Data Visualisation

Week 3 – D3 Basics

Basic Template

```
<!DOCTYPE  html>
<html>
  <head>
    <meta  charset="utf-8">
  </head>
  <body>
    <svg id="..."></svg>
    <script  src="http://d3js.org/d3.v3.min.js"></script>
    <script>
    // Write your visualization here.
    </script>
  </body>
</html>
```

SVG for drawing visualisation

Load D3 library (must before your code)

SVG: Where You Visualise

- Two way to create:
  - HTML
    - Write a HTML tag: `<svg width="700" height="400"></svg>`
    - Select in JavaScript: `var svg = d3.select("svg");`
  - JavaScript
    - ```
      var svg = d3.select("body").append("svg")
                .attr("width",  700)
                .attr("height", 400);
      ```
      ```

Imperative vs. Declarative Programming

- Imperative (step-by-step) – Traditional

```
for (var i = 0; i < data.length; i++) {
    // create a circle
    // set radius
    // do other things…
}
```

- Declarative Programming (describe what you want) – D3
  - I have these data points […]
  - Please create a circle for each data point
  - For each circle, set radius as X…

Data

- What is the data in your visualisation?
- For example, in the Exercise 1:
  - `[44, 47, 48, 49, 52, 57, 61, 63, 65, 67]`
- In Exercise 2: The CSV file

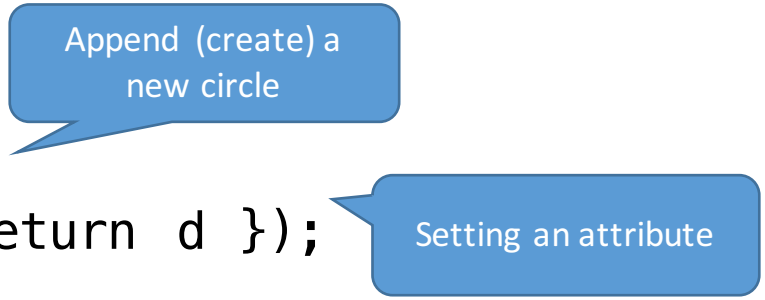
D3 Pipelines: Enter, Update, Exit

- Pass data points to D3 using `.data(…)`
- D3 will identify the differences between the last dataset (if exists) and the new data.
- Three possible outcomes:
  - For new data point: Run Enter -> then Update
  - For existing data point: Run Update
  - For redundant data point: Run Exit

Example:

- Pass data:
  - `var circles = svg.selectAll("circle").data(data);`

- Enter pipeline:
  - `circles.enter().append("circle")`
    `.attr("r", function (d) { return d });`



- In general, attributes that will **NOT** be changed in the lifetime of the visualisation will live in the Enter pipeline.

Setting Attributes

- Fix value
  - `.attr("y", "100");`

- Dynamic value (depends on the data point):
  - `.attr("r", function (d) { return d });`

Example:

- Update pipeline:

```
circles
  .attr("cx", function (d) {
     var n = cx; cx += d; return n; })
  .attr("cy", "200")
  .attr("fill", "lightgreen")
  .attr("stroke", "green");
```

- In general, attributes that will be changed in the lifetime of the visualisation will live in the Update pipeline.

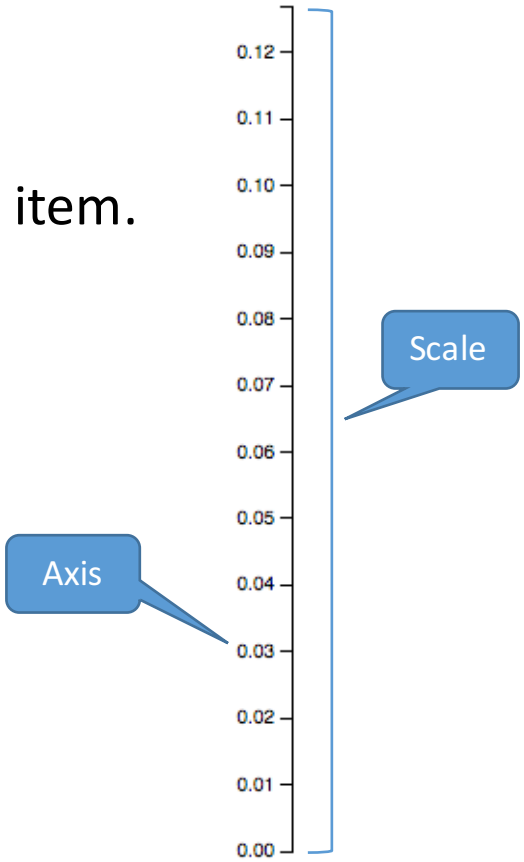
Example:

- Exit pipeline:

```
circles.exit().remove();
```

Drawing Axis

- Scale: Use to calculate the space between each item.
- Axis: Draw actual text labels.



Drawing Axis

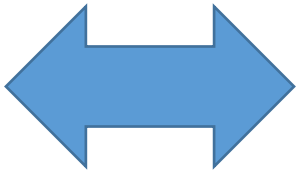
```
var x = d3.scale.ordinal()
    .rangeRoundBands([0, width], 0.1);
var y = d3.scale.linear()
    .range([height, 0]);
var xAxis = d3.svg.axis()
    .scale(x)
    .orient("bottom");
var yAxis = d3.svg.axis()
    .scale(y)
    .orient("left");
```

Load Remote CSV


```
d3.csv("data.csv", type, function (error, data) {
    // Draw visualisation when the CSV is loaded.
}
```

data.csv

```
name,value
A,.08167
B,.01492
C,.02782
D,.04253
E,.12702
```



name	value
A	.08167
B	.01492
C	.02782
D	.04253
E	.12702

Converting Data Type

- Everything loaded from the CSV file is text (string).
- We need to convert it to numbers.

```
function type(d) {
  d.value = +d.value;
  return d;
}
```

After Loading File

- Assign text labels to axis.

```
// Pass the first column as X axis labels.
x.domain(data.map(function (d) { return d.name; }));
// Pass the second column as Y axis.
y.domain([0, d3.max(data, function (d) { return d.value;
})]);
```

After Loading File

- Draw both X and Y axes.

```
// Draw the X axis.
chart.append("g")
     .attr("class", "axis")
     .attr("transform", "translate(0," + height + ")")
     .call(xAxis);

// Draw the Y axis.
chart.append("g")
     .attr("class", "axis")
     .call(yAxis);
```

After Loading File

- Use data to draw bars – like drawing circles in Exercise 1.

```
chart.selectAll("rect")
          .data(data)
       .enter().append("rect")
          .attr("x", function (d) { return x(d.name); })
          .attr("y", function (d) { return y(d.value); })
          .attr("height", function (d) { return height - y(d.value); })
          .attr("width", x.rangeBand())
          .attr("fill", function (d, i) {
              return i % 2 ? "steelblue" : "lightpink"});
```