

Visualization on the Web

SVG

SCALABLE VECTOR GRAPHICS

Introducing SVG

- Descriptive tags for images
- Based on vector graphics
- D3.js can manage the creation and modification of tags

SVG External Resources

- SVG Specification
 - <http://www.w3.org/TR/SVG/>
- Mozilla Developer Network
 - <https://developer.mozilla.org/en/SVG>
- D3.js API Reference
 - <https://github.com/mbostock/d3/wiki/SVG-Shapes>

ObservableHQ Notebook available

- <https://observablehq.com/collection/@rinziv/va602aa>

Hello World Example

```
<!DOCTYPE html>
```

```
<meta charset="utf-8">
```

```
[...]
```

```
<svg width="960" height="500">
```

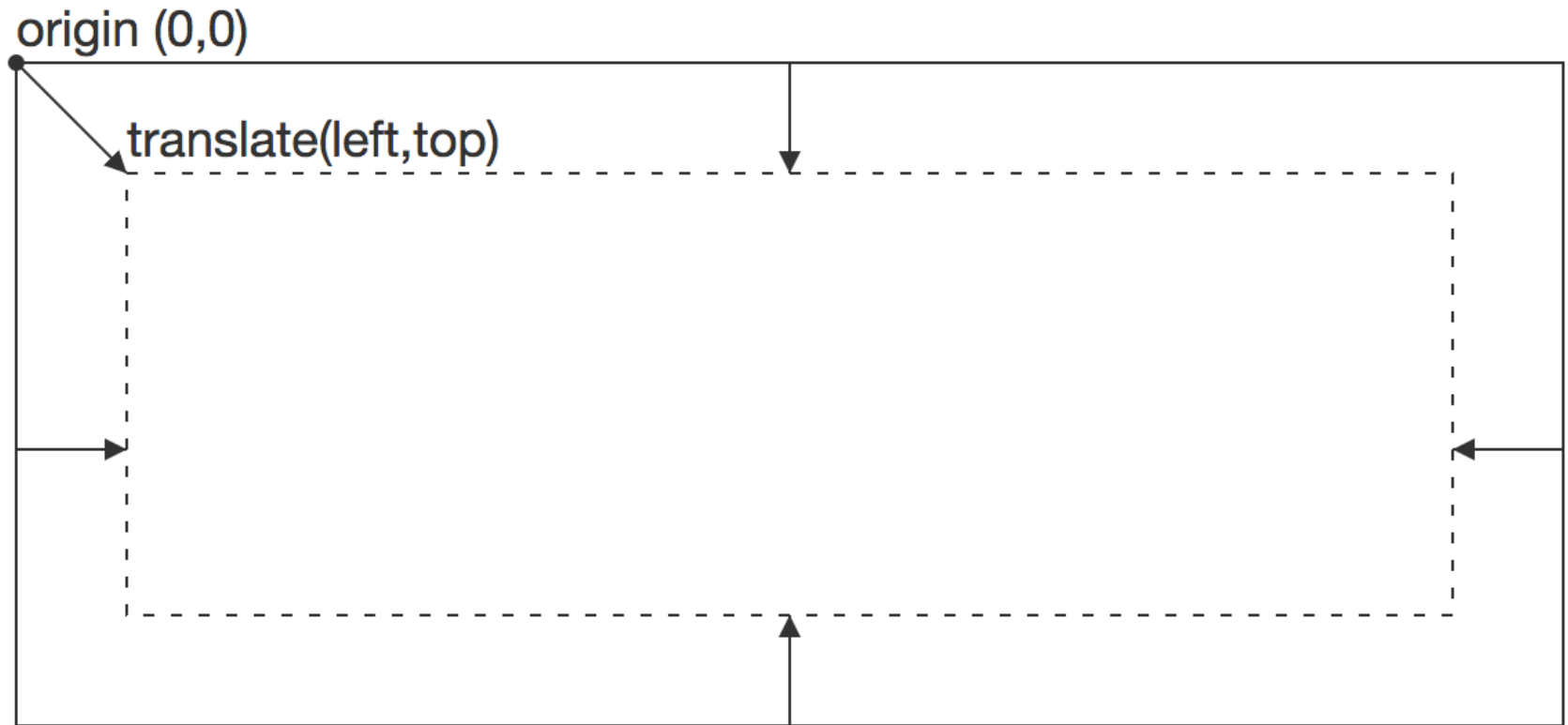
```
  <circle cx="480" cy="250">
```

```
  </circle>
```

```
</svg>
```

```
[...]
```

Coordinate System



SVG – BASIC SHAPES

Rectangle

```
<svg>  
  <rect width="200" height="100" fill="#BBC42A" />  
</svg>
```



Circle

```
<svg>  
  <circle cx="75" cy="75" r="75" fill="#ED6E46" />  
</svg>
```



Ellipse

```
<svg>  
  <ellipse cx="100" cy="100" rx="100" ry="50"  
  fill="#7AA20D" />  
</svg>
```



Line

```
<svg>  
  <line x1="5" y1="5" x2="100" y2="100"  
stroke="#765373" stroke-width="8"/>  
</svg>
```



Polyline

```
<svg>  
  <polyline points="0,40 40,40 40,80 80,80 80,120 120,120 120,160"  
  fill="white" stroke="#BBC42A" stroke-width="6" />  
</svg>
```



Polygon

```
<svg>  
  <polygon points="50,5 100,5 125,30 125,80 100,105  
50,105 25,80 25,30" fill="#ED6E46" />  
</svg>
```



Path

```
<svg width="258px" height="184px">  
  <path fill="#7AA20D" stroke="#7AA20D" stroke-width="9" stroke-  
linejoin="round"  
d="M248.761,92c0,9.801-7.93,17.731-17.71,17.731c-0.319,0-0.617,0-0.935-0  
.021c-10.035,37.291-51.174,65.206-100.414,65.206  
c-49.261,0-90.443-27.979-100.435-65.334c-0.765,0.106-1.531,0.149-2.317,0  
.149c-9.78,0-17.71-7.93-17.71-17.731  
c0-9.78,7.93-17.71,17.71-17.71c0.787,0,1.552,0.042,2.317,0.149C39.238,37  
.084,80.419,9.083,129.702,9.083  
c49.24,0,90.379,27.937,100.414,65.228h0.021c0.298-0.021,0.617-0.021,0.91  
4-0.021C240.831,74.29,248.761,82.22,248.761,92z" />  
</svg>
```



Path specifications

- Definition of a path is done within a path element
 - `<path d="{specifics of the path}" />`
- The specifics of path are instructions to move a virtual pen over the graphics
 - **Move to** (M or m). Go to coordinates lifting the pen, without a trace
 - **Line to** (L or l). Draw a line from the last point to the new coordinates
 - **Vertical or Horizontal lines** (H or h, V or v). Draw a line parallel to one of the axis
 - **Close path** (Z or z)

Path commands – Uppercase vs lowercase commands

- An uppercase letter indicates absolute coordinates will follow
- A lowercase letter indicates a relative coordinate

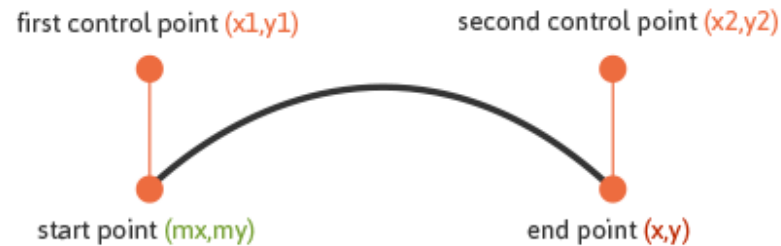
Path – Cubic Bezier

```
<svg>
```

```
  <path fill="none" stroke="#333333" stroke-width="3"  
d="M10,55 C10,5 100,5 100,55" />
```

```
</svg>
```

Cubic Bézier Curve



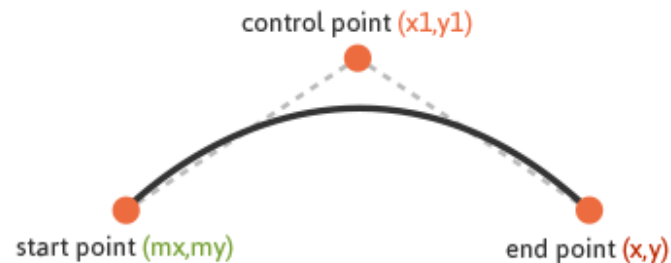
Path – Quadratic bezier Curve

```
<svg>
```

```
  <path fill="none" stroke="#333333" stroke-width="3"  
d="M20,50 Q40,5 100,50" />
```

```
</svg>
```

Quadratic Bézier Curve



Example - Stairways with path

```
<!--  
  Stairways example using path  
  -->  
  <svg width="200" height="200">  
    <path d="M0,40 L40,40 L40,80 L80,80 L80,120 L120,120  
L120,160" fill="white" stroke="#BBC42A" stroke-width="6" />  
  </svg>
```

Example - Stairways with path

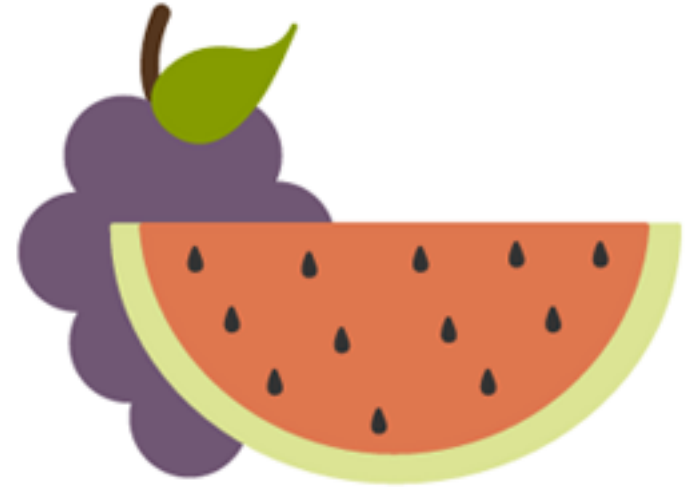
```
<!--  
  Stairways example using path with H and V commands  
  -->  
<svg width="200" height="200">  
  <path d="M0,40 H40 V80 H80 V120 H120 V160"  
fill="white" stroke="#BBC42A" stroke-width="6" />  
</svg>
```

Example - Stairways with path

```
<!--  
    Stairways example using path with relative coordinates  
(h and v)  
-->  
<svg width="200" height="200">  
    <path d="M0,40 h40 v40 h40 v40 h40 v40" fill="white"  
stroke="#BBC42A" stroke-width="6" />  
</svg>
```

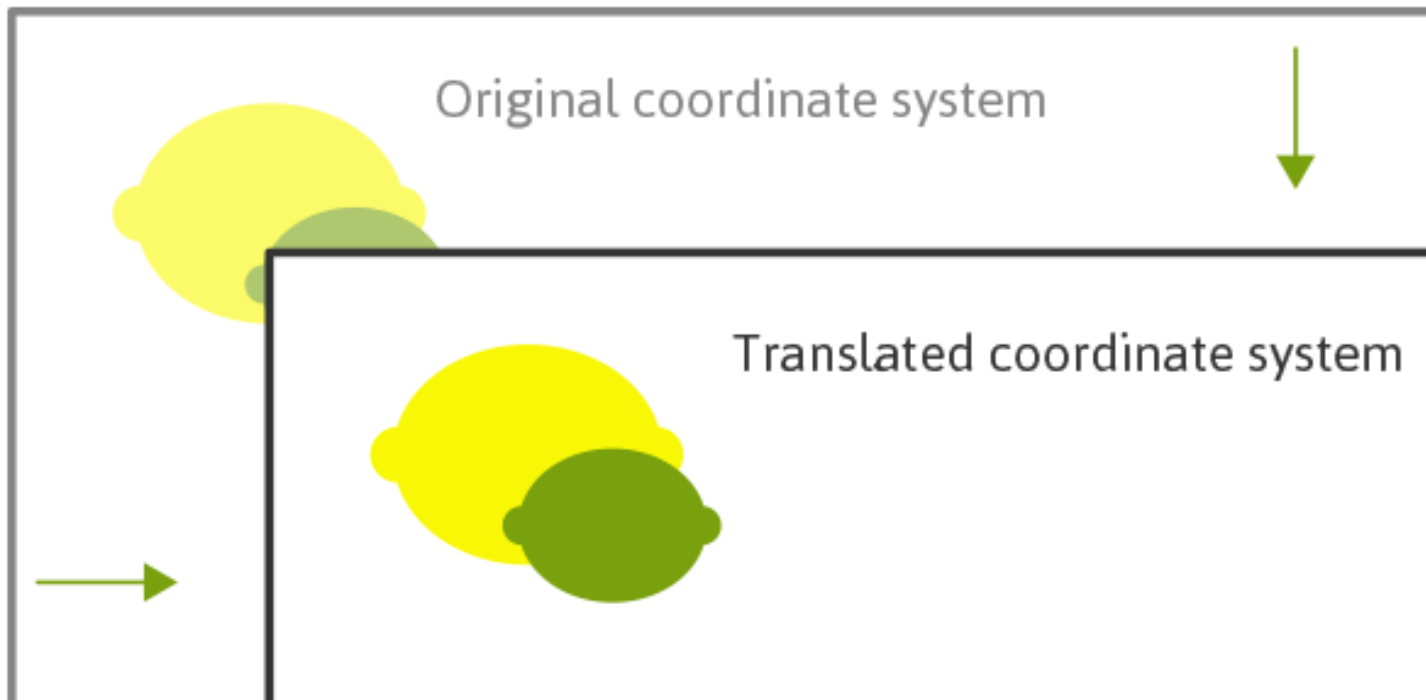
Stacking ordering

```
<svg>  
  <g class="grapes">  
    <!--<path <stem path> />-->  
    <!--<path <grapes path> />-->  
    <!--<path <leaf path> />-->  
  </g>  
  <g class="watermelon">  
    <!--<path <outside path> />-->  
    <!--<path <inside path> />-->  
    <!--<path <seeds path> />-->  
  </g>  
</svg>
```



Coordinate System Transform

`transform="translate(<tx>,<ty>) rotate(<rotation angle>)"`



Transformations

- Translate
 - `transform="translate(<tx>,<ty>)"`
- Rotate
 - `transform="rotate(<rotation angle>)"`
 - `transform=rotate(<rotation angle> [<cx>,<cy>])"`
- Scale
 - `transform="scale(<sx> [<sy>])"`
- Skew
 - `transform="skewX(20)"`

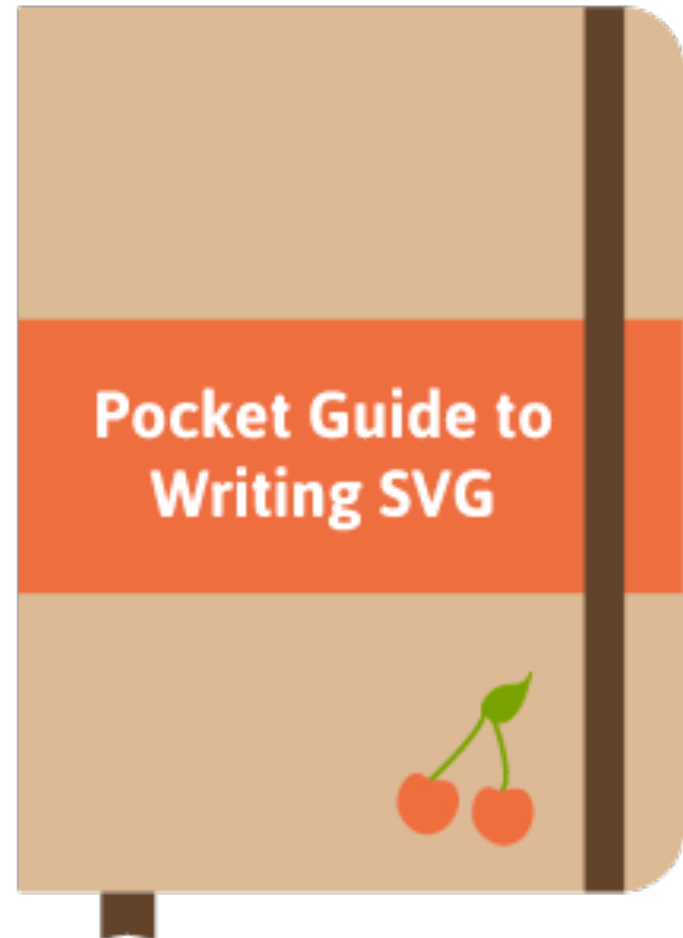
Circle example with translation

```
<!--  
    Draw a circle in the center of the element  
    using relative coordinates after a translation  
-->  
<svg width="200" height="100">  
  <g transform="translate(100,50)">  
    <circle r="50"/>  
    <circle r="20" style="fill:#fdbb84"/>  
  </g>  
</svg>
```

- Live example at:
- <http://jsbin.com/kiwukat/2/edit?html,output>

Pocket Guide to Writing SVG

<http://svgpocketguide.com/book/>





CANVAS ELEMENT

Canvas

- A **canvas** element is a container for raster graphics
- Within the canvas, a **context** provide the functions to draw visual elements
- Two different context types:
 - “2d”
 - “webgl”

Canvas - Example

<p>Before canvas.</p>

```
<canvas width="120" height="60"></canvas>
```

<p>After canvas.</p>

```
<script>
```

```
    var canvas =  
document.querySelector("canvas");  
    var context = canvas.getContext("2d");  
    context.fillStyle = "red";  
    context.fillRect(10, 10, 100, 50);  
</script>
```

Canvas - Path

```
<canvas></canvas>
<script>
  var cx =
document.querySelector("canvas").getContext("2d");
  cx.beginPath();
  for (var y = 10; y < 100; y += 10) {
    cx.moveTo(10, y);
    cx.lineTo(90, y);
  }
  cx.stroke();
</script>
```

Canvas - Curves

```
<canvas></canvas>
<script>
  var cx =
document.querySelector("canvas").getContext("2d");
  cx.beginPath();
  cx.moveTo(10, 90);
  // control=(60,10) goal=(90,90)
  cx.quadraticCurveTo(60, 10, 90, 90);
  cx.lineTo(60, 10);
  cx.closePath();
  cx.stroke();
</script>
```

Canvas - Curves

```
<canvas></canvas>
<script>
  var cx =
document.querySelector("canvas").getContext("2d");
  cx.beginPath();
  cx.moveTo(10, 90);
  // control1=(10,10) control2=(90,10) goal=(50,90)
  cx.bezierCurveTo(10, 10, 90, 10, 50, 90);
  cx.lineTo(90, 10);
  cx.lineTo(10, 10);
  cx.closePath();
  cx.stroke();
</script>
```