Data-Driven Documents: A JavaScript Library
By Zachary Boulanger

About me
Undergraduate (Class of 2017)
Studying Computer Science, Minors in German and Multidisciplinary Design
Completed the Multidisciplinary Design Program in 2016
Front-End Developer on the ProQuest Search Visualization team
http://zachboulanger.me/

Outline
JS/D3 Introduction
D3 Features
Example Walkthrough

JavaScript - What is it?
Dynamic, weakly-typed scripting language
Can be Object-Oriented, Functional, Procedural, etc
HTML DOM (Document Object Model)
Tree-like structure of a web page
Very important for D3!
http://www.w3schools.com/js/js_htmldom.asp
JavaScript - Why/How is it used?

Mainly used in Web Dev, also in App Dev

Mainly Client-Side, also Server-Side

Very powerful and widely used, but need to be careful
  Security, load times, etc very important to think about

Many frameworks and libraries!
  jQuery, Node.js, Bootstrap, D3, etc

D3.js - According to https://d3js.org/

JavaScript Library combining:
  powerful visualization components
  a data-driven approach to DOM manipulation

Extremely fast

Supports large data sets and dynamic behaviors for interaction/animation

Basically, allows you to easily and dynamically create HTML, SVGs, and CSS to fit your data

D3.js - What can it do?

Create charts and visualizations based on your data
  Anything from simple html tables to complex interactive zooming bubble charts with SVGs

Allows you to create many similar HTML structures at once
  E.g. create many divs with the same CSS class to house your content/data
  Similar to creating a function to contain code that you reuse a lot

D3.js - Examples
Many other examples on the site!

All show the (possibly interactive) example and code

See more here:


---

**Example Walkthrough - basics**

"Normal" Javascript:

```javascript
var paragraphs = document.getElementsByTagName("p");
for (var i = 0; i < paragraphs.length; i++) {
    var paragraph = paragraphs.item(i);
    paragraph.style.setProperty("color", "white", null);
}
```

D3 Equivalent:

```javascript
d3.selectAll("p").style("color", "white");
```

---

**Example Walkthrough - basics**

This code:

1. Selects the "body" html tag
2. Selects all "p" tags in body (if there are any)
3. Defines the data points (array of ints)
4. Applies this text value to each p tag, creating more p tags if there aren't enough.
I highly suggest getting the walkthrough code from the website:
http://bl.ocks.org/dbuezas/9306799

And then following along with the tutorial as I explain, that way you don’t have to try and write it all out while also listening.

1. Create a new file: index.html
   a. This can be done by opening your text editor of choice, e.g. notepad
   b. Make sure to save as .html not .txt

2. Add in this basic html webpage template
   a. Mostly just template stuff for now
   b. We add in our randomize button, the only HTML we’ll hardcode
   c. We also add in the D3 js library to use

3. Note: This uses version 3 of D3.
   a. They have just released version 4, and while some things are different, v3 still exists and this example will still work

```html
<!DOCTYPE html>

<body>

<button class="randomize">randomize</button>

<script src="http://d3js.org/d3.v3.min.js"></script>

</body>
```

1. Add this code between the script tags:
2. The first chunk goes into the body tag and adds in an SVG object, then a g object within that.
   a. Note: D3 will still execute the code, even though it’s in a variable. The variable allows us to save our spot, like a bookmark, so that our later code can reference this area.
3. We then add in three more g objects and give each of them a class.
   a. If you were to look at our site now, you would see nothing, but could view source and see the objects we created as html.
Example Walkthrough - Animated Pie Chart

1. Underneath our last code, add this:

```javascript
<script>
...<br><br>
var width = 960;<br>height = 450;<br>radius = Math.min(width, height) / 2;<br><br>var pie = d3.layout.pie().sort(null).value(function(d) { return d.value; });
</script>
```

2. The first chunk is just some variable declaration for later.

3. The next part allows us to create the “pie”:
   a. D3.layout contains all the pre-made charts and such, e.g. pie, pack, tree, etc
   b. We save our place again with a variable for reference later
   c. We override the default value function to match the way our data will be set up (more info on this later)

```javascript
var arc = d3.svg.arc().outerRadius(radius * 0.8).innerRadius(radius * 0.4);
var outerArc = d3.svg.arc().innerRadius(radius * 0.9).outerRadius(radius * 0.9);
```

```javascript
svg.attr("transform", "translate(" + width / 2 + "," + height / 2 + ")");
```

```javascript
function randomData() {
  var labels = color.domain();
  return labels.map(function(label) {
    return { label: label, value: Math.random() }
  });
}<script>

function change(data) {
  /* ------- PIE SLICES -------*/
  var slice = svg.select("<paths>").selectAll("path.slice")
    .data(pie(data), key);
    slice.enter().insert("path")
      .attr("fill", function(d) { return color(d.data.label); })
      .attr("class", "slice");
    slice.transition().duration(1000)
      .attrTween("d", function(d) {
        return arc bloggers(interpolate(d));
      });
  slice.exit().remove();
</script>
```
Example Walkthrough - Animated Pie Chart

function midAngle(d) {
  return d.startAngle + (d.endAngle - d.startAngle)/2;
}

/* ------ TEXT LABELS ------*/
var text =
  svg.select(".labels").selectAll("text")
    .data(pie(data), key);

  text.enter()
    .append("text")
    .attr("dy", ".35em")
    .text(function(d) {
      return d.data.label;
    });

  function midAngle(d){
    return d.startAngle + (d.endAngle - d.startAngle)/2;
  }

  text.transition().duration(1000)
    .attrTween("transform", function(d) {
      this._current = this._current || d;
      var interpolate =
        d3.interpolate(this._current, d);
      this._current = interpolate(0);
      return function(t) {
        var d2 = interpolate(t);
        var pos = outerArc.centroid(d2);
        var pos = outArc.centroid(d2);
        pos[0] = radius * 0.95 * 
          (midAngle(d2) < Math.PI ? 1 : -1);
        return [arc.centroid(d2),
          outerArc.centroid(d2), pos];
      };
    };

  text.exit()
    .remove();

Example Walkthrough - Animated Pie Chart

/* ------ SLICE TO TEXT POLYLINES ------*/
var polyline =
  svg.select(".lines").selectAll("polyline")
    .data(pie(data), key);

  polyline.enter()
    .append("polyline")
    .attr("points", function(d) {
      return d.data.label;
    });

  return function(t) {
    var d2 = interpolate(t);
    var pos = outArc.centroid(d2);
    var pos = outArc.centroid(d2);
    pos[0] = radius * 0.95 * 
      (midAngle(d2) < Math.PI ? 1 : -1);
    return [arc.centroid(d2),
      outerArc.centroid(d2), pos];
  };

  polyline.exit()
    .remove();

Example Walkthrough - Animated Pie Chart

Example Walkthrough - Animated Pie Chart

That doesn't look right...

We need CSS!
Example Walkthrough - Animated Pie Chart

Thank you!

Any Questions?