DEPARTMENT OF
COMPUTER SCIENCE
計算機科學系

SVG AND D3.JS

ITEC2016 - DATA-DRIVEN VISUALIZATION FOR THE WEB

CHAPTER 5 - 2018 - HKBU

DR. MARTIN CHOY

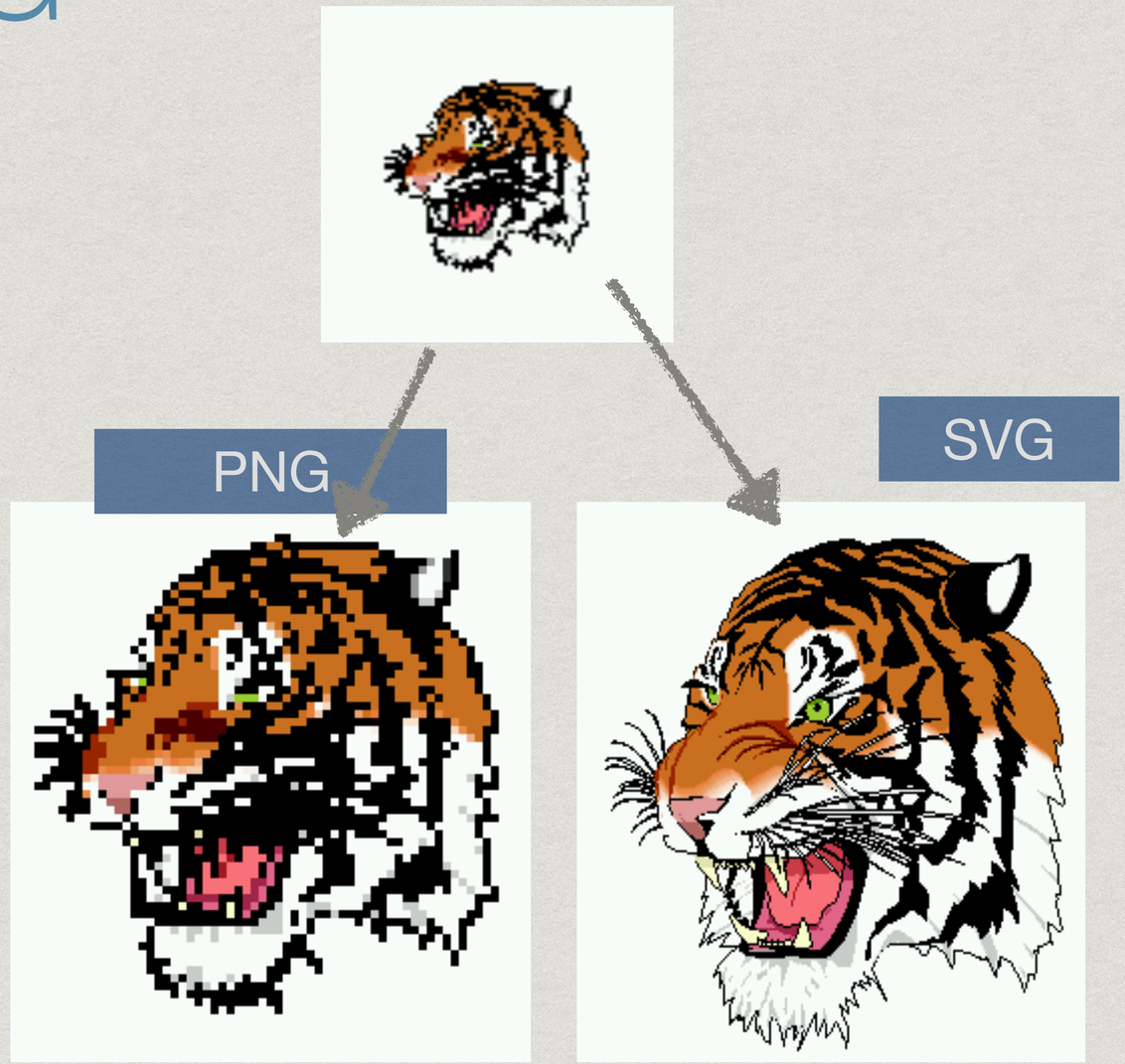
What is SVG ?

Type of content	Markup	Scripting
Text, image, etc…	HTML	JavaScript
Vector graphics	SVG	D3.js

- SVG stands for **S**calable **V**ector **G**raphics
- SVG is a language for describing 2D vector and mixed vector/raster graphics in **XML**.
- SVG drawings can be dynamic and **interactive**.
- Vector graphics **animation via scripting** (D3.js, etc…)

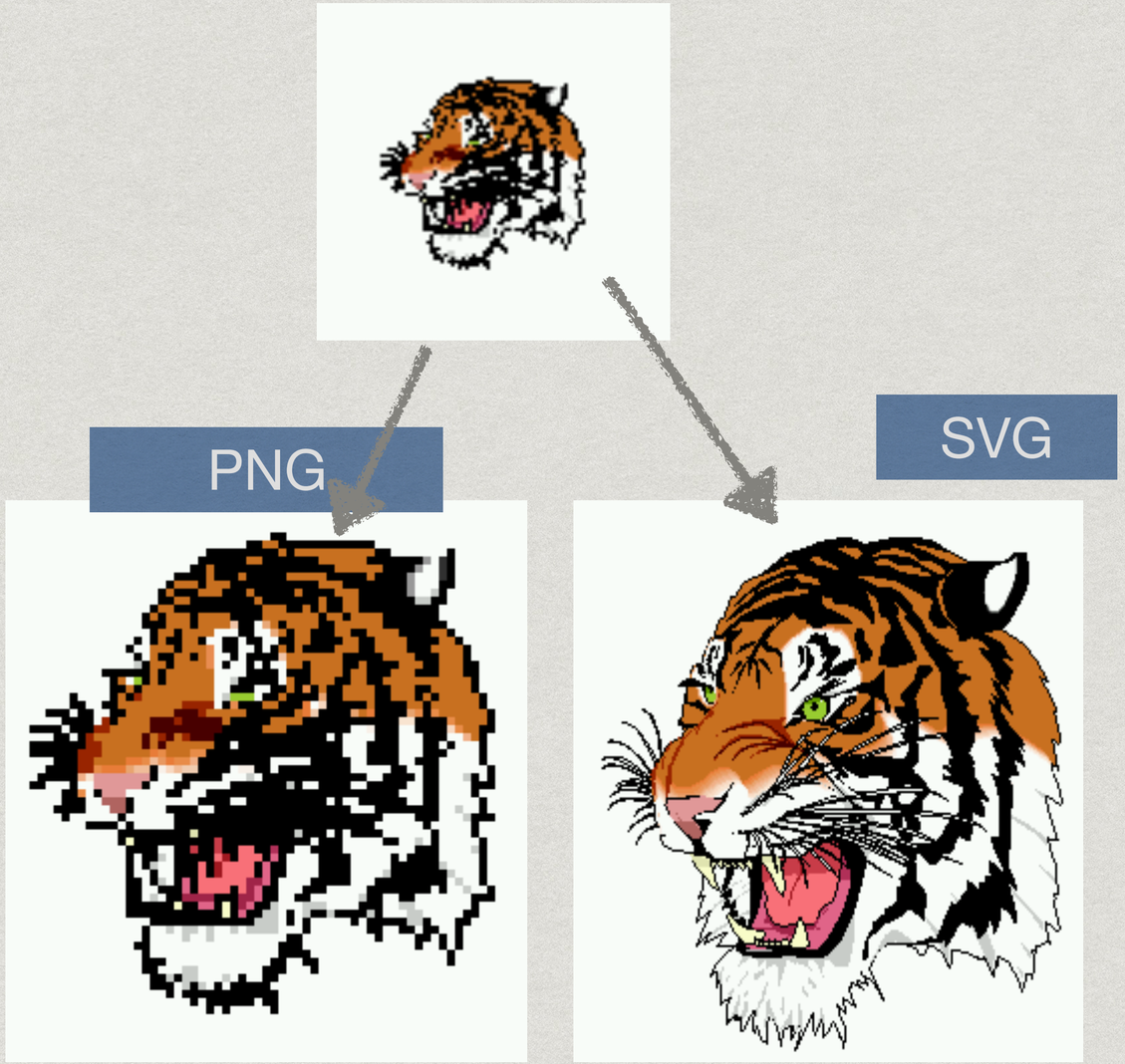
Motivation for SVG

- Text based language.
- Simple to program.
- SVG graphics is scalable to different display **resolutions** and color spaces.
  - SVG graphics can be magnified to see fine detail.



Bitmap Graphics

- **Bitmap images** are widely used over the Internet: attached to HTML documents using **<img>** tag.
- Main formats: GIF, JPEG, PNG.
- Represented in Binary format.



Vectors vs. Bitmaps (1)

- Format representation - Bitmaps are binary files, **vector based graphics can be represented as text files**.
  - Size : **Vectors cost much less to represent than bitmaps**.
- **Drawing instruction include text**, and so are selectable and searchable. Links can be created to any part of a vector based image.
  - Flexibility - **Vectors are much more flexibly to changes** (transformations and different text styles).

Vectors vs. Bitmaps (2)

- **Animation** is much simpler using vectors.
  - Accessibility to editing - **easier to edit quickly a textual file** than a binary file.
  - Interactivity – the ease of using **scripts allows very good interactivity**.

SVG is XML

- SVG is defined by **XML, a markup language similar to HTML**.
- XML stands for **EXtensible Markup Language**
- **XML was designed to describe data**
- XML tags are not predefined in XML. You must define your own tags

XML Document

- All XML elements must have **a closing tag**.
- Tags are Case sensitive
- Elements must be properly nested
- **Attribute values must be always be quoted**
- Comments in XML
  <!-- This is your comment -->

```
<svg width="300" height="200">
    <desc>
        <!-- put a description here -->
    </desc>
    <g>
        <!-- your graphic here -->
    </g>
</svg>
```

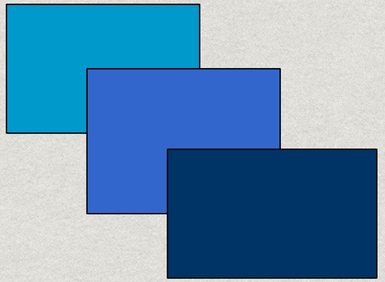
SVG BASIC SYNTAX AND STRUCTURE

SVG Rendering

- Draw a simple text “Hello, World!”

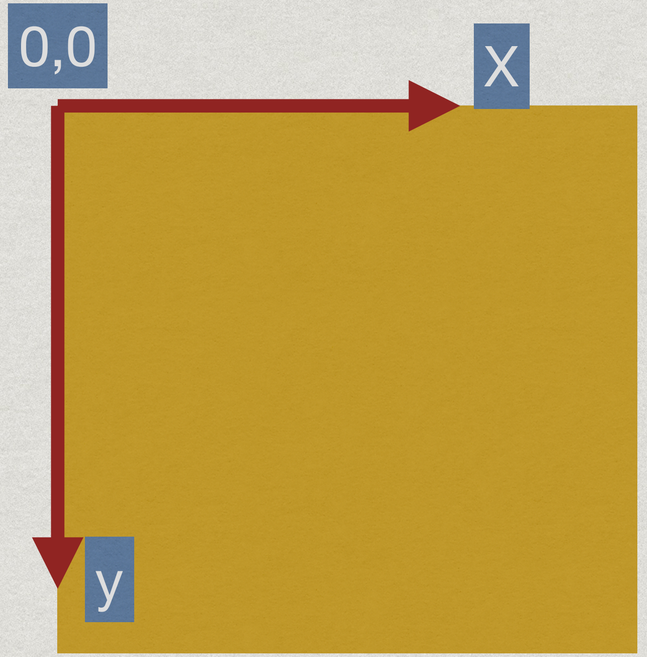
```
<svg width="300" height="300">
    <text x="100" y="100">Hello, World!</text>
</svg>
```

- SVG uses a "painters model" of rendering:
  - **Rendering order: First Comes First Painted**

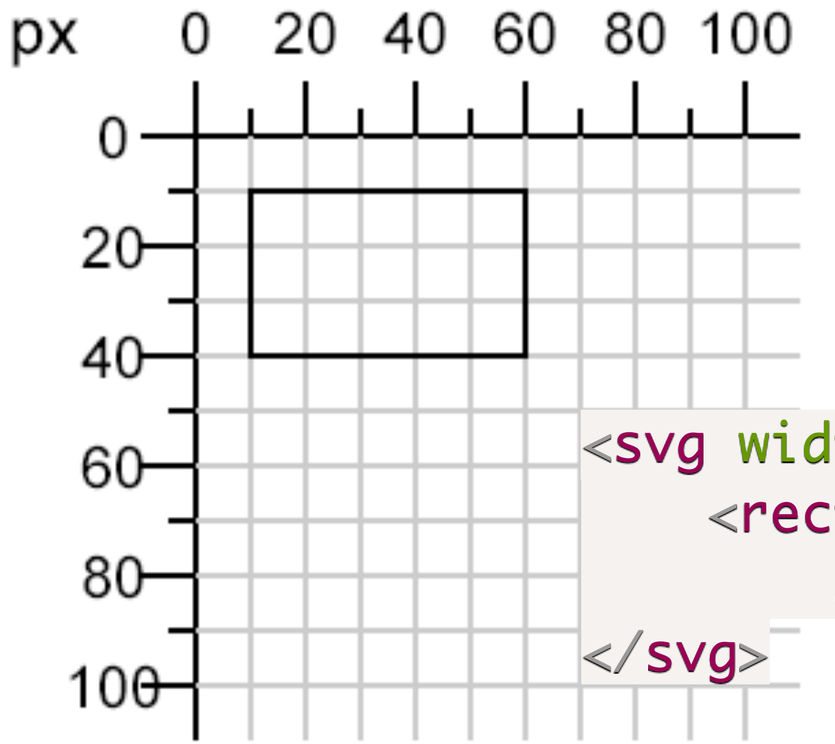


SVG Coordinate Systems

- SVG documents have **their origin at the top left**
- Major supported units of measure:
  - pixels (px)
  - percent (%)

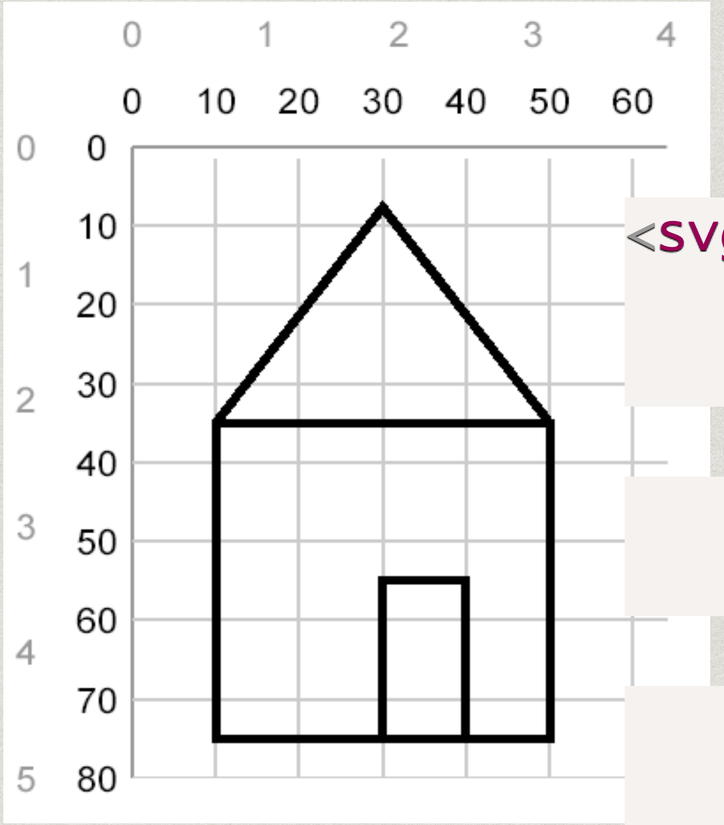


Examples




```svg
<svg width="200" height="200">
    <rect x="10" y="10" width="50" height="30"
        style="stroke: black; fill: none;" />
</svg>
```

Examples

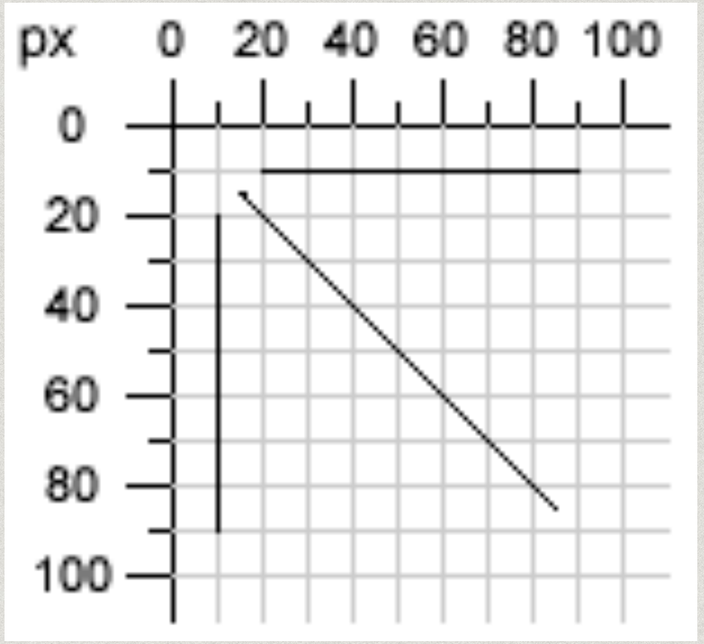



```
<svg width="200" height="200">
    <rect x="10" y="35" width="40" height="40"
        style="stroke:black; fill: none;" />

    <polyline points="10 35, 30 7.68, 50 35"
        style="stroke:black; fill: none;" />

    <polyline points="30 75, 30 55, 40 55, 40 75"
        style="stroke:black; fill: none;" />
</svg>
```

Basic Shapes supported

- All **basic shapes** in SVG are represented as **elements**, with attributes corresponding to the shape.
  - <rect>, <circle>, <ellipse>, <line>.
- <polyline> - defines a set of connected straight lines.
- <polygon> - defines **a closed shape** consisting of a set of connected straight line segments.
- Each shape can be **stroked, filled and transformed**.

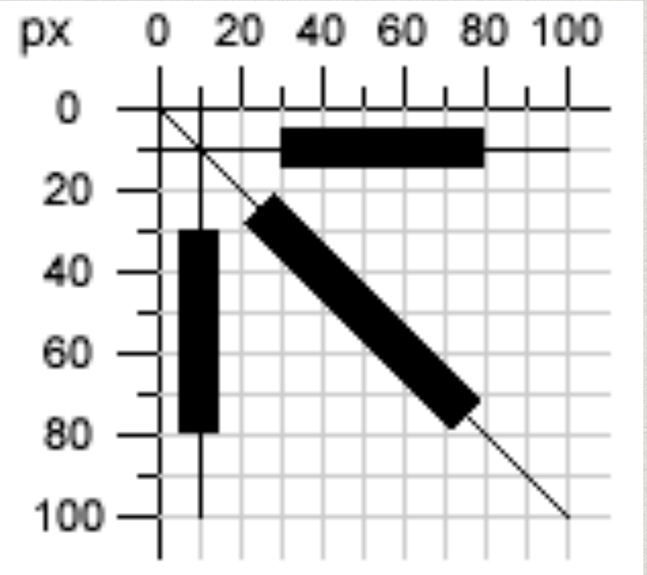
Line




```
<svg width="200" height="200">

    <!-- horizontal line -->
    <line x1="20" y1="10" x2="90" y2="10" style="stroke: black;" />

    <!-- vertical line -->
    <line x1="10" y1="20" x2="10" y2="90" style="stroke: black;" />

    <!-- diagonal line -->
    <line x1="15" y1="15" x2="85" y2="85" style="stroke: black;" />

</svg>
```

Line stroke-width

```
    <!-- horizontal line -->
    <line x1="30" y1="10" x2="80" y2="10"
        style="stroke-width: 10; stroke: black;" />

    <!-- vertical line -->
    <line x1="10" y1="30" x2="10" y2="80"
        style="stroke-width: 10; stroke: black;" />

    <!-- diagonal line -->
    <line x1="25" y1="25" x2="75" y2="75"
        style="stroke-width: 10; stroke: black;" />
```



Line stroke-color

```
<svg width="200" height="200">

    <!-- red -->
    <line x1="10" y1="10" x2="50" y2="10" style="stroke: red; stroke-width: 5;" />

    <!-- light green -->
    <line x1="10" y1="20" x2="50" y2="20" style="stroke: #9f9; stroke-width: 5;" />

    <!-- light blue -->
    <line x1="10" y1="30" x2="50" y2="30" style="stroke: #9999ff; stroke-width: 5;" />

    <!-- medium orange -->
    <line x1="10" y1="40" x2="50" y2="40"
                                style="stroke: rgb(255, 128, 64); stroke-width: 5;" />

    <!-- deep purple -->
    <line x1="10" y1="50" x2="50" y2="50"
                               style="stroke: rgb(60%, 20%, 60%); stroke-width: 5;" />

</svg>
```

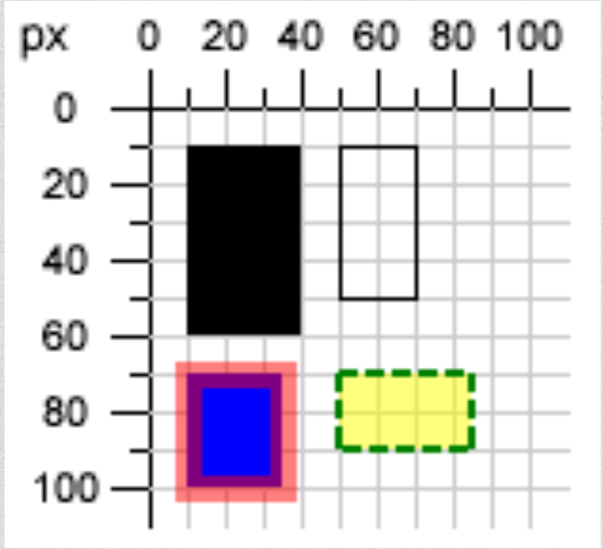
Control the Opacity

```
<svg width="200" height="200">
    <line x1="10" y1="10" x2="50" y2="10"
        style="stroke-opacity: 0.2; stroke: black; stroke-width: 5;" />

    <line x1="10" y1="20" x2="50" y2="20"
        style="stroke-opacity: 0.4; stroke: black; stroke-width: 5;" />

    <line x1="10" y1="30" x2="50" y2="30"
        style="stroke-opacity: 0.6; stroke: black; stroke-width: 5;" />

    <line x1="10" y1="40" x2="50" y2="40"
        style="stroke-opacity: 0.8; stroke: black; stroke-width: 5;" />

    <line x1="10" y1="50" x2="50" y2="50"
        style="stroke-opacity: 1.0; stroke: black; stroke-width: 5;" />
</svg>
```

Dot Lines

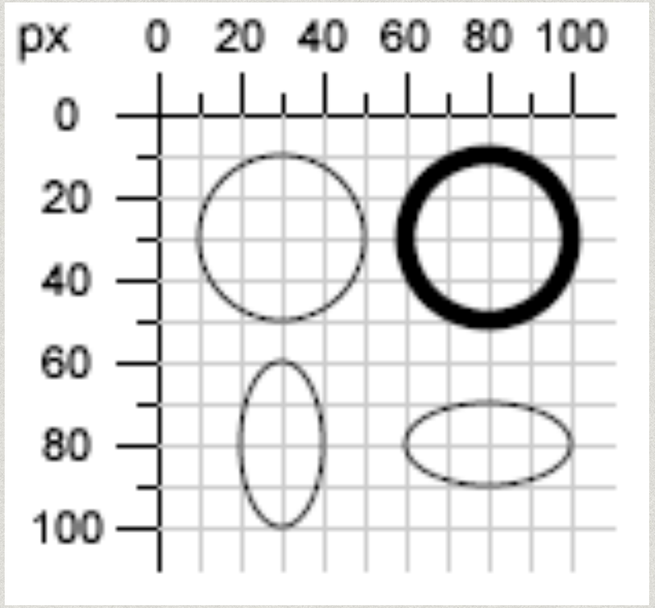
```
<svg width="200" height="200">
    <line x1="10" y1="10" x2="100" y2="10"
        style="stroke-dasharray: 9, 5; stroke: black; stroke-width: 2;" />

    <line x1="10" y1="20" x2="100" y2="20"
        style="stroke-dasharray: 5, 3, 9, 2; stroke: black; stroke-width: 2;" />

    <line x1="10" y1="30" x2="100" y2="30"
        style="stroke-dasharray: 9, 3, 5; stroke: black; stroke-width: 2;" />
</svg>
```

Rectangles




```
<svg width="200" height="200">
    <rect x="10" y="10" width="30" height="50" />

    <rect x="50" y="10" width="20" height="40"
        style="fill: none; stroke: black;" />

    <rect x="10" y="70" width="25" height="30"
        style="fill: #0000ff; stroke: red; stroke-width: 7; stroke-opacity: 0.5;" />

    <rect x="50" y="70" width="35" height="20"
        style="fill: yellow; fill-opacity: 0.5;
            stroke: green; stroke-width: 2; stroke-dasharray: 5 2" />
</svg>
```

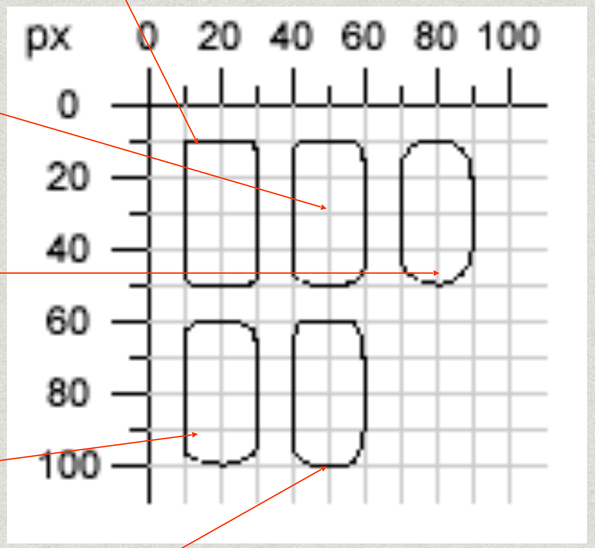
Circles and Ellipses



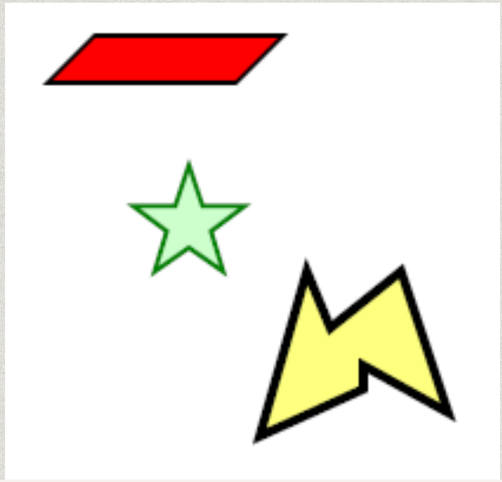

```
<svg width="200px" height="200px">
    <circle cx="30" cy="30" r="20" style="stroke: black; fill: none;"/>

    <circle cx="80" cy="30" r="20"
                    style="stroke-width: 5; stroke: black; fill: none;"/>

    <ellipse cx="30" cy="80" rx="10" ry="20" style="stroke: black; fill: none;"/>

    <ellipse cx="80" cy="80" rx="20" ry="10" style="stroke: black; fill: none;"/>
</svg>
```

Rectangles with rounded corners

```
<svg width="200px" height="200px">
    <rect x="10" y="10" width="20" height="40" rx="2" ry="2"
                        style="stroke: black; fill: none;"/>

    <rect x="40" y="10" width="20" height="40" rx="5"
                        style="stroke: black; fill: none;"/>

    <rect x="70" y="10" width="20" height="40" ry="10"
                        style="stroke: black; fill: none;"/>

    <rect x="10" y="60" width="20" height="40" rx="10" ry="5"
                        style="stroke: black; fill: none;"/>

    <rect x="40" y="60" width="20" height="40" rx="5" ry="10"
                        style="stroke: black; fill: none;"/>
</svg>
```



Polygon



```
<svg width="200px" height="200px">

    <!-- parallelogram -->
    <polygon points="15,10 55,10 45,20 5,20" style="fill: red; stroke: black;" />

    <!-- star -->
    <polygon points="35,37.5 37.9,46.1 46.9,46.1 39.7,51.5 42.3,60.1
       35,55 27.7,60.1 30.3,51.5 23.1,46.1 32.1,46.1"
        style="fill: #ccffcc; stroke: green;" />

    <!-- weird shape -->
    <polygon points="60 60, 65 72, 80 60, 90 90, 72 80, 72 85, 50 95"
        style="fill: yellow; fill-opacity: 0.5; stroke: black; stroke-width: 2;" />

</svg>
```

Star with crossing lines

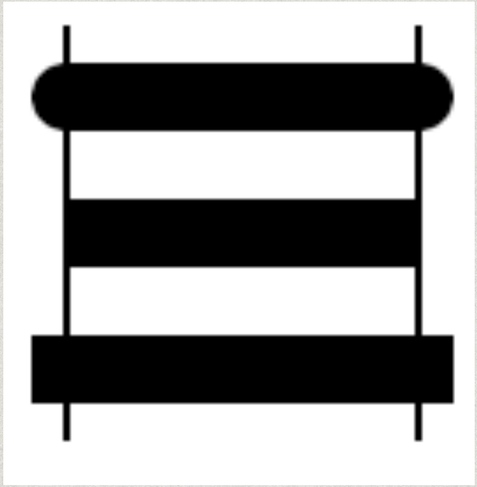
```
<svg width="200px" height="200px">

    <polygon points="48,16  16,96  96,48  0,48  80,96"
             style="stroke: black; fill: none;" />

</svg>
```

Polyline

```
<svg width="200px" height="200px">

    <polyline points="5 20, 20 20, 25 10, 35 30, 45 10,
                      55 30, 65 10, 75 30, 80 20, 95 20"
              style="stroke: black; stroke-width: 3; fill: none;" />

</svg>
```

Line Caps

```
<svg width="200px" height="200px">

    <line x1="30" y1="10" x2="80" y2="10"
        style="stroke-linecap: round; stroke-width: 10; stroke: black;" />

    <line x1="30" y1="30" x2="80" y2="30"
        style="stroke-linecap: butt; stroke-width: 10; stroke: black;" />

    <line x1="30" y1="50" x2="80" y2="50"
        style="stroke-linecap: square; stroke-width: 10; stroke: black;" />

    <line x1="30" y1="0" x2="30" y2="60"
        style="stroke-width: 1; stroke: black;" />

    <line x1="80" y1="0" x2="80" y2="60"
        style="stroke-width: 1; stroke: black;" />

</svg>
```

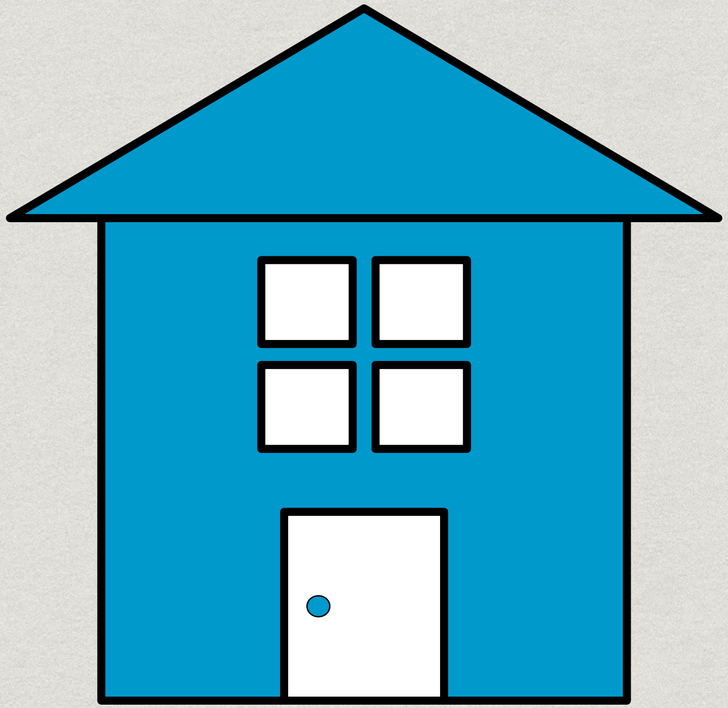


Line Joins

```
<svg width="200" height="200">

    <polyline style="stroke-linejoin: miter; stroke: black; stroke-width: 12; fill: none;"
        points="30 30, 45 15, 60 30" />

    <polyline style="stroke-linejoin: round; stroke: black; stroke-width: 12; fill: none;"
        points="90 30, 105 15, 120 30" />

    <polyline style="stroke-linejoin: bevel; stroke-width: 12; stroke: black; fill: none;"
        points="150 30, 165 15, 180 30" />

</svg>
```

Activity

- Use **rectangles**, **polygon** and **lines** element to draw the image
  - Time: **< 25mins**



D3.JS

http://annatedata.co.uk/articles/d3/whatisd3/

D3.js

- **D3 is a JavaScript library** for building **custom data visualizations**.
  - It's behind many of the groundbreaking visualizations seen in **leading publications**.
- However it **isn't a charting library** like Charts.js.
  - Rather than providing ready built charts, it provides **building blocks** for charts.

Building Blocks - Scale

- Scale Functions

```
// Create a scale function for population
var populationScale = d3.scaleLinear().domain([0, 1367030000]).range([0, 600]);

populationScale(1367030000); // returns 600  (China)
populationScale(64105654); // returns 28.13  (UK)
```

- Color could also be scaled

```
var colorScale = d3.scaleLinear().domain([0, 10]).range(['#ddd', 'red']);

myColourScale( 0 ); // returns '#dddddd' (grey)
myColourScale( 50 ); // returns '#ee6f6f' (pale red)
myColourScale( 100 ); // returns '#ff0000' (red)
```

Building Blocks - Selection

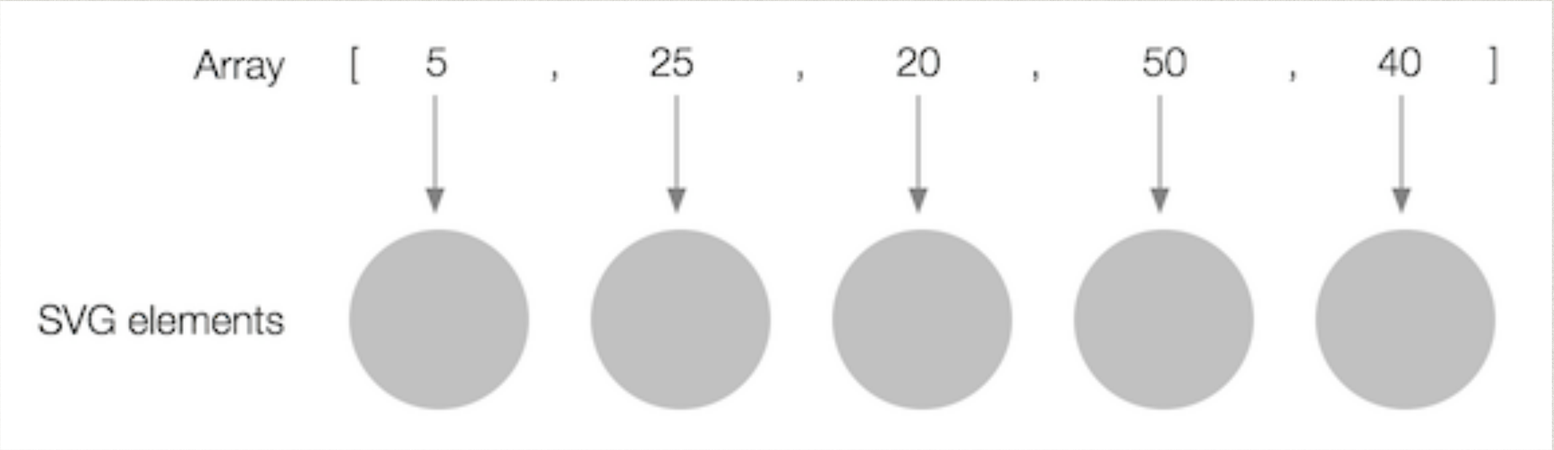
```
d3.select('body');
d3.select('#my-chart');
d3.selectAll('.bars');
```

- Selection
  - Making a selection, takes a **CSS3 selector** as input.
- Manipulation

Function	Description	Example
.style()	Add/modify CSS style declaration	d3.selectAll(‘circle’).style(‘fill’, ‘blue’);
.attr()	Add/modify element attributes	d3.selectAll(‘circle’).attr(‘r’, ‘100’);
.classed()	Add/remove a CSS class	d3.select(‘.tooltip’).classed(‘visible’, true);
.text()	Set an element’s text content	d3.select(‘.tooltip’).text(‘Dave’);
.html()	Set an element’s HTML content	d3.select(‘.tooltip’).html(‘<h1>Dave<\h1>’);

Building Blocks - Data Joins

- Given an **array of data** and a **D3 selection of HTML or SVG elements**, we can attach or **‘join’ each array element to each element of the selection**.



Building Blocks - Data Joins

- There are two functions in D3 which **join data to a d3 selection**:
  - **datum()** and **data()**.

```
var s = d3.selectAll('circle');
s.datum(100);

// The first circle will be assigned with data value 100
```

```
var s = d3.selectAll('circle');
s.data([5, 25, 20, 50, 40]);

s.attr('r', function(d) {
  return d;
});
```

Interaction and Tooltip

- A **D3 selection of HTML or SVG elements** could listen to particular events.

```
circles
    .on("mouseover", function () {
        tooltip.style("display", "block");
    })
    .on("mouseout", function () {
        tooltip.style("display", "none");
    })
    .on("mousemove", function (d) {
        tooltip
            .style("left", Math.max(0, d3.event.pageX + 20) + "px")
            .style("top", (d3.event.pageY + 20) + "px")
            .text(d.district);
    });
```