



VISUALIZATION ON THE WEB

tableau.com



DATA ANALYSIS SOFTWARE

START YOUR FREE TRIAL

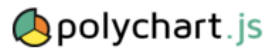
Full-version trial. No credit card required.



Kibana GA



Polychart.js



[DEMO](#)

[DOCUMENTATION](#)

[SOURCE](#)

[FORUM](#)

[BUY](#)

With Polychart.js charts are defined **declaratively** and **component-wise**, by combining:



1) Data

A JSON, CSV or AJAX data source.



2) Layers

One or more chart types, with features (like colors, sizes) bound directly to data.



3) Guides

Automatically generated axes and legends.



4) Interactions

A flexible, event-based interaction model.

NVD3.js

NVD3.js

Home

Examples

Live Code

Source

Blog

Downloads:

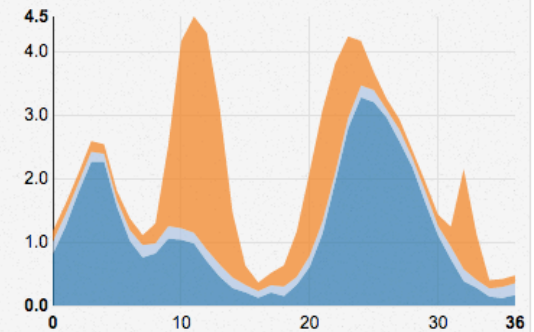
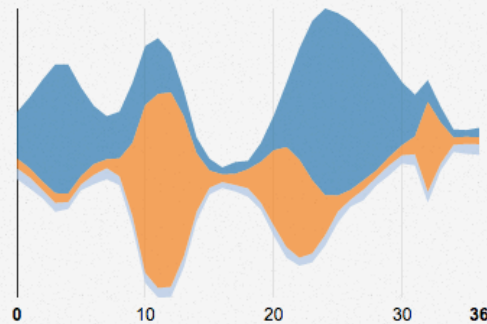
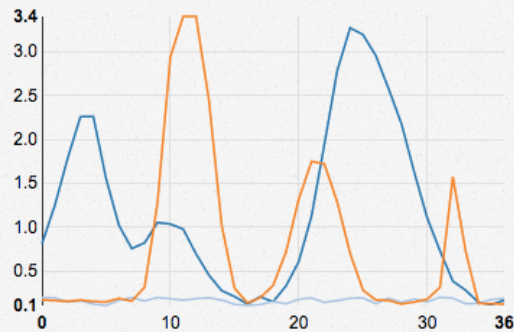
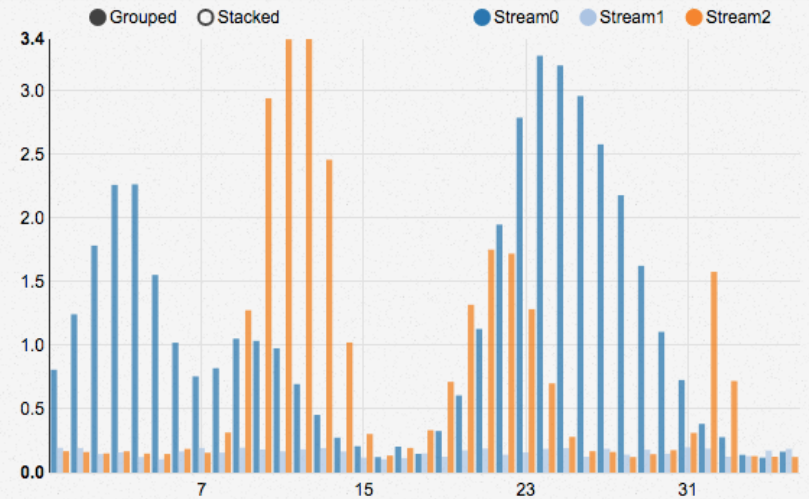
ZIP TAR.GZ

NVD3 Re-usable charts for d3.js

This project is an attempt to build re-usable charts and chart components for d3.js without taking away the power that d3.js gives you. This is a very young collection of components, with the goal of keeping these components very customizable, staying away from your standard cookie cutter solutions.

[View more examples »](#)

[GitHub Repo](#)



Visualize Data, Together

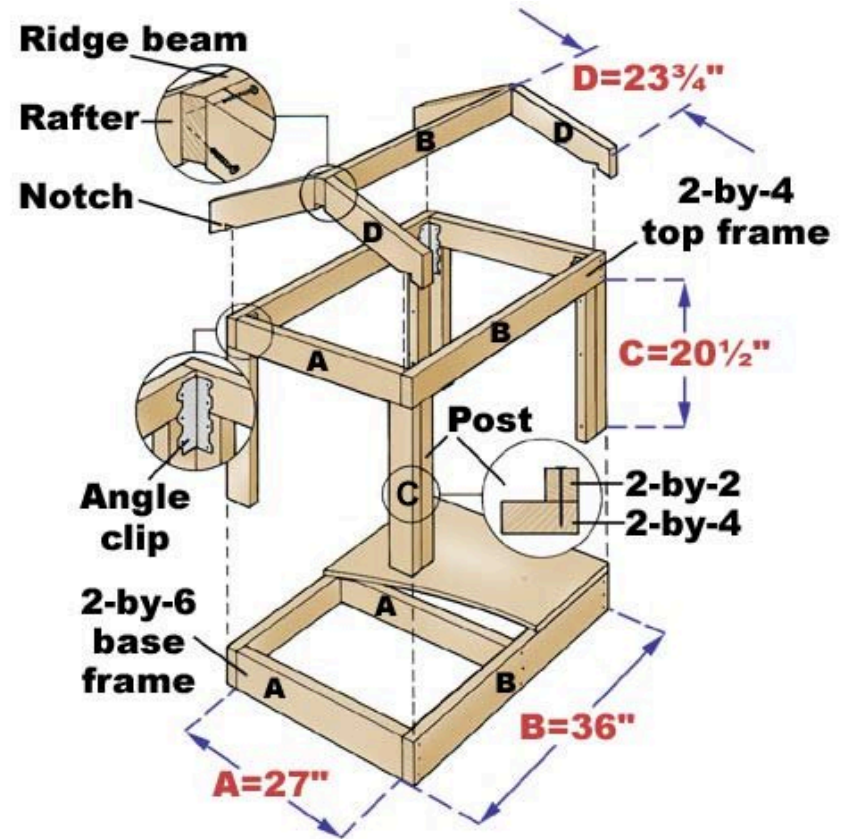
I want to make a...

[New chart](#)

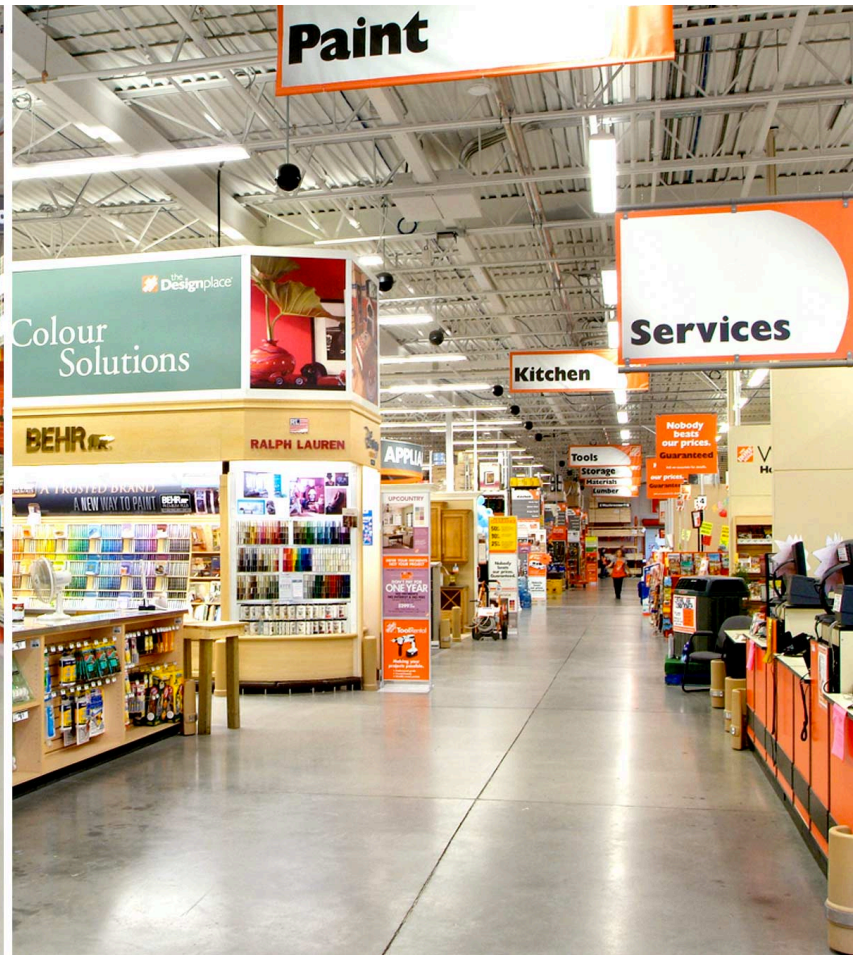
[Dashboard](#)

Compatible with a variety of tools

What is D3?



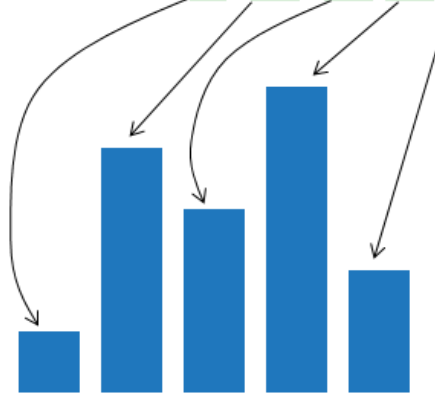
What is D3?



What is D3?

- JavaScript library to make beautiful, interactive, browser-based data visualizations.
- D3 stands for **D**ata **D**riven **D**ocuments
- D3.js is a low level visualization library based on Web standards (HTML, CSS, JS, SVG)
- D3.js is Open Source library written by Mike Bostok
- [Mike Bostock Github Profile](#)
- d3js.org

```
var data=[1, 4, 3, 5, 2];
```




Visualization and Data Graphics

Data Types

- Categorical
- Ordinal
- Quantitative

Visual Variables

position 

length 

area 

angle 

shape 

hue 

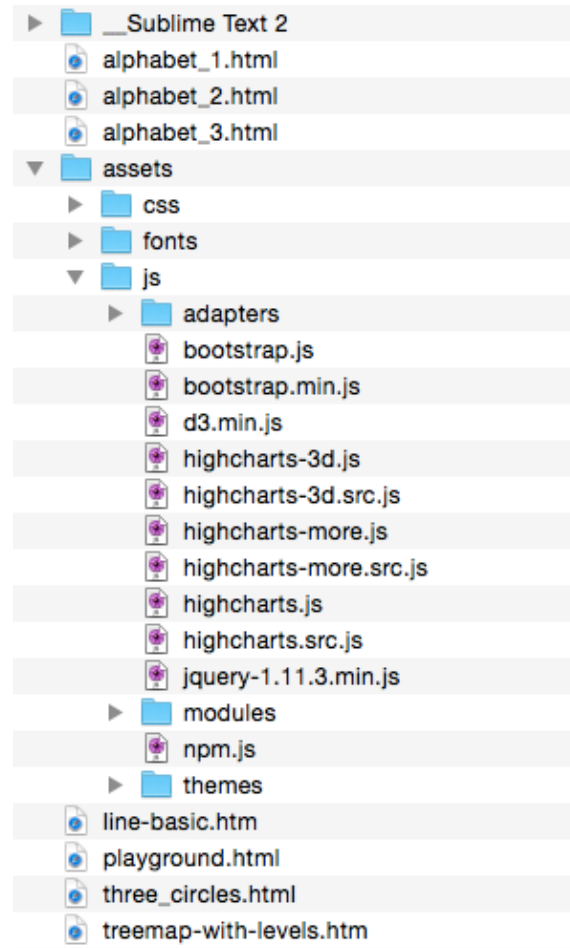
Visual Variables -> Documents

- Datum -> Element
 - Associate a graphical mark to each data point
- Data Attribute -> Element Attribute
 - Adjust properties of mark to encode properties of datum



GETTING STARTED

Your toolbox



Web Server

- A web server is required when loading external data
- Python
 - `> python -m SimpleHTTPServer 8888`
- LAMP/MAMP/WAMP
- Other webserver...

Developer Tools (Safari, Chrome, Firefox)

The screenshot displays the developer tools interface with the following components:

- Elements Panel:** Shows the DOM tree with the following structure:

```
<!DOCTYPE html>
<html>
  <head>...</head>
  <body class="overview" style="margin-top: -347.5px; margin-bottom: -347.5px; height: 6492.5px; ">
    <section class="stack" style="z-index: 10; -webkit-transform: translate3d(0px, 0px, 0px); ">
      <h1 class="blue">D3 Workshop</h1>
      <h2>...</h2>
    </section>
    <section class="stack" style="z-index: 9; opacity: 0; ">
      <h1>...</h1>
    </section>
    <section class="stack" style="z-index: 8; ">
      <h1>1. DOM Manipulation.</h1>
    </section>
    <section class="stack" style="z-index: 7; ">
      <h1>...</h1>
      <h2 class="green">Document Object Model</h2>
    </section>
    <section class="stack" style="z-index: 6; ">
      <pre>...</pre>
    </section>
    <section class="stack" style="z-index: 5; ">
      <h1>...</h1>
      <h2>A hierarchical, abstract representation of an image.</h2>
    </section>
    <section class="stack active" style="z-index: 4; -webkit-transform: translate3d(0px, 0px, 0px); ">
      <h1>Developer Tools</h1>
    </section>
    <section class="stack" style="z-index: 3; opacity: 0; ">
      <pre>...</pre>
      <h2> (= jQuery)</h2>
    </section>
    <section class="stack" style="z-index: 2; ">
      <h1>2. DOM Generation.</h1>
      <h2>Document Object Model + Data</h2>
    </section>
    <section class="stack" style="z-index: 1; ">
      <h1>...</h1>
      <h2>Document Object Model + Data + Visual Encodings</h2>
    </section>
    <script src="d3.v2.js"></script>
    <script src="stack.v0.js"></script>
    <script src="highlight.v0.js"></script>
  </body>
</html>
```
- Styles Panel:** Shows the computed style for the selected `h1` element:

```
Computed Style
Styles
element.style {
}

Matched CSS Rules
.blue, .html .tag, .css .tag, style.css:76
.javascript .keyword {
  color: #6BAED6;
}

h1 {
  top: 160px;
}

h1 {
  font-size: 160px;
}

h1, h2, h3, h4 {
  position: absolute;
  font-family: "Yanone Kaffeesatz";
  line-height: 1em;
  margin: 0;
}

:-webkit-user agent stylesheet
any(article,aside,nav,section) h1 {
  font-size: 1.5em;
  -webkit-margin-before: 0.83em;
  -webkit-margin-after: 0.83em;
}

h1 {
  display: block;
  font-size: 2em;
  -webkit-margin-before: 0.67em;
  -webkit-margin-after: 0.67em;
  -webkit-margin-start: 0px;
  -webkit-margin-end: 0px;
  font-weight: bold;
}

Inherited from section.stack
.stack {
  color: white;
  font-size: 36px;
  line-height: 1.5em;
}

Inherited from body.overview
body {
}
```


Getting started – Anatomy of an HTML Page

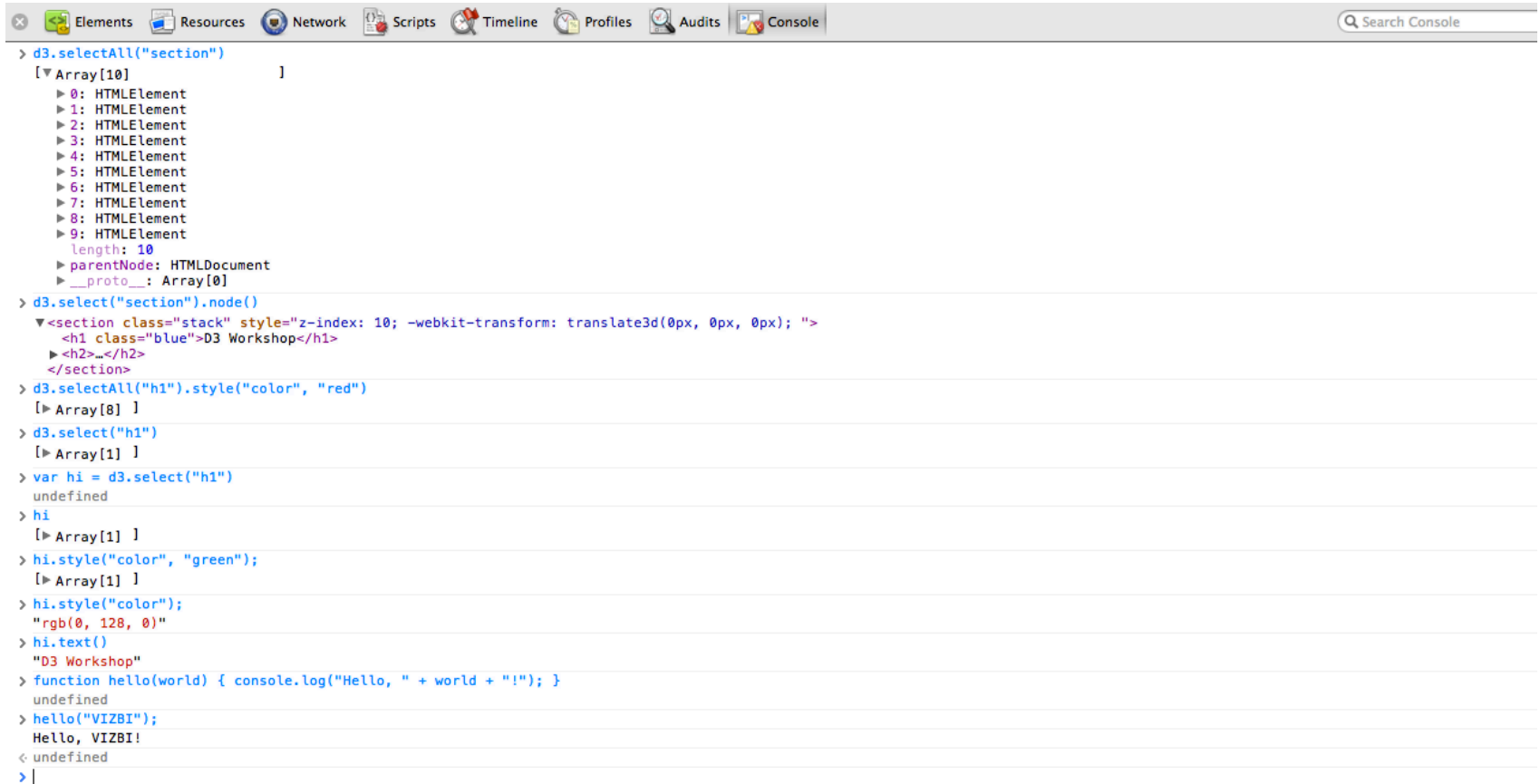
```
<!DOCTYPE html>
<html>
  <head>
    <script src="http://d3js.org/d3.v3.min.js"></script>
    <script src="js/main.js" ></script>
  </head>
  <body>
    ...
  </body>
</html>
```



main.js

```
d3.select("body").append("p").text("Hello
World!");
```

Javascript Console (Safari, Chrome, Firefox)



The screenshot shows a browser's developer console with the following code and output:

```
> d3.selectAll("section")
[▼ Array[10] ]
  ▶ 0: HTMLElement
  ▶ 1: HTMLElement
  ▶ 2: HTMLElement
  ▶ 3: HTMLElement
  ▶ 4: HTMLElement
  ▶ 5: HTMLElement
  ▶ 6: HTMLElement
  ▶ 7: HTMLElement
  ▶ 8: HTMLElement
  ▶ 9: HTMLElement
  length: 10
  parentNode: HTMLDocument
  __proto__: Array[0]
> d3.select("section").node()
▼<section class="stack" style="z-index: 10; -webkit-transform: translate3d(0px, 0px, 0px); ">
  <h1 class="blue">D3 Workshop</h1>
  ▶<h2>...</h2>
  </section>
> d3.selectAll("h1").style("color", "red")
[▶ Array[8] ]
> d3.select("h1")
[▶ Array[1] ]
> var hi = d3.select("h1")
undefined
> hi
[▶ Array[1] ]
> hi.style("color", "green");
[▶ Array[1] ]
> hi.style("color");
"rgb(0, 128, 0)"
> hi.text()
"D3 Workshop"
> function hello(world) { console.log("Hello, " + world + "!"); }
undefined
> hello("VIZBI");
Hello, VIZBI!
< undefined
> |
```



SELECTIONS

CSS Selectors

- CSS provides an efficient way to refer to specific elements in a DOM
- `#foo` // `<any id="foo">`
- `foo` // `<foo>...</foo>`
- `.foo` // `<any class="foo">`
- `[foo=bar]` // `<any foo="bar">`
- `foo bar` // `<foo><bar/></foo>`

Selector Functions

W3C

- `document.querySelectorAll("h1")`

D3.js / JQuery

- `d3.selectAll("h1")`

Selections are Arrays.

Explore selections with Developer Tools

attr and style methods

```
// select all <h1> elements  
var H1s = d3.selectAll("H1");  
  
H1s.attr("class", "newClass");  
H1s.style("fill", "yellow");  
H1s.style("font-color", "black");
```

Chaining methods

```
d3.selectAll("H1")  
  .attr("class", "newClass")  
  .style("fill", "yellow")  
  .style("font-color", "black");
```

Append new elements

```
var body = d3.select("body");
```

```
var h1 = body.append("h1");
```

```
h1.text("Hello!");
```


Modify existing elements

```
var section = d3.selectAll("section");
```

```
var h1 = section.append("h1");
```

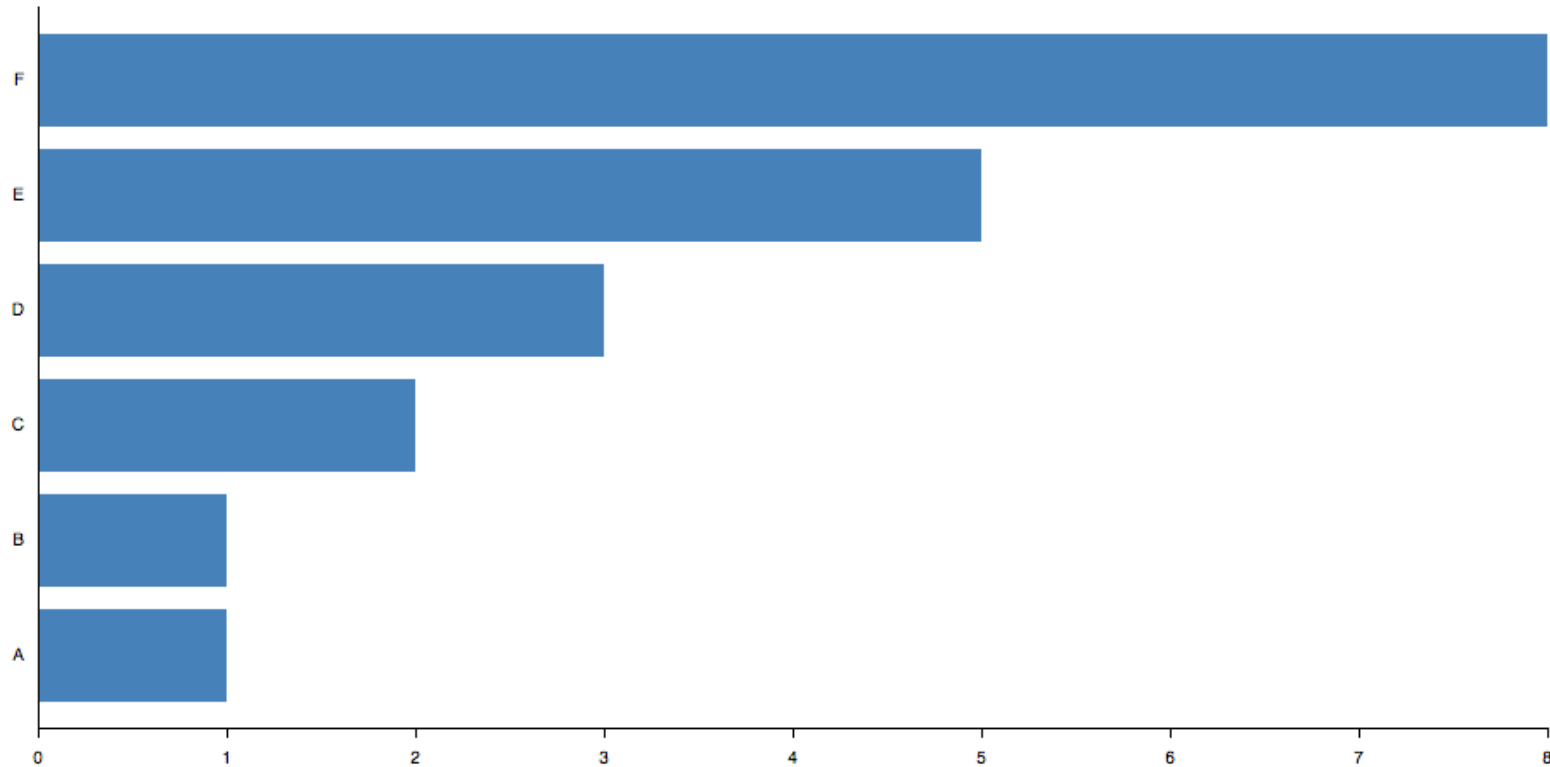
```
h1.text("Hello!");
```

Exercise: Playground

See example [playground.html](#)

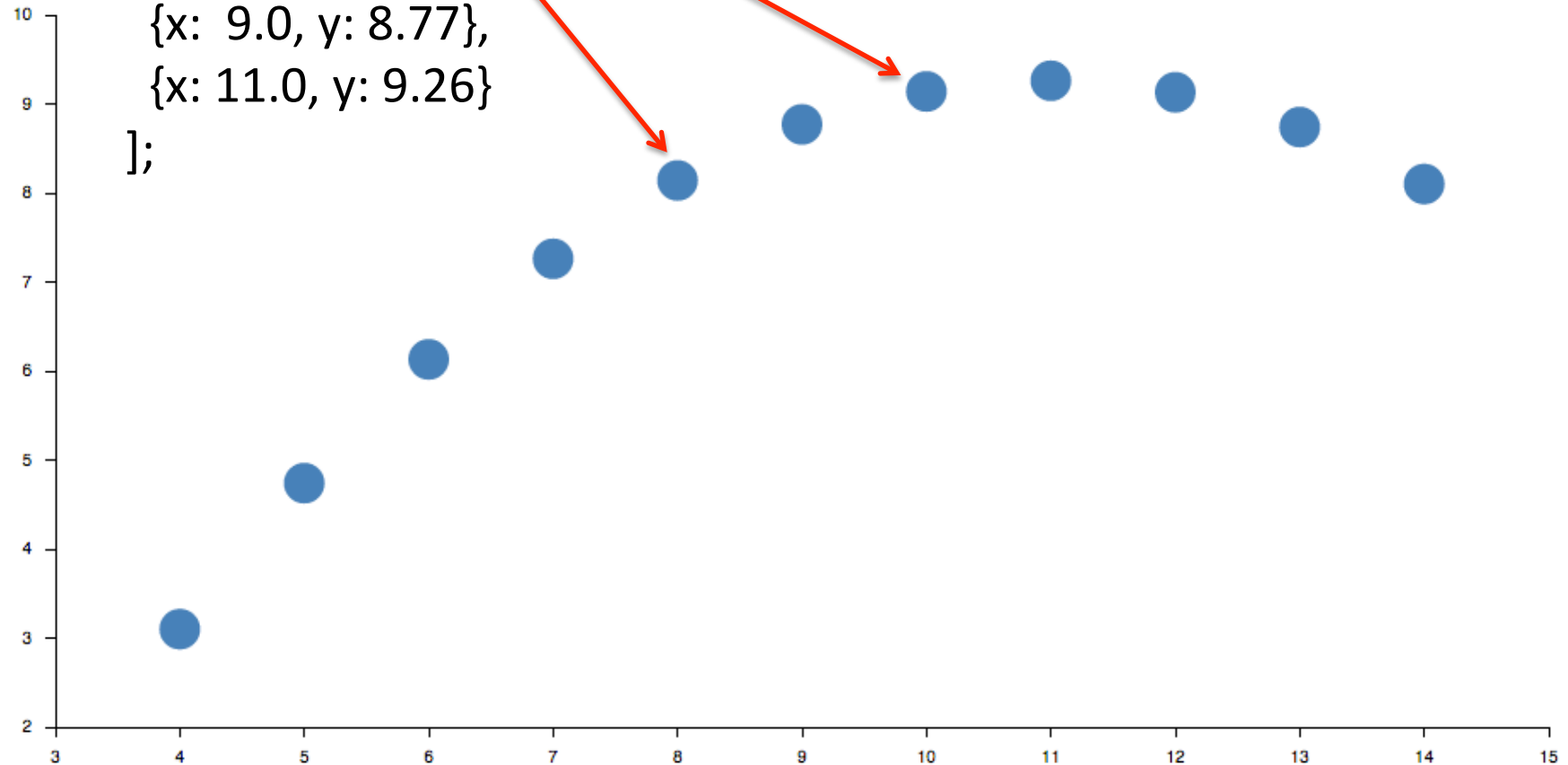
Data can be numbers

```
var numbers= [1, 1, 2, 3, 5, 8];
```



Data can be objects.

```
var data = [  
  {x: 10.0, y: 9.14},  
  {x: 8.0, y: 8.14},  
  {x: 13.0, y: 8.74},  
  {x: 9.0, y: 8.77},  
  {x: 11.0, y: 9.26}  
];
```





DATA TO ELEMENTS

Selection should correspond to data

```
svg.selectAll("circle")  
  .data(data)  
  .enter().append("circle")  
  .attr("cx", x)  
  .attr("cy", y)  
  .attr("r", 2.5);
```

Selection should correspond to data

```
svg.selectAll("circle")  
  .data(data)  
  .enter().append("circle")  
  .attr("cx", x)  
  .attr("cy", y)  
  .attr("r", 2.5);
```

Method [data](#) joins data with document elements

Selection should correspond to data

```
svg.selectAll("circle")  
  .data(data)  
  .enter().append("circle")  
  .attr("cx", x)  
  .attr("cy", y)  
  .attr("r", 2.5);
```

Method `enter` specifies the action for missing elements

Selection should correspond to data

```
svg.selectAll("circle")  
  .data(data)  
  .enter().append("circle")  
  .attr("cx", x)  
  .attr("cy", y)  
  .attr("r", 2.5);
```

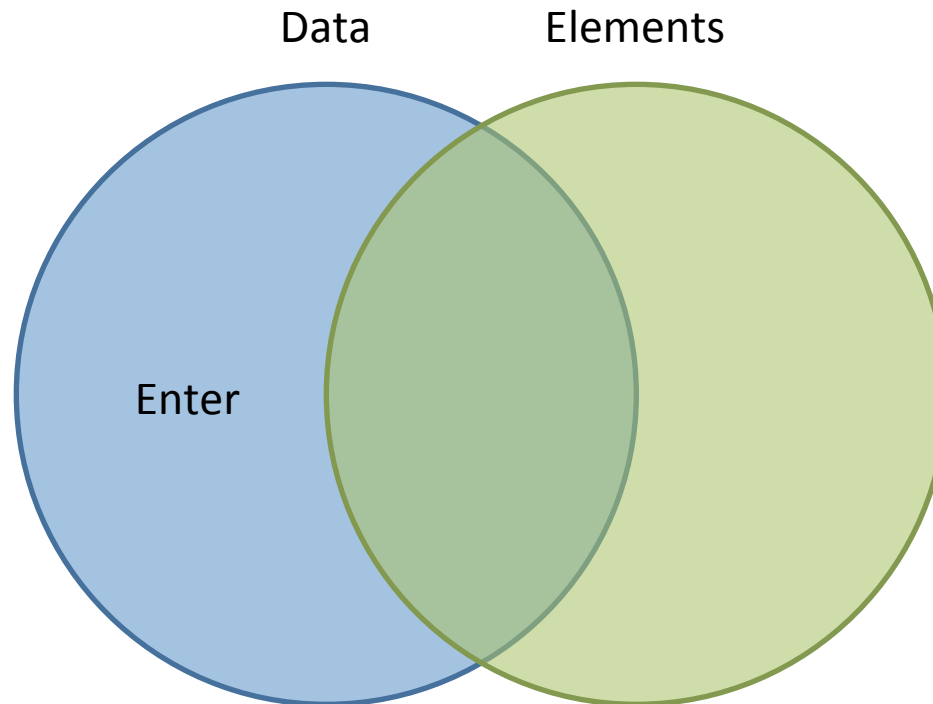
The new elements are bound to data. Data can be used to compute attributes

Thinking with Joins

ENTER, UPDATE, AND EXIT

Enter

- New data, for which there were no existing elements.

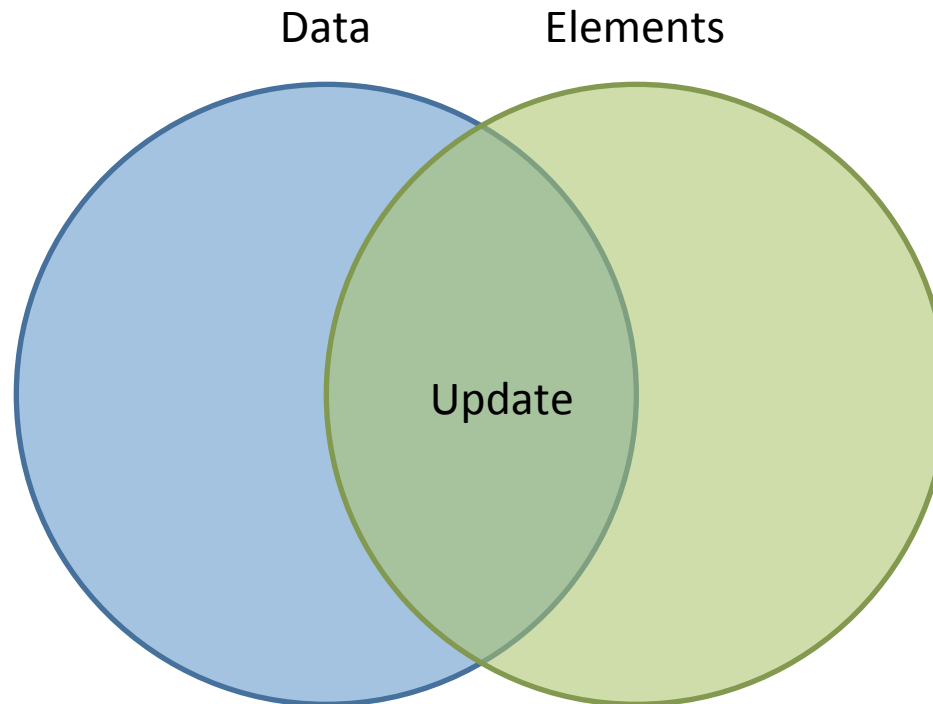


Entering new elements

```
svg.selectAll("circle")  
  .data(data)  
  .enter().append("circle")  
  .attr("cx", x)  
  .attr("cy", y)  
  .attr("r", 2.5);
```

Update

- Data that was joined previously to an existing element

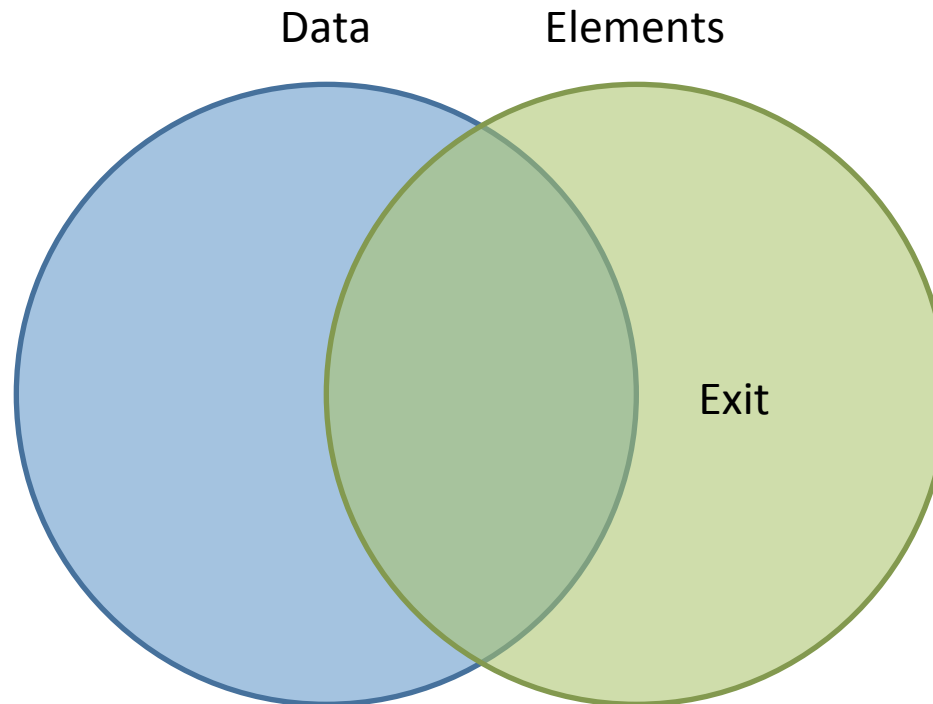


Updating existing elements

```
svg.selectAll("circle")  
  .data(data)  
  .attr("cx", x)  
  .attr("cy", y)  
  .attr("r", 2.5);
```

Exit

- Data that is not associated to any data

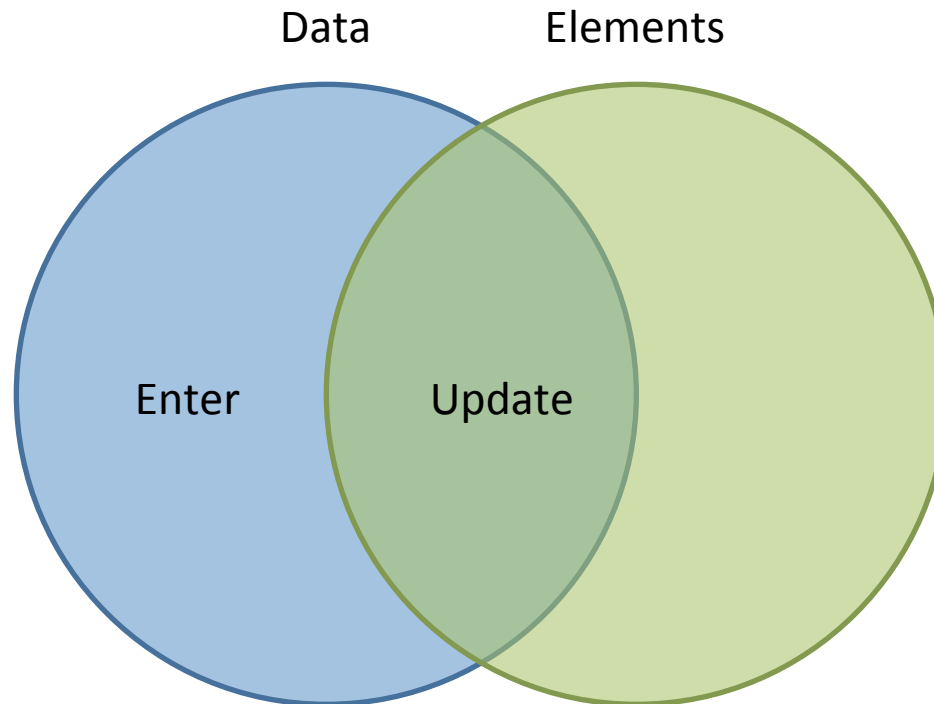


Removing unlinked elements

```
svg.selectAll("circle")  
  .data(data)  
  .exit().remove();
```


Enter + Update

- Often, we need to perform both Enter and Update



Entering and then update

```
// perform join and add missing elements
var circles = svg.selectAll("circle")
    .data(data)
    .enter().append("circle");
// Update previous and new elements at
once
circles.attr("cx", x)
    .attr("cy", y)
    .attr("r", 2.5);
```

Playground.html text example

Exercise