Scott Murray
code artist
Assistant Professor, Design
University of San Francisco
University of San Francisco's primary visual identification uses the university's full name spelled out. This logo should be used as the first identification where appropriate in all external communications, whether print, web, or electronic materials, and merchandising items.

Note: Use the tagline logo with discretion as to context. Do not use the tagline logo if "Change the world from here" has an unintended associated meaning, for instance on a student comedy night poster or a bookstore promotion, in which case use the full name logo without tagline, see next page.

The Full Name Logo with Tagline 2-Lines is the recommended identification. Use this version whenever possible.

Alternates: Use the Full Name Logo with Tagline 1-Line for applications where space parameters dictates an extreme horizontal format.

The Full Name Centered Logo with Tagline is advisable for vertical or square formats. All Full Name Logos are available as 3-color logo—USF Green, USF Yellow, and USF Gray; 2-color logo—USF Green and USF Yellow; or one-color logo—black and white.
Rue de Paris, temps de pluie

*Paris Street, Rainy Day*

Gustave Caillebotte, 1877
void setup() {
    size(1920, 1080);
    frameRate(30);
    smooth();
}

void draw() {
    translate(width/2, height/2);
    for (int i = 0; i < 100; i++) {
        ellipse(0, 0, i * 5, i * 5);
    }
}
Rotating Type
with Jan Kubasiewicz, 2011
IBM Smarter Planet

with Mirada, 2011
Smarter business for a Smarter Planet:

**How to build a car fueled by software.**

When you look at the Chevrolet Volt, are you looking at steel and plastic, or are you looking at software? The Volt, an electric car with gas-powered extended range, contains over 10 million lines of code, more software than you’d find in the avionics and navigation systems of a modern fighter jet. Chevrolet turned to IBM to help them design the control typically taken by the driver. Using the Rational platform to design the car allowed engineers around the world to collaborate in software, systems and services.

Let’s build a smarter planet. ibm.com/collaborate
Smarter business for a Smarter Planet:

The cloud that’s transforming an industry, one fish at a time.

At the University of Bari, a new computing model is creating new business models. Using an IBM SmartCloud™ their team built a solution that allows local fishermen to auction their catch while still at sea. By creating more demand for the fishermen's product, the cloud has increased income by 25% while reducing time to market by 70%. Now the team is scaling the solution to create new business models for the winemaking and transportation industries. What can cloud do for your business? A smarter planet is built on smarter software, systems and services.

Let’s build a smarter planet. ibm.com/cloudsolutions
Viégas, Wattenberg, van Ham, Kriss, and McKeon, “Many Eyes: A Site for Visualization at Internet Scale”
Cui, Zhou, Qu, Wong, and Li, “Geometry-Based Edge Clustering for Graph Visualization”
Holten, “Hierarchical Edge Bundles: Visualization of Adjacency Relations in Hierarchical Data”
Phan, Xiao, Yeh, Hanrahan, and Winograd, “Flow Map Layout”
About these tutorials

Last updated 2012 December 30

These tutorials aim to be

- Brief
- Focused, each addressing a single topic
- Modular, so you can reference only the topics relevant to your goals
- Complete, with sample code illustrating each topic
- Dynamic, updated and expanded as needed
- Free, licensed so you can use the code however you wish

Philosophy

These tutorials evolved out of my own process of learning how to use D3. You already know that D3 is an extraordinary tool for mapping data within web pages, written by Mike Bostock. Many people, including myself, come to D3 with backgrounds in design, mapping, and data visualization, but not programming and computer science.

Yet D3 employs advanced JavaScript techniques, so learning to use D3 often means learning a lot about JavaScript. For many datavis folks, D3 is their introduction to JavaScript. It’s hard enough to learn a new programming language, let alone a new tool built on that language. D3 is amazing and powerful.
Interactive Data Visualization for the Web

Scott Murray

March 2013
Introducing D3 for Visualization
Assumptions

— You are somewhat familiar with HTML & CSS
— You have a little programming experience

To follow along, get sample code files from:
https://github.com/alignedleft/crazy-data-circles
What is d3.js?
D3.js is a JavaScript library for manipulating documents based on data. D3 helps you bring data to life using HTML, SVG and CSS. D3’s emphasis on web standards gives you the full capabilities of modern browsers without tying yourself to a proprietary framework, combining powerful visualization components and a data-driven approach to DOM manipulation.
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>D3 Page Template</title>
  <script type="text/javascript" src="d3.v3.js"></script>
</head>
<body>
  <script type="text/javascript">
    // Your beautiful D3 code
    // can go here
  </script>
</body>
</html>
d3 = function() {
  var π = Math.PI, ε = 1e-6, d3 = {
    version: "3.8.6"
  }, d3.radians = π / 180, d3.degrees = 180 / π, d3_document = document, d3_window = window;
  function d3_target(d) {
    return d.target;
  }
  function d3_source(d) {
    return d.source;
  }
  var d3_format_decimalPoint = ".", d3_format_thousandsSeparator = ",", d3_format_grouping = [ 3, 3 ];
  if (!Date.now) Date.now = function() {
    return +new Date();
  };
  try {
    d3_document.createElement("div").style.setProperty("opacity", 0, "");
  } catch (error) {
    var d3_style_prototype = d3_window.CSSStyleDeclaration.prototype, d3_style_setProperty = d3_style_prototype.setProperty;
    d3_style_prototype.setProperty = function(name, value, priority) {
      d3_style_setProperty.call(this, name, value + ",", priority);
    };
  }
  function d3_class(ctor, properties) {
    try {
      for (var key in properties) {
        Object.defineProperty(ctor.prototype, key, {
          value: properties[key],
          enumerable: false
        });
      }
    } catch (e) {
      ctor.prototype = properties;
    }
  }
  var d3_array = d3_arraySlice;
  function d3_arrayCopy(pseudoarray) {
    var i = -1, n = pseudoarray.length, array = [];
    while (++i < n) array.push(pseudoarray[i]);
    return array;
  }
  function d3_arraySlice(pseudoarray) {
    return Array.prototype.slice.call(pseudoarray);
  }
  try {
    d3_array(d3_document.documentElement.childNodes)[0].nodeType;
  } catch (e) {
    d3_array = d3_arrayCopy;
  }
}
<table>
<thead>
<tr>
<th>HTML</th>
<th>Hypertext Markup Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSS</td>
<td>Cascading Style Sheets</td>
</tr>
<tr>
<td>JS</td>
<td>JavaScript</td>
</tr>
<tr>
<td>SVG</td>
<td>Scalable Vector Graphics</td>
</tr>
<tr>
<td>DOM</td>
<td>The Document Object Model</td>
</tr>
</tbody>
</table>

all of the above == web standards
Learning D3 is a process of “learning the web”
Selecting & generating elements
Binding data / data joins
// JavaScript arrays!

var dataset = [5, 10, 20, 15, 18];
// Data joins!

var dataset = [ 5, 10, 20, 15, 18 ];

d3.select("svg").selectAll("circle")
  .data(dataset)
  .enter()
  .append("circle");
/ Data joins!

var dataset = [ 5, 10, 20, 15, 18 ];

d3.select("svg").selectAll("circle")
  .data(dataset)
  .enter()
  .append("circle");
// Data joins!

var dataset = [ 5, 10, 20, 15, 18 ];

d3.select("svg").selectAll("circle")
  .data(dataset)
  .enter()
  .append("circle");

(empty selection)
// Data joins!

var dataset = [5, 10, 20, 15, 18];

d3.select("svg").selectAll("circle")
  .data(dataset)
  .enter()
  .append("circle");

5 values 0 circles

Room for 5 new circles!
// Data joins!

var dataset = [ 5, 10, 20, 15, 18 ];

d3.select("svg").selectAll("circle")
 .data(dataset)
 .enter()
 .append("circle");
// Data joins!

var dataset = [ 5, 10, 20, 15, 18 ];

d3.select("svg").selectAll("circle")
  .data(dataset)
  .enter()
  .append("circle");
// Setting attributes from data!
d3.selectAll("circle")
  .attr("r", function(d) {
    return d;
  });
// Binding data to elements

// 1. Lets you reference values later
// 2. Prevents need to “redraw” elements
Transitions
// Transitions and motion
Scales
// Scale values

```javascript
var scale = d3.scale.linear()
  .domain([200, 1000])
  .range([0, 500]);

scale(600);  // Returns 250
```

scale(600);  // Returns 250
Axes
// Generate axes

var axis = d3.svg.axis()
    .scale(scale);

svg.append("g")
    .call(axis);

svg.append("g")
    .call(axis);
What else can D3 do for me?
// Layouts
More mapping and projections
D3.js is a JavaScript library for manipulating documents based on data. D3 helps you bring data to life using HTML, SVG and CSS. D3’s emphasis on web standards gives you the full capabilities of modern browsers without tying yourself to a proprietary framework, combining powerful visualization components and a data-driven approach to DOM manipulation.