CSE 412 - Intro to Data Visualization

Intro to D3.js

Jane Hoffswell  University of Washington
Introducing Observable Plot: a new Javascript library for exploratory data visualization #72

It has been. ONE. DAY. And it seems that Monday’s Visualization Tools lecture slides are already out of date.

Introducing Observable Plot! A new open-source Javascript library for exploratory data visualization.

"Plot is informed by ten years of maintaining D3 but does not replace it. We continue to support and develop D3, and recommend its low-level approach for bespoke explanatory visualizations and as a foundation for higher-level exploratory visualization tools. In fact, Plot is built on D3! Observable Plot is more akin to Vega-Lite, another great tool for exploration. We designed Plot to pair beautifully with Observable: to leverage Observable dataflow for fluid exploration and interaction. However, Plot does not depend on Observable; use it wherever you like."
**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.

Download the latest version (6.5.0) here:

- d3.zip

To link directly to the latest release, copy this snippet:
**D³: Data-Driven Documents**

Michael Bostock, Vadim Ogievetsky and Jeffrey Heer

---

Fig. 1. Interactive visualizations built with D3, running inside Google Chrome. From left to right: calendar view, chord diagram, choropleth map, hierarchical edge bundling, scatterplot matrix, grouped & stacked bars, force-directed graph clusters, Voronoi tessellation.

**Abstract**—Data-Driven Documents (D3) is a novel representation-transparent approach to visualization for the web. Rather than hide the underlying scenegraph within a toolkit-specific abstraction, D3 enables direct inspection and manipulation of a native representation: the standard document object model (DOM). With D3, designers selectively bind input data to arbitrary document elements, applying dynamic transforms to both generate and modify content. We show how representational transparency improves expressiveness and better integrates with developer tools than prior approaches, while offering comparable notational efficiency and retaining powerful declarative components. Immediate evaluation of operators further simplifies debugging and allows iterative development. Additionally, we demonstrate how D3 transforms naturally enable animation and interaction with dramatic performance improvements over intermediate representations.

**Index Terms**—Information visualization, user interfaces, toolkits, 2D graphics.
D3 allows you to bind arbitrary data to a Document Object Model (DOM), and then apply data-driven transformations to the document. For example, you can use D3 to generate an HTML table from an array of numbers. Or, use the same data to create an interactive SVG bar chart with smooth transitions and interaction.

D3 is not a monolithic framework that seeks to provide every conceivable feature. Instead, D3 solves the crux of the problem: efficient manipulation of documents based on data. This avoids proprietary representation and affords extraordinary flexibility, exposing the full capabilities of web standards such as HTML, SVG, and CSS. With minimal overhead, D3 is extremely fast, supporting large datasets and dynamic behaviors for interaction and animation. D3’s functional style allows code reuse through a diverse collection of official and community-developed modules.

Quoted from the D3.js homepage, color emphasis added here.
Designing Visualizations for the Web

Today - Intro to D3.js
Overview of important D3 concepts.

Thursday Quiz Section - Intro to Idyll
Hands on experience with Idyll and project template.

Friday Lecture - D3 Tutorial, Part 1
Hands on experience with a static D3 chart.

Week 7, Thursday Quiz Section - D3 Tutorial, Part 2
Hands on experience with interaction & animation.
D3 is a declarative language.
What is a Declarative Language?

Programming by describing what, not how

Separate specification (what you want) from execution (how it should be computed)

In contrast to imperative programming, where you must give explicit steps.
Declarative Programming in D3

d3.selectAll("p").style("color", "blue");

"I want all paragraphs to have the color blue."
Declarative Programming in D3

d3.selectAll("p").style("color", "blue");

"I want all paragraphs to have the color blue."

Compared to imperative programming:

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

"Get all the paragraphs, then take each one one-at-a-time and set the color to be blue."
```
Why Declarative Languages?

Faster iteration. Less code. Larger user base.

Better visualization. Smart defaults.

Reuse. Write-once, then re-apply.

Performance. Optimization, scalability.

Portability. Multiple devices, renderers, inputs.

Programmatic generation.
Write programs which output visualizations.

Automated search & recommendation.
D3 leverages web standards.
"D3 allows you to bind arbitrary data to a Document Object Model"

**HTML:** Structure of items on a page.

```html
<h1>My First Heading</h1>
<p>My first paragraph.</p>
<input type="text" value="Type text here.">
<input type="submit">
```

**CSS:** Visual style of items on a page.

```css
body {
  border: blue 2px dashed;
  padding: 10px;
  width: 225px;
}

h1 {
  font-family: sans-serif;
  font-size: 16pt;
  margin: 0px;
}

p {
  color: red;
  font-family: sans-serif;
  font-style: italic;
  margin: 5px 0px;
}
```

Click title elements for web resources.
"D3 allows you to bind arbitrary data to a Document Object Model"

**HTML for Course Website:**

```html
documents.html
<html xmlns="en" lang="en">
  <head></head>
  <body data-spy="scroll" data-target="#mainbar" id="main" style="">
    <div class="content">
      <div class="title">
        <h1 class="title"></h1>
      </div>
    </div>
    <div class="content">
      <div class="sidebar">
        <h2 id="texts">Textbooks</h2>
        <h2 id="objectives">Learning Goals & Objectives</h2>
        This course is designed to provide students with the foundation of data visualization. By the end of the course, students will ...
      </div>
      <div class="article">
        <h2 id="texts">Schedule & Readings</h2>
        <script type="text/javascript"></script>
        <h3 id="week1">Week 1</h3>
        <div class="day" id="lec-value"></div>
        <div class="day" id="lec-model"></div>
        <div class="day" id="lec-ela"></div>
        <h3 id="week2">Week 2</h3>
        <div class="day" id="lec-value"></div>
        <div class="day" id="lec-model"></div>
        <div class="day" id="lec-ela"></div>
      </div>
    </div>
  </body>
</html>
```

**CSS for Course Website:**

```css
section small {
  padding: 0.5px 10.5px;
  background:  □ #7533f4;
}

.optional small {
  color: □ #666;
}

.duedate {
  font-size: 0.85em;
  color: □ red;
}

.duedate small {
  position: relative;
  top: -0.15em;
  font-size: 0.75em;
  background: □ red;
  padding: 0.5px 3.5px;
  border-radius: 4px;
  color: □ white;
}
```

Hint: You can use the Chrome Developer Tools to view webpage details, inspect or modify the structure, and debug using the JavaScript console. From Chrome, select "View" > "Developer" > "Inspect Elements" to see the HTML.
"D3 allows you to bind arbitrary data to a Document Object Model"

**SVG**: Scalable Vector Graphics - shapes and lines!

```xml
<svg version="1.1"
     baseProfile="full"
     width="300" height="200"
     xmlns="http://www.w3.org/2000/svg">

<rect width="100%" height="100%" fill="red" />

<circle cx="150" cy="100" r="80" fill="green" />

<text x="150" y="125" font-size="60" text-anchor="middle" fill="white">SVG</text>

</svg>
```
The core abstraction in D3 is a selection.
D3 Selections

The core abstraction in D3 is a selection.
D3 Selections

The core abstraction in D3 is a *selection*.

```javascript
// Add and configure an SVG element (<svg width="500" height="300">)
var svg = d3.create("svg")  // add new SVG to page body
  .attr("width", 500)       // set SVG width to 500px
  .attr("height", 300);    // set SVG height to 300px
```
D3 Selections

The core abstraction in D3 is a *selection*.

// Add and configure an SVG element (<svg width="500" height="300">)

```javascript
var svg = d3.create("svg")  // add new SVG to page body
    .attr("width", 500)  // set SVG width to 500px
    .attr("height", 300);  // set SVG height to 300px

// Select & update existing rectangles contained in the SVG element

svg.selectAll("rect")  // select all SVG rectangles
    .attr("width", 100)  // set rect widths to 100px
    .style("fill", "steelblue");  // set rect fill colors
```
D3 allows you to bind arbitrary data to the DOM.
Data Binding

Selections can *bind* data and DOM elements.

```javascript
var values = [ {...}, {...}, {...}, ... ]; // input data as JS objects
```
Data Binding

Selections can bind data and DOM elements.

```javascript
var values = [
    {"x": 0, "y": 28},
    {"x": 1, "y": 55},
    {"x": 2, "y": 43},
    {"x": 3, "y": 91},
    {"x": 4, "y": 81},
    {"x": 5, "y": 53},
    {"x": 6, "y": 19},
    {"x": 7, "y": 87},
    {"x": 8, "y": 52}
];
```

// input data as JS objects
Data Binding

Selections can **bind** data and DOM elements.

```javascript
var values = [ {...}, {...}, {...}, ... ]; // input data as JS objects

// Select SVG rectangles and bind them to data values.
var bars = svg.selectAll("rect.bars").data(values);
```
The Data Join

**Enter**
Data values without matching DOM elements.

**Update**
Existing DOM elements, bound to valid data.

**Exit**
DOM elements whose bound data has gone “stale”.

DATA VALUES

ELEMENTS
Data Binding

Selections can *bind* data and DOM elements.

```javascript
var values = [ {...}, {...}, {...}, ... ]; // input data as JS objects

// Select SVG rectangles and bind them to data values.
var bars = svg.selectAll("rect.bars").data(values)
    .join(
        // create new SVG rect marks with class "bars"
        enter => enter.append("rect").attr("class","bars"),
        // update the existing marks to change their style
        update => update,
        // remove outdated marks from the view
        exit => exit.remove()
    )
```

---

Data Binding
<svg width="500" height="300">
  <rect class="bars"></rect>
  <rect class="bars"></rect>
  <rect class="bars"></rect>
  <rect class="bars"></rect>
  <rect class="bars"></rect>
  <rect class="bars"></rect>
  <rect class="bars"></rect>
  <rect class="bars"></rect>
  <rect class="bars"></rect>
  <rect class="bars"></rect>
</svg>
Updating Data Example

// round the y value down to the tens place (e.g., 58 => 50, 91 => 90)
function tensplace(array) {
    return array.map(function(obj) {
        return {
            "x": obj.x,
            "y": obj.y - obj.y%10
        }
    });
}

// randomly shuffle the order of the input array
function shuffle(array) {...

// update our data values
values = shuffle(tensplace(values))
shuffle(tensplace(values))
Data Binding

Selections can *bind* data and DOM elements.

```javascript
var values = [ {…}, {…}, {…}, ... ]; // input data as JS objects

// Select SVG rectangles and bind them to data values.
var bars = svg.selectAll("rect.bars")
    .data(values)
    .join(
        // create new SVG rect marks with class "bars"
        enter => enter.append("rect").attr("class","bars"),
        // update the existing marks to change their style
        update => update,
        // remove outdated marks from the view
        exit => exit.remove()
    )
```
Data Binding

Selections can **bind** data and DOM elements.

```javascript
var values = [ {…}, {…}, {…}, ... ]; // input data as JS objects

// Select SVG rectangles and bind them to data values.
var bars = svg.selectAll("rect.bars")
  .data(values)
  .join("rect")
  .attr("class", "bars")
  .attr("x", d => xscale(d.x))
// more code for styling the bars...
```
Selections can *bind* data and DOM elements.

```javascript
var values = [ {...}, {...}, {...}, ... ]; // input data as JS objects

// Select SVG rectangles and bind them to data values.
var bars = svg.selectAll("rect.bars")
  .data(values, d => d.x)
  .join("rect")
  .attr("class", "bars")
  .attr("x", d => xscale(d.x))
// more code for styling the bars...
```

Data Binding with Key Functions
shuffle(tensplace(values))
Administrivia
Reminders!

Final Project Proposal Due **Fri 5/7, 11:59pm PT**
https://courses.cs.washington.edu/courses/cse412/21sp/fp.html

Four Peer Evaluations Due **Mon 5/10, 11:59pm PT**
https://courses.cs.washington.edu/courses/cse412/21sp/a3b.html
https://courses.cs.washington.edu/courses/cse412/21sp/uwnetid/a3peereval/
Zoom Poll: Final Project Progress
A3 Assignment Peer Critiques

Review **four** A3 submissions (assigned on Canvas)

Submit **four** critique forms by **Mon 5/10, 11:59pm**

You will be assigned **at least one ethical and one deceptive visualization** to review; the other two visualizations will be randomly assigned.

Follow **I like / I wish / What if?** format for critiques

Be positive! Be constructive! Share crazy ideas!
Required Reading for Fri 5/7

No required readings for Friday!
Required Reading for Mon 5/10

Notebook: Cartographic Visualization.

```javascript
vl.markGeoshape()
data(vl.topojson(world).feature('countries')).render()
```
Optional Readings for Week 7

**MON** Chapter 11: The Cartogram: Value-by-Area Mapping.

**MON** Chapter 14 in textbook.

**WED** Mapping Text with Phrase Nets
D3 is modular.
D3 Modules

Data Parsing / Formatting (JSON, CSV, …)
Shape Helpers (arcs, curves, areas, symbols, …)
Scale Transforms (linear, log, ordinal, …)
Color Spaces (RGB, HSL, LAB, …)
Animated Transitions (tweening, easing, …)
Geographic Mapping (projections, clipping, …)
Layout Algorithms (stack, pie, force, trees, …)
Interactive Behaviors (brush, zoom, drag, …)

Many of these correspond to future lecture topics!
Data Parsing / Formatting

Load file and process data in callback function.

d3.csv("path/to/file.csv", function(data) { ... });
d3.json("path/to/file.json", function(data) { ... });
d3.tsv("path/to/file.tsv", function(data) { ... });
d3.xml("path/to/file.xml", function(data) { ... });
Shape Helpers: Arc, Curve, etc.

"Graphical primitives for visualization"

d3-shape
Graphical primitives for visualization, such as lines and areas.

Showing all 29 listings
Scale Transforms

"Encoding that map abstract data to visual"

d3-scale

Encodings that map abstract data to visual representation.

Showing all 15 listings

Time thresholds for d3.bin

Color Legend

scale.ticks

d3.scaleBand

Sequential scales

Introduction to D3's scales

d3.scaleOrdinal

Diverging scales
Color Spaces

"RGB, HSL, Cubehelix, Lab (CIELAB) and HCL"

**d3-color**

Color spaces! RGB, HSL, Cubehelix, Lab (CIELAB) and HCL (CIELCH).

Showing all 3 listings

Sequential scales

Fil in D3

Jun 28, 2019 • 22

Achromatic Interpolation

Mike Bostock in D3

Apr 17, 2018 • 6

Working with Color

Mike Bostock in D3

May 7, 2018 • 59
**Animated Transitions**

**d3-transition**

Animated transitions for D3 selections.

**transition.easeVarying**

Fil in D3  
Aug 23, 2020 • ❤️ 3

**Streamgraph Transitions**

Mike Bostock in D3  
Sep 6, 2018 • ❤️ 52

**transition.textTwee**

Mike Bostock in D3  
Nov 17, 2019 • ❤️ 15 • 📊 3

**Easing Animations**

Mike Bostock in D3  
Aug 17, 2019 • ❤️ 29

**Stacked-to-Grouped Bars**

Mike Bostock in D3  
Oct 22, 2018 • ❤️ 74

**transition.end**

Mike Bostock in D3  
Jan 24, 2019 • ❤️ 27
Geographic Mapping

**d3-geo**
- World Airports
  - Mike Bostock in D3
  - Nov 3, 2018 • ❤ 6

**d3-geo-polygon**
- Two Point Equidistant
  - Fil in D3
  - May 22, 2020 • ❤ 6

**d3-geo-projection**
- Hufnagel
  - Fil in D3
  - Jul 27, 2019

- World Airports Voronoi
  - Mike Bostock in D3
  - Aug 28, 2018 • ❤ 28

- Lee’s Tetrahedral
  - Mike Bostock in D3
  - Mar 15, 2019 • ❤ 1

- Waterman’s Butterfly
  - Mike Bostock in D3
  - Mar 15, 2019 • ❤ 4
## Layout Algorithms

### d3-force
Force-directed graph layout using velocity Verlet integration.

Showing all 16 listings

<table>
<thead>
<tr>
<th>Name</th>
<th>Date</th>
<th>Author</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collision Detection</td>
<td>Sep 15</td>
<td>Mike Bostock</td>
<td>38</td>
</tr>
<tr>
<td>forceCenter.strength</td>
<td>Aug 29</td>
<td>Mike Bostock</td>
<td>11</td>
</tr>
<tr>
<td>Force-Directed Lattice</td>
<td>Sep 1</td>
<td>Mike Bostock</td>
<td>13</td>
</tr>
<tr>
<td>Collision Detection</td>
<td>Sep 1</td>
<td>Mike Bostock</td>
<td>6</td>
</tr>
<tr>
<td>Temporal Force-Directed Graph</td>
<td>Jul 8, 2020</td>
<td>Mike Bostock</td>
<td>41</td>
</tr>
<tr>
<td>Sticky Force Layout</td>
<td>Sep 2</td>
<td>Mike Bostock</td>
<td>15</td>
</tr>
</tbody>
</table>

### d3-hexbin
Group two-dimensional points into hexagonal bins.

<table>
<thead>
<tr>
<th>Name</th>
<th>Date</th>
<th>Author</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hexbin Map</td>
<td>Feb 26, 2019</td>
<td>Mike Bostock</td>
<td>56</td>
</tr>
<tr>
<td>Hexbin (Area)</td>
<td>Oct 21, 2018</td>
<td>Mike Bostock</td>
<td>13</td>
</tr>
<tr>
<td>Hexbin</td>
<td>Oct 21, 2018</td>
<td>Mike Bostock</td>
<td>42</td>
</tr>
</tbody>
</table>

### d3-hierarchy
2D layout algorithms for visualizing hierarchical data.

<table>
<thead>
<tr>
<th>Name</th>
<th>Date</th>
<th>Author</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>d3 groups as a hierarchy</td>
<td>Sep 15</td>
<td>Mike Bostock</td>
<td>38</td>
</tr>
<tr>
<td>Random Tree</td>
<td>Sep 29, 2018</td>
<td>Mike Bostock</td>
<td>28</td>
</tr>
<tr>
<td>Hierarchy traversal, animated</td>
<td>Jul 6, 2010</td>
<td>Mike Bostock</td>
<td>12</td>
</tr>
</tbody>
</table>

### d3-sankey
Visualize flow between nodes in a directed acyclic network.

<table>
<thead>
<tr>
<th>Name</th>
<th>Date</th>
<th>Author</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nike Quarterly Statement</td>
<td>Mar 3, 2019</td>
<td>Mike Bostock</td>
<td>7</td>
</tr>
<tr>
<td>Brexit Voting</td>
<td>Mar 11, 2010</td>
<td>Mike Bostock</td>
<td>16</td>
</tr>
<tr>
<td>Parallel Sets</td>
<td>Mar 11, 2010</td>
<td>Mike Bostock</td>
<td>18</td>
</tr>
</tbody>
</table>
Interactive Behaviors

**d3-brush**

Select a one- or two-dimensional region using the mouse or touch.

Showing all 15 listings

- **Brushable Parallel Coordinates** by Kerry Rodden in D3
  - Sep 10 - 18
- **Quadtree Brush** by Fil in D3
  - Sep 2 - 14
- **brush.filter** by Mike Bostock in D3
  - Aug 20, 2019 - 12
- **Mona Lisa Histogram** by Mike Bostock in D3
  - Aug 8, 2019 - 11

**d3-zoom**

Pan and zoom SVG, HTML or Canvas using mouse or touch input.

Showing all 20 listings

- ** delaunay.find & zoom** by Fil in D3
  - Oct 14 - 28
- **X/Y Zoom** by Fil in D3
  - Jun 23, 2020 - 12
- **Scatterplot Tour** by Mike Bostock in D3
  - Apr 3, 2020 - 17
- **Zoomable Area Chart** by Mike Bostock in D3
  - Jan 14, 2020 - 28
D3 is flexible!
Quiz Section: Intro to Idyll

Tomorrow, Thursday May 6th

Introduction to Idyll
Useful skills for getting started with the final project
Overview of template and visualization embedding

Up Next: Jane's Office Hour (link on Canvas)