CSCI-UA-4-005

Intro to Web Design + Computer Principles

Vector Graphics

Professor Emily Zhao

M/W 12:30PM – 1:45PM

Agenda

- Midterm Information
- **Review Vector Graphics Basics**
  - SVG as an XML-based image format
  - Coding + Styling SVGs
- Paths
- <defs>
- Gradients
- Introduce Assignment #5

Midterm

Midterm

Date: Monday, March 11

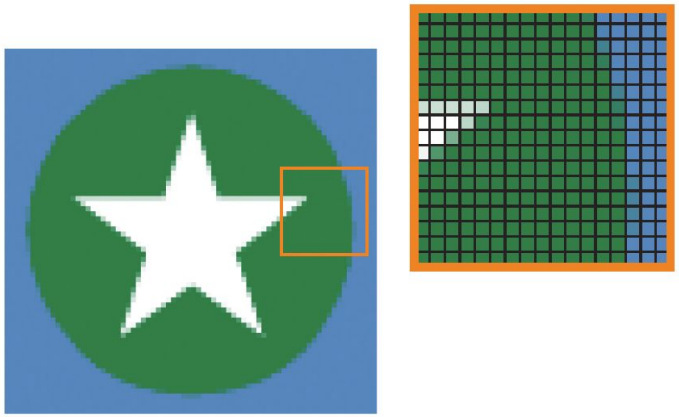
Format: Multiple Choice

Topics Covered: Computer Principles, The Internet, Unix, HTML, CSS, Web Graphics

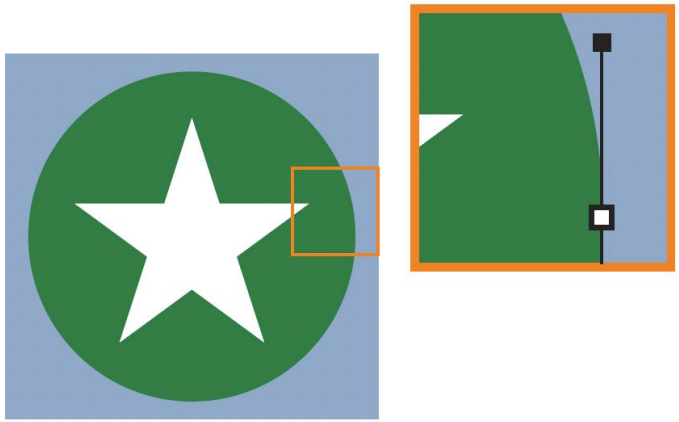
- Paper exam; no laptops/internet
- Open note (single cheat sheet front + back)
- 5-10 multiple choice questions per unit
- 25-35 multiple choice questions in reference to attached code

Vector Graphics

Raster vs Vector



Bitmapped images are made up of a grid of variously colored pixels, like a mosaic.



Vector images use mathematical equations to define shapes.

Vector Graphics

- Vector graphics contain geometric objects, such as lines and curves.
- Images can be scaled up or down without a loss of quality because the software can recalculate the shapes based on the new size.
- Since all modern displays are raster-oriented, the difference between raster-only and vector graphics comes down to where they are rasterized.
- Vector graphics are “rasterized” client side; raster graphics are, by nature, already rasterized on the server.

Scalable Vector Graphics

- Scalable Vector Graphics (SVG) is a markup language for describing two-dimensional graphics.
- SVG allows for three types of graphic objects: vector graphic shapes, images, and text.
- SVG drawings can be interactive and even styled with CSS.
- SVG defines vector graphics in XML format.

XML (eXtensible Markup Language)

- XML is a general-purpose markup language used to structure data in a way that's both human-readable and machine-readable.
- It doesn't define how data should be presented; instead, it defines the data's structure and hierarchy.
- In XML, you define your own tags and document structure; they are “extensible.” XML doesn't have predefined tags like HTML.
- SVG provides a rich, structured description of vector and mixed vector/raster graphics with pure XML.

Example XML

```
<person>
    <name>John Doe</name>
    <age>30</age>
    <address>
        <street>123 Main St</street>
        <city>New York</city>
    </address>
</person>
```

* Tags and structure are user-defined.
* XML doesn't define how this data should be displayed; it’s only used for structuring data.
* XML is flexible/extensible; HTML is specific to web page content + presentation.

Scalability

- To be scalable, means to increase or decrease uniformly.
- In terms of graphics, it means not being limited to a single, fixed, pixel size.
- On the web, scalability means that a particular technology can grow over time.
- SVG is scalable in both senses of the word.

Advantages of SVG

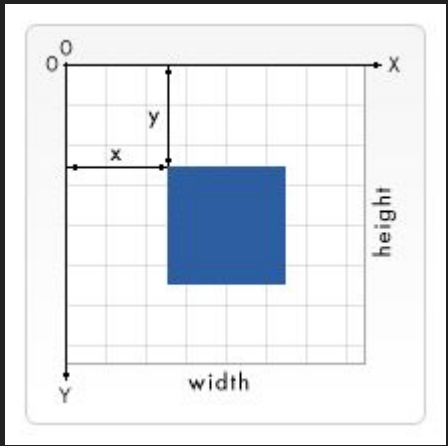
- SVG images can be created and edited with any text editor.
- SVG images can be searched, indexed, scripted, and compressed.
- SVG images are scalable, can be printed at any resolution, and are zoomable without degradation.
- SVG is an open standard!

Okay, I get it, SVGs are great – how do I make them?

How to create SVGs

1) Text Editor (Code them!)
2) Vector Graphics Software
    a) Adobe Illustrator
    b) Inkscape (open source)

SVG Code




```
<svg width="75" height="75">
    <rect x="25" y="25" width="30" height="30" fill="blue" />
</svg>
```

SVG Drawing Elements

Rectangle

Specify attributes for top, left point of rect (x, y) and size (width and height)

```
<rect x="100" y="100" width="100" height="100" />
```

Circle

Specify attributes for center point (cx, cy) and radius (r)

```
<circle cx="100" cy="100" r="50"/>
```

Line

Specify attributes for 2 points (x1, y1) and (x2,y2) as well as the line color (stroke)

```
<line x1="0" y1="80" x2="100" y2="20" stroke="black" />
```

SVG Drawing Elements

Polygon

Specify attribute for points. Each coordinate pair is separated by a space with a comma between the x and y coordinate. Creates closed shapes.

```
<polygon points="0,100 50,25 50,75 100,0" />
```

Polyline

Same as polygon, but creates open shapes; doesn't connect first point to last point.

```
<polyline points="10,100 50,25 50,75 100,0"/>
```

Text

Specify attributes for bottom-left point of text (x, y)

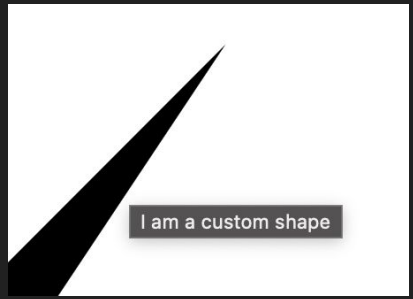
```
<text x="20" y="35">Hi there!</text>
```

SVG Drawing Elements

title

Provides an accessible, short-text description of any SVG; not rendered as part of graphic but displayed rather as a tooltip

```
<title> This is a description </title>
```



group

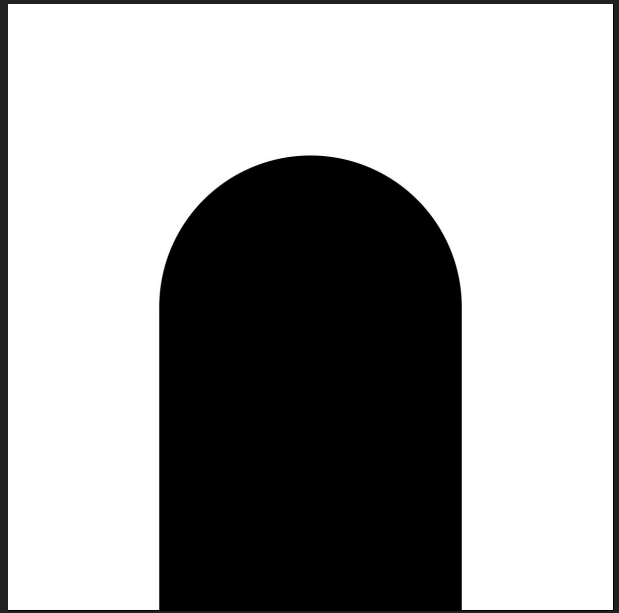
Used as a container to group SVG elements

```
<g>
    <circle cx="40" cy="40" r="25" />
    <circle cx="60" cy="60" r="25" />
</g>
```

SVG Exercise: Draw this shape!

This SVG is 200px by 200px.

- Add a title to your SVG so that on hover, a dialogue box appears with a description
- *Hint: Can you separate the larger shape into smaller shapes?*



SVG Viewbox

`viewBox` defines the logical coordinate system and aspect ratio for the SVG content, allowing for flexible and responsive scaling. Rectangular are is specified in user coordinates (`x y width height`)

```
<svg viewBox="0 0 100 100">
    <!-- SVG content goes here -->
</svg>
```

`width` and `height` set the physical dimensions of the SVG element on the screen or within the document but may not preserve the content's aspect ratio. It's typically used for fixed-size SVGs.

```
<svg width="200" height="100">
    <!-- SVG content goes here -->
</svg>
```

Styling SVGs

Common SVG Styling Properties

`fill`
sets the color inside the shape/object

`stroke`
sets the color of the line drawn around the shape/object

`stroke-width`
defines the width of the stroke
supply a value that is a number; don’t use px units!

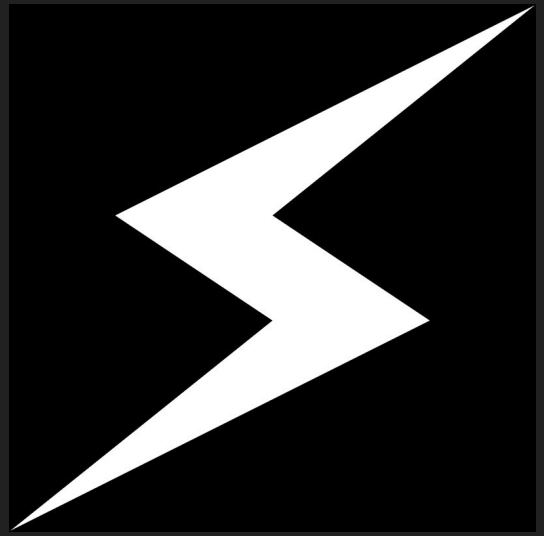
`opacity`
specifies the opacity/transparency of a shape/object
supply a value that is a floating point number from 0 to 1 (i.e. 0.5)

Using CSS Pseudo-classes

- Pseudo-classes are used to style elements that cannot be targeted using only standard element selectors.
- Pseudo-classes are denoted by a colon (":") followed by their name.
- This should look familiar to how we have styled different link <a> states
- They can be applied to SVGs as well

```
circle:hover {
    opacity: 0.4;
}
```

SVG Exercise: Draw this shape!



Create this shape using the `polygon` tag

- Make the background color of your webpage `black`
- Change the fill of your polygon to `white`
- On hover, the shape should turn `yellow`; on active the shape should turn `cyan`

: hover

: active

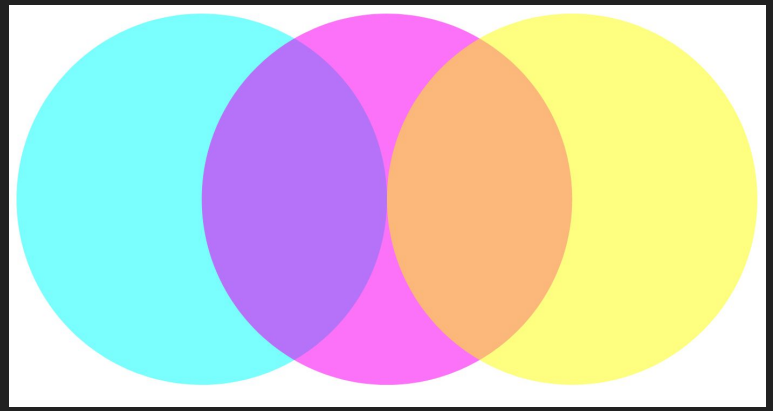
Making SVGs interactive

Making SVGs clickable

We can nest SVG drawing elements within <a> HTML elements

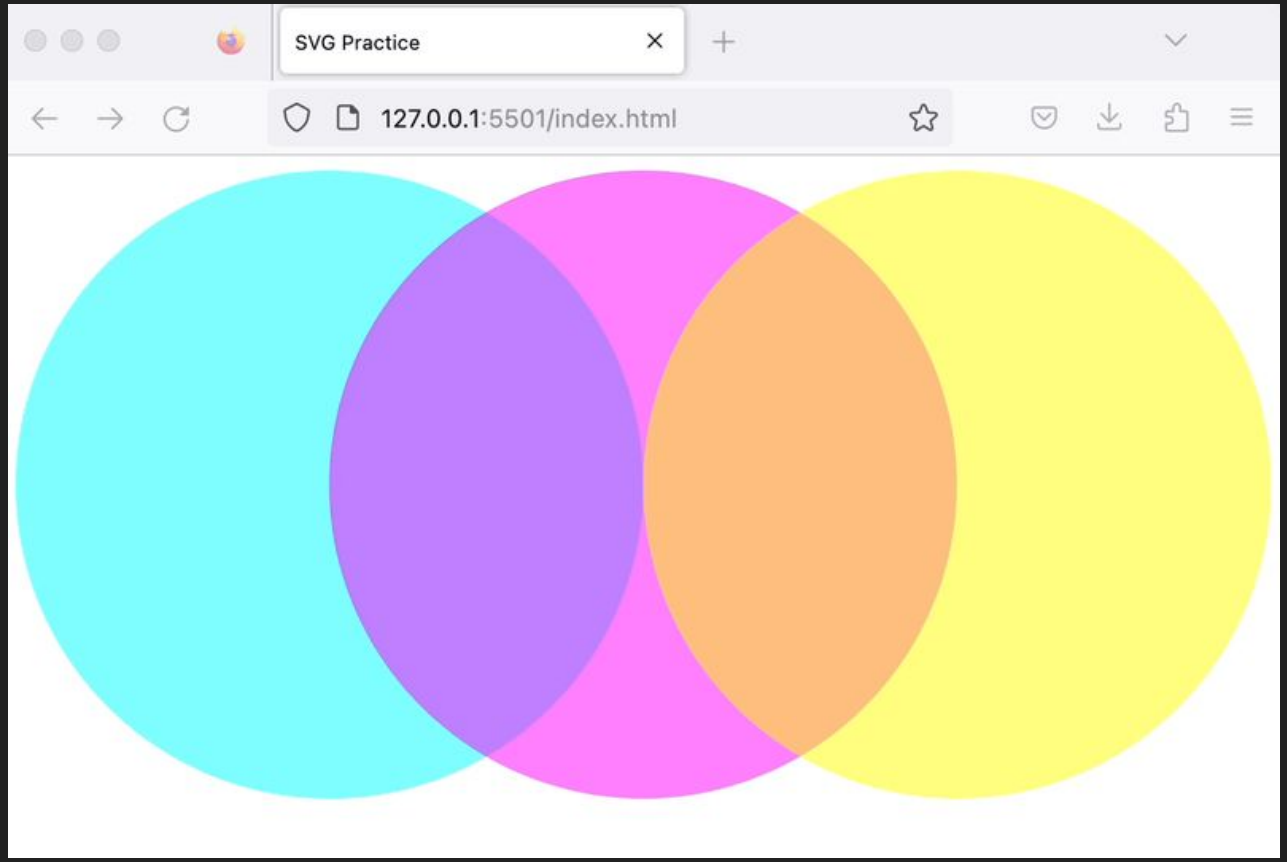
```
<svg width="100" height="100">
    <a href="www.google.com">
        <rect x="50" y="50"width="100" height="50" />
    </a>
</svg>
```

SVG Exercise: Make an SVG website



Create three circles

- The first circle should be filled with `cyan`; the second `magenta`; the third `yellow`
- All circles should have an opacity of 0.5
- On hover, the circles should become full opacity (1)
- The first circle should link to our class website; the second to exercise 1 (the door shape); the third to exercise 2 (the lightning bolt)

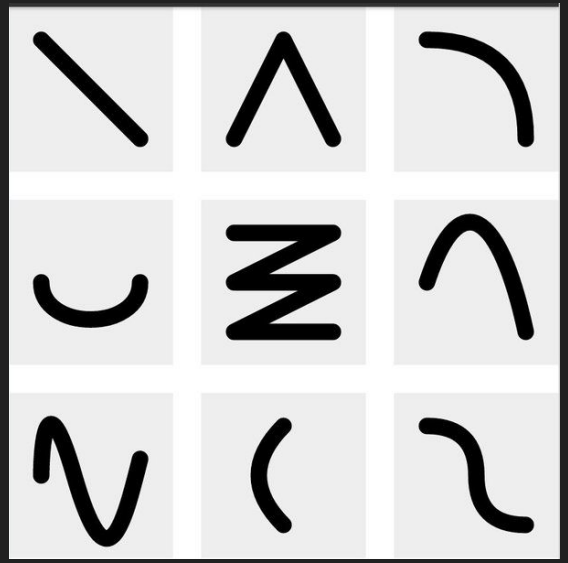

SVG Practice
127.0.0.1:5501/index.html

Path

<path>

Most powerful element in the SVG library of basic shapes

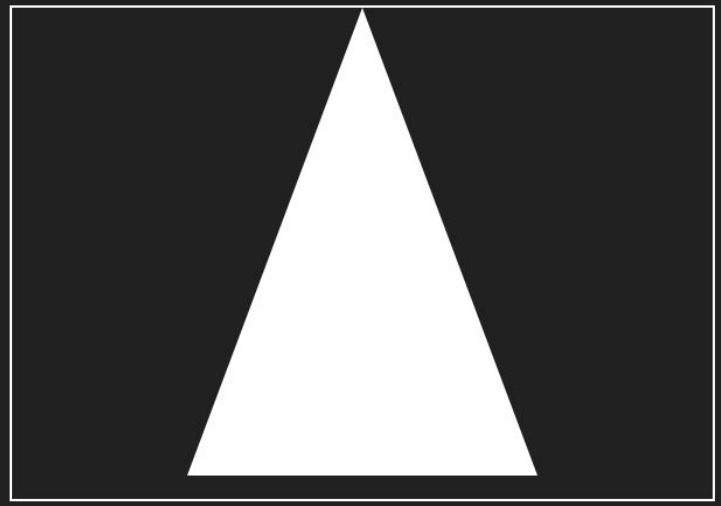
- Can be used to create rectangles, circles, ellipses, polylines, and polygons.
- Can create lines, curves, arcs, and basically any other shape, too.



<path>

Defined by one attribute d

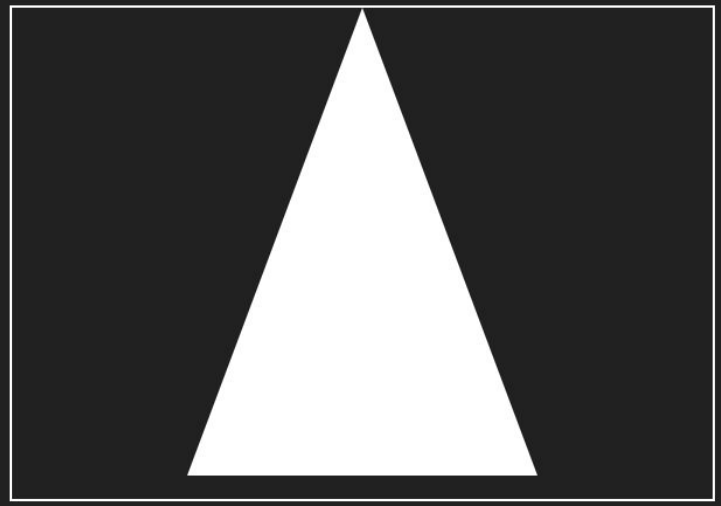
- The d attribute contains a series of commands and parameters used by those commands.
- All of the commands also come in two variants: an uppercase letter specifies absolute coordinates; a lowercase letter specifies relative coordinates.



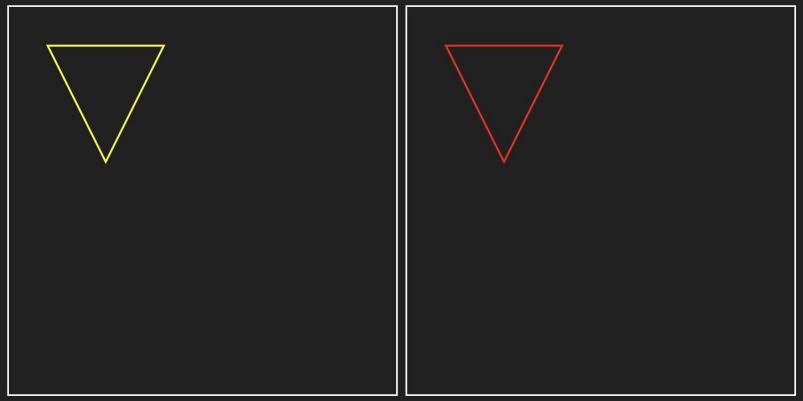
```
<svg height="210" width="300">
  <path d="M150 0 L75 200 L225 200 Z" />
</svg>
```

`<path>` commands

M = moveto
L = lineto
H = horizontal lineto
V = vertical lineto
C = curveto
S = smooth curveto
Q = quadratic Bézier curve
T = smooth quadratic Bézier curveto
A = elliptical Arc
Z = closepath



```
<svg height="210" width="300">
  <path d="M150 0 L75 200 L225 200 Z" />
</svg>
```



```
<svg width="200" height="200">
     <!-- Uppercase "M" command to move to (20, 20) -->
     <path d="M 20 20 L 80 20 L 50 80 Z" stroke="yellow"/>
</svg>

<svg width="200" height="200">
    <!-- Lowercase "m" command to move relative to the current position -->
    <path d="m 20 20 l 60 0 l -30 60 z" stroke="red"/>
</svg>
```

xmlns

`xmlns`

- An XML Namespace is a way to avoid naming conflicts when elements and attributes in an XML document
- Namespaces are identified by a unique URI (Uniform Resource Identifier)

```
<library xmlns:books="http://example.com/books">
   <books:book>
      <books:title>XML for Beginners</books:title>
      <books:author>Jane Doe</books:author>
   </books:book>
</library>
```

→ *Good explanation from StackOverflow*

<defs>

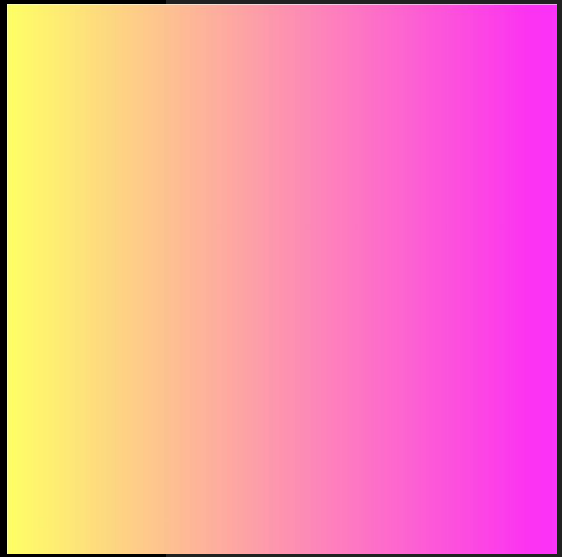
`<defs>`

- You define reusable elements, patterns, gradients, filters, and masks that can be referenced and applied within an SVG document
- `<defs>` is used to separate and store these definitions, making the SVG document more organized and efficient
- After you define elements within the `<defs>` section, you can reference and apply them in the main body of your SVG document using elements like `<use>`, `<linearGradient>`, `<radialGradient>`, `<pattern>`, `<filter>`, or `<mask>`

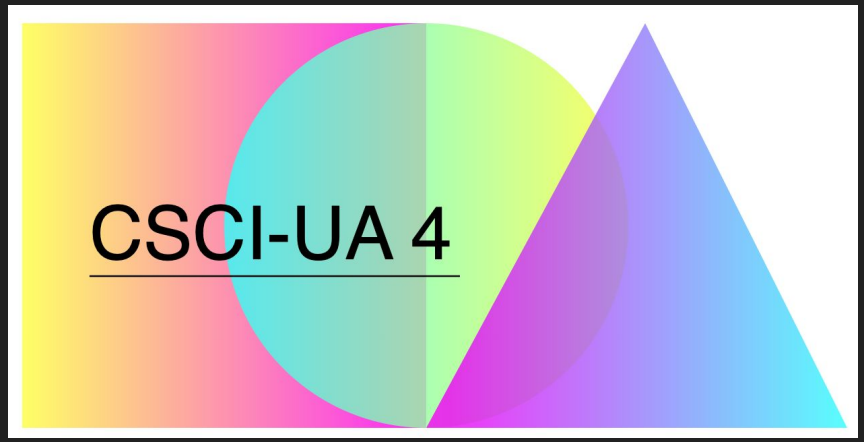
Gradients

```
<svg width="550" height="300" xmlns="http://www.w3.org/2000/svg">
     <defs>
         <linearGradient id="gradient1">
           <stop offset="0%" stop-color="yellow" />
           <stop offset="100%" stop-color="magenta" />
         </linearGradient>
     </defs>

    <rect
     x="30"
     y="30"
     width="240"
     height="240"
     fill="url(#gradient1)"
     opacity="0.8"
    />
</svg>
```



Create the following logo



Create the following five elements:

(1) square
(2) circle
(3) polygon [triangle]
(4) text
(5) line

Uses <defs> to define your gradients

Square: yellow → magenta

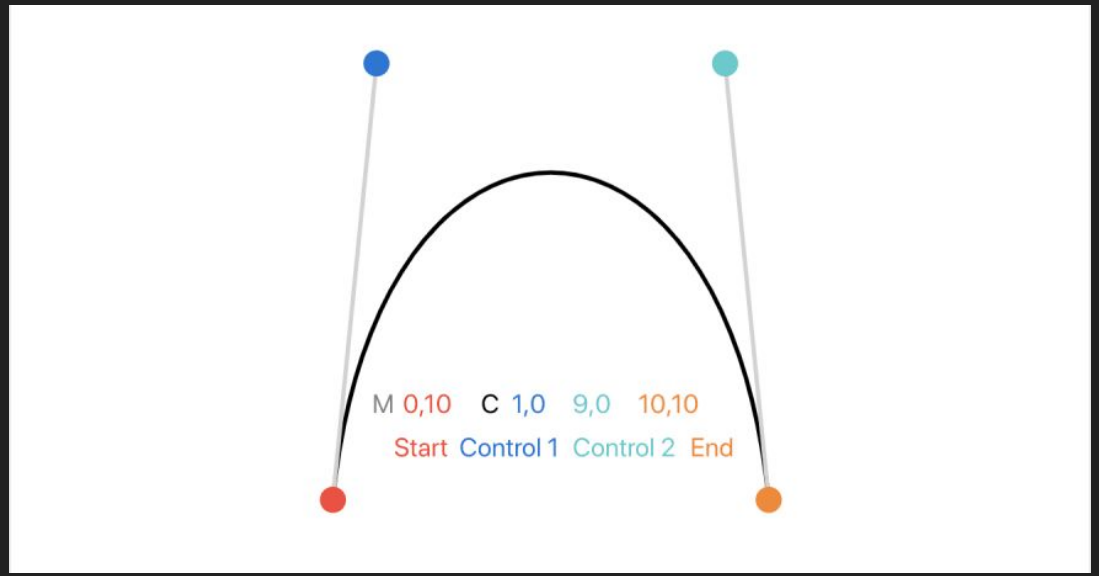
Circle: cyan → yellow

Triangle: magenta → cyan

All the ways we can embed SVGs

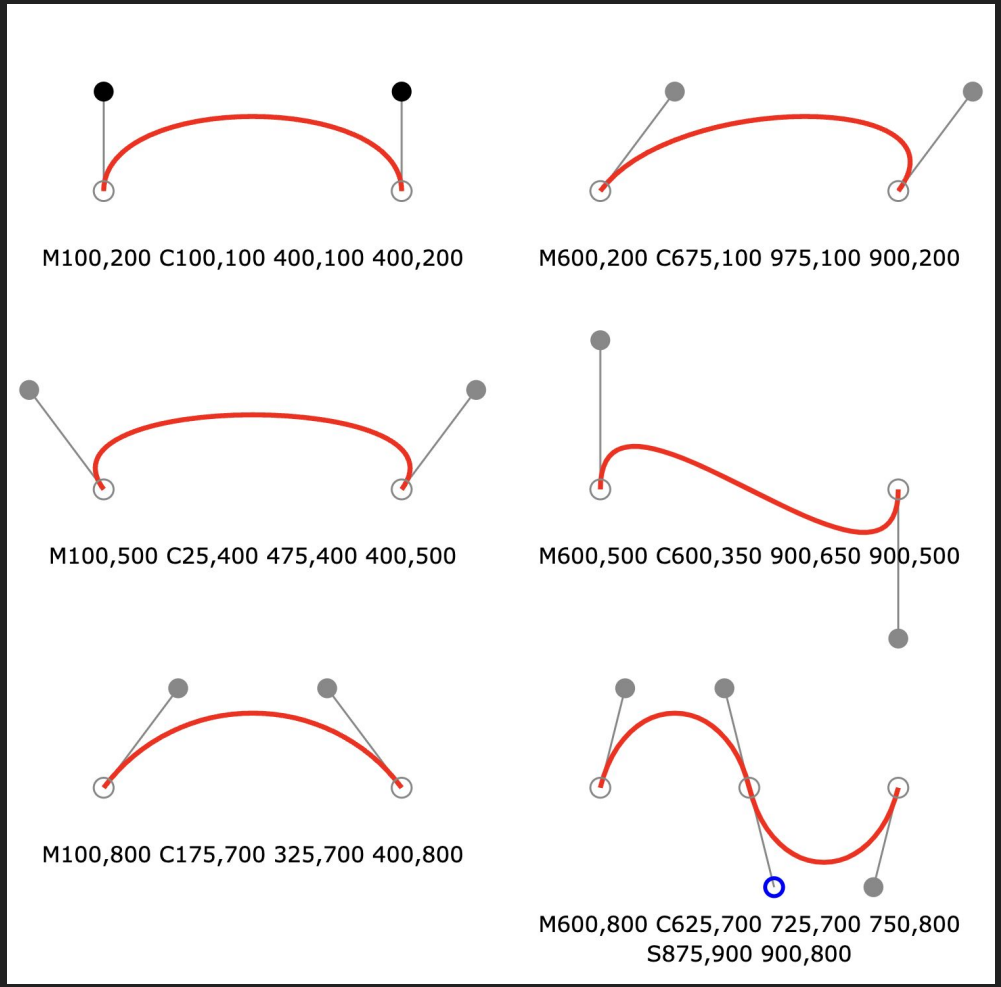
- Inline with HTML
- External link using the HTML <a> element
- Embedding by reference using the HTML <img> element
- Referenced from a CSS property (i.e. background image)
- A stand-alone SVG web page

Bezier curve




```
<path d="M 0 10 C 1 0, 9 0, 10 10" />
```

Bezier curves



Creating SVGs with Illustrator + Inkscape