VISUALIZATION ON THE WEB
DATA ANALYSIS SOFTWARE

START YOUR FREE TRIAL

Full-version trial. No credit card required.
Kibana GA
Kibi

https://siren.solutions/kibi/
World's Bank Data

Most Populated Countries

<table>
<thead>
<tr>
<th>country_name</th>
<th>sum_SP_POP_TOTM</th>
</tr>
</thead>
<tbody>
<tr>
<td>China</td>
<td>1.36G</td>
</tr>
<tr>
<td>India</td>
<td>1.30G</td>
</tr>
<tr>
<td>United States</td>
<td>319M</td>
</tr>
<tr>
<td>Indonesia</td>
<td>254M</td>
</tr>
<tr>
<td>Brazil</td>
<td>206M</td>
</tr>
<tr>
<td>Pakistan</td>
<td>185M</td>
</tr>
<tr>
<td>Nigeria</td>
<td>177M</td>
</tr>
<tr>
<td>Bangladesh</td>
<td>159M</td>
</tr>
<tr>
<td>Russian Fed.</td>
<td>144M</td>
</tr>
<tr>
<td>Japan</td>
<td>127M</td>
</tr>
<tr>
<td>Mexico</td>
<td>125M</td>
</tr>
<tr>
<td>Philippines</td>
<td>99.1M</td>
</tr>
<tr>
<td>Ethiopia</td>
<td>97.0M</td>
</tr>
<tr>
<td>Vietnam</td>
<td>90.7M</td>
</tr>
<tr>
<td>Egypt, Arab Rep.</td>
<td>89.6M</td>
</tr>
<tr>
<td>Germany</td>
<td>80.9M</td>
</tr>
<tr>
<td>Iran, Islamic Rep.</td>
<td>78.1M</td>
</tr>
<tr>
<td>Turkey</td>
<td>75.9M</td>
</tr>
<tr>
<td>Congo, Dem.</td>
<td>74.9M</td>
</tr>
</tbody>
</table>

https://github.com/airbnb/superset
NVD3.js

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
Vega-Lite is a high-level grammar of interactive graphics. It provides a concise JSON syntax for rapidly generating visualizations to support analysis. Vega-Lite specifications can be compiled to Vega specifications.

Vega-Lite specifications describe visualizations as mappings from data to properties of graphical marks (e.g., points or bars). The Vega-Lite compiler automatically produces visualization components including axes, legends, and scales. It then determines properties of these components based on a set of carefully designed rules. This approach allows specifications to be succinct and expressive, but also provide user control. As Vega-Lite is designed for analysis, it supports data transformations such as aggregation, binning, filtering, sorting, and visual transformations including stacking and faceting. Moreover, Vega-Lite specifications can be composed into layered and multi-view displays, and made interactive with selections.
VISUAL ANALYTICS
D3.JS
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](https://github.com/d3)
- [d3js.org](https://d3js.org)
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
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

```javascript
var body = d3.select("body");
var h1 = body.append("h1");
h1.text("Hello!");
```
Modify existing elements

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

var h1 = section.append("h1");
h1.text("Hello!");
```
Exercise #1

- Create the ladder design of the previous lesson, using only D3.js manipulation of DOM

```html
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>Stairs example - Multiple implementation</title>
<style>
  svg{
    background:#fff;
  }

  svg circle{
    fill:#e34a33
  }
</style>
</head>
<body>
<svg width="200" height="200">
  <polyline points="0,40 40,40 40,80 80,80 80,120 120,120 120,160" fill="white" stroke="#BBC42A" stroke-width="6" />
</svg>
</body>
</html>
```
DATA TO ELEMENTS
Selection should correspond to data

```javascript
var numbers = [5, 10, 15, 20, 25];
var lines = svg.selectAll("line")
    .data(numbers)
    .enter().append("line");
```
Selection should correspond to data

```javascript
var numbers = [5, 10, 15, 20, 25];
var lines = svg.selectAll("line").data(numbers).enter().append("line");
```

Method `data` joins data with document elements.
Selection should correspond to data

```javascript
var numbers = [5,10,15,20,25];
var lines = svg.selectAll("line").data(numbers).enter().append("line");
```

Method `enter` specifies the action for missing elements
Selection should correspond to data

```javascript
var numbers = [5,10,15,20,25];
var lines = svg.selectAll("line")
    .data(numbers)
    .enter().append("line");
```

<table>
<thead>
<tr>
<th>Data</th>
<th>SVG</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>25</td>
<td>25</td>
</tr>
</tbody>
</table>
Selection should correspond to data

```
var numbers = [5,10,15,20,25];
var lines = svg.selectAll("line")
  .data(numbers)
  .enter().append("line");

lines.attr("x1",10)
  .attr("y1",posy(d,i))
  .attr("x2",posx(d,i))
  .attr("y2",posy(d,i))

<table>
<thead>
<tr>
<th>Data</th>
<th>SVG</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>25</td>
<td>25</td>
</tr>
</tbody>
</table>
```

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

```javascript
lines.attr("x1", 10)
  .attr("y1", posy(d, i))
  .attr("x2", posx(d, i))
  .attr("y2", posy(d, i));

var posy = function(d, i) {
  return i*10;
}

var posx = function(d, i) {
  return d * 10;
}
```

The `attr` functions takes in input a constant value or a function. The function is called automatically by d3, passing the data (`__data__`) bound to the element and a progressive counter.

<table>
<thead>
<tr>
<th>Data</th>
<th>SVG</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>25</td>
<td>25</td>
</tr>
</tbody>
</table>
Exercise #2

- Use length visual variable to represent a set of numbers
  - Map numbers to a set of lines
  - Make each line length proportional to the number it represents
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}
];
Thinking with Joins

ENTER, EXIT, AND UPDATE
- New data, for which there were no existing elements.
Entering new elements

```javascript
var numbers = [5,10,15,20,25];
var lines = svg.selectAll("line").data(numbers);
lines.enter().append("line");
```

<table>
<thead>
<tr>
<th>Data</th>
<th>SVG</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
</tr>
</tbody>
</table>
Entering new elements

```javascript
var numbers = [5,10,15,20,25];
var lines = svg.selectAll("line").data(numbers);
  .enter().append("line");
```
- Elements that are associated with no data
Exiting unnecessary elements

var numbers = [5,10,15];
var lines = 
  svg.selectAll("line")
    .data(numbers);
lines
  .exit().remove();
var numbers = [5,10,15,20,25];
var lines = svg.selectAll("line")
  .data(numbers);

lines.exit().remove();
Step 2
DATA ATTRIBUTES TO ELEMENTS ATTRIBUTES
- Data already joined with previous elements
Update existing and new elements with new data

```javascript
var numbers = [5,10,15,20,25];
var lines = svg.selectAll("line")
    .data(numbers);

lines = lines.enter()
    .append("line")
    .merge(lines);

lines.attr("x1",10)
    .attr("y1",posy(d,i))
    .attr("x2",posx(d,i))
    .attr("y2",posy(d,i));
```

<table>
<thead>
<tr>
<th>Data</th>
<th>SVG</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>15</td>
<td>15</td>
</tr>
</tbody>
</table>
Joining with key function

```javascript
var data = [  
    {name: "Locke", number: 4},  
    {name: "Reyes", number: 8},  
    {name: "Ford", number: 15},  
    {name: "Jarrah", number: 16},  
    {name: "Shephard", number: 31},  
    {name: "Kwon", number: 34}  
];

d3.selectAll("div")
    .data(data, function(d) { return d ? d.name : this.id; })
    .text(function(d) { return d.number; });
```
Useful resources

- https://d3js.org
- https://www.dashingd3js.com/
- Tutorial by Mike Bostok