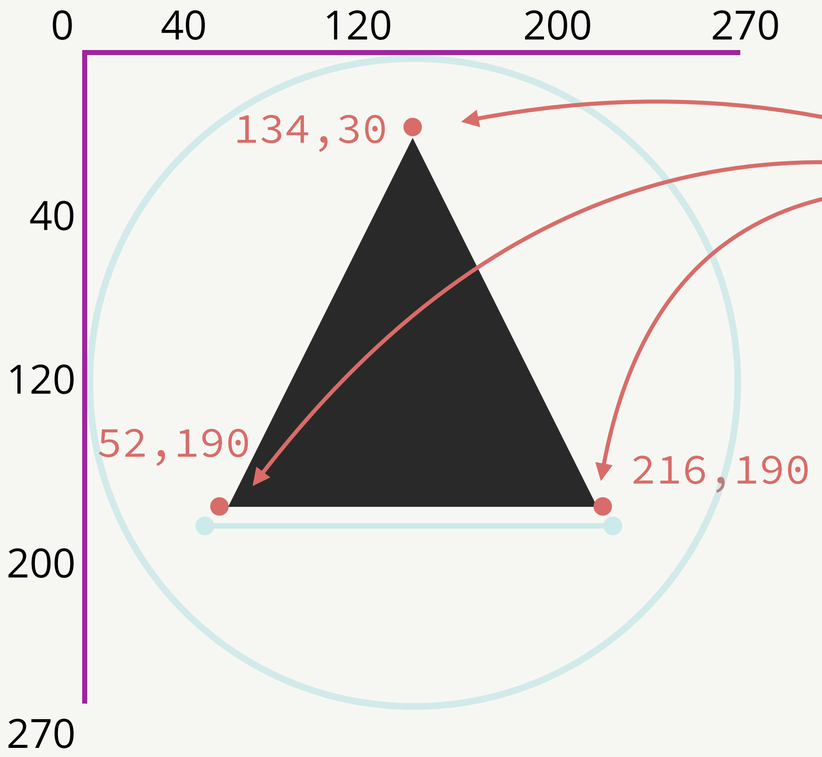
All we need now is the triangle in the background!





Introducing the SVG Polygon Element

The SVG <polygon> element is used to draw shapes with multiple (three or more) sides.



index.html **x,y for each point!**

```
<polygon points="52,190 134,30 216,190"/>
```

Polygon connects the x,y points to draw the shape and connects the last point to the first point.

Styling Our Polygon

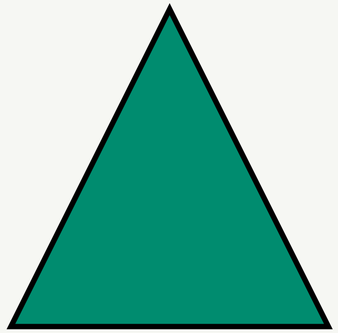
Now inside the stylesheet, we can give the polygon a fill, stroke, and stroke width so it looks as the badge example does.

index.html

```
<polygon points="52,190 134,30 216,190"/>
```

style.css

```
...
polygon {
  fill: #008B6F;
  stroke: black;
  stroke-width: 2px;
}
```



black 2px stroke

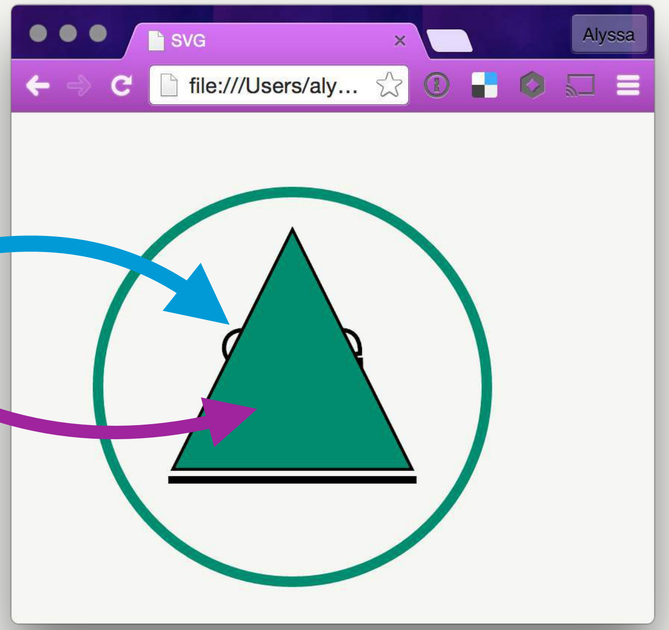
blue/green fill

Adding the Polygon to Our SVG

index.html

```
<!DOCTYPE html>
<html>
  <head>...</head>
  <body>
    <svg ...>

      <circle r="130" cx="134" cy="134"/>
      <line x1="47" y1="198" x2="221" y2="198"/>
      <text x="134" y="142">SVG</text>
      <polygon points="52,190 134,30 216,190"/>

    </svg>
  </body>
</html>
```

What in Schmuffle Land is happening?!



The triangle is plop right on top of our text.

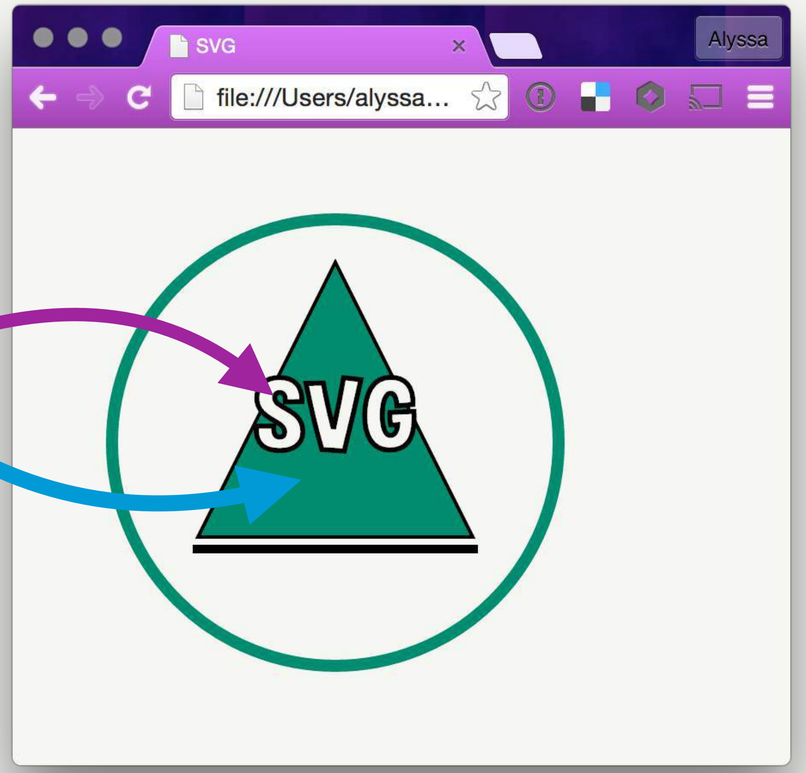
Fixing the Order

For our badge, the polygon needs to go first in the markup so it is drawn first. HTML elements stack this way too!

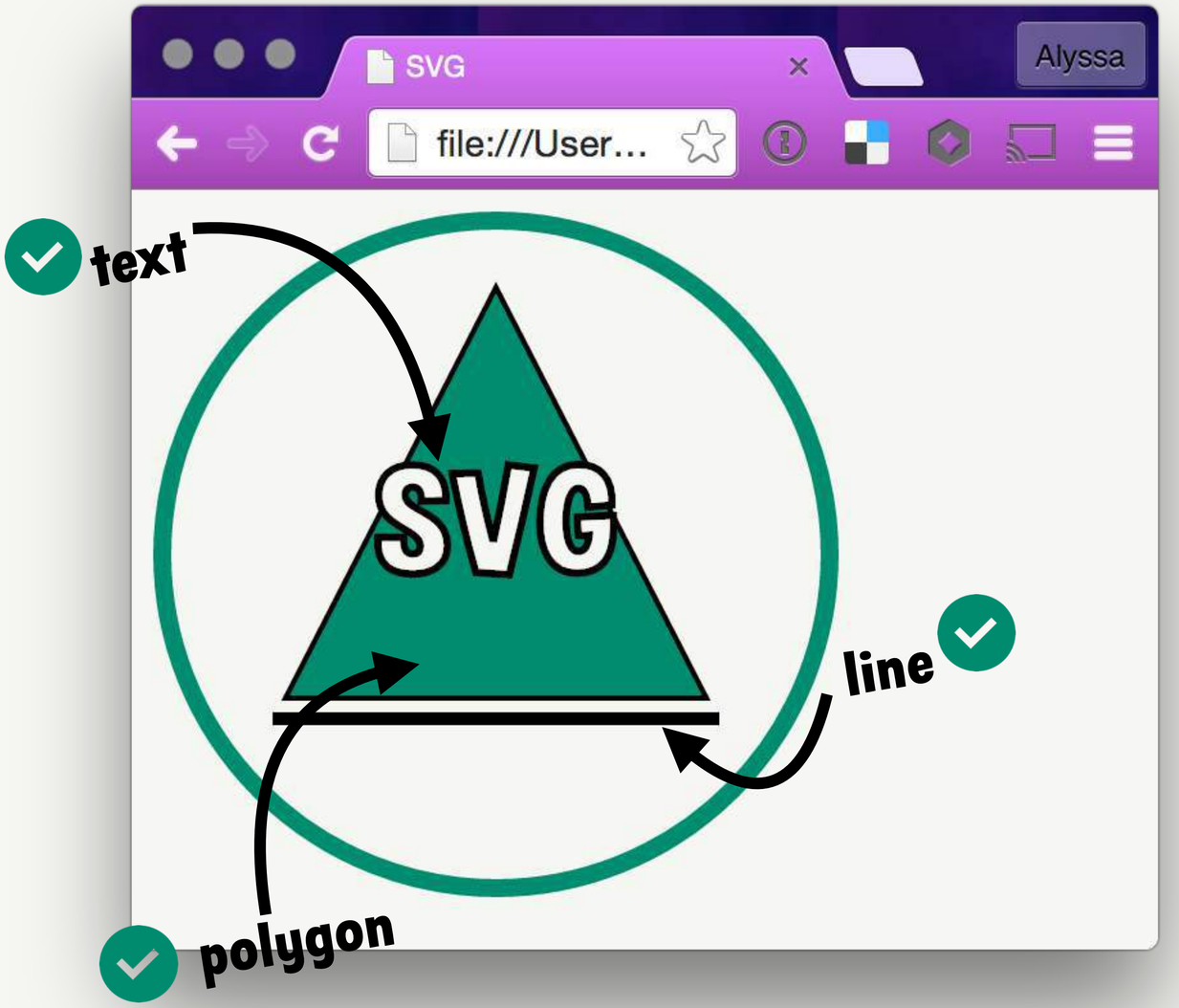
index.html

```
<!DOCTYPE html>
<html>
  <head>...</head>
  <body>
    <svg ...>

      <circle r="130" cx="134" cy="134"/>
      <line x1="47" y1="198" x2="221" y2="198"/>
      <polygon points="52,190 134,30 216,190"/>
      <text x="134" y="142">SVG</text>

    </svg>
  </body>
</html>
```

Order matters



Finished Icon



Level 3

Group de Loop

Section 1 – Groups Anyone?

Where We Left Off

index.html

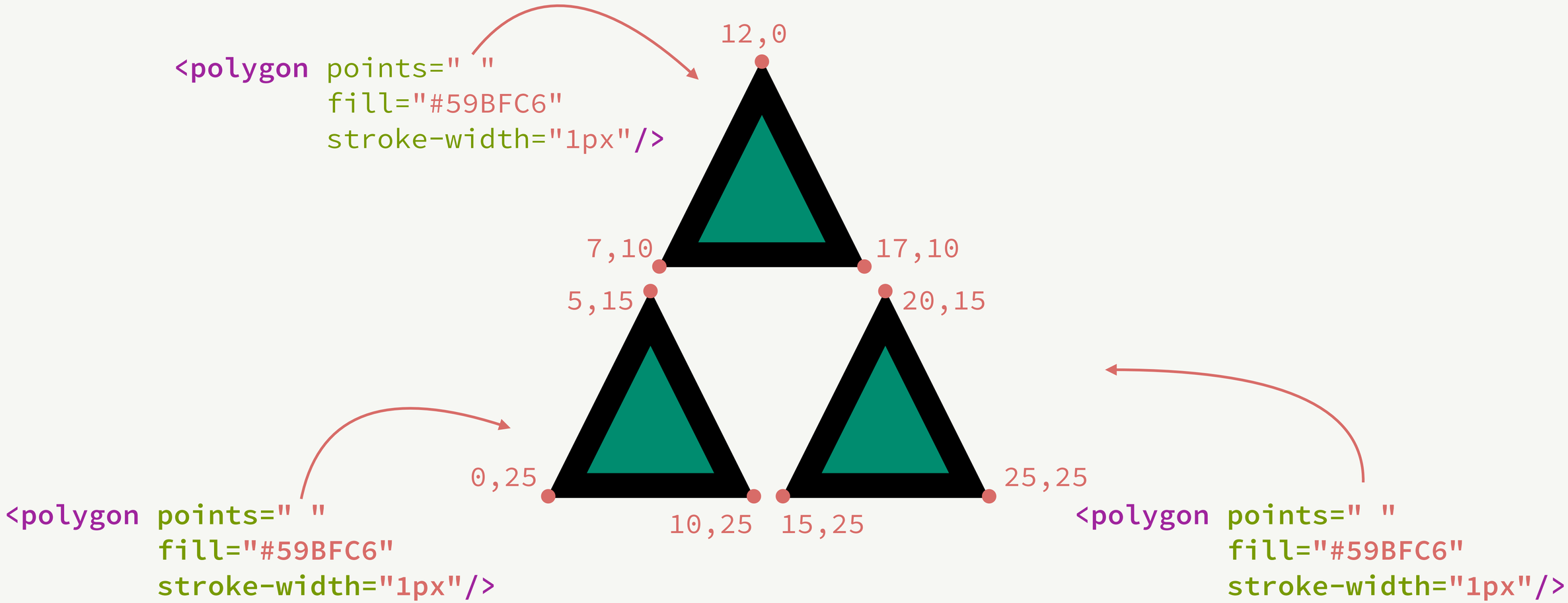
```
<!DOCTYPE html>
<html>
  <head>...</head>
  <body>
    <svg ...>

      <circle r="130" cx="134" cy="134"/>
      <line x1="47" y1="198" x2="221" y2="198"/>
      <polygon points="52,190 134,30 216,190"/>
      <text x="134" y="142">SVG</text>

    </svg>
  </body>
</html>
```

Adding Some Detail to Our Badge

Our badge is still looking a little plain. How could we add some detail (like below) to our badge?

Drawing the Triforce

We would start off by drawing and positioning three of the triangles.

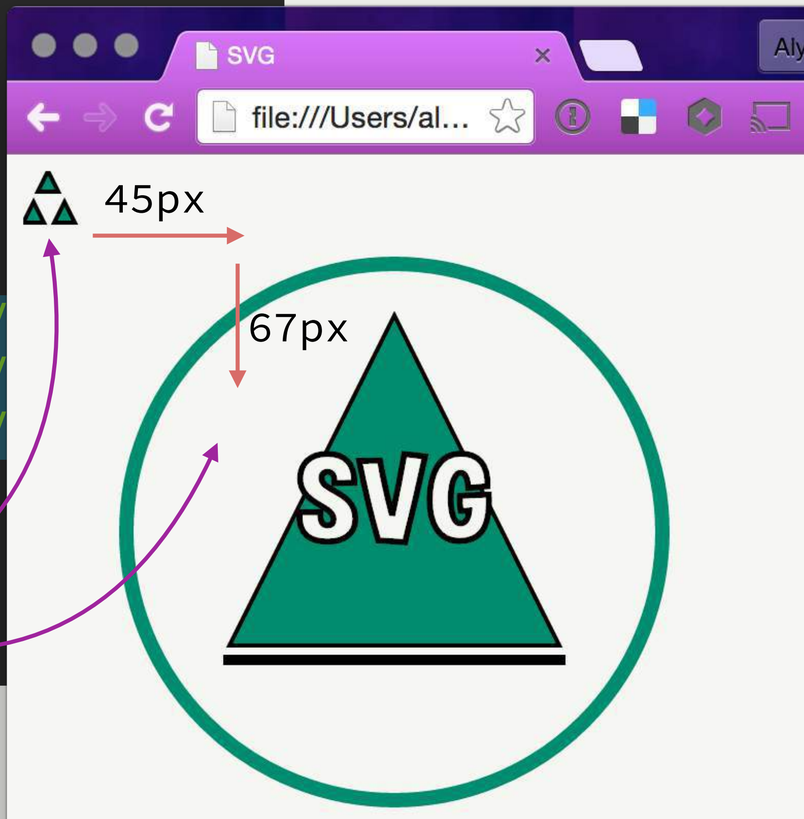


But Our Triangles Are Not on the Badge

index.html

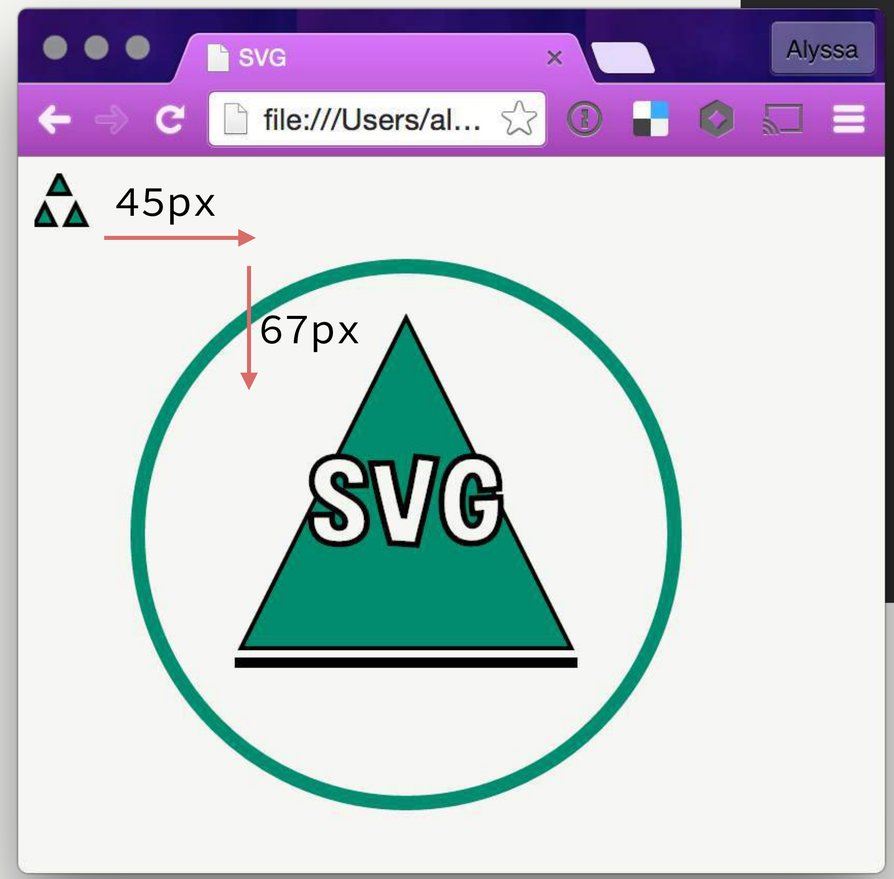
```
<!DOCTYPE html>
<html>
  <head>...</head>
  <body>
    <svg xmlns="..." xmlns:svg="...">

      <circle r="130" cx="134" cy="134"/>
      <line x1="47" y1="198" x2="221" y2="198"/>
      <polygon points="52,190 134,30 216,190"/>
      <text x="134" y="142">SVG</text>
      <polygon points="7,10 12,0 17,10"   fill="#59BFC6" .../
      <polygon points="0,25 5,15 10,25"   fill="#59BFC6" .../
      <polygon points="15,25 20,15 25,25" fill="#59BFC6" .../
    </svg>
  </body>
</html>
```

How are we going to get each triangle into the badge?



Moving Them Into the Badge

We could add to the x and y to fix the position.




```
...
                                +45px +67px +45px +67px +45px +67px
<polygon points="7,10 12,0 17,10"    fill="#59BFC6" .../>

                                +45px +67px +45px +67px +45px +67px
<polygon points="0,25 5,15 10,25"    fill="#59BFC6" .../>

                                +45px +67px +45px +67px +45px +67px
<polygon points="15,25 20,15 25,25" fill="#59BFC6" .../>
```

But there is a better way!

Group to the Rescue!

By grouping these shapes together, we now have the option to transform.

```
<g>
  <polygon points="7,10 12,0 17,10"   fill="#59BFC6" .../>
  <polygon points="0,25 5,15 10,25"   fill="#59BFC6" .../>
  <polygon points="15,25 20,15 25,25" fill="#59BFC6" .../>
</g>
```

This will enable us to move the group as a whole over and down 45,67.

Translating (Moving) the Group

The SVG transform attribute allows us to do multiple things, like translate, rotate, and scale. In order to move the group, we will need to use transform's translate option.

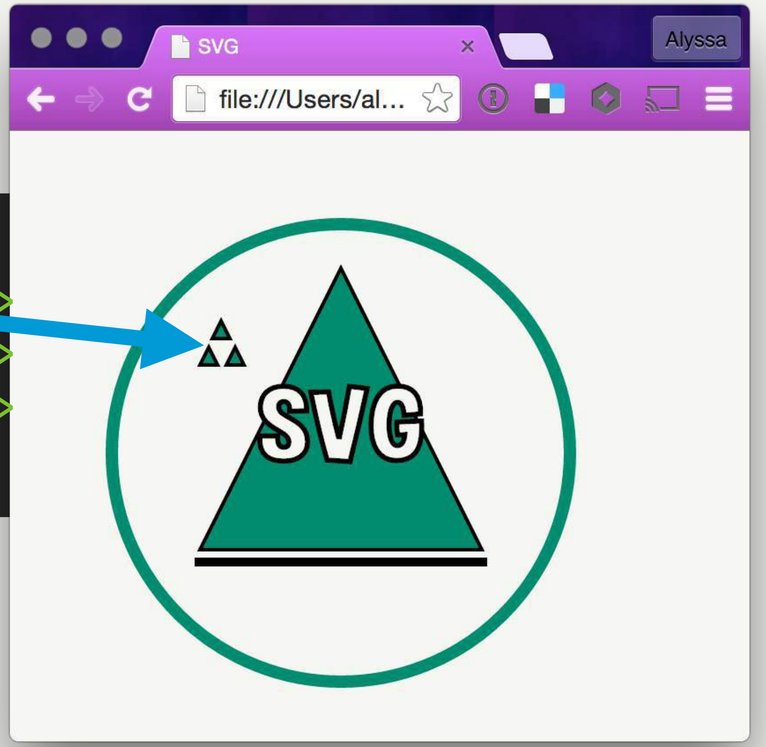
Translate takes two values that will *move* the x and y of the group by the values specified.

Translate means "to move."

move x

move y

```
<g transform="translate(45,67)">
  <polygon points="7,10 12,0 17,10"    fill="#59BFC6" .../>
  <polygon points="0,25 5,15 10,25"    fill="#59BFC6" .../>
  <polygon points="15,25 20,15 25,25" fill="#59BFC6" .../>
</g>
```

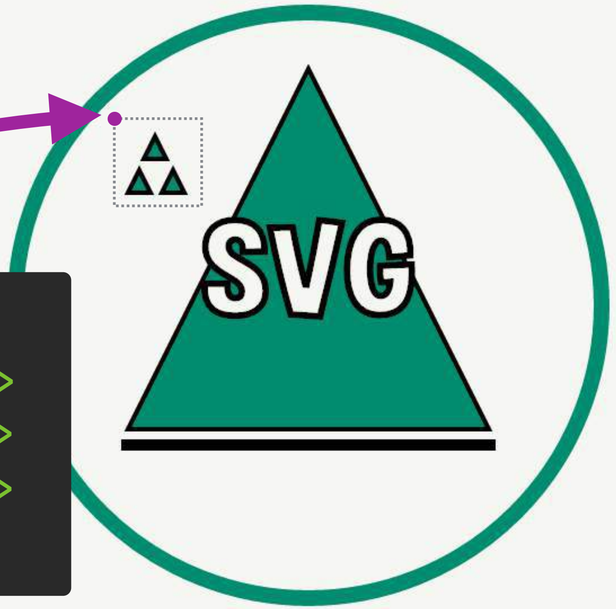


Group Items Have a New Origin!

When you group items, you effectively give them a new 0,0 — the top left of the group!

These numbers are now starting from the groups origin.

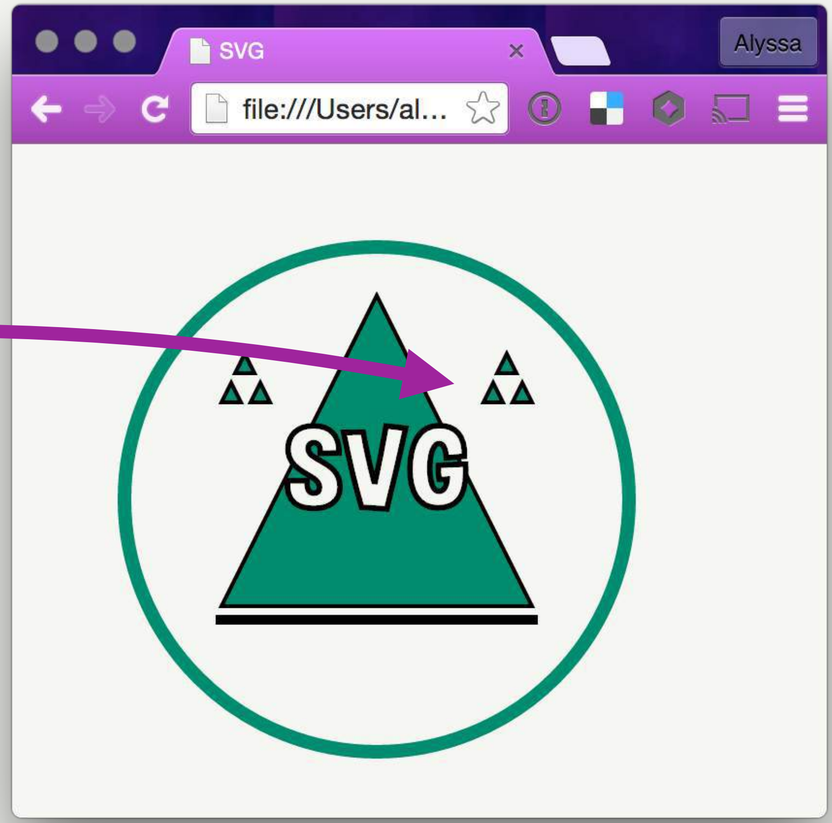
```
<g transform="translate(45,67)">
  <polygon points="7,10 12,0 17,10"   fill="#59BFC6" .../>
  <polygon points="0,25 5,15 10,25"   fill="#59BFC6" .../>
  <polygon points="15,25 20,15 25,25" fill="#59BFC6" .../>
</g>
```



The Second Group

The second group needs to be 198 from the left edge of the viewport and 67 from the top.

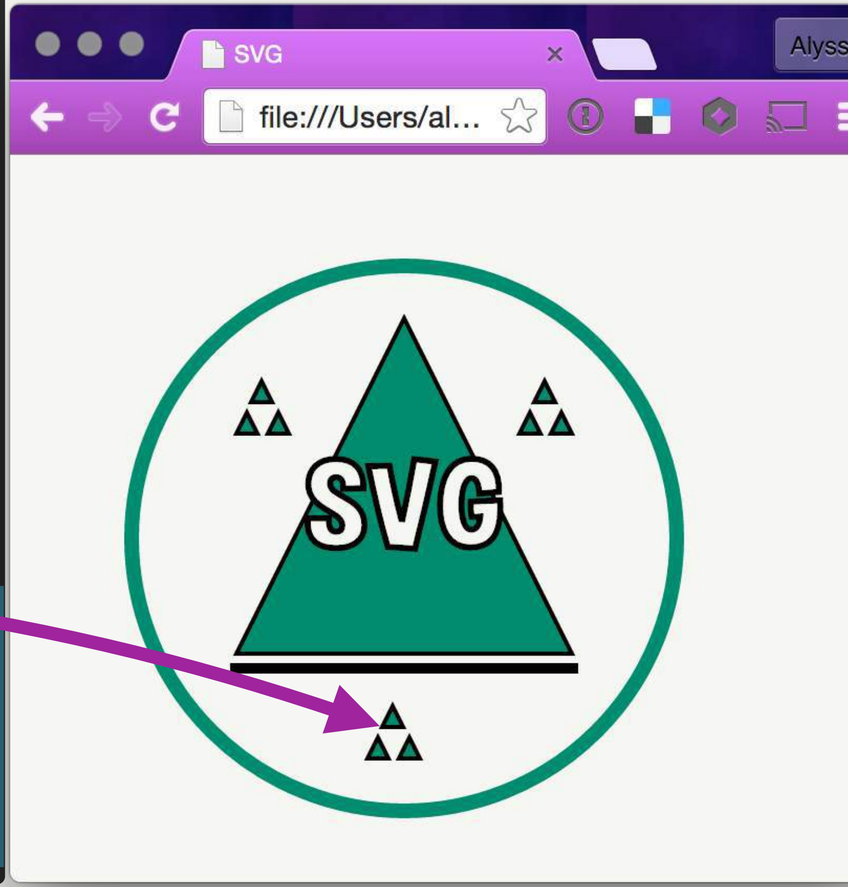
```
<g transform="translate(45,67)">
  <polygon points="7,10 12,0 17,10"   fill="#59BFC6" .../>
  <polygon points="0,25 5,15 10,25"   fill="#59BFC6" .../>
  <polygon points="15,25 20,15 25,25" fill="#59BFC6" .../>
</g>
<g transform="translate(198,67)">
  <polygon points="7,10 12,0 17,10"   fill="#59BFC6" .../>
  <polygon points="0,25 5,15 10,25"   fill="#59BFC6" .../>
  <polygon points="15,25 20,15 25,25" fill="#59BFC6" .../>
</g>
```



The Third Group

The last group needs to be 121.5 from the left edge of the viewport and 211 from the top.

```
<g transform="translate(45,67)">
  <polygon points="7,10 12,0 17,10"   fill="#59BFC6" .../>
  <polygon points="0,25 5,15 10,25"   fill="#59BFC6" .../>
  <polygon points="15,25 20,15 25,25" fill="#59BFC6" .../>
</g>
<g transform="translate(198,67)">
  <polygon points="7,10 12,0 17,10"   fill="#59BFC6" .../>
  <polygon points="0,25 5,15 10,25"   fill="#59BFC6" .../>
  <polygon points="15,25 20,15 25,25" fill="#59BFC6" .../>
</g>
 <g transform="translate(121.5,211)">
   <polygon points="7,10 12,0 17,10"   fill="#59BFC6" .../>
   <polygon points="0,25 5,15 10,25"   fill="#59BFC6" .../>
   <polygon points="15,25 20,15 25,25" fill="#59BFC6" .../>
 </g>
```



SVG coordinates take decimals!

These Styles Are Duplicated

```
<g class="triangle_group" transform="translate(45,67)">
  <polygon points="7,10 12,0 17,10"   fill="#59BFC6" stroke-width="1"/>
  <polygon points="0,25 5,15 10,25"   fill="#59BFC6" stroke-width="1"/>
  <polygon points="15,25 20,15 25,25" fill="#59BFC6" stroke-width="1"/>
</g>

<g class="triangle_group" transform="translate(198,67)">
  <polygon points="7,10 12,0 17,10"   fill="#59BFC6" stroke-width="1"/>
  <polygon points="0,25 5,15 10,25"   fill="#59BFC6" stroke-width="1"/>
  <polygon points="15,25 20,15 25,25" fill="#59BFC6" stroke-width="1"/>
</g>

<g class="triangle_group" transform="translate(121.5,211)">
  <polygon points="7,10 12,0 17,10"   fill="#59BFC6" stroke-width="1"/>
  <polygon points="0,25 5,15 10,25"   fill="#59BFC6" stroke-width="1"/>
  <polygon points="15,25 20,15 25,25" fill="#59BFC6" stroke-width="1"/>
</g>
```

We will give each group a class and get rid of these styles from our HTML.

Adding the Style to Group Inside CSS

Groups let us add styles so that they trickle down to the inner elements.

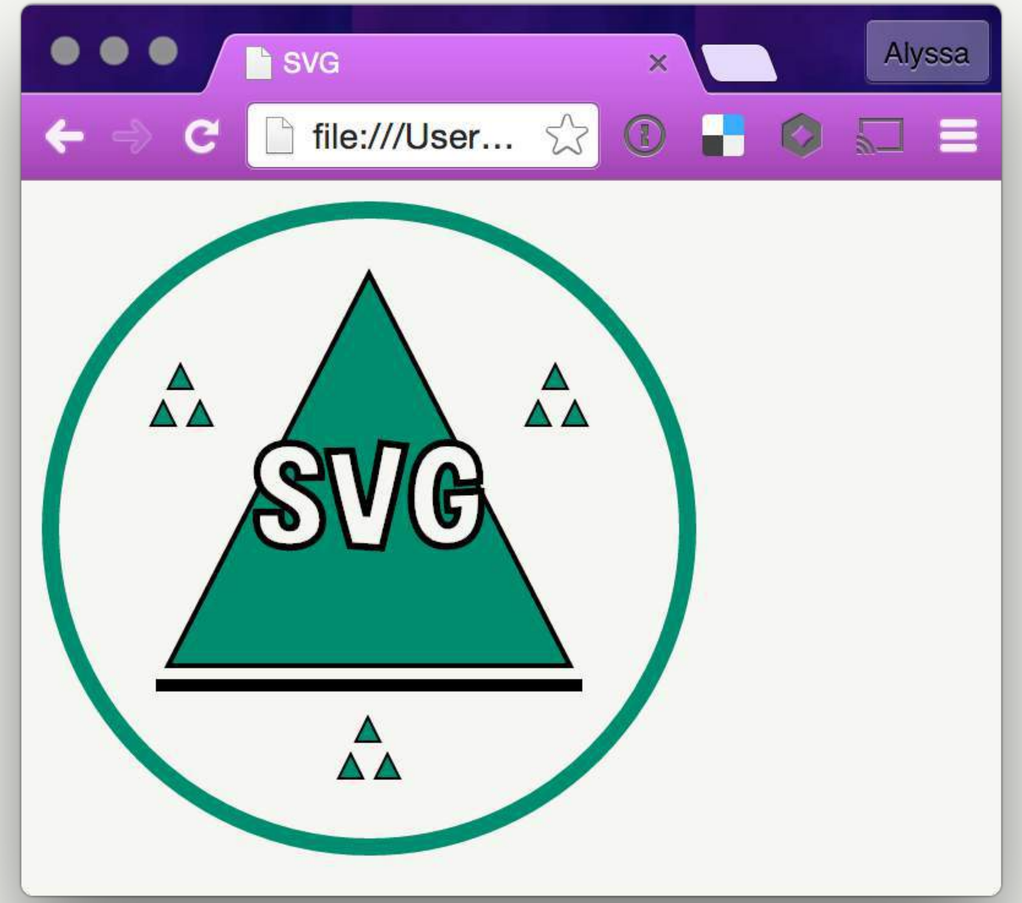
```svg
<g class="triangle_group" transform="translate(45,67)">
  <polygon points="7,10 12,0 17,10" />
  <polygon points="0,25 5,15 10,25" />
  <polygon points="15,25 20,15 25,25" />
</g>

<g class="triangle_group" transform="translate(198,67)">
  <polygon points="7,10 12,0 17,10" />
  <polygon points="0,25 5,15 10,25" />
  <polygon points="15,25 20,15 25,25" />
</g>

<g class="triangle_group" transform="translate(121.
  <polygon points="7,10 12,0 17,10" />
  <polygon points="0,25 5,15 10,25" />
  <polygon points="15,25 20,15 25,25" />
</g>
```

style.css

```css
polygon {
  fill: #008B6F;
  stroke: #000;
  stroke-width: 2px;
}

.triangle_group polygon {
  stroke-width: 1px;
}
```

The Badge Is Looking Great!

Groups allowed us to translate the entire group to its desired position.




```
<g class="triangle_group"
   transform="translate(45,67)">
  ...
</g>

<g class="triangle_group"
   transform="translate(198,67)">
  ...
</g>

<g class="triangle_group"
   transform="translate(121.5,211)">
  ...
</g>
```

We also used groups to add semantic classes and moved our styles over to a stylesheet!

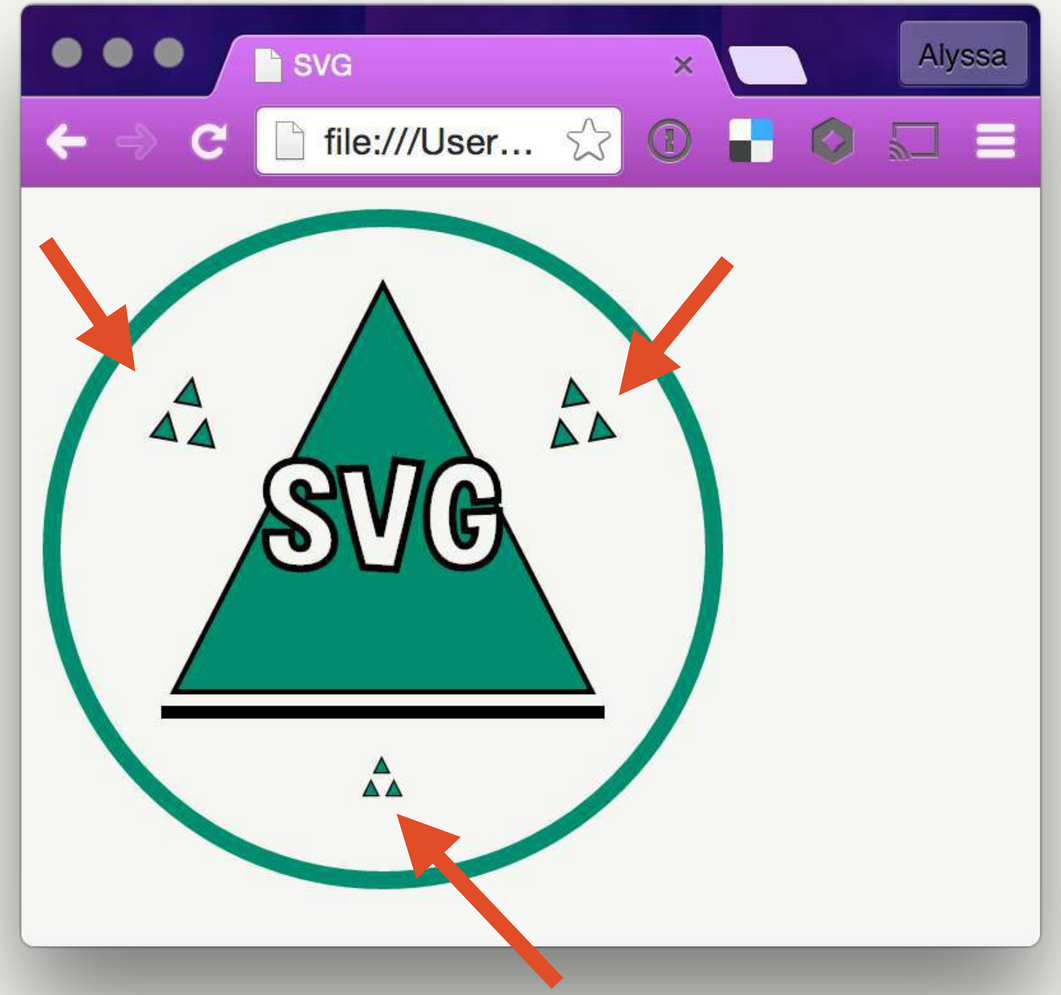
Level 3

Group de Loop

Section 2 – Transform to the Rescue

Other Ways to Use Transform

You can also rotate and scale with transform!



Rotating With Transform

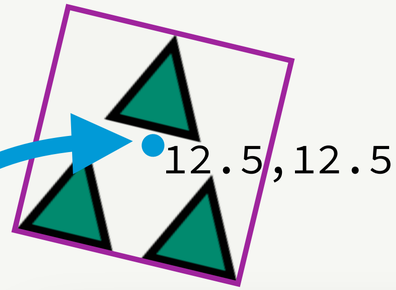
To rotate this top left group of polygons, we use transform's rotate property.

Rotate takes three values: the degrees to rotate and the x,y to rotate around.

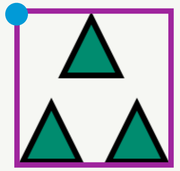
index.html

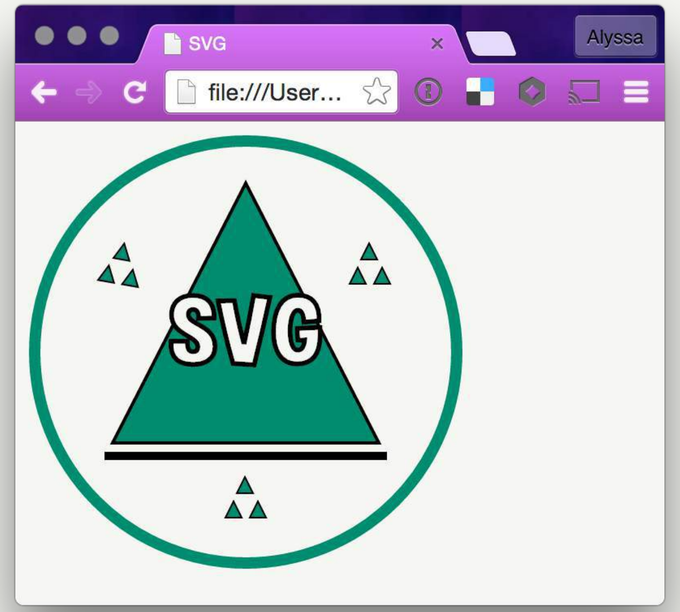
```
<g class="first triangle_group"
   transform="translate(45, 67) rotate(10 12.5 12.5) ">
  <polygon points="7,10 12,0 17,10"/>
  <polygon points="0,25 5,15 10,25"/>
  <polygon points="15,25 20,15 25,25"/>
</g>
```

degrees x origin y origin



If you just specify the degrees, the rotation will default to rotate around 0,0.





Rotating the Second Triangle Group

We rotate around the center 12.5,12.5 again for the second group, but this time we will rotate it -10 degrees.

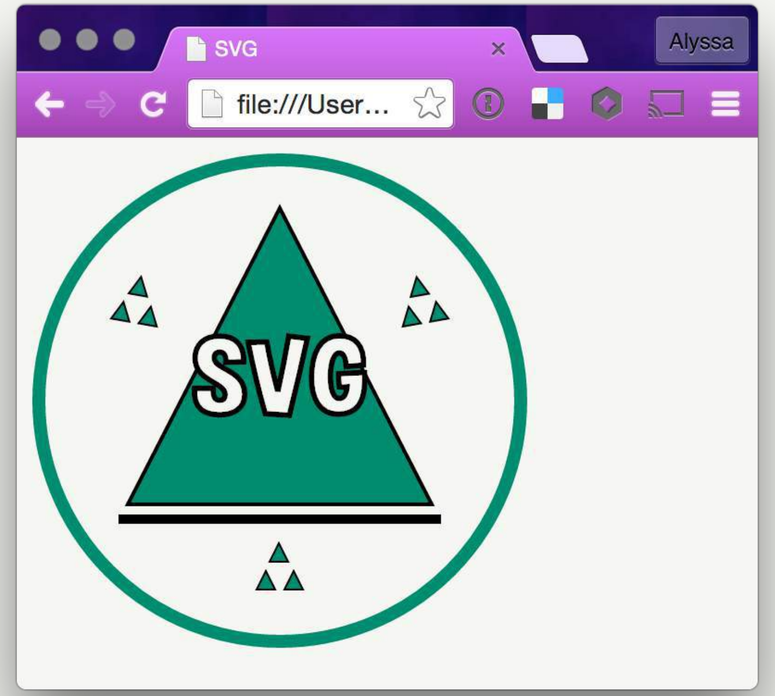
index.html

```
<g class="second triangle_group"
   transform="translate(198, 67) rotate(-10 12.5 12.5)">
  <polygon points="7,10 12,0 17,10"/>
  <polygon points="0,25 5,15 10,25"/>
  <polygon points="15,25 20,15 25,25"/>
</g>
```

degrees

x origin

y origin



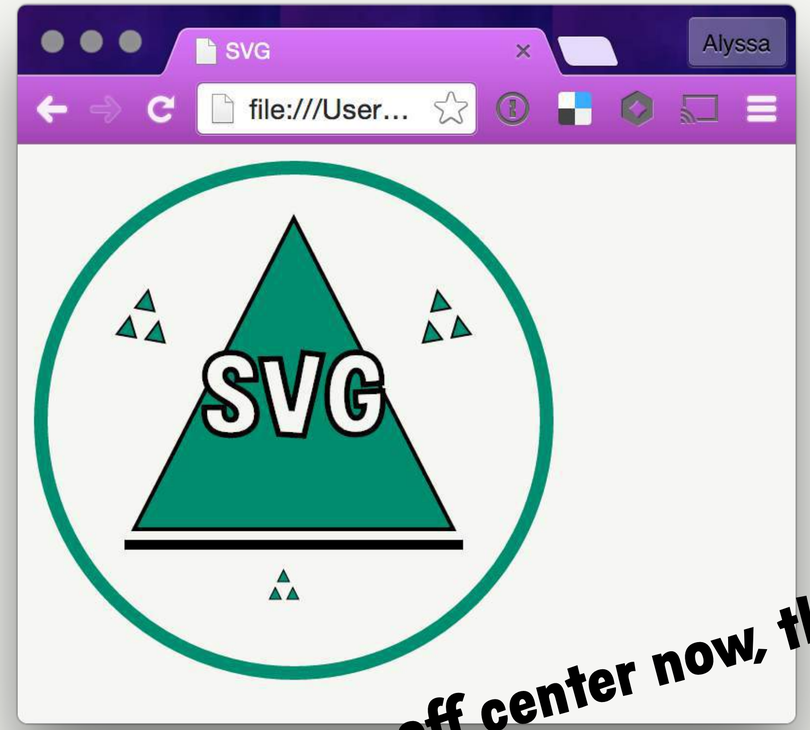
Static rotations like this will stay between 360 to -360.

Scaling the Third Triangle Group

Next, we can shrink this bottom group using transform's scale property. Scale takes a value to multiply the size by. So scale(1) is normal; scale(2) is twice the size.

index.html

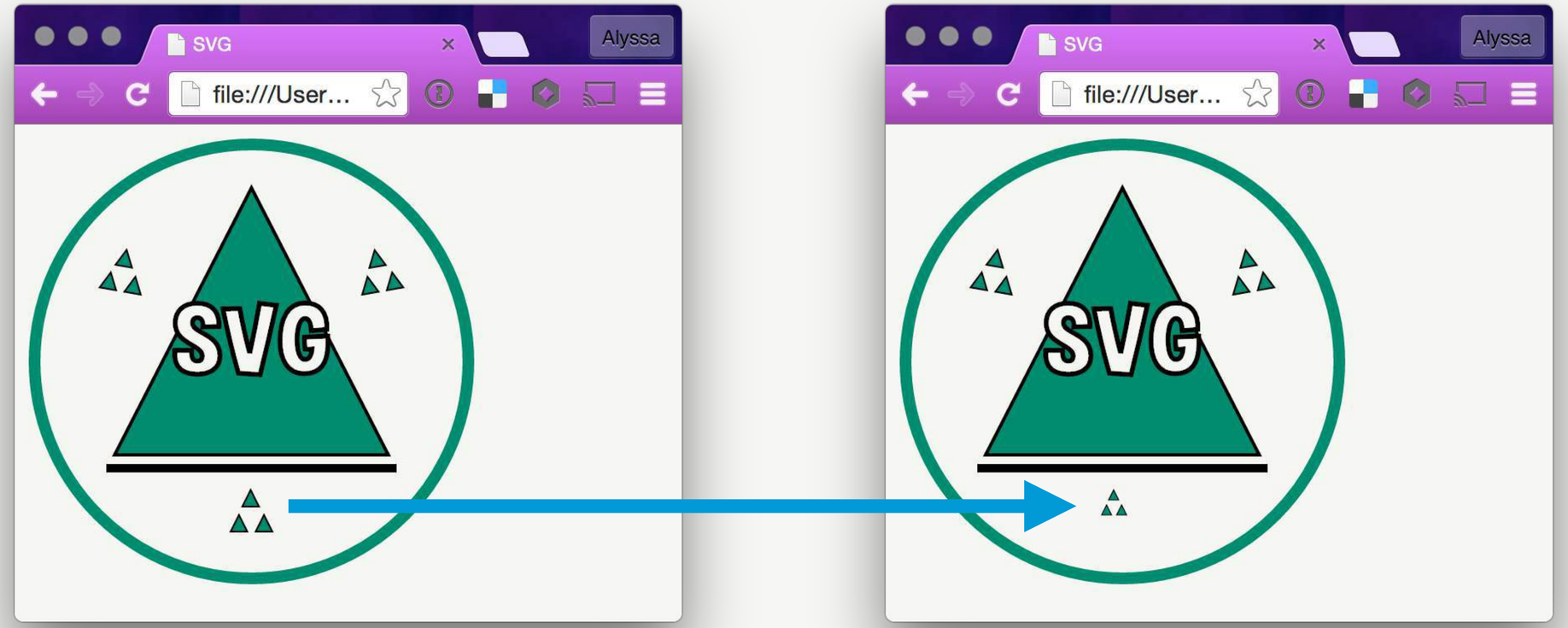
```
<g class="third triangle_group"
   transform="translate(121.5,211) scale(0.6) ">
  <polygon points="7,10 12,0 17,10"/>
  <polygon points="0,25 5,15 10,25"/>
  <polygon points="15,25 20,15 25,25"/>
</g>
```

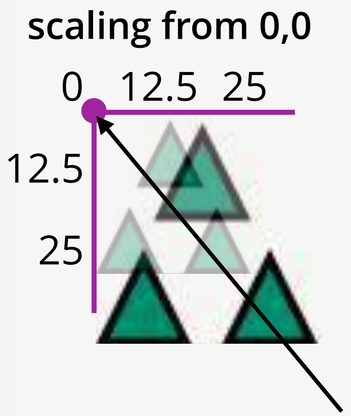


Looks off center now, though...

Scale's Top Left Origin

The group is scaling from 0,0.



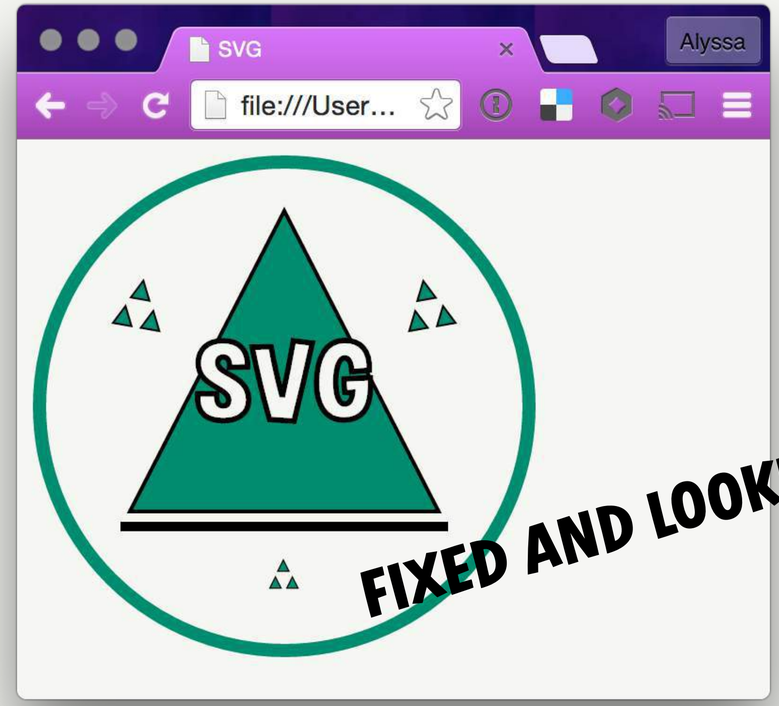


Adjusting Coordinates Because of Scale

Chaining another translation to move the group down and right will fix this problem.

Every time will be different. We need to move ours 8,8.

```
<g class="third triangle_group" transform="translate(121.5,211)
                                           scale(0.6)
                                           translate(8,8)">
```



We now know how to not only translate, but rotate, scale, and chain transforms!

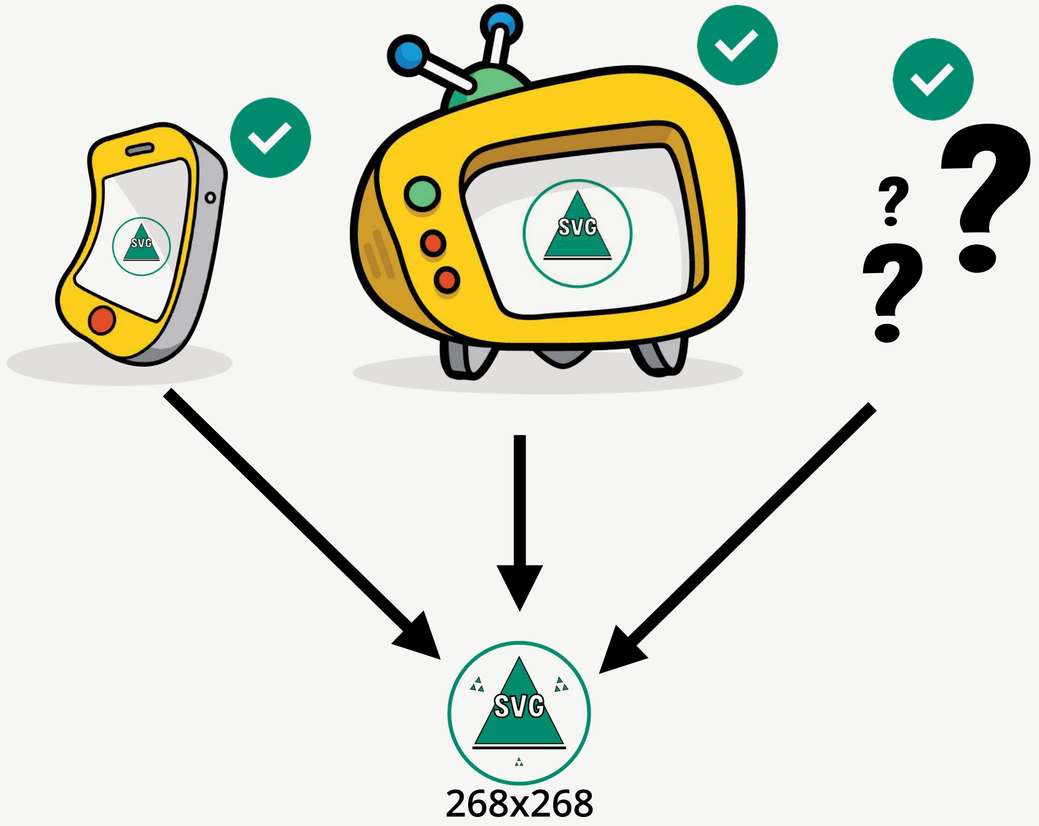
Level 3

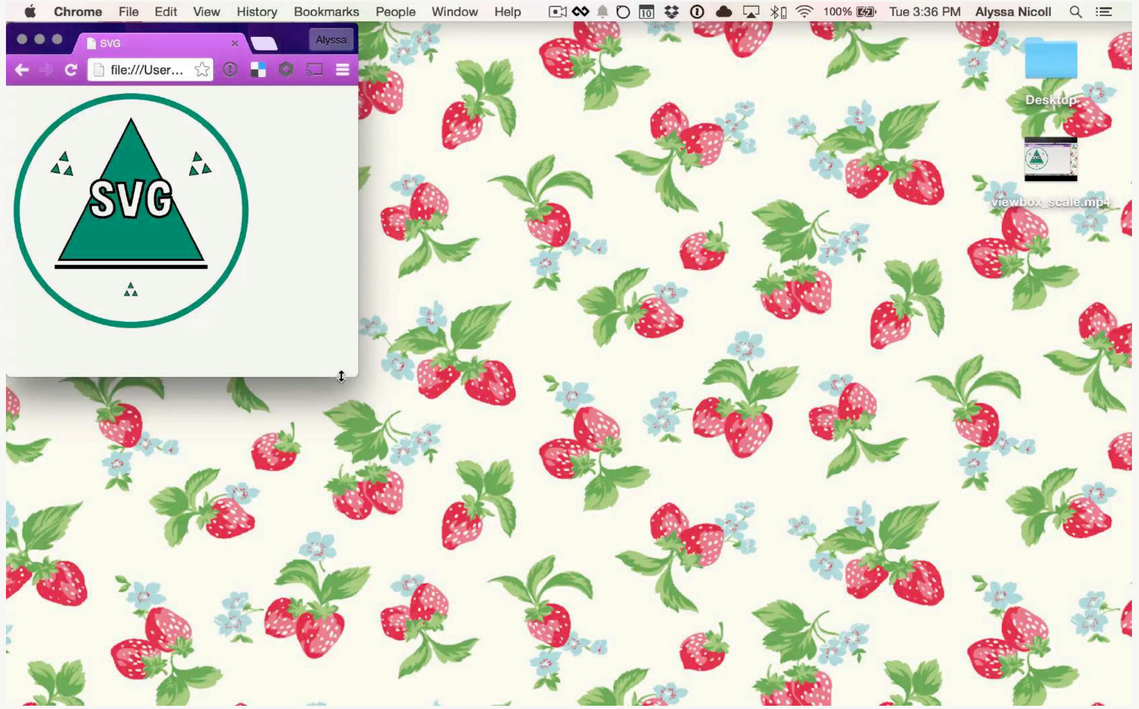
Group de Loop

Section 3 – Responsively

Truly Scalable Graphics?

We are looking good, but our SVG is still not very responsive based on the screen size.





100%

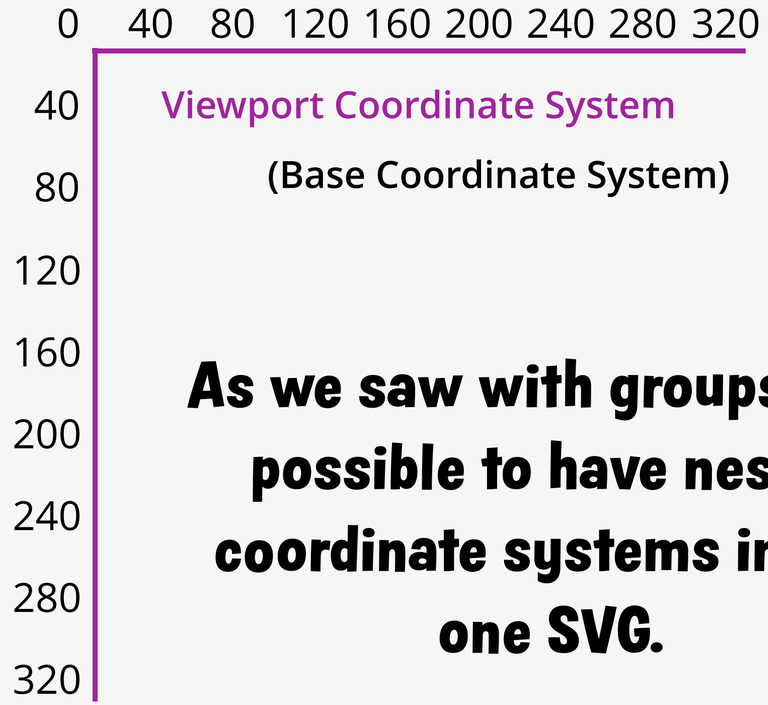
Wouldn’t it be better if we could set our SVG’s width to a percentage of the screen width?!

Viewport Is Our Base Coordinate System

SVG height and width is called our **viewport**.

index.html

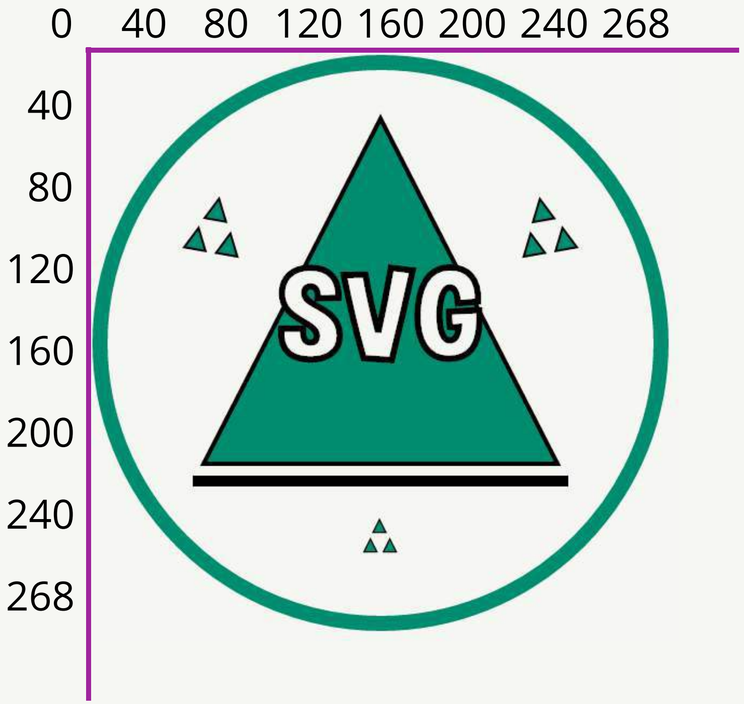
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>SVG</title>
  </head>
  <body>
     <svg  height="268"
           width="268"
           version="1.1"
           xmlns="http://www.w3.org/2000/svg">
        ...
     </svg>
   </body>
 </html>
```



As we saw with groups, it is possible to have nested coordinate systems inside one SVG.

Using a ViewBox

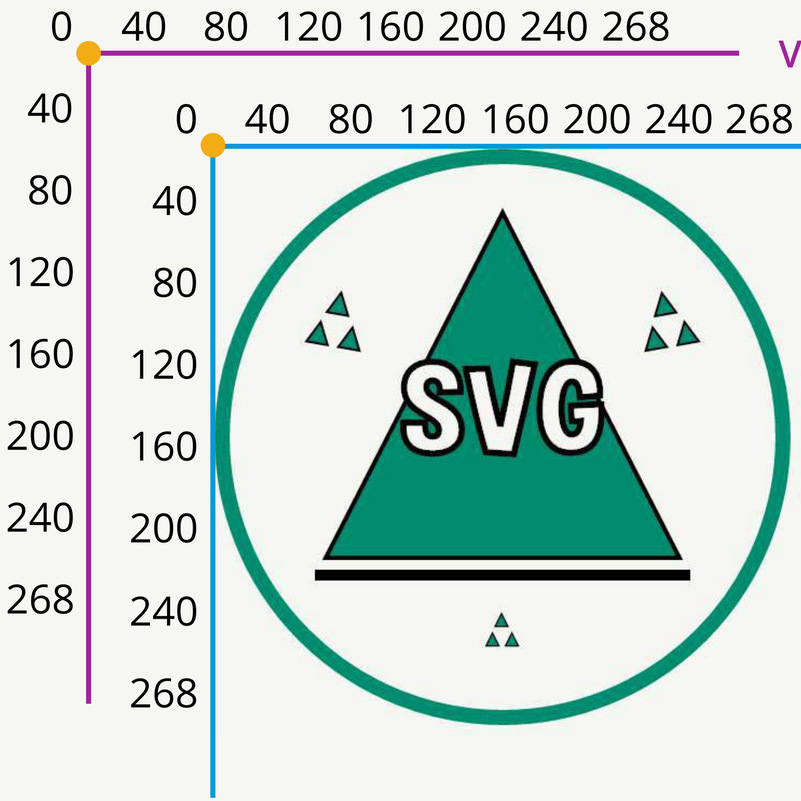
1. Copy our viewport values into a nested coordinate system called the **viewBox**.



Viewport Coordinate System height: 268 width: 268

Using a ViewBox

1. Copy our viewport values into a nested coordinate system called the **viewBox**.



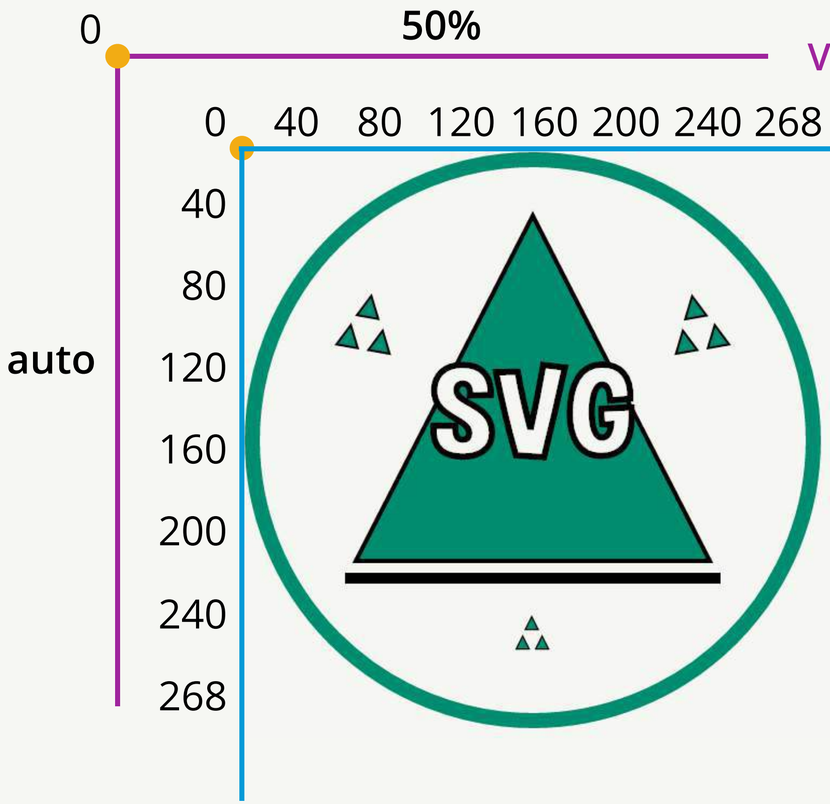
Viewport Coordinate System height: 268 width: 268

ViewBox Coordinate System height: 268 width: 268

These two coordinate systems will lay right on top of each other.

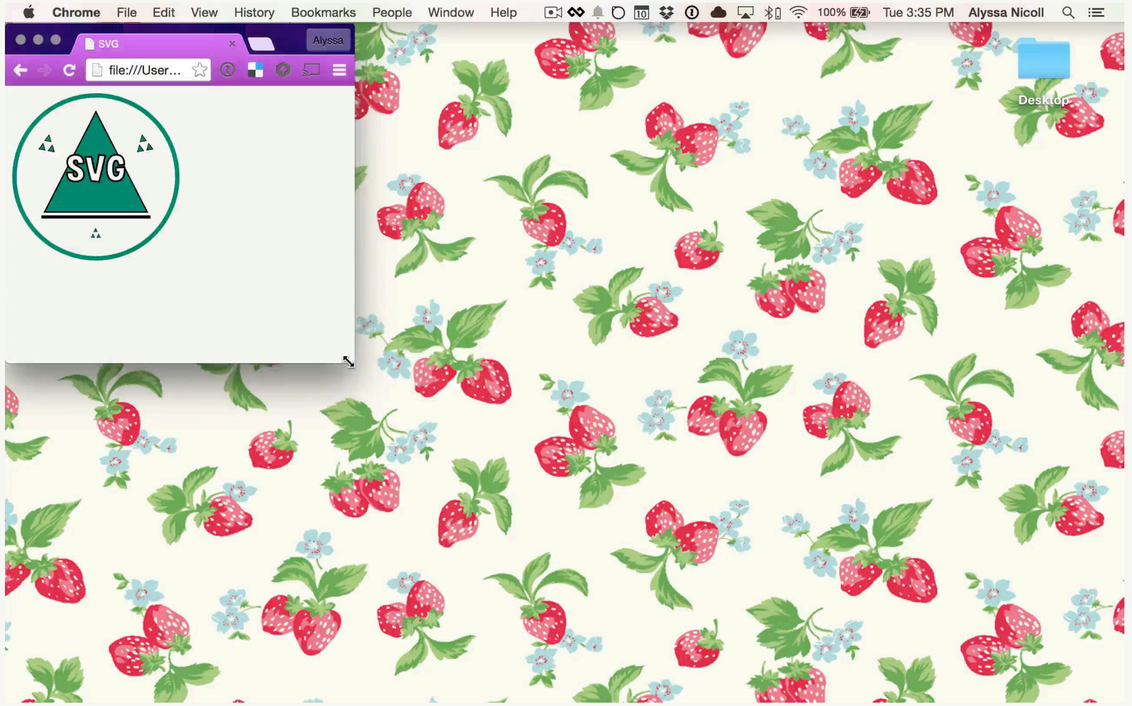
Using Responsive Values

1. Copy our viewport values into a nested coordinate system called the **viewBox**.
2. Give our viewport responsive values for height and width.



Viewport Coordinate System height: auto width: 50%

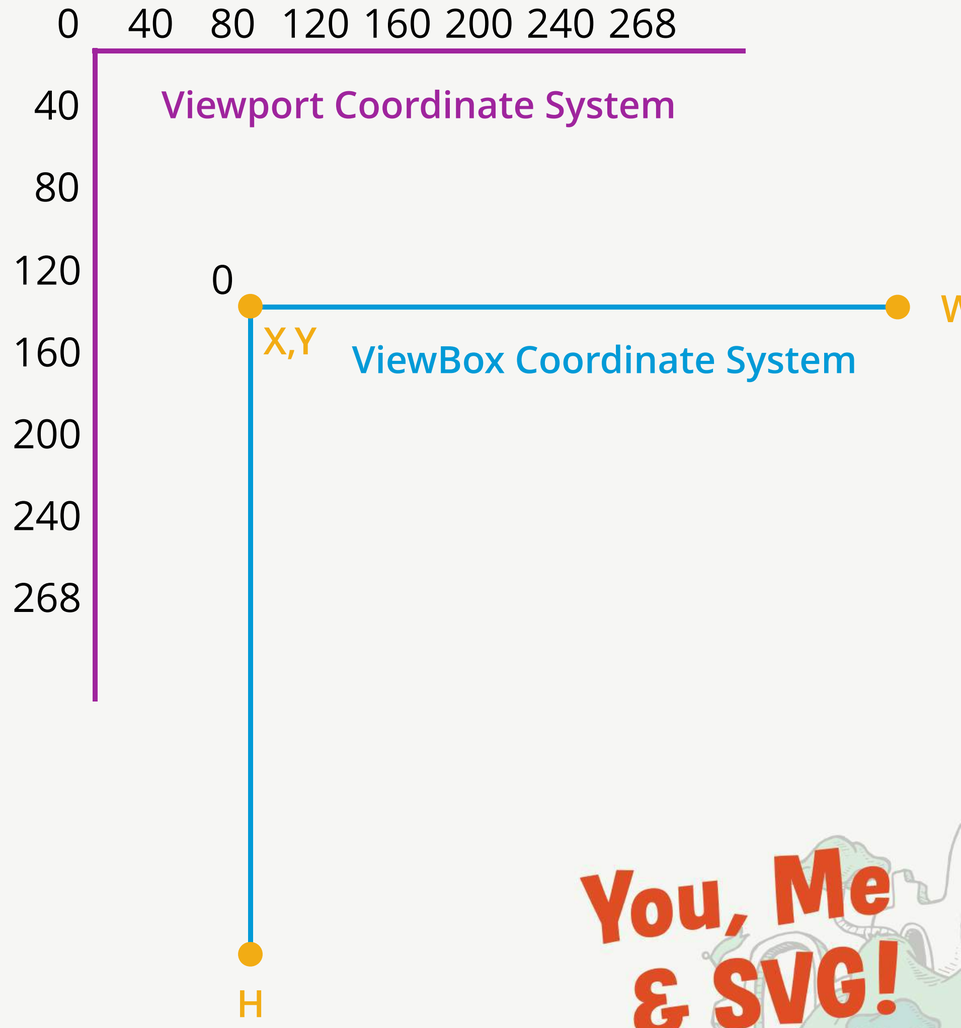
ViewBox Coordinate System height: 268 width: 268



Moving Viewport Values to ViewBox

index.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>SVG</title>
  </head>
  <body>
     <svg  height="268"
           width="268"
           version="1.1"
           xmlns="http://www.w3.org/2000/svg">
           viewBox="            ">
        ...
     </svg>
   </body>
 </html>
```



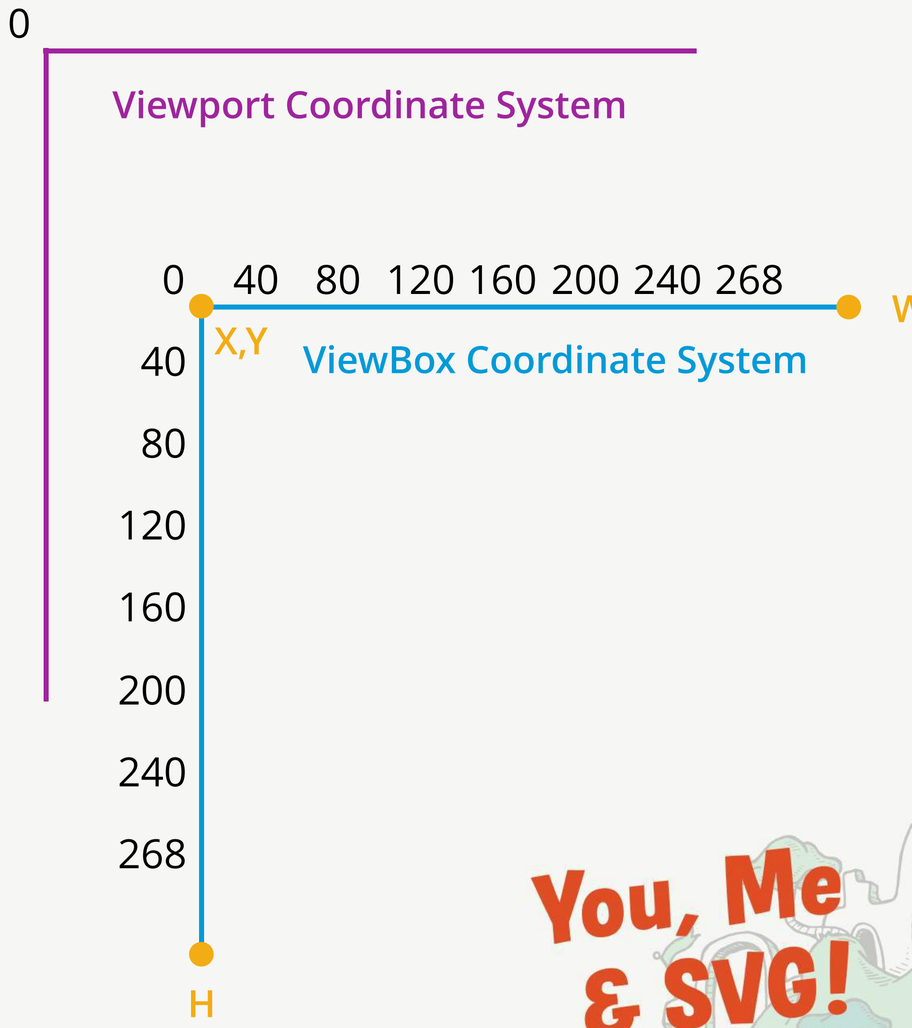
Moving Viewport Values to ViewBox

index.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>SVG</title>
  </head>
  <body>
     <svg  version="1.1"
           xmlns="http://www.w3.org/2000/svg">
           viewBox="    268 268 ">
        ...
    </svg>
   </body>
 </html>
```

Width and Height

We will set the static size of our asset(268x268) on the viewBox.



Same Origin for Both Coordinate Systems

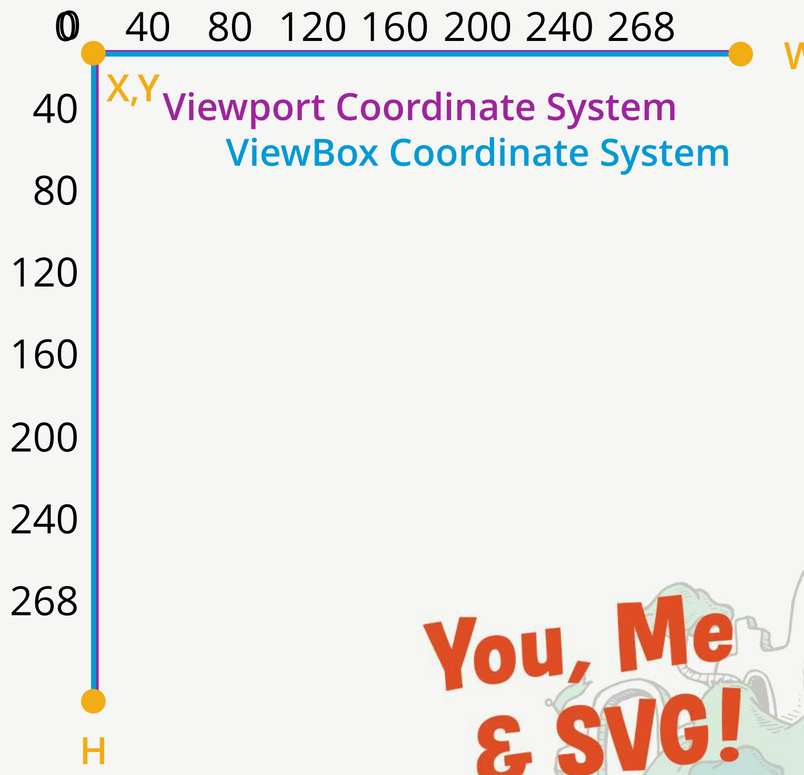
index.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>SVG</title>
  </head>
  <body>
     <svg version="1.1"
          xmlns="http://www.w3.org/2000/svg">
          viewBox="0 0 268 268">
        ...
    </svg>
  </body>
</html>
```

ViewBox Origin X,Y

Width & Height

For this example, our coordinate systems will have the same origin: 0,0.



Giving Viewport Responsive Values

Now all we need to do is set our viewport height and width to responsive sizes. You need to do this in the CSS:

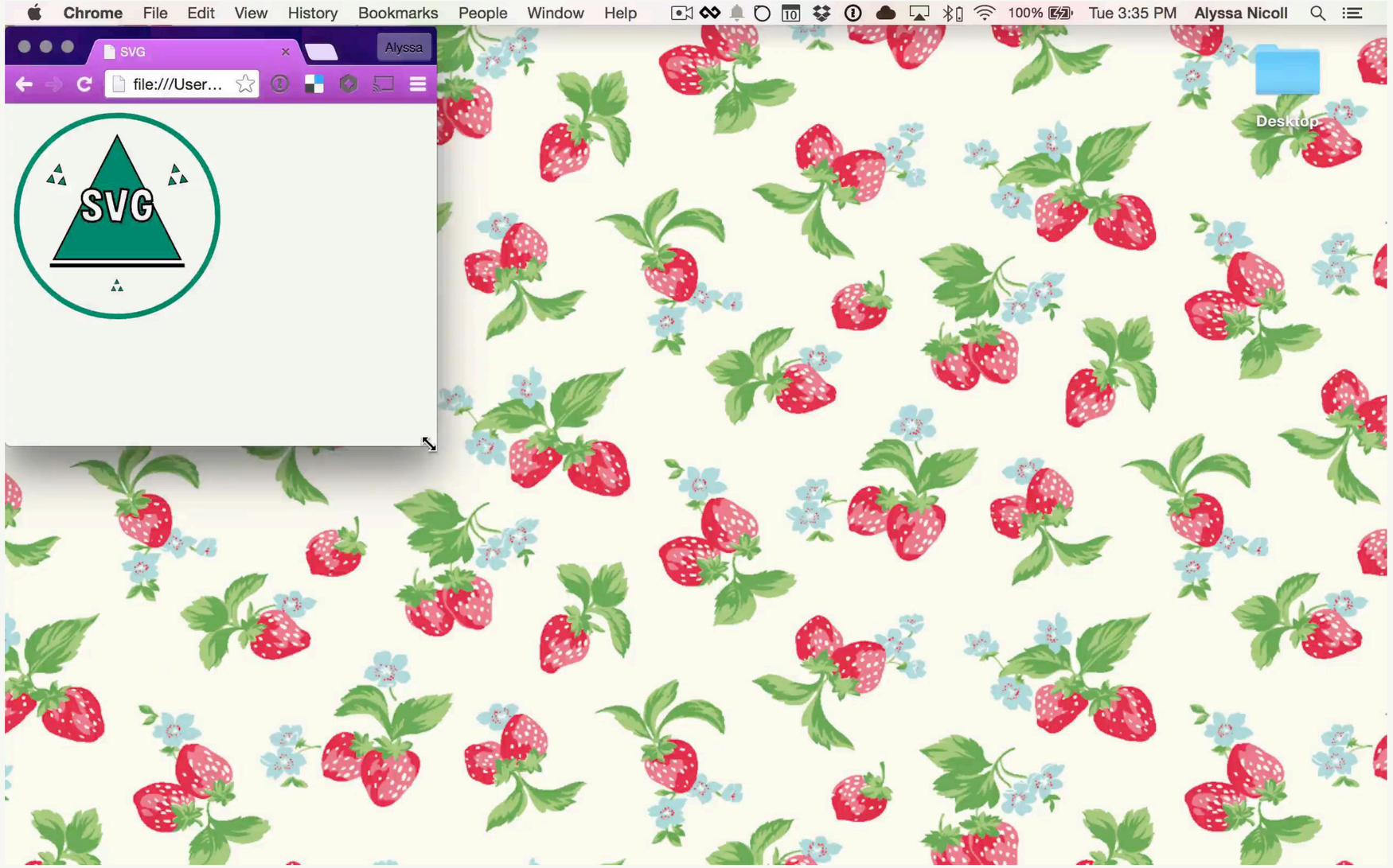
index.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>SVG</title>
  </head>
  <body>
     <svg  version="1.1"
           xmlns="http://www.w3.org/2000/svg">
           viewBox="0 0 268 268">
        ...
     </svg>
   </body>
 </html>
```

style.css

```
svg {
  height: auto;
  width: 50%;
}
```

Responsive Scalable Graphics — Wow!



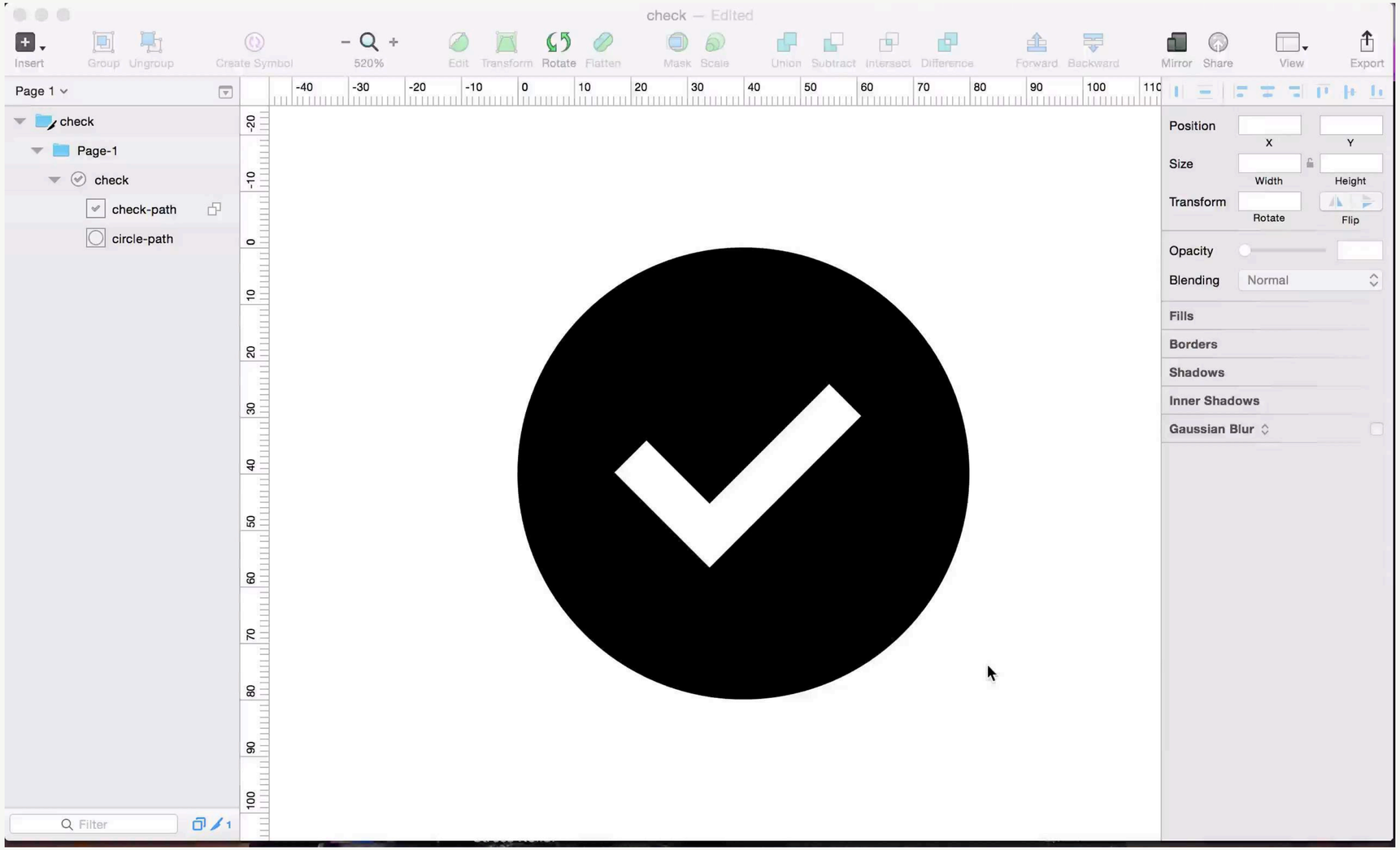
Level 4

SVG Encore!

Section 1 – Paths Are Fun

Exporting an SVG From a Drawing Tool

Here’s a check we drew in Sketch. Let’s export it as an SVG!



Looking at an Exported SVG

Whether exporting from a program or found online somewhere, SVG assets can have some funky code...

check.svg

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<svg width="80px" height="80px" viewBox="0 0 80 80" version="1.1" xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:sketch="http://www.bohemiancoding.com/sketch/ns">
    <!-- Generator: Sketch 3.3.3 (12072) - http://www.bohemiancoding.com/sketch -->
    <title>check</title>
    <desc>Created with Sketch.</desc>
    <defs></defs>
    <g id="Page-1" stroke="none" stroke-width="1" fill="none" fill-rule="evenodd" sketch:type="MSPage">
        <path d="M40,0 C17.909,0 0,17.909 0,40 C0,62.091 17.909,80 40,80 C62.091,80 80,62.091 80,40
C80,17.909 62.091,0 40,0 L40,0 Z M34,56.657 L17.172,39.829 L22.828,34.171 L34,45.343 L55.172,24.171
L60.828,29.829 L34,56.657 L34,56.657 Z" id="check" fill="#000000" sketch:type="MSShapeGroup"></path>
    </g>
</svg>
```

Scary long path

check.svg looks like this: 

What is going on here?

Understanding Paths

Paths are very powerful for creating complicated SVGs, but they're better suited for creation by software.

```
<path d="..."></path>
```

Draws an object by following path instructions you send it

We'll show you the basics, but you typically wouldn't write this by hand.

Comparing Path vs. Polygon

What would a triangle from our badge look like as a path?

As a polygon:

```
<polygon points="7,10 12,0 17,10"/>
```

Just another way to draw a shape!

As a path:

```
<path d="M7,10 L7,10 L12,0 L17,10 Z"></path>
```

start path

draw line between points

close path

MLZ are all path commands that will draw straight lines.

Cubic Bézier Path

You can use `C` in your path to denote a cubic Bézier curve.

Cubic Bézier

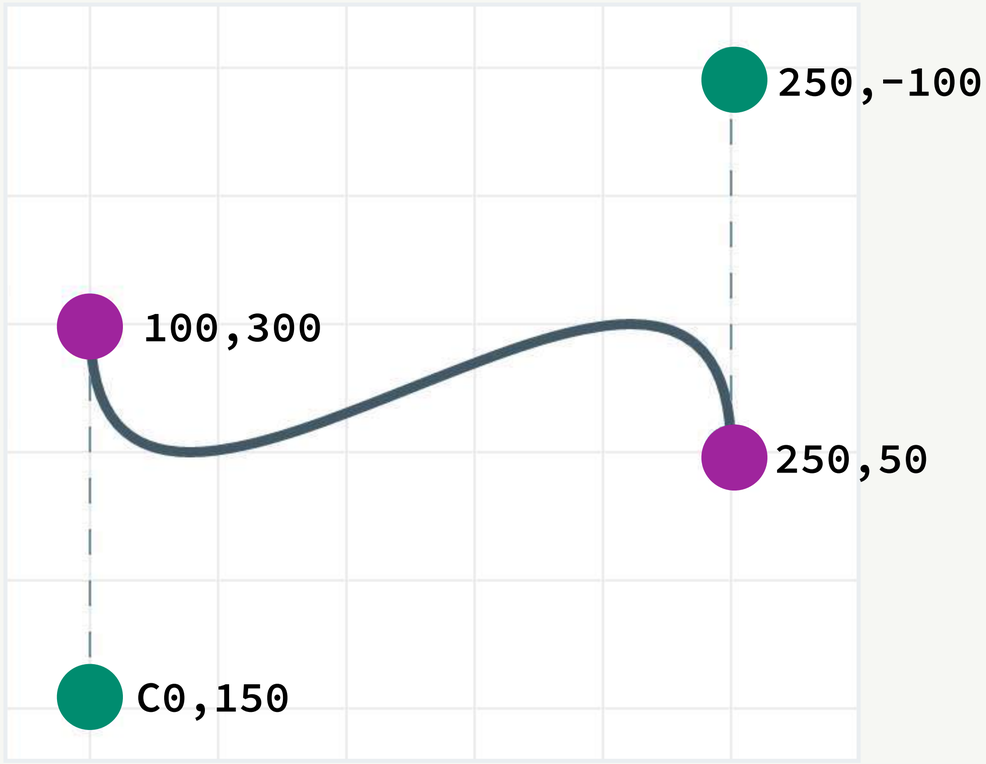
```
<path d="M100,300 C0,150 250,-100 250,50"/>
```

x,y first point

x,y first handle

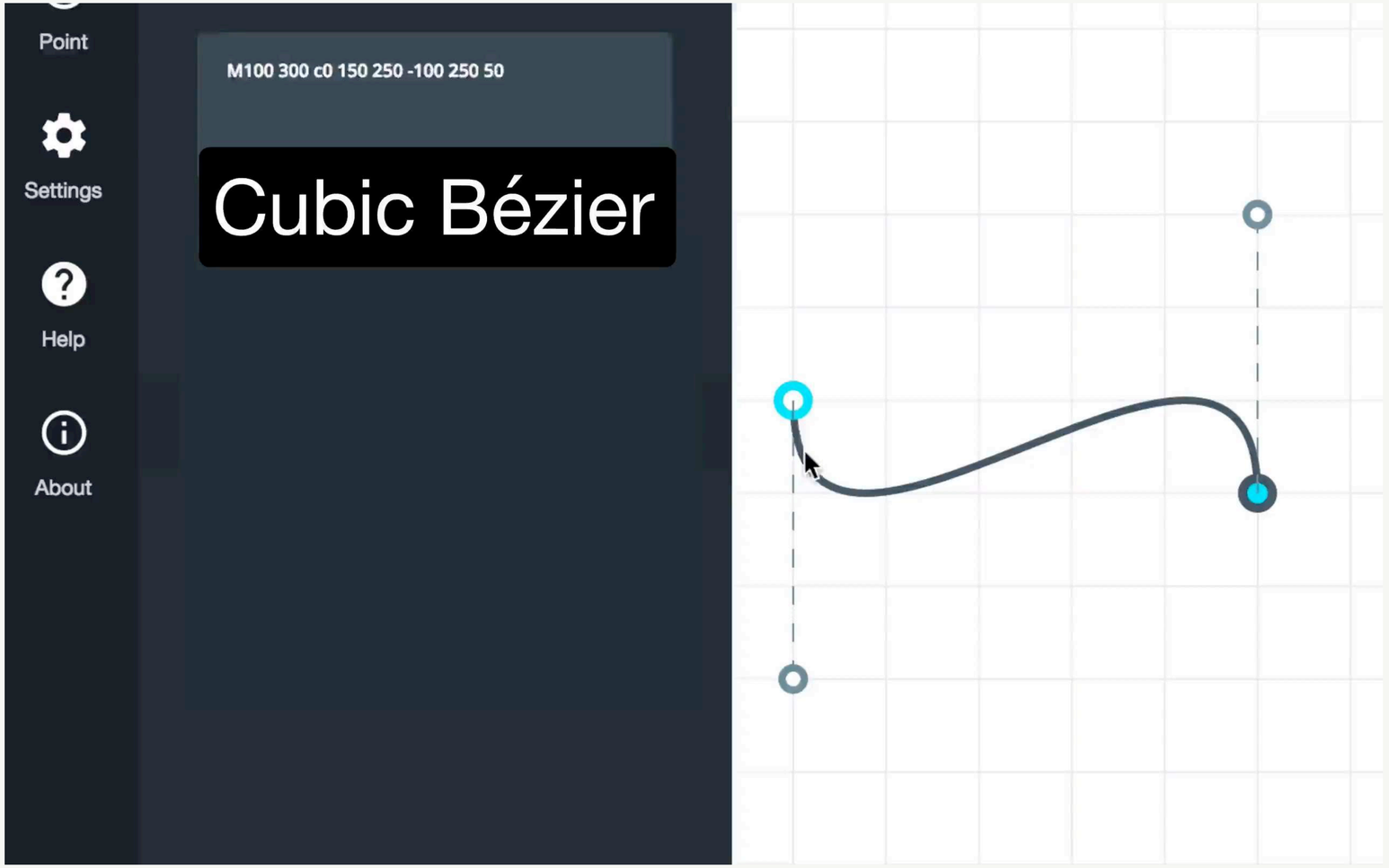
x,y second handle

x,y second point



Great article about paths: ***http://www.sitepoint.com/closer-look-svg-path-data/***

Playing Around With Cubic Bézier



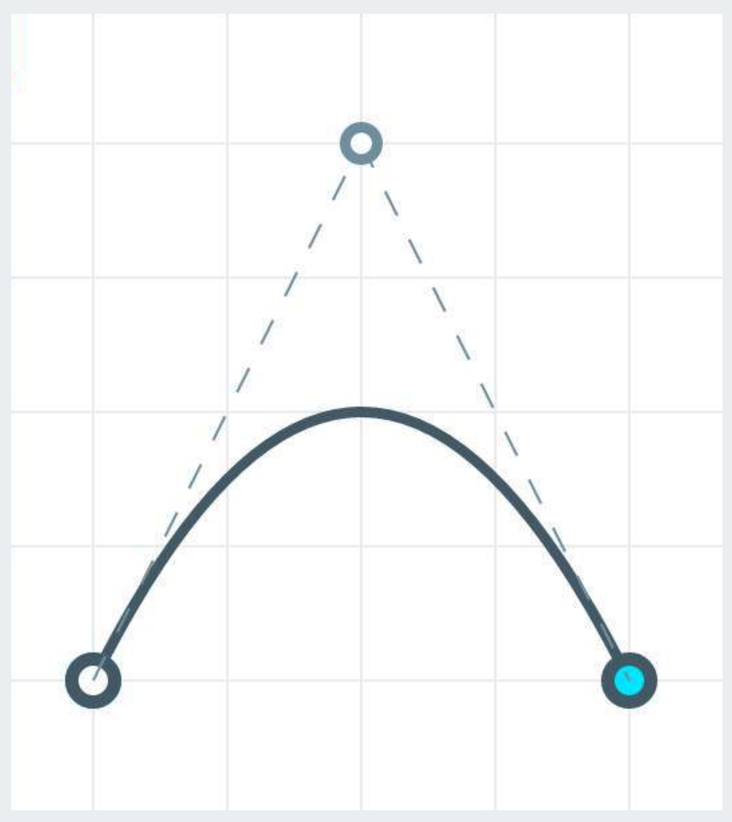
Play around with paths here: ***http://anthonydugois.com/svg-path-builder/***

Quadratic Bézier Curve

Similarly, you can use `Q` to denote a quadratic Bézier curve.

Quadratic Bézier

```
<path d="M100 200 Q200 0 300 200"/>
```

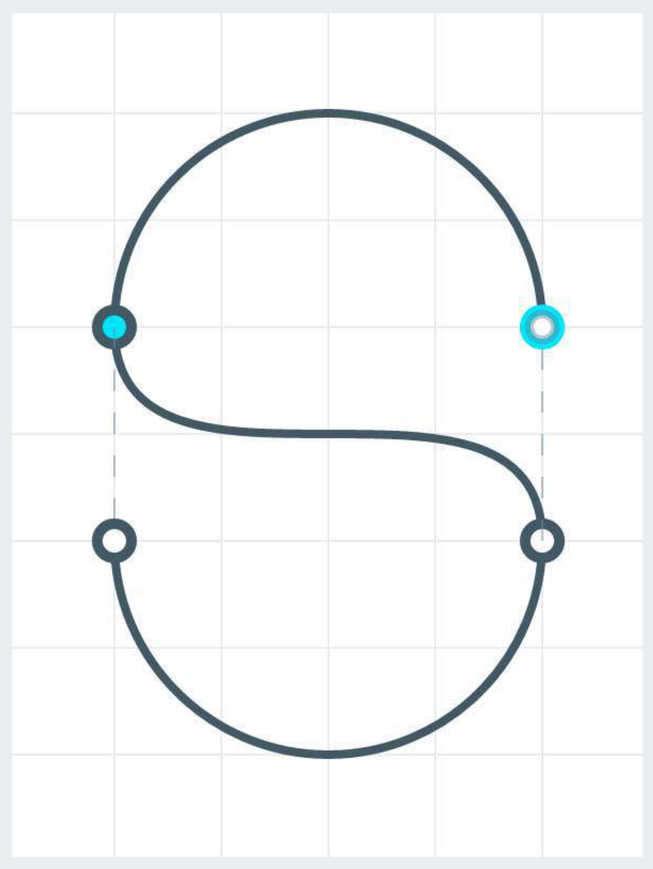


Elliptical Arc Curve

You can denote an elliptical arc curve with a leading `A`. This one has the most parameters:

Elliptical arc

```
<path d="M350 300 A50 50 0 1 0 150 300 C150 400 350 300 350 400 A50 50 0 1 1 150 400"/>
```



Styling Paths

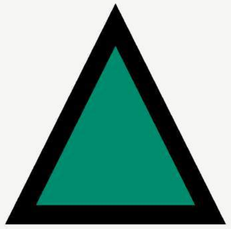
Paths can be styled or animated just like any other SVG element!

```
<path d="M7,10 L12,0 L17,10 L7,10 Z" fill="#008B6F" stroke="black" stroke-width="1">
</path>
```

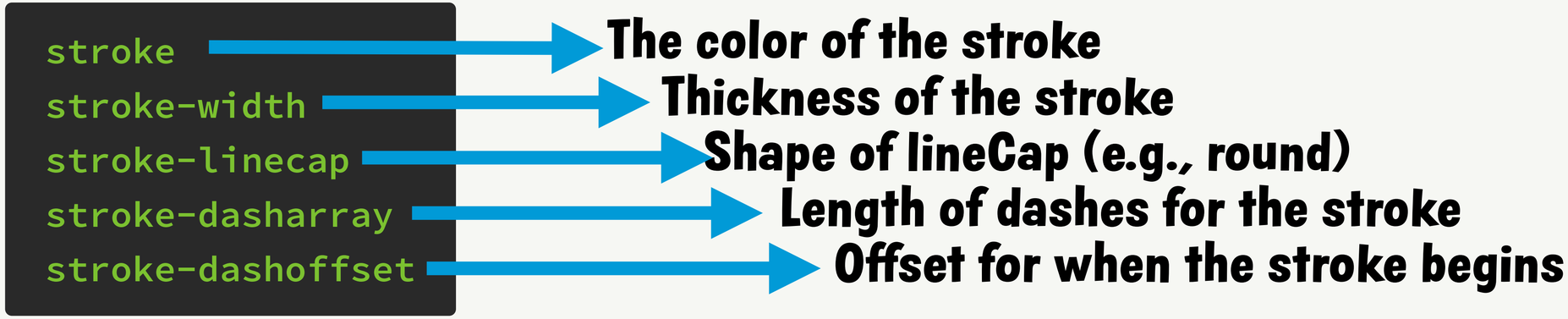
You can also do these styles in CSS.

```
<path d="M7,10 L12,0 L17,10 L7,10 Z"></path>
```

```
path {
  fill: #008B6F;
  stroke: #000;
  stroke-width: 2px;
}
```



These attributes exist to style the path:

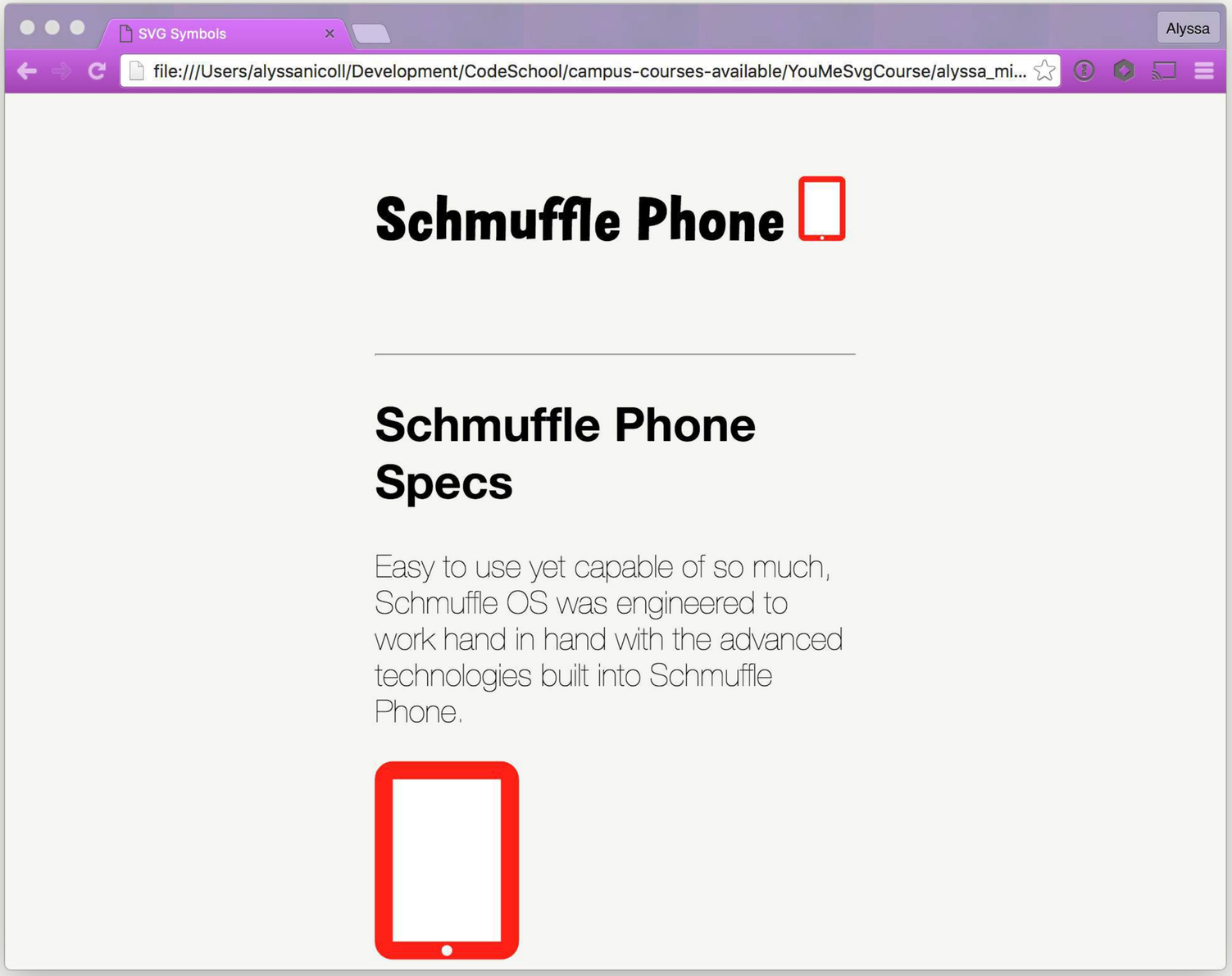


Level 4

SVG Encore!

Section 2 – Symbols to Use

Icons Everywhere



Icons Everywhere

```
<!DOCTYPE html>
<html>
  <head>...</head>
    <body>
      <svg xmlns="http://www.w3.org/2000/svg" class="defined-icon">
        <rect x="5" y="5" width="70" height="100" rx="5"/>
        <circle r="3" cy="105" cx="40"/>
      </svg>
      <h2>Schmuffle-Phone Specs</h2>
      <p>Easy to use yet capable of so much, Schmuffle OS
was engineered to work hand in hand with the advanced
technologies built into Schmuffle Phone.</p>
      <svg xmlns="http://www.w3.org/2000/svg">
        <rect height="100" width="70" x="5" y="5" rx="5"/>
        <circle r="3" cy="105" cx="40"/>
      </svg>
    ...
  </body>
</html>
```

...If you need to use an icon in multiple places on your page, the duplicate SVG can get a bit out of control!

Symbol Is for Reusable Elements!

The symbol element stores the SVG for later **use**.

```
<!DOCTYPE html>
<html>
  <head>...
  </head>
  <body>
    <svg xmlns="http://www.w3.org/2000/svg" class="defined-icon">
      <symbol id="phone">
        <rect x="5" y="5" width="70" height="100" rx="5"/>
        <circle r="3" cy="105" cx="40"/>
      </symbol>
    </svg>
  </body>
</html>
```

Code to draw icon goes inside symbol

style.css

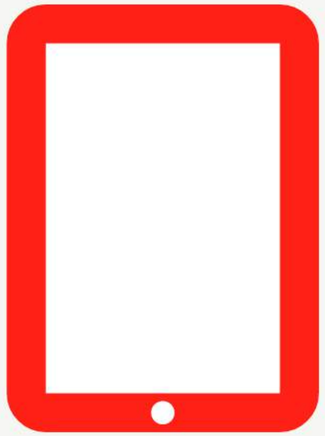
```
.defined-icon {
  display: none;
}
```

Just define the icon — do not display it.

Styles Still Being Applied Through CSS

```html
<!DOCTYPE html>
<html>
  <head>...
    <link rel="stylesheet" href="style.css">
  </head>
  <body>
    <svg xmlns="http://www.w3.org/2000/svg" class="defined-icon">
      <symbol id="phone">
        <rect x="5" y="5" width="70" height="100" rx="5"/>
        <circle r="3" cy="105" cx="40"/>
      </symbol>
    </svg>
  </body>
</html>
```

Styles are already being applied with existing CSS



style.css

```css
#phone rect, #phone circle {
  fill: white;
}

#phone rect {
  stroke: #FF2626;
  stroke-width: 10px;
}
```

Displaying the Icon With <use>

The use tag references the id of an element, group, or symbol and displays it inline where it is.

```
...
<svg xmlns="http://www.w3.org/2000/svg" class="defined-icon">
  <symbol id="phone">
    <rect x="5" y="5" width="70" height="100" rx="5"/>
    <circle r="3" cy="105" cx="40"/>
  </symbol>
</svg>
...
<svg xmlns="http://www.w3.org/2000/svg"
     viewBox="0 0 80 110"
     class="displayed-icon">
     version="1.1"
     xmlns:xlink="http://www.w3.org/1999/xlink">
  <use xlink:href="#phone"/>
</svg>
...
```

viewbox goes on second svg

Must be an id, not a class!

In order to use XLink, you need to specify an XLink namespace on the SVG that will be using XLink!

Use Tag for External Sources

The use tag's XLink points to a named anchor. This can also be an outside source (like a file):

```
<use xlink:href="path-to-file.svg#phone" />
```

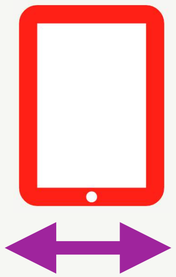
Unfortunately, external references don't work in IE10 and below.

Give the Displayed Icon Responsive Width

```
...
<svg xmlns="http://www.w3.org/2000/svg" class="defined-icon">
  <symbol id="phone">
    <rect x="5" y="5" width="70" height="100" rx="5"/>
    <circle r="3" cy="105" cx="40"/>
  </symbol>
</svg>
...
<svg xmlns="http://www.w3.org/2000/svg"
     viewBox="0 0 80 110"
     class="displayed-icon">
  <use xlink:href="#phone"/>
</svg>
...
```

style.css

```
#phone rect, #phone circle {
  fill: white;
}

#phone rect {
  stroke: #FF2626;
  stroke-width: 10px;
}
...
.displayed-icon {
  height: auto;
  width: 30%;
}
```



We are using the symbol now, but let's go ahead and give it a responsive width!

index.html — /Users/alyssanicoll/Development/CodeSchool/campus-courses-available/YouMeSvgCourse/alyssa_mini_course_repos/level_4_course

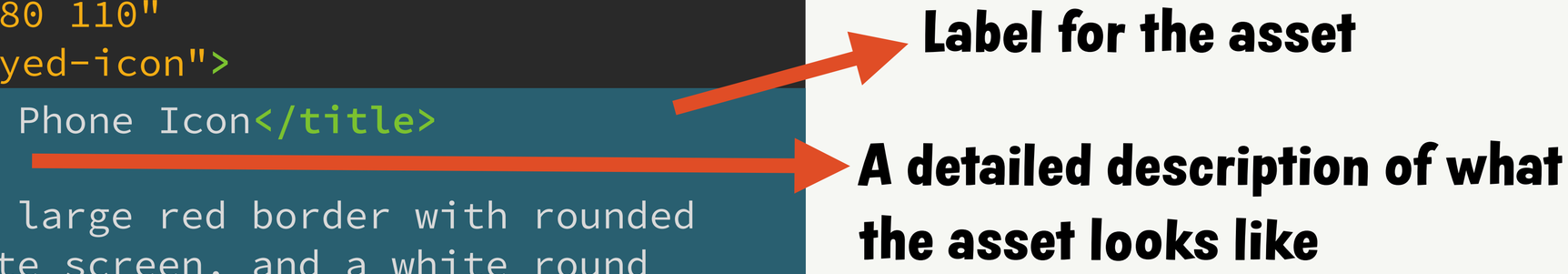
index.html style.css

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>SVG Symbols</title>
    <link rel="stylesheet" href="style.css" charset="utf-8">
  </head>
  <body>
    <div class="wrapper">
      <svg xmlns="http://www.w3.org/2000/svg" version="1.1" class="defined-icon">
        <symbol id="phone">
          <rect height="100" width="70" x="5" y="5" rx="5"/>
          <circle r="3" cy="105" cx="40"/>
        <symbol/>
      </svg>

      <h1>Schmuffle Phone
        <svg xmlns="http://www.w3.org/2000/svg"
             xmlns:xlink="http://www.w3.org/1999/xlink"
             viewbox="0 0 80 110"
             class="top-displayed-icon">
          <use xlink:href="#phone"/>
        </svg>
      </h1>

      <hr>
```

SVG Accessibility

Two SVG elements that we can use here to make our SVG more meaningful and accessible:

```
...
<svg xmlns="http://www.w3.org/2000/svg"
     xmlns:xlink="http://www.w3.org/1999/xlink"
     viewBox="0 0 80 110"
     class="displayed-icon">
  <title>Schmuffle Phone Icon</title>
  <desc>
    A phone with a large red border with rounded
    corners, a white screen, and a white round
    button centered below the screen.
  </desc>
  <use xlink:href="#phone"/>
</svg>
```



Now Screenreaders Can Describe the SVG