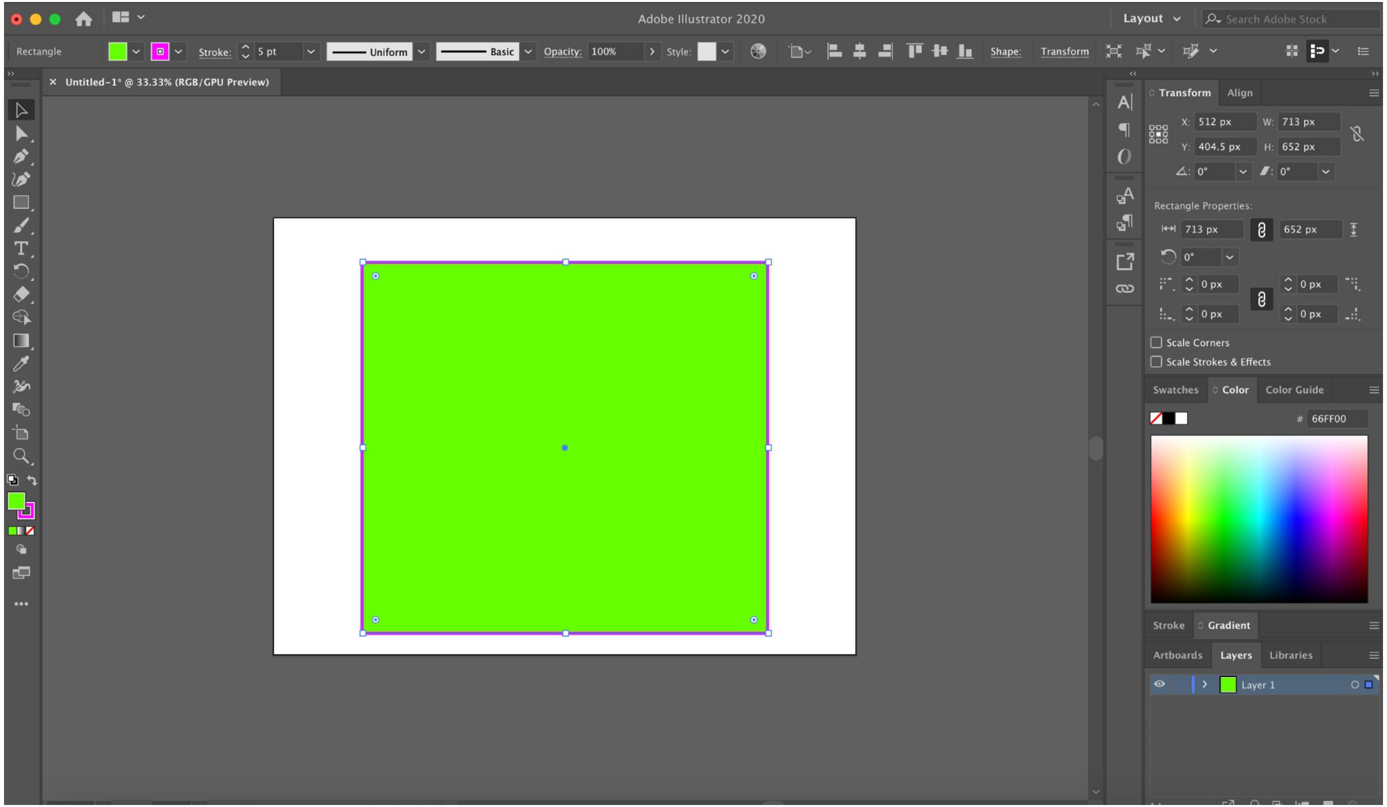
SVG GRAPHICS

2 methods

USING ADOBE ILLUSTRATOR



Adobe Illustrator 2020

Untitled-1* @ 33.33% (RGB/GPU Preview)

Rectangle

Stroke: 5 pt

Uniform

Basic

Opacity: 100%

Style:

Shape:

Transform

X: 512 px

W: 713 px

Y: 404.5 px

H: 652 px

0°

0°

Rectangle Properties:

713 px

652 px

0°

0 px

0 px

0 px

0 px

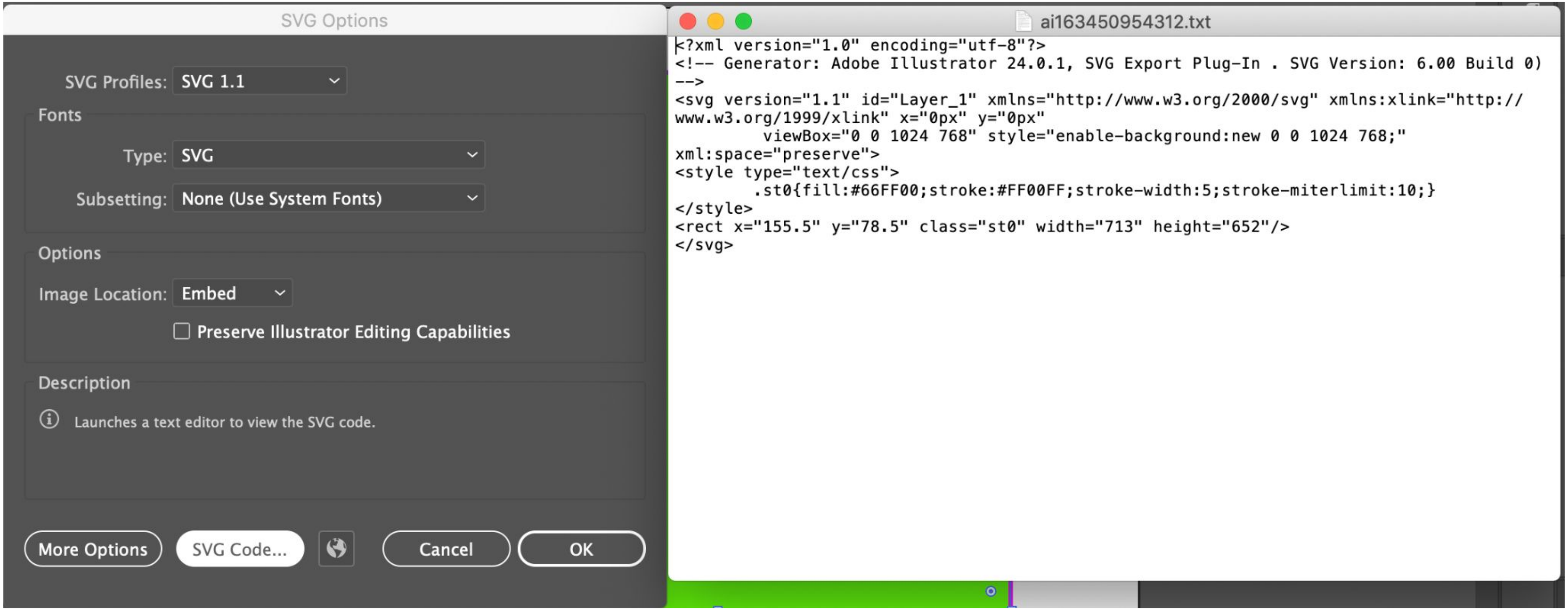
□ Scale Corners

□ Scale Strokes & Effects

66FF00

Layer 1

save AS "SVG" - uncheck preserve Illustrator editing capabilities



SVG Options

SVG Profiles: SVG 1.1

Fonts

Type: SVG

Subsetting: None (Use System Fonts)

Options

Image Location: Embed

☐ Preserve Illustrator Editing Capabilities

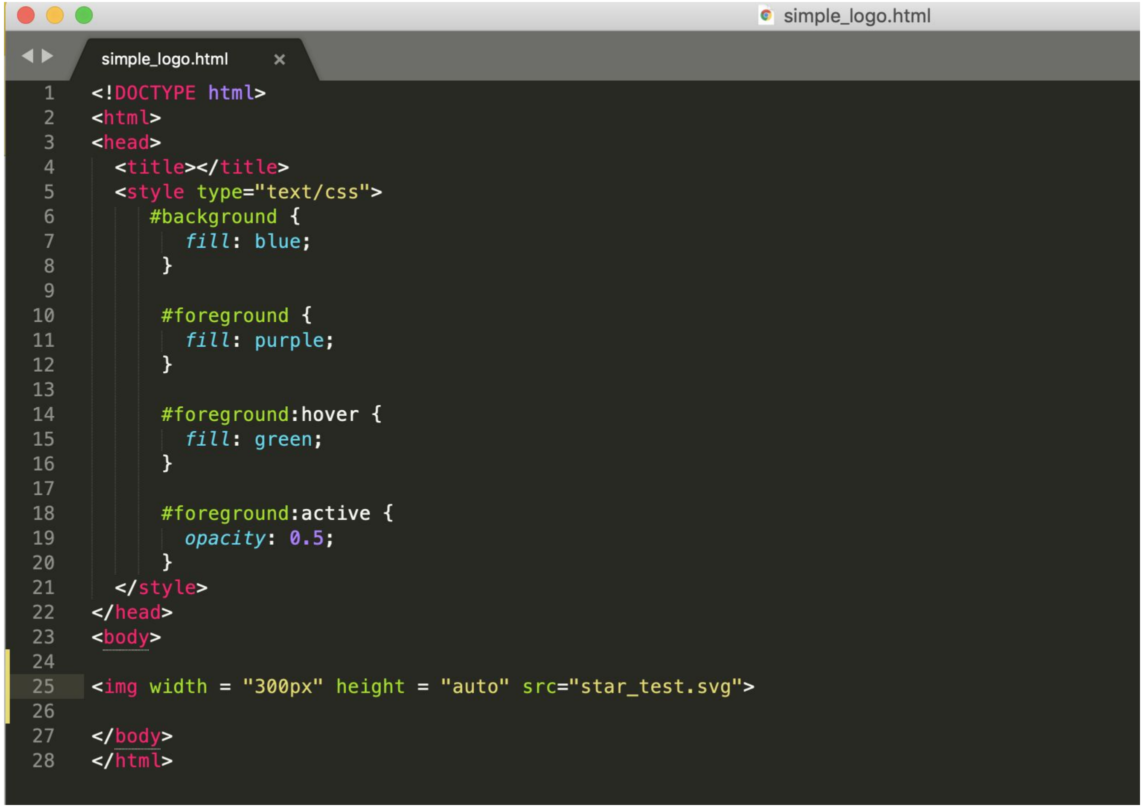
Description

ⓘ Launches a text editor to view the SVG code.

ai163450954312.txt

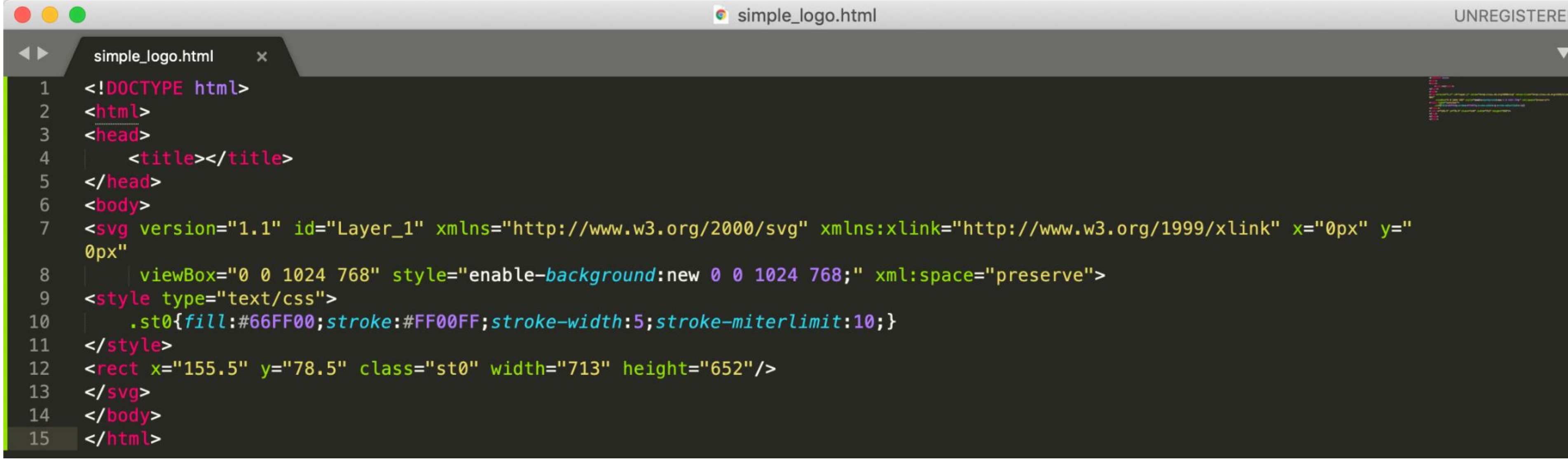
```
<?xml version="1.0" encoding="utf-8"?>
<!-- Generator: Adobe Illustrator 24.0.1, SVG Export Plug-In . SVG Version: 6.00 Build 0)
-->
<svg version="1.1" id="Layer_1" xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://
www.w3.org/1999/xlink" x="0px" y="0px"
         viewBox="0 0 1024 768" style="enable-background:new 0 0 1024 768;"
xml:space="preserve">
<style type="text/css">
        .st0{fill:#66FF00;stroke:#FF00FF;stroke-width:5;stroke-miterlimit:10;}
</style>
<rect x="155.5" y="78.5" class="st0" width="713" height="652"/>
</svg>
```

YOU CAN LINK



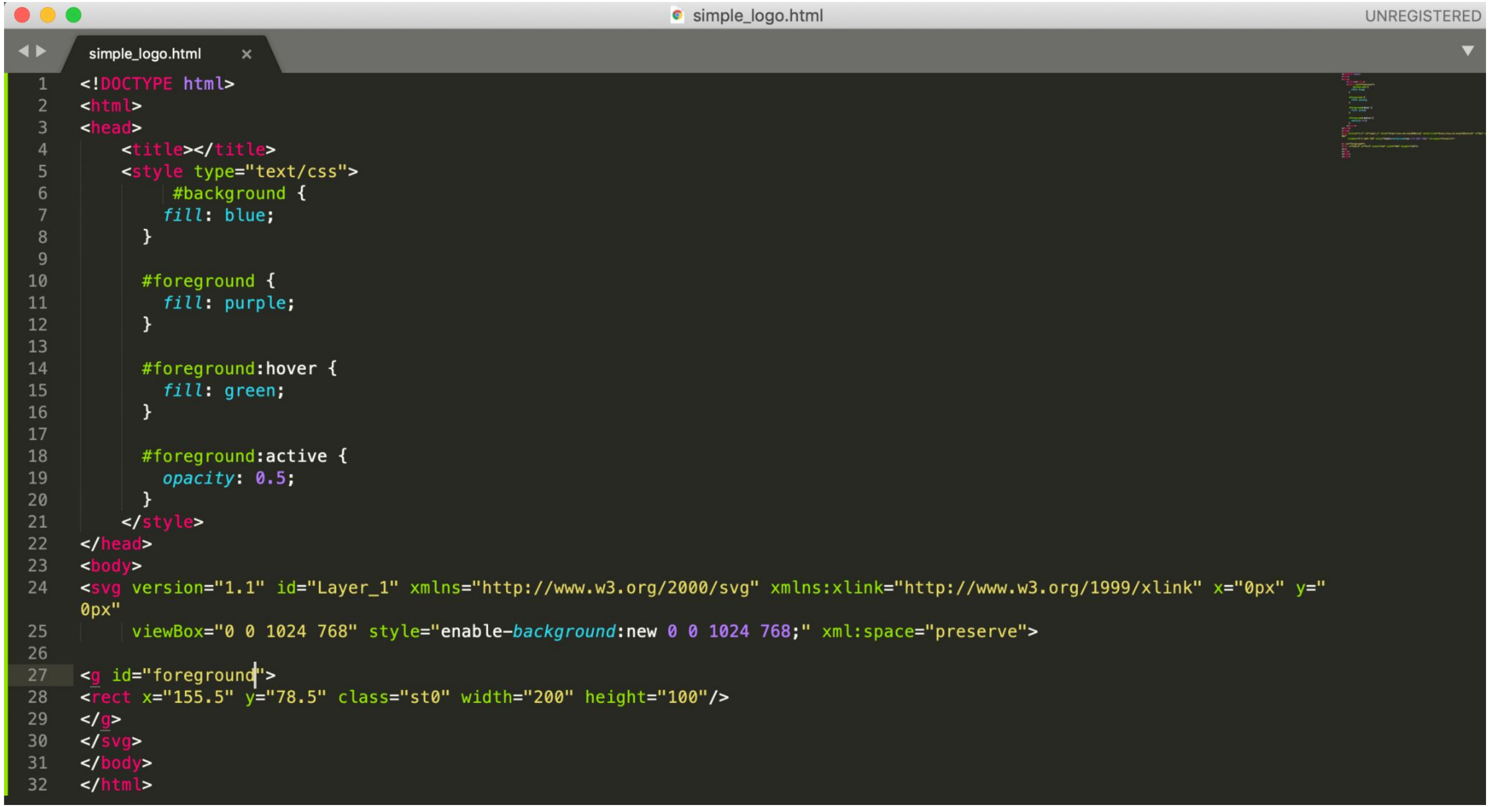
```
<!DOCTYPE html>
<html>
<head>
    <title></title>
    <style type="text/css">
        #background {
            fill: blue;
        }

        #foreground {
            fill: purple;
        }

        #foreground:hover {
            fill: green;
        }

        #foreground:active {
            opacity: 0.5;
        }
    </style>
</head>
<body>

<img width = "300px" height = "auto" src="star_test.svg">

</body>
</html>
```

OR EMBED ONTO A WEBPAGE



```html
<!DOCTYPE html>
<html>
<head>
    <title></title>
</head>
<body>
<svg version="1.1" id="Layer_1" xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" x="0px" y="0px"
     viewBox="0 0 1024 768" style="enable-background:new 0 0 1024 768;" xml:space="preserve">
<style type="text/css">
    .st0{fill:#66FF00;stroke:#FF00FF;stroke-width:5;stroke-miterlimit:10;}
</style>
<rect x="155.5" y="78.5" class="st0" width="713" height="652"/>
</svg>
</body>
</html>
```

ADD STYLING ON THE CSS END

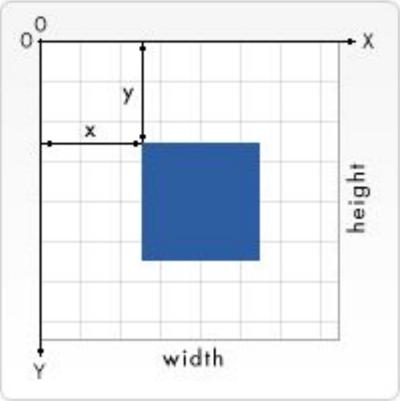


```
<!DOCTYPE html>
<html>
<head>
    <title></title>
    <style type="text/css">
         #background {
        fill: blue;
      }

      #foreground {
        fill: purple;
      }

      #foreground:hover {
        fill: green;
      }

      #foreground:active {
        opacity: 0.5;
      }
    </style>
</head>
<body>
<svg version="1.1" id="Layer_1" xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" x="0px" y="0px"
     viewBox="0 0 1024 768" style="enable-background:new 0 0 1024 768;" xml:space="preserve">

<g id="foreground">
<rect x="155.5" y="78.5" class="st0" width="200" height="100"/>
</g>
</svg>
</body>
</html>
```

PURE CODE SOLUTIONS

Scalable Vector Graphics, SVG, is a W3C XML dialect to mark up graphics.

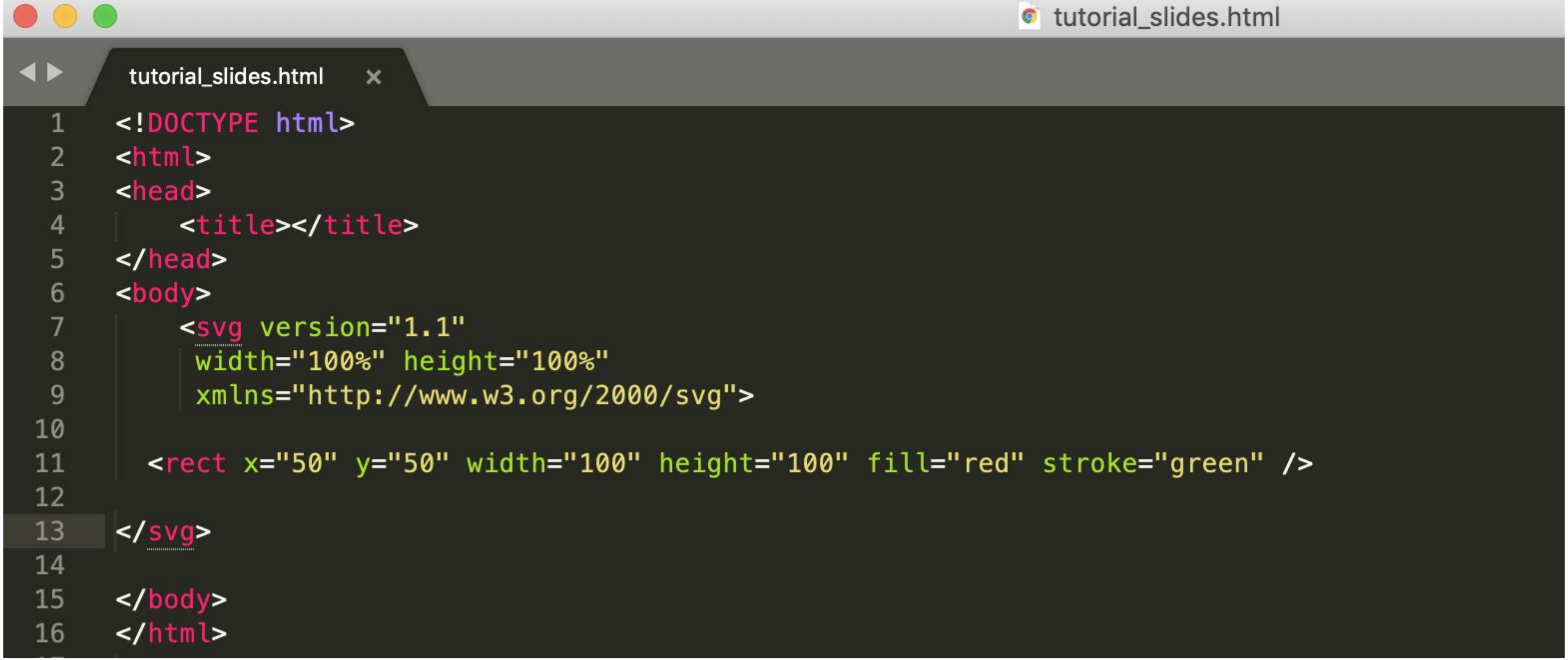
POSITIONS



For all elements, SVG uses a coordinate system or **grid** system similar to the one used by canvas (and by a whole lot of other computer drawing routines). That is, the top left corner of the document is considered to be the point (0,0), or point of origin. Positions are then measured in pixels from the top left corner, with the positive x direction being to the right, and the positive y direction being to the bottom.

Note that this is slightly different than the way you're taught to graph as a kid (y axis is flipped). However, this is the same way elements in HTML are positioned (By default, LTR documents are considered not the RTL documents which position X from right-to-left).

example



```
<!DOCTYPE html>
<html>
<head>
    <title></title>
</head>
<body>
    <svg version="1.1"
     width="100%" height="100%"
     xmlns="http://www.w3.org/2000/svg">

  <rect x="50" y="50" width="100" height="100" fill="red" stroke="green" />

</svg>

</body>
</html>
```

shape primitives

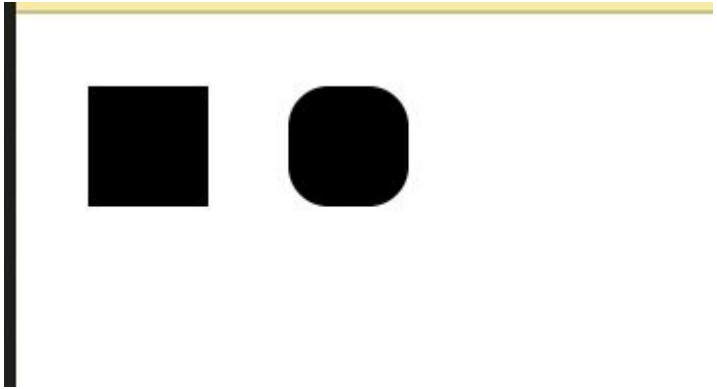
- rectangle : `<rect x y width height/>`
- circle : `<circle cx cy radius>`
- ellipse : `<ellipse cx cy rx ry/>`
- line : `<line x1 x2 y1 y2 stroke stroke-width/>`
- polyline : `<polyline points/>`
- polygon : `<polygon points/>`
- path : `<path d fill stroke stroke-width/>`

rectangle

The `<rect>` element draws a rectangle on the screen. There are 6 basic attributes that control the position and shape of the rectangles on screen. The one on the right has its `rx` and `ry` parameters set, giving it rounded corners. If they're not set, they default to `0`.

```
<rect x="10" y="10" width="30" height="30"/>
```

```
<rect x="60" y="10" rx="10" ry="10" width="30" height="30"/>
```



`x`

The x position of the top left corner of the rectangle.

`y`

The y position of the top left corner of the rectangle.

`width`

The width of the rectangle

`height`

The height of the rectangle

`rx`

The x radius of the corners of the rectangle

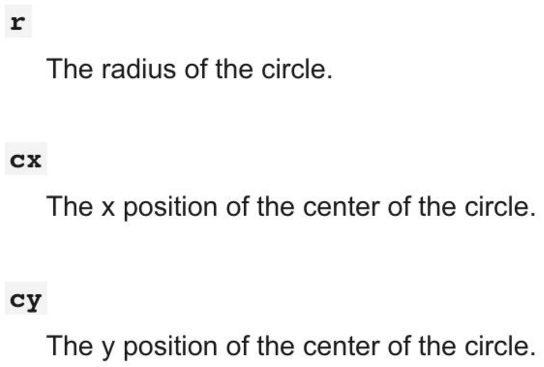
`ry`

The y radius of the corners of the rectangle

circle

The <circle> element draws a circle on the screen. It takes 3 basic parameters to determine the shape and size of the element.

```
<circle cx="25" cy="75" r="20"/>
```



`r`

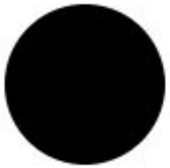
The radius of the circle.

`cx`

The x position of the center of the circle.

`cy`

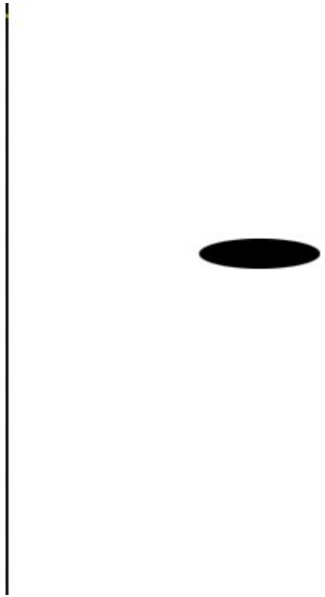
The y position of the center of the circle.

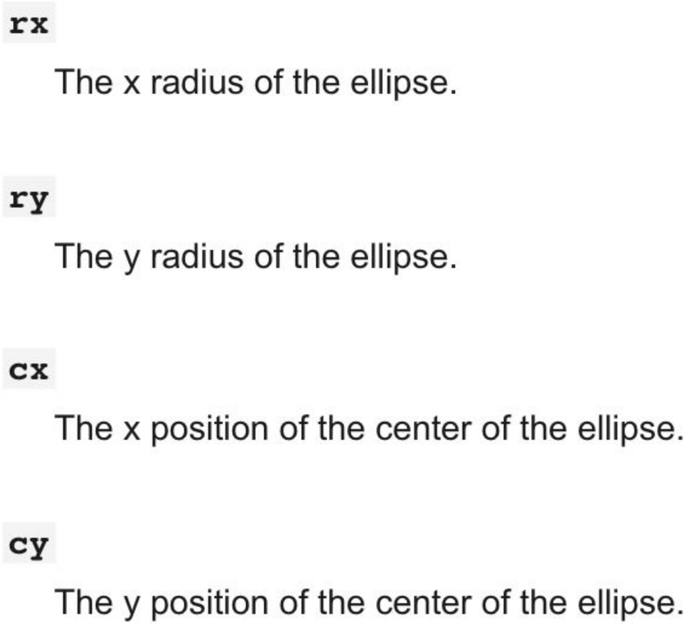


ellipse

An `<ellipse>` is a more general form of the `<circle>` element, where you can scale the x and y radius (commonly referred to as the *semimajor* and *semiminor* axes in maths) of the circle separately.

```
<ellipse cx="75" cy="75" rx="20" ry="5"/>
```





rx

The x radius of the ellipse.

ry

The y radius of the ellipse.

cx

The x position of the center of the ellipse.

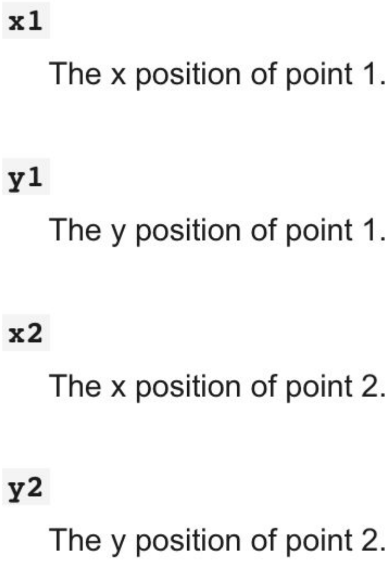
cy

The y position of the center of the ellipse.

Line

The `<line>` element takes the positions of two points as parameters and draws a straight line between them.

```
<line x1="10" x2="50" y1="110" y2="150" stroke="black" stroke-width="5"/>
```



`x1`

The x position of point 1.

`y1`

The y position of point 1.

`x2`

The x position of point 2.

`y2`

The y position of point 2.

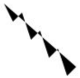
POLYLINE

A `<polyline>` is a group of connected straight lines. Since the list of points can get quite long, all the points are included in one attribute:

```
<polyline points="60, 110 65, 120 70, 115 75, 130 80, 125 85, 140 90, 135 95, 150 100, 145"/>
```

`points`

A list of points. Each number must be separated by a space, comma, EOL, or a line feed character. Each point must contain two numbers: an x coordinate and a y coordinate. So, the list `(0,0)`, `(1,1)`, and `(2,2)` would be written as `0, 0 1, 1 2, 2`.



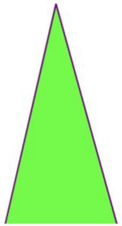
POLYGON

A `<polygon>` is similar to a `<polyline>`, in that it is composed of straight line segments connecting a list of points. For polygons though, the path automatically connects the last point with the first, creating a closed shape.

```
<polygon points="200,10 250,190 160,210" style="fill:lime;stroke:purple;stroke-width:1" />
```

`points`

A list of points, each number separated by a space, comma, EOL, or a line feed character. Each point must contain two numbers: an x coordinate and a y coordinate. So, the list `(0,0)`, `(1,1)`, and `(2,2)` would be written as `0, 0 1, 1 2, 2`. The drawing then closes the path, so a final straight line would be drawn from `(2,2)` to `(0,0)`.



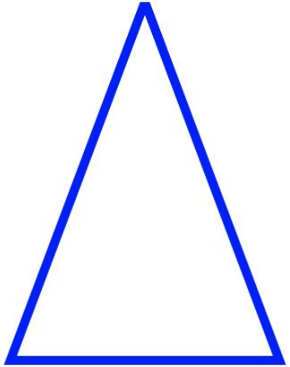
PATH

A `<path>` is the most general shape that can be used in SVG. Using a `path` element, you can draw rectangles (with or without rounded corners), circles, ellipses, polylines, and polygons. Basically any of the other types of shapes, bezier curves, quadratic curves, and many more.

```
<path d="M20,230 Q40,205 50,230 T90,230" fill="none" stroke="blue" stroke-width="5"/>
```

d

A list of points and other information about how to draw the path.



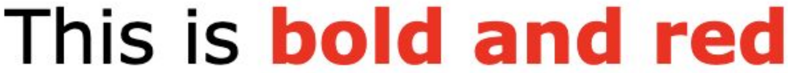
Text

The `text` element can be used to put arbitrary text in SVG documents:

```
<text x="10" y="10">Hello World!</text>
```

The `x` and `y` attributes determine, where in the viewport the text will appear. The attribute [`text-anchor`](), which can have the values "start", "middle", "end" or "inherit", decides in which direction the text flows from this point. The attribute [`dominant-baseline`]() decides the vertical alignment.

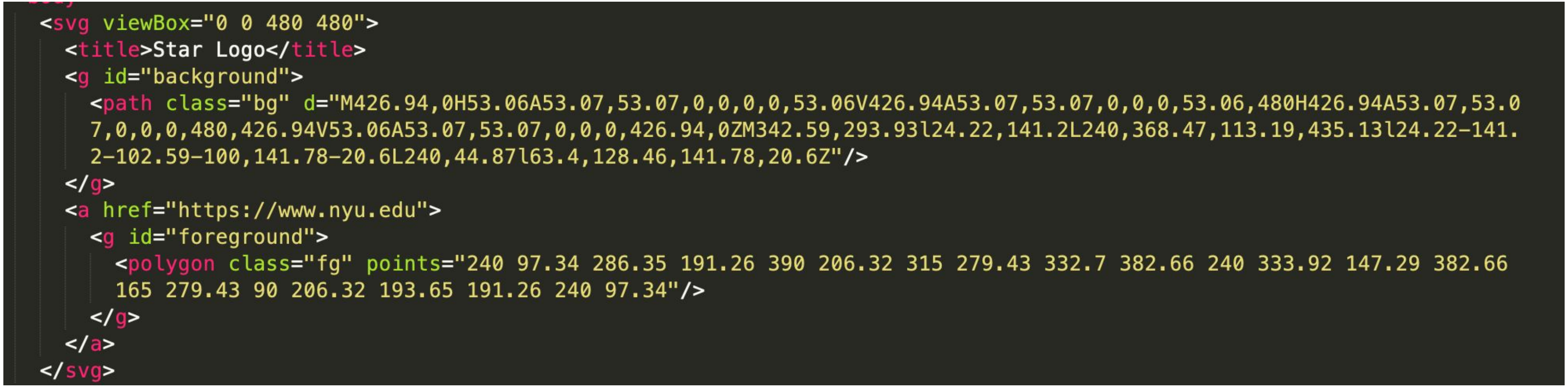
Like with the shape elements text can be colorized with the `fill` attribute and given a stroke with the `stroke` attribute. Both may also refer to gradients or patterns, which makes simple coloring text in SVG very powerful compared to CSS 2.1.



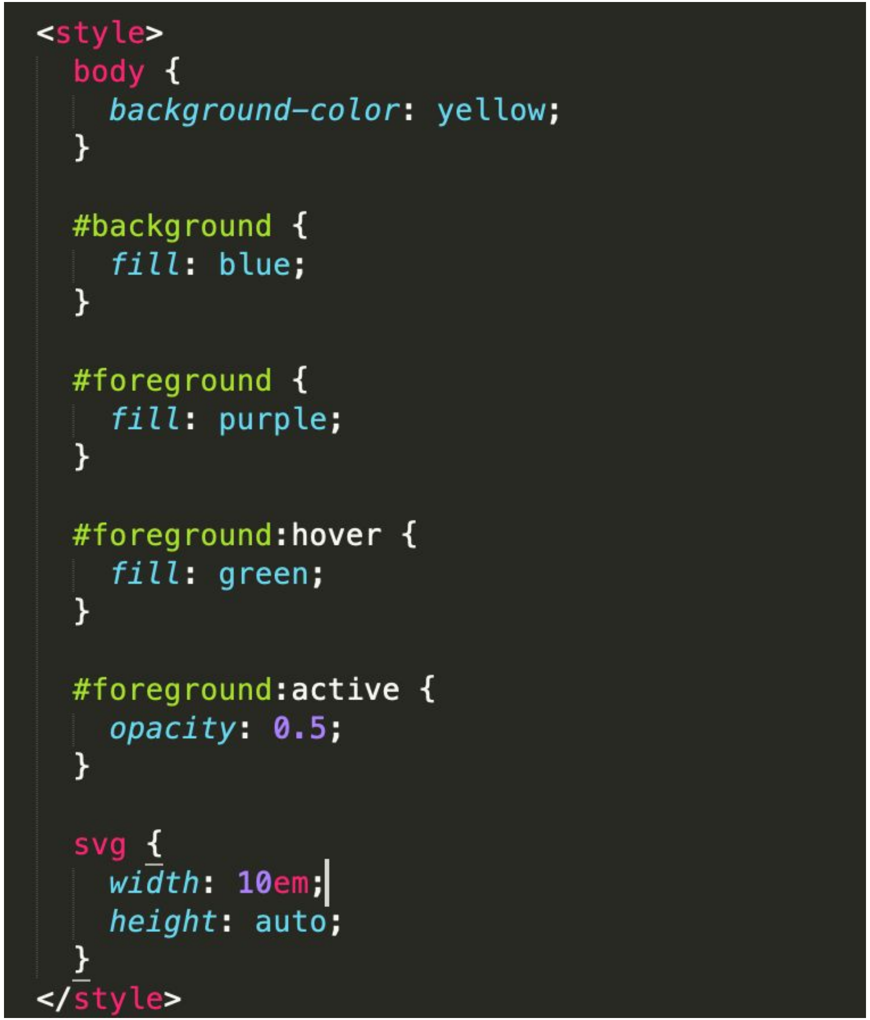
This is **bold and red**

ADDING CSS

STYLES and PSEUDO-CLASSES

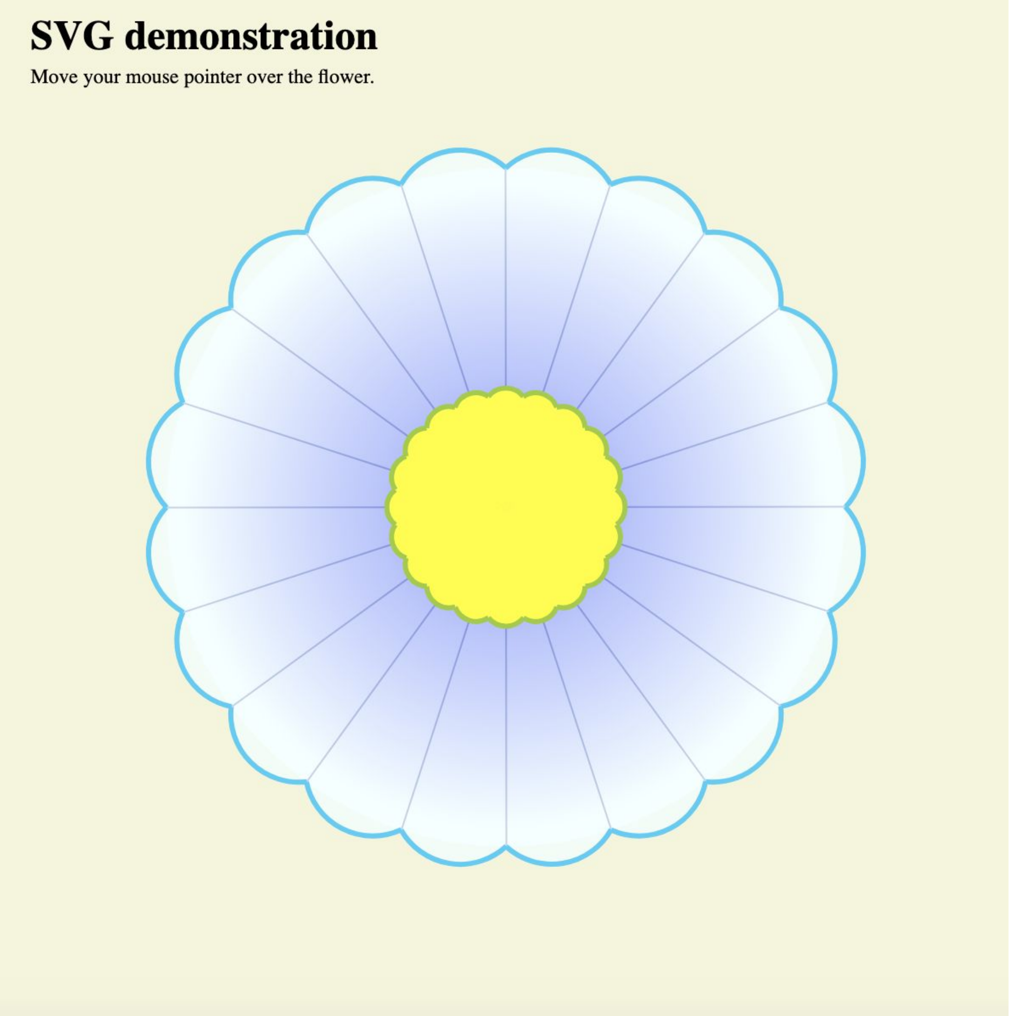


```
<svg viewBox="0 0 480 480">
  <title>Star Logo</title>
  <g id="background">
    <path class="bg" d="M426.94,0H53.06A53.07,53.07,0,0,0,0,53.06V426.94A53.07,53.07,0,0,0,53.06,480H426.94A53.07,53.0
    7,0,0,0,480,426.94V53.06A53.07,53.07,0,0,0,426.94,0ZM342.59,293.93l24.22,141.2L240,368.47,113.19,435.13l24.22-141.
    2-102.59-100,141.78-20.6L240,44.87l63.4,128.46,141.78,20.6Z"/>
  </g>
  <a href="https://www.nyu.edu">
    <g id="foreground">
      <polygon class="fg" points="240 97.34 286.35 191.26 390 206.32 315 279.43 332.7 382.66 240 333.92 147.29 382.66
      165 279.43 90 206.32 193.65 191.26 240 97.34"/>
    </g>
  </a>
</svg>
```



```
<style>
  body {
    background-color: yellow;
  }

  #background {
    fill: blue;
  }

  #foreground {
    fill: purple;
  }

  #foreground:hover {
    fill: green;
  }

  #foreground:active {
    opacity: 0.5;
  }

  svg {
    width: 10em;
    height: auto;
  }
</style>
```

https://i6.cims.nyu.edu/~mr6465/web_design/svg/star-logo.html



SVG demonstration

Move your mouse pointer over the flower.

https://i6.cims.nyu.edu/~mr6465/web_design/svg/example.html