# Scalable Vector Graphics

## Like Canvas, but prettier and slower

# Table of Contents

* SVG Overview
* Vector Graphics Overview
* Basic SVG Shapes
  + Rectangular
  + Ellipse
  + Path
* Drawing Text

# SVG Overview (What is SVG?)

# SVG Overview

* SVG is a technology for describing two dimensional vector graphics
  + Uses an extension of XML
* SVG stands for Scalable Vector Graphics
* SVG is platform independent
  + Understood by most browsers

# Vector Graphics Overview

* Vector graphics are based on mathematical expressions
  + The same on any resolution and zoom level and are not pixelated
* Consist of geometrical primitives such as:
  + Points
  + Lines and curves
  + Shapes or polygons
* Represent images in computer graphics
* Vectors are locations in a dimensional space

# Using SVG in a Web Page

* To use SVG you need to simply open the **<svg>** element and to start defining your shapes using XML notation

<svg width="300" height="450">

<rect x="50" y="50" width="150" height="150" fill="lightblue"></rect>

<circle cx="125" cy="125" r="60" stroke="none" fill="darkblue"></circle>

</svg>

# *Simple SVG - Live Demo*

# SVG Shapes (What can SVG do?)

# SVG Shapes

* As mentioned, vector graphics are built from graphic primitives
  + Points
  + Lines and curves
  + Shapes: rectangular, circle, etc…
* SVG supports most of the basic shapes
  + More complex shapes can be created using the basic ones

# SVG Shapes: Line

* <line> is the most basic shape in SVG
* Creates a line between two points
  + **Lines** are defined by start and end points
  + Each points has a coordinates: (x, y)

<line x1="0" y1="0" x2="300" y2="450" stroke="black" />

<line x1="300" y1="0" x2="0" y2="450" stroke="black" />

<line x1="0" y1="150" x2="300" y2="150" stroke="black" />

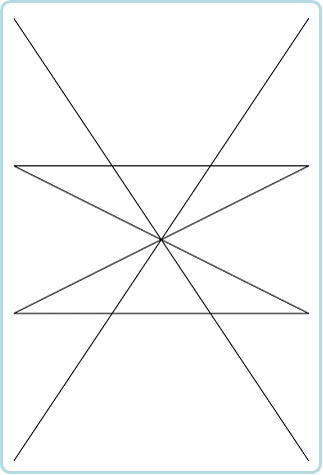
<line x1="0" y1="300" x2="300" y2="300" stroke="black" />

<line x1="0" y1="150" x2="300" y2="300" stroke="black" />

<line x1="0" y1="300" x2="300" y2="150" stroke="black" />

Stroke sets the **color** of the line

# *Defining Lines with SVG - Live Demo*



# SVG Shapes: Rects and Circles

* <rect> creates a rectangular with a **top-left position**, **width** and **height**
* <circle> creates a circle with **center position** and **radius**

<rect x="10" y="10" width="280" height="280" fill="#222"/>

<circle cx="150" cy="150" r="135" fill="#333"/>

<rect x="55" y="55" width="190" height="190" fill="#444"/>

<circle cx="150" cy="150" r="95" fill="#555"/>

<rect x="85" y="85" width="130" height="130" fill="#666"/>

<circle cx="150" cy="150" r="65" fill="#777"/>

<rect x="105" y="105" width="90" height="90" fill="#888"/>

<circle cx="150" cy="150" r="45" fill="#999"/>

# *Circles and Rectangles with SVG - Live Demo*

# SVG Paths (Defining more than rects)

# SVG Paths

* SVG can define more complex shapes using the path
  + Create straight line from a point to other point
  + Create a curve between two points
  + Used with the element <path>
    - **Add commands** and **points** for the lines using the "d" attribute`

<path d="M 50 50 L 175 310 H210"></path>

# SVG Paths: Commands

* The path commands are as follows:
  + M x y or m x y
    - **Moves** the path marker to **position (x, y)**
  + L x y or l x y
    - Creates a **straight line** between the **marker point** and **point (x, y)**
  + (H x or h x) and (V y or v y)
    - Creates a **horizontal**/**vertical** line from the **marker point** to the **given point**
  + Z or z
    - **Closes the path**, connects the first and last pointsю

# SVG Paths: Line Commands

* Paths example
  + Drawing the letter **"R"**

<path stroke="yellowgreen" fill="none"

d="M 375 50 H 450 M 375 50 V 150 H 450 M 375 100 H 430" />

* Drawing the letter **"E"**

<path stroke="yellowgreen" fill="none"

d="M 475 50 V 150 M 475 50 H 525 L 550 75 V 100 H 475 L 550 150" />

*SVG Paths: Line Commands - Live Demo*

# SVG Paths: Curves

* C x1 y1 x2 y2 x y
  + Cubic Bezier curve
  + **Two control** points: (x1, y1) and (x2, y2)
  + **Ending point** at (x, y)
  + S x2 y2 x y continues the curve
* Q x1 y1 x y
  + Quadratic Bezier curve
  + **One control** point: (x1, y1)
  + **Ending point** at (x, y)
  + T x y continues the curve

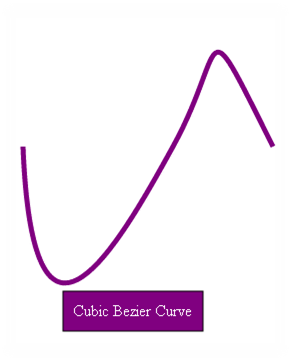
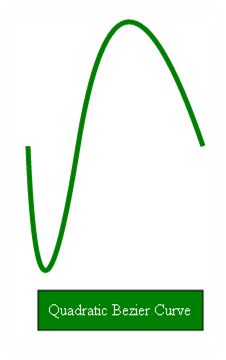
# SVG Paths: Curves Example

* Cubic Bezier Curve:

<path d="M50 200 C60 450 145 300 200 200 S225 50 300 200" />

* Quadratic Bezier Curve:

<path d="M350 200 Q360 450 400 200 T525 200" />

 ***SVG Paths: Curves - Live Demo***

# SVG DOM API (SVG with JavaScript)

# SVG DOM API

* SVG is plain old XML
  + SVG elements \*can be selected8 just as DOM elements
  + getElementsByXXX(…) and querySelector(…)
  + SVG elements can be **created dynamically**
    - document.createElement('rect')

var svgNS = 'http://www.w3.org/2000/svg';

var rect = document.createElementNS(svgNs, 'rect');

rect.setAttribute('x', x);

rect.setAttribute('y', y);

rect.setAttribute('width', width);

rect.setAttribute('height', height);

document.getElementById('the-svg').appendChild(rect);

# SVG Styles

* SVG elements can also be styled with CSS:

<!-- SVG -->

<svg id="the-svg" …="">

<rect …=""></rect>

<circle …=""></circle>

</svg>

/\* CSS \*/

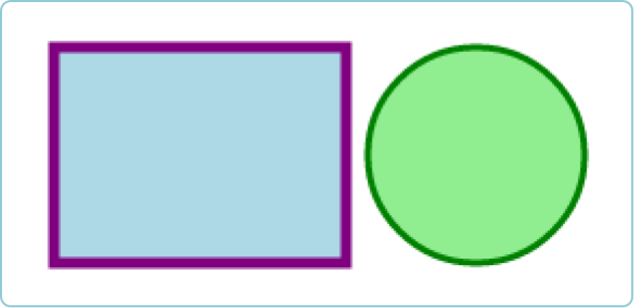
#the-svg rect{

fill: 'white';

stroke: 'purple';

stroke-width: '5';

}

******

***SVG DOM and Styles - Live Demo***