



VISUALIZATION ON THE WEB

tableau.com



DATA ANALYSIS SOFTWARE

START YOUR FREE TRIAL

Full-version trial. No credit card required.



Kibana GA



Kibi

The screenshot displays the Kibi web application interface. The top navigation bar includes 'Discover', 'Visualize', 'Dashboard', and 'Settings'. Below this, there are tabs for 'Molecules', 'Activities (13520737)', 'Assays (1148941)', 'Targets (10775)', and 'Papers (59610)'. The main content area is divided into three sections:

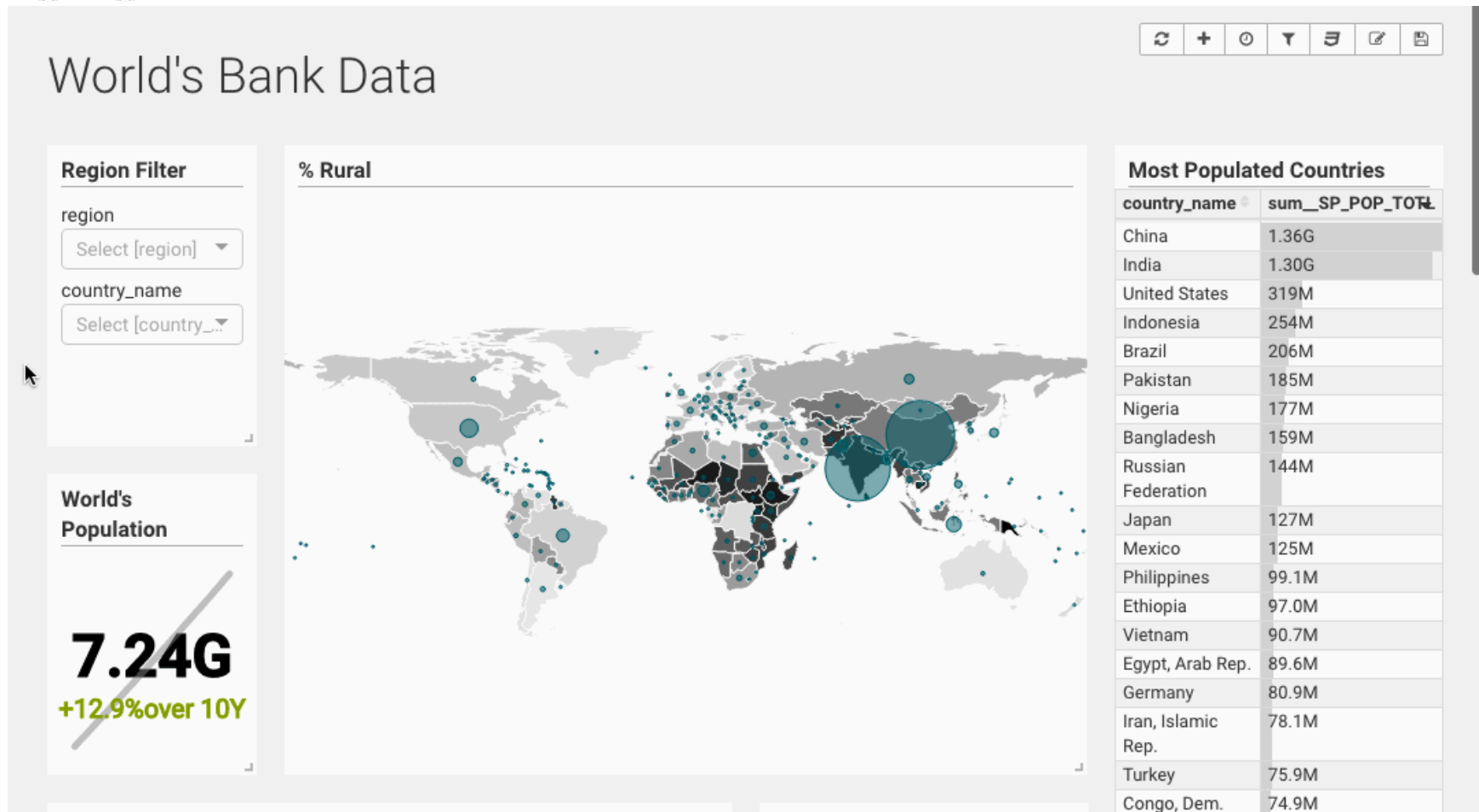
- Molecule type:** A sidebar showing a list of molecule types and their counts. The top 10 are: Small molecule (1,437,508), Protein (19,405), Unknown (5,379), Antibody (718), Enzyme (88), Oligonucleotide (86), Oligosaccharide (60), Cell (22), and Unclassified (6).
- Indication Class:** A sidebar showing a list of indication classes and their counts. The top 500 are: Antibacterial (319), Antineoplastic (167), Antidepressant (99), Antihypertensive (97), Anti-inflammatory (89), Analgesic (81), Antipsychotic (80), and Radioactive Agent (73).
- molecules search:** A table displaying search results. The table has columns for 'molregno', 'pref_name *', 'molecule_type', 'availability_type', 'synonyms', and 'chirality'. The results are paginated, showing items 1 through 10.

On the right side of the interface, there are two additional sections:

- Relational Button Activities:** A button labeled '... show related activities (13520737)'.
- Therapeutic vs Non (Chirality):** Two pie charts. The top chart is a green pie chart representing a single category. The bottom chart is a multi-colored pie chart (purple, blue, teal) representing multiple categories. A legend to the right of the charts indicates the color coding for values -1, 1, 2, and 0.

<https://siren.solutions/kibi/>

Superset



NVD3.js

NVD3.js

Home

Examples

Live Code

Source

Blog

Downloads:

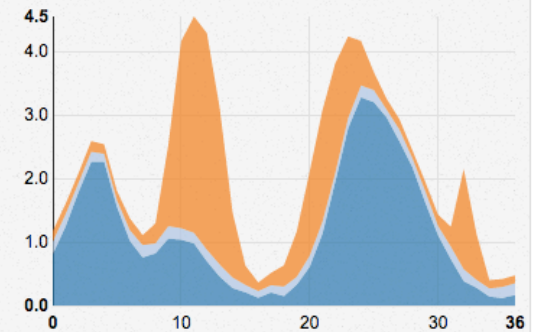
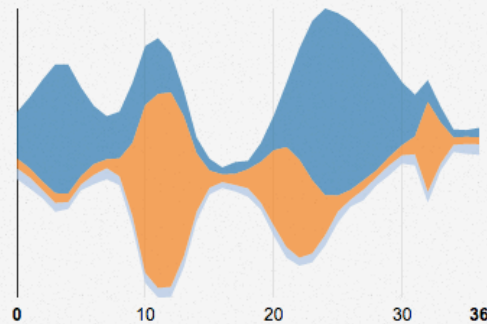
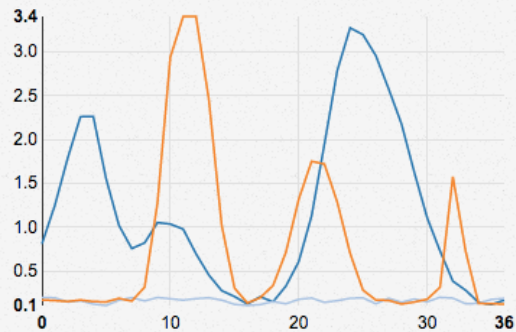
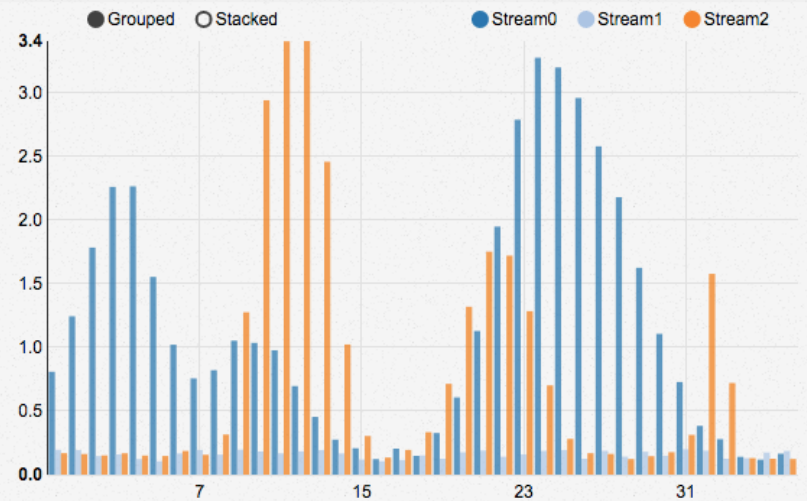
ZIP TAR.GZ

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

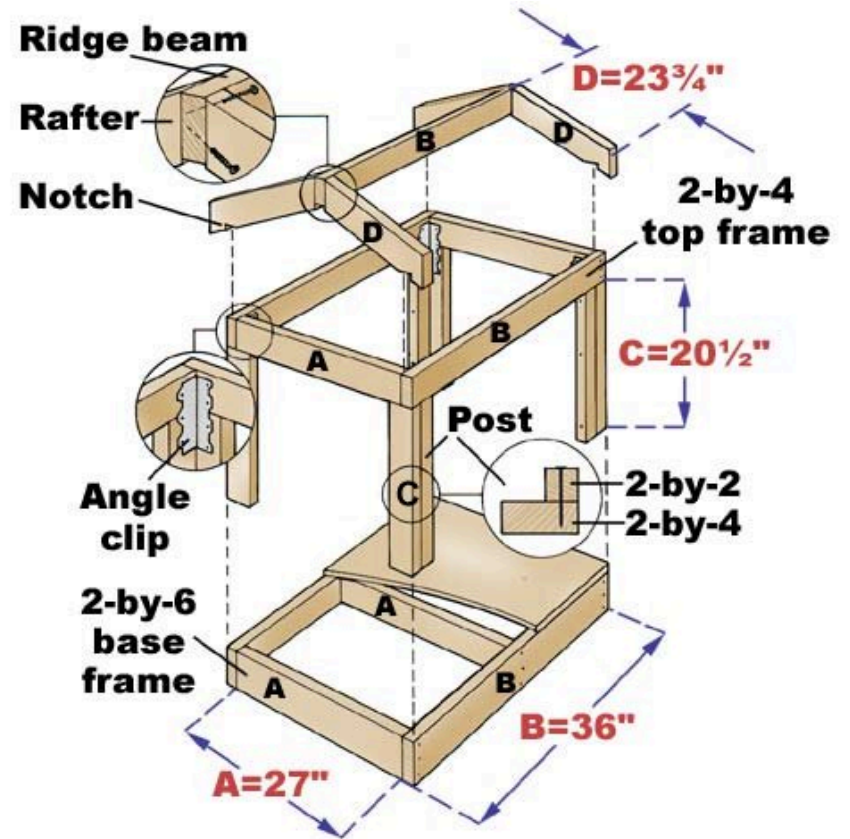
I want to make a...

[New chart](#)

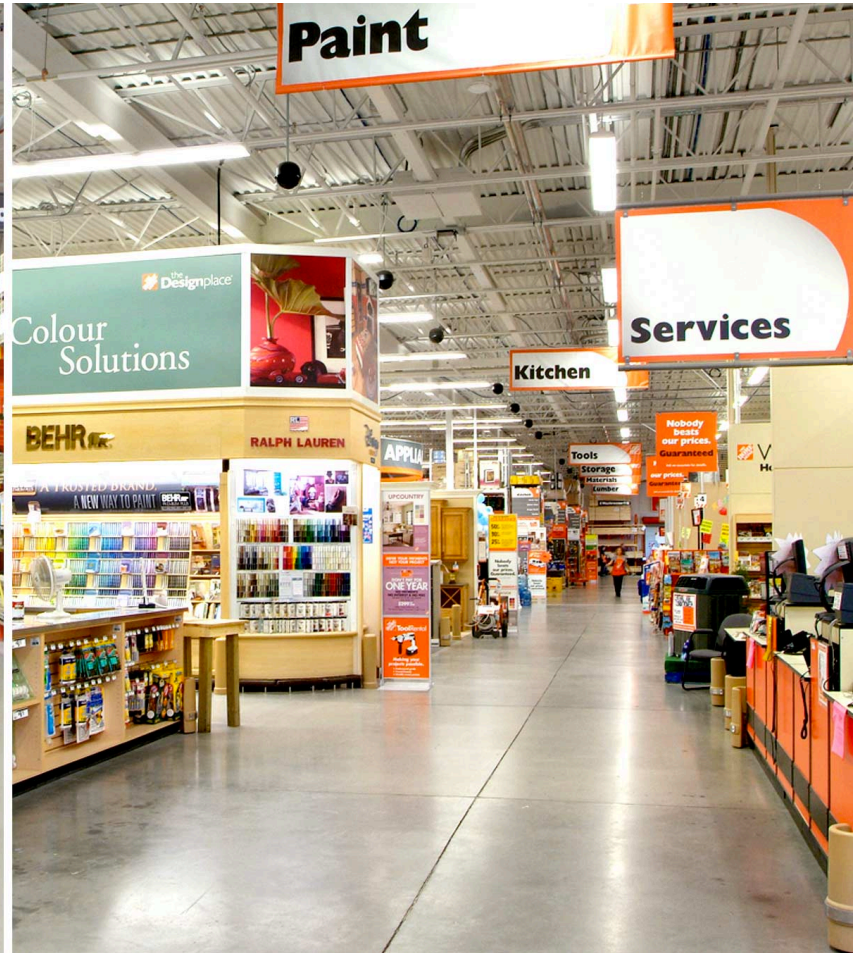
[Dashboard](#)

Compatible with a variety of tools

What is D3?



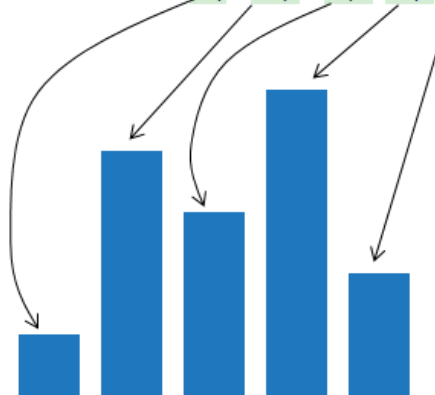
What is D3?



What is D3?

- JavaScript library to make beautiful, interactive, browser-based data visualizations.
- D3 stands for **Data Driven Documents**
- 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 

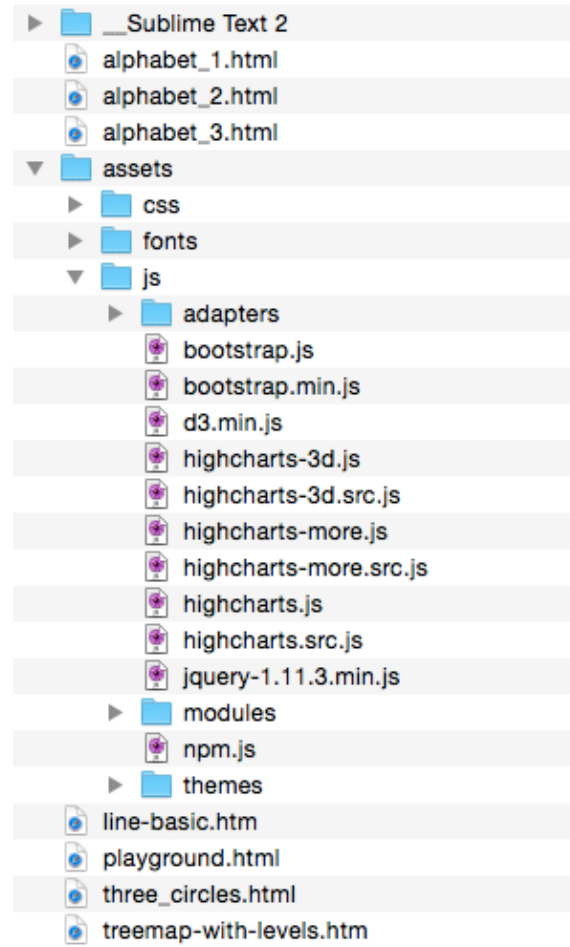
Visual Variables -> Documents

- 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 a web browser's developer tools interface. The left pane 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>
      <script src="d3.v2.js"></script>
      <script src="stack.v0.js"></script>
      <script src="highlight.v0.js"></script>
    </section>
  </body>
</html>
```

The right pane shows the **Computed Style** panel for the selected `h1` element. It displays the following styles:

- element.style {**
- Matched CSS Rules**
- `.blue, .html .tag, .css .tag, .javascript .keyword {` [style.css:76](#)
 - `color: #6BAED6;`
- `h1 {` [style.css:49](#)
 - `top: 160px;`
- `h1 {` [style.css:45](#)
 - `font-size: 160px;`
- `h1, h2, h3, h4 {` [style.css:38](#)
 - `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 {` [user agent stylesheet](#)
 - `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** [style.css:17](#)
 - `.stack {`
 - `color: white;`
 - `font-size: 36px;`
 - `line-height: 1.5em;`
- Inherited from body.overview** [style.css:1](#)
 - `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)

```
Elements Resources Network Scripts Timeline Profiles Audits Console Search Console
> 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");
```

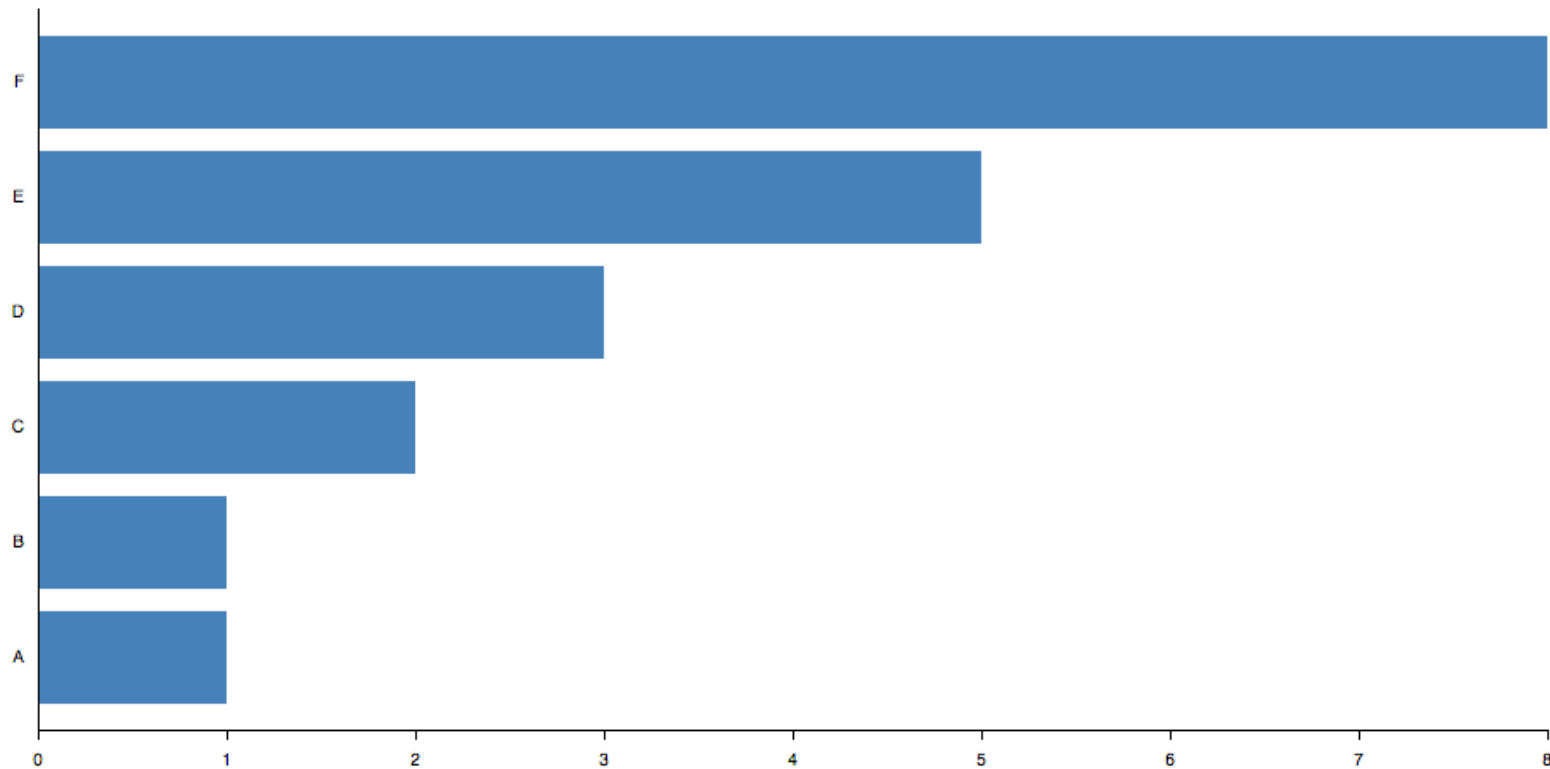
```
h1.text("Hello!");
```

Exercise: Playground

See example [playground.html](#)

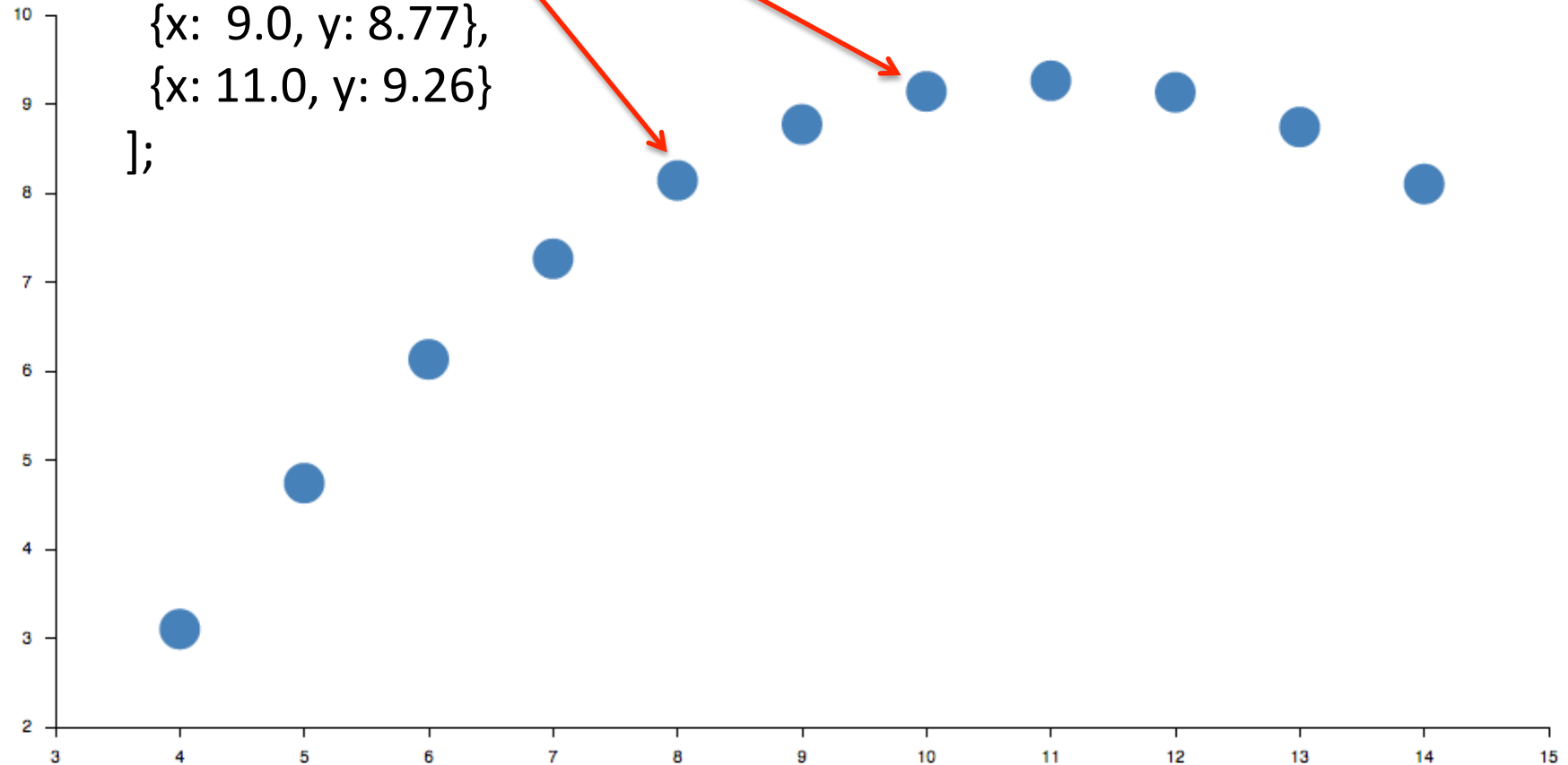
Data can be numbers

```
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);
```

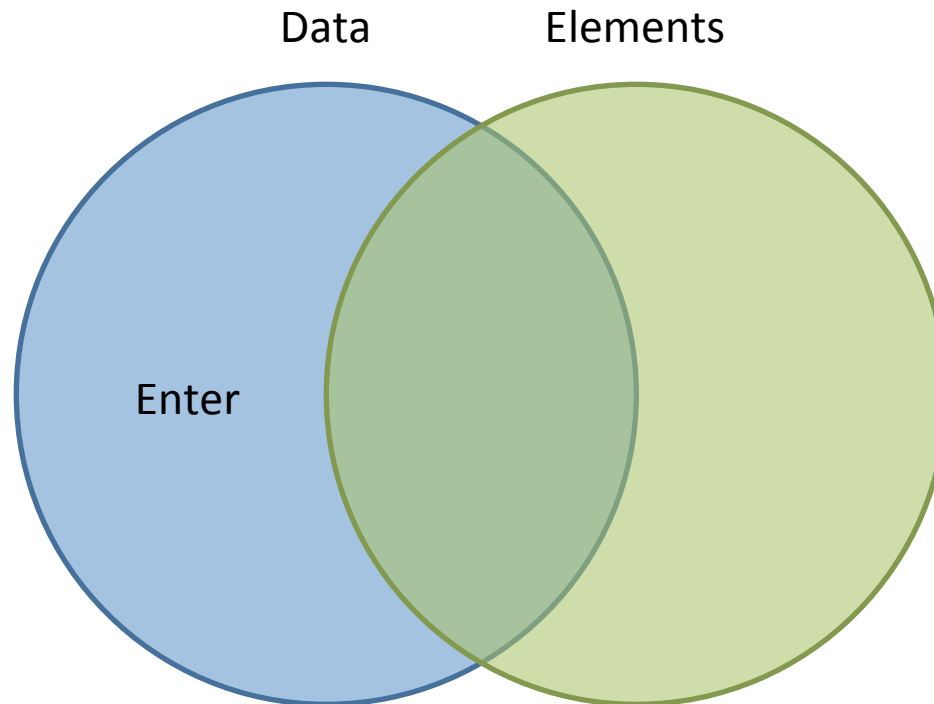
The new elements are bound to data. Data can be used to compute attributes

Thinking with Joins

ENTER, UPDATE, AND EXIT

Enter

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

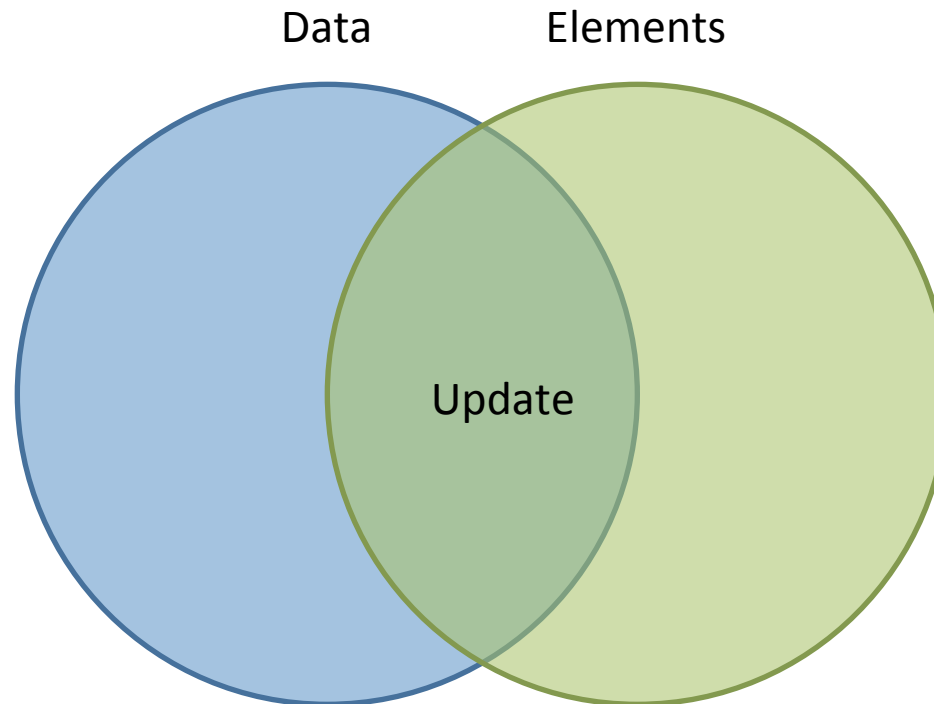


Entering new elements

```
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

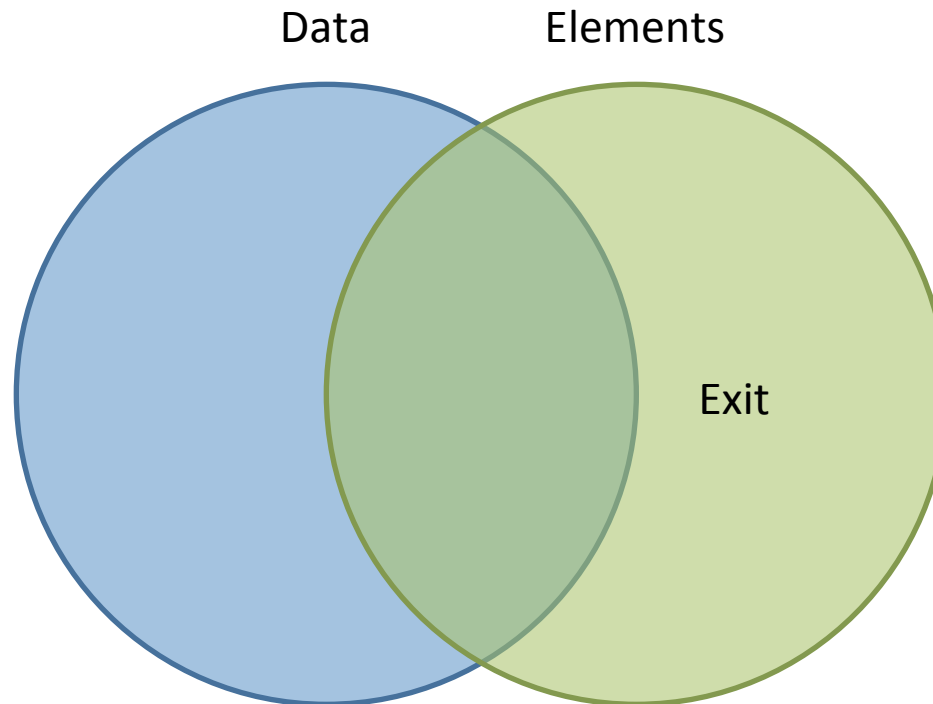


Updating existing elements

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

Exit

- Data that is not associated to any data

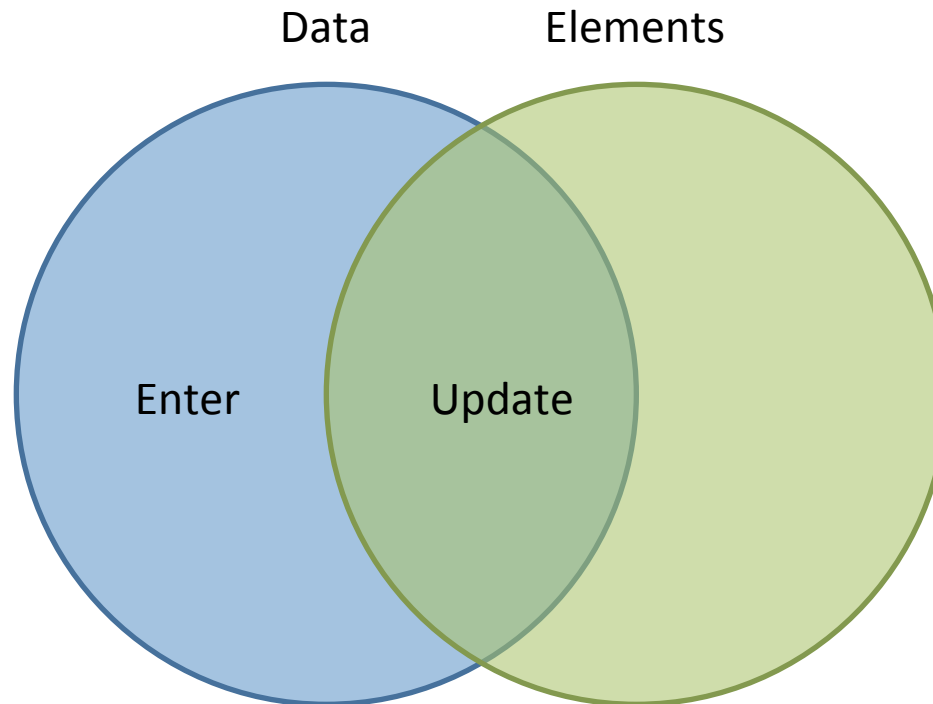


Removing unlinked elements

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

Playground.html text example

Exercise