Tutoriais | Intro

D3 + Observable

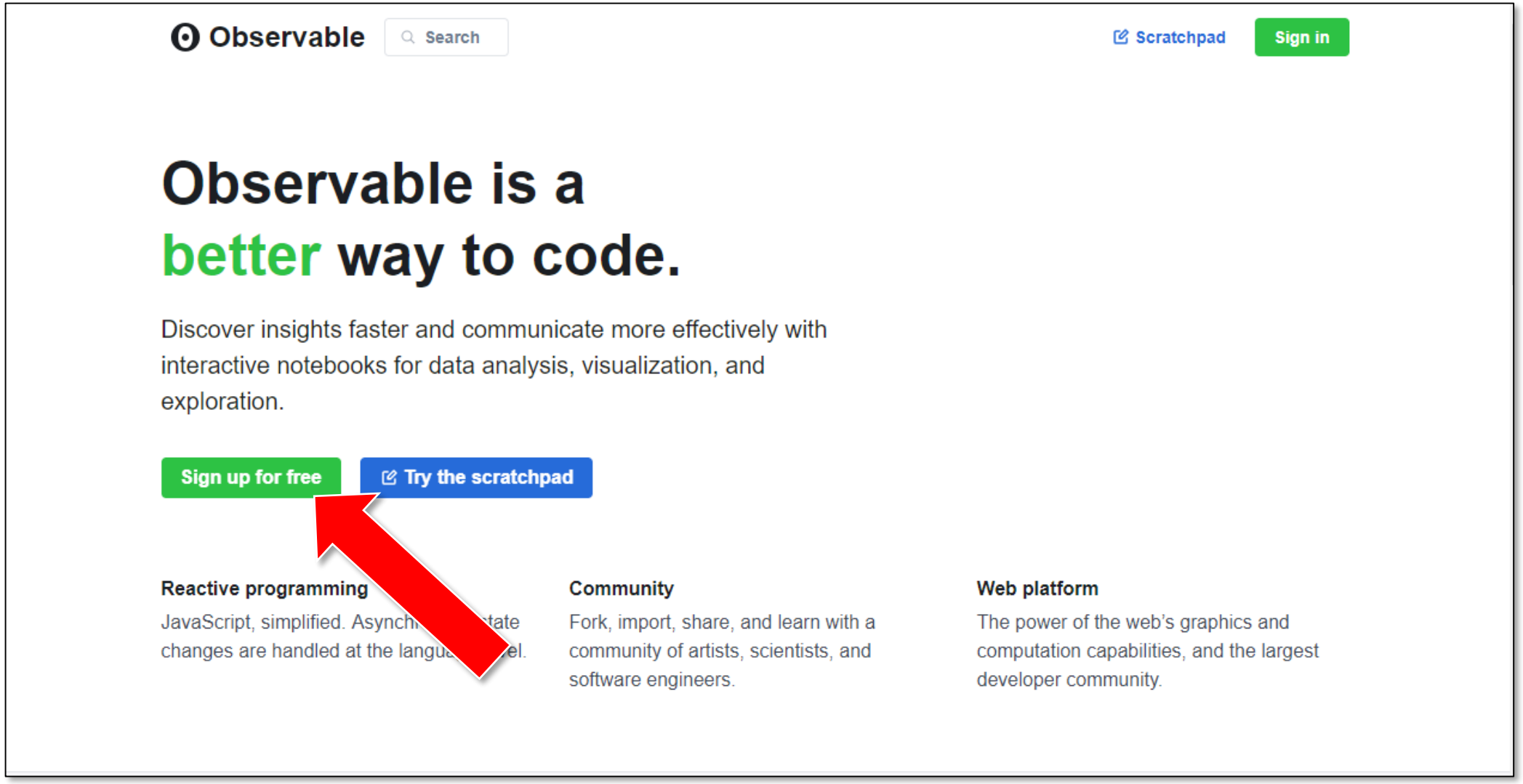
01 / 10 (segunda-feira)

python

Bokeh

08 / 10

beta.observablehq.com

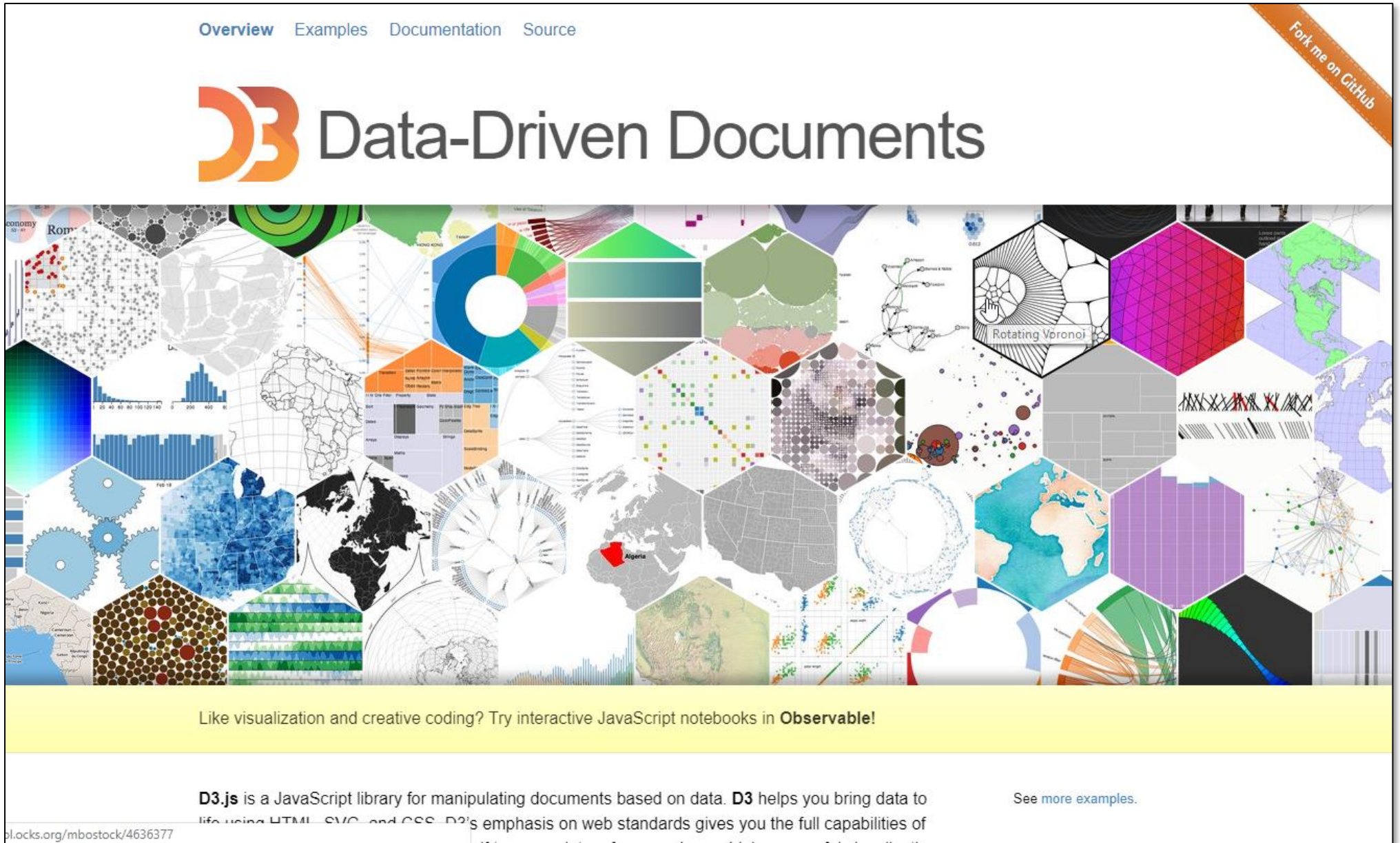


Content

https://beta.observablehq.com/d/0add62c9618c972f

Content

Why D3?


Overview
Examples
Documentation
Source
Fork me on GitHub
Data-Driven Documents
Rotating Voronoi
Algeria
Like visualization and creative coding? Try interactive JavaScript notebooks in **Observable!**
D3.js is a JavaScript library for manipulating documents based on data. **D3** helps you bring data to
See more examples.

D3.js

- ✓ "D3" stands for **Data-Driven Documents**
- ✓ is a JavaScript library for manipulating documents based on data
- ✓ works with web standards (HTML, SVG, and CSS)
  - ✓ HyperText Markup Language (HTML)
  - ✓ Scalable Vector Graphics (SVG); XML-based vector image format for graphics
  - ✓ Cascading Style Sheets (CSS); separation of document content from document presentation
- ✓ Runs in modern browsers, no proprietary framework

Before showing beautiful graphics…

What D3 is all about…?

Data

D3.js

interaction with:

- ✓ The Document Object Model (DOM)
- ✓ SVG (Scalable Vector Graphics)

HTML page

```
<html>
<head>….</head>
<body>

<div>
…
</div>

<div>
…
</div>

.
.
.

</body>
</html>
```

History

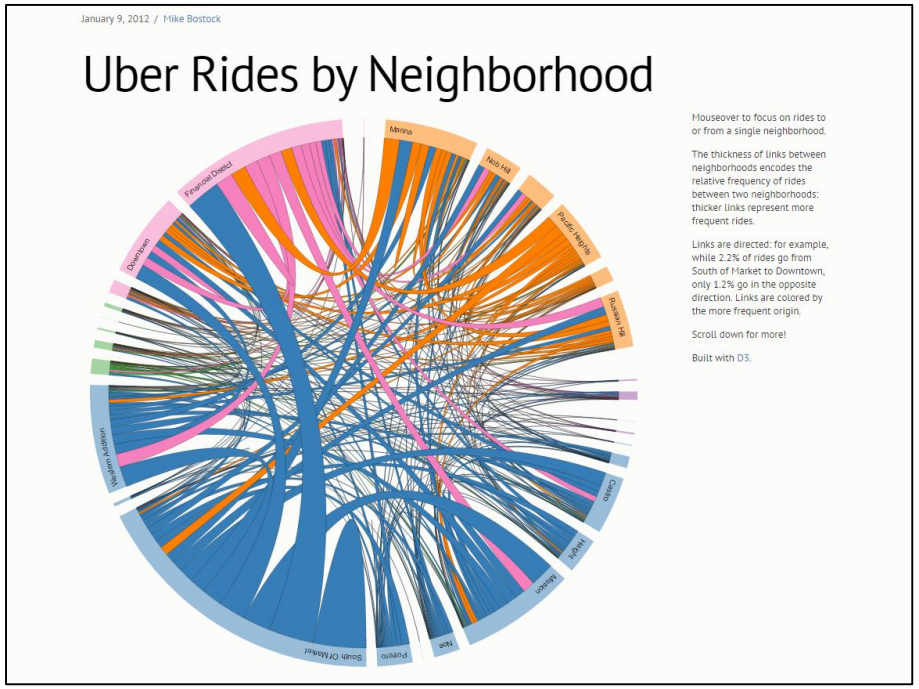
Rich internet applications

Java applet

Flash

HTML5 + JavaScript (D3 also..) + CSS

- ✓No need to install any plugin
- ✓ Run on any platform (iOS, Android, Desktops, ..)



http://bost.ocks.org/mike/uberdata/

Content

Observable https://beta.observablehq.com/@mbostock/introduction-to-notebooks

Observable Search ··· Fork + New

Mike Bostock · Dec 21, 2017
Building a better computational medium. Founder @observablehq.
Creator @d3. Former @nytgraphics. Pronounced BOSS-tock.

25

Featured in Introduction 6 forks

Introduction to Notebooks

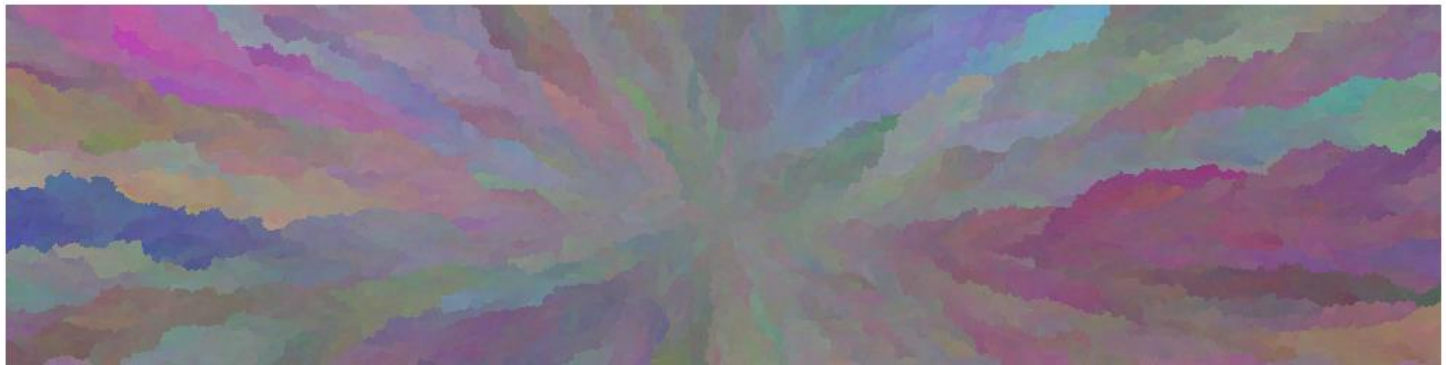
This is a *notebook*—an interactive, editable document defined by code. It's a computer program, but one that's designed to be easier to read and write by humans.

A notebook can be used as a scratchpad for exploring data, or to explain quantitative phenomena. Because the code runs live in your browser, notebooks are also a powerful medium for studying dynamic systems and algorithms. (See Stephen Wolfram's What Is a Computational Essay? for more.)

Notebooks are comprised of cells. A *cell* can be pretty much anything. For example, a cell might contain a few paragraphs of text, like this one. Or a cell might contain images, videos, charts, or other graphical and interactive elements. A cell can also contain data structures, like numbers, strings, arrays and objects. Functions, too.

[Sem título]

image



Observable https://beta.observablehq.com/@mbostock/predator-and-prey

Observable Search

Mike Bostock · Jun 8, 2018
Building a better computational medium. Founder @observablehq.
Creator @d3. Former @nytgraphics. Pronounced BOSS-tock.

Featured in Explorables 5 forks

Predator and Prey

Imagine an island of cats 🐈 and mice 🐁. If mice are plentiful, the cats eat well and reproduce rapidly. But as the cats multiply, they eat more mice. Soon the mouse population is decimated and the cats begin to starve. With fewer cats, the mice recover, and the cycle repeats.

The Lotka–Volterra equations model such a dynamic predator–prey interaction. The model is a pair of (ordinary differential) equations where 🐁 is the number of prey and 🐈 is the number of predators:

$$\frac{\mathrm{d}🐁}{\mathrm{d}t} = \alpha🐁 - \beta🐁🐈$$

$$\frac{\mathrm{d}🐈}{\mathrm{d}t} = \delta🐁🐈 - \gamma🐈$$

In this model, the prey have unlimited food and grow exponentially (α); however, they are sometimes eaten (β) by predators. The predators have unlimited appetite, but eat only prey, limiting their growth (δ); their decay is exponential (γ). The parameters α, β, γ, and δ thus describe how the two populations behave, while $\frac{\mathrm{d}🐁}{\mathrm{d}t}$ and $\frac{\mathrm{d}🐈}{\mathrm{d}t}$ are the rates at which each population is increasing or decreasing.

By iteratively evaluating these equations—by starting with some initial populations, calculating the rates, adjusting the populations accordingly, then repeating this process many times—we can compute the populations over time as shown in the plot below. Drag the cat 🐈 or the mouse 🐁 to change the initial populations. (You can't have fractional mice, so interpret the numbers as thousands or similar.)

Observable

https://beta.observablehq.com/@jashkenas/inputs

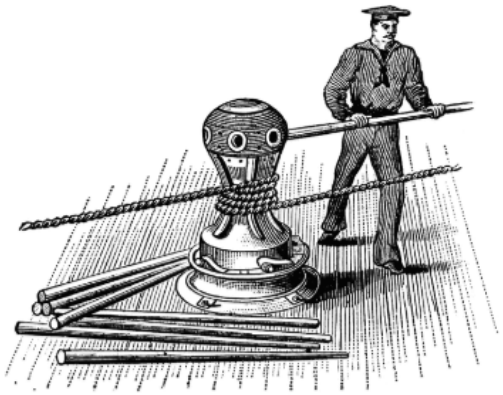
Observable Search

Jeremy Ashkenas · Jan 24, 2018
Working on @observablehq.

Featured in Libraries and Techniques 14 forks

Inputs

a.k.a "The Grand Input Bazaar"



A collection of assorted fancy inputs, odds and ends — with which to produce values to feed your burgeoning sketches. All inputs support optional **titles** and **descriptions**; where it makes sense, inputs also support a **submit** option, which allows you to prevent the value from updating until the input has been finalized.

Wares we have on offer:

- `slider`
- `button`
- `select`
- `color`
- `date`

Content

D3 “chain syntax”

```
d3.select("body").append("p").text("New paragraph!");
```

..or for better legibility:

```
d3.select("body")
    .append("p")
    .text("New paragraph!");
```

JavaScript, like HTML, doesn’t care about whitespace and line breaks.

Now, the same code without chain syntax:

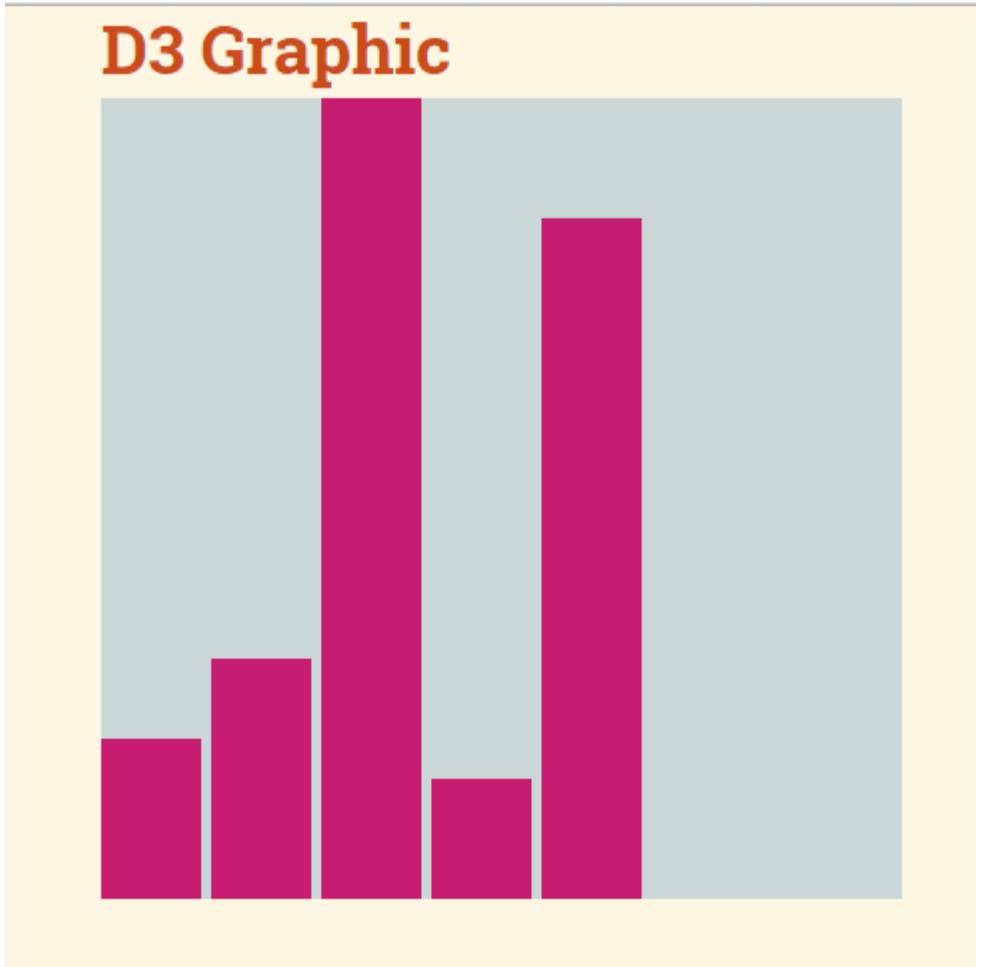
```
var body = d3.select("body");
var p = body.append("p");
p.text("New paragraph!");
```

Reference:
http://alignedleft.com/tutorials/d3/chaining-methods

Loading data

Arrays

```
var bardata = [20, 30, 100, 15, 85];

var height = 400,
    width = 400,
    barWidth = 50,
    barOffset = 5;

var yScale = d3.scale.linear()
     . domain([0, d3.max(bardata)])
     . range([0, height])

d3.select('#chart').append('svg')
    .attr('width', width)
    .attr('height', height)
    .style('background', '#C9D7D6')
    .selectAll('rect').data(bardata)
    .enter().append('rect')
        .style('fill', '#C61C6F')
        .attr('width', barWidth)
        .attr('height', function(d) {
            return yScale(d);
        })
        .attr('x', function(d,i) {
            return i * (barWidth + barOffset);
        })
        .attr('y', function(d) {
            return height - yScale(d);
        })
```



Flexibility for varying sizes of input data

Short example – data driven approach

```
var bardata = [20, 30, 100, 15, 85];
```

“Empty” selection

```
var myChart = d3.select('#chart').append('svg')
    .attr('width', width)
    .attr('height', height)
    .selectAll('rect').data(bardata)
    .enter().append('rect')
        .style('fill', function(d,i) {
            return colors(i);
        })
        .attr('width', xScale.rangeBand())
        .attr('x', function(d,i) {
            return xScale(i);
        })
        .attr('height', 0)
        .attr('y', height)
```

Adding elements

Looping through data (but not with an explicit “for” loop).

Loading data

CSV files

```
d3.csv("somefiles.csv", function(data) {doSomethingWithData(data)});
```

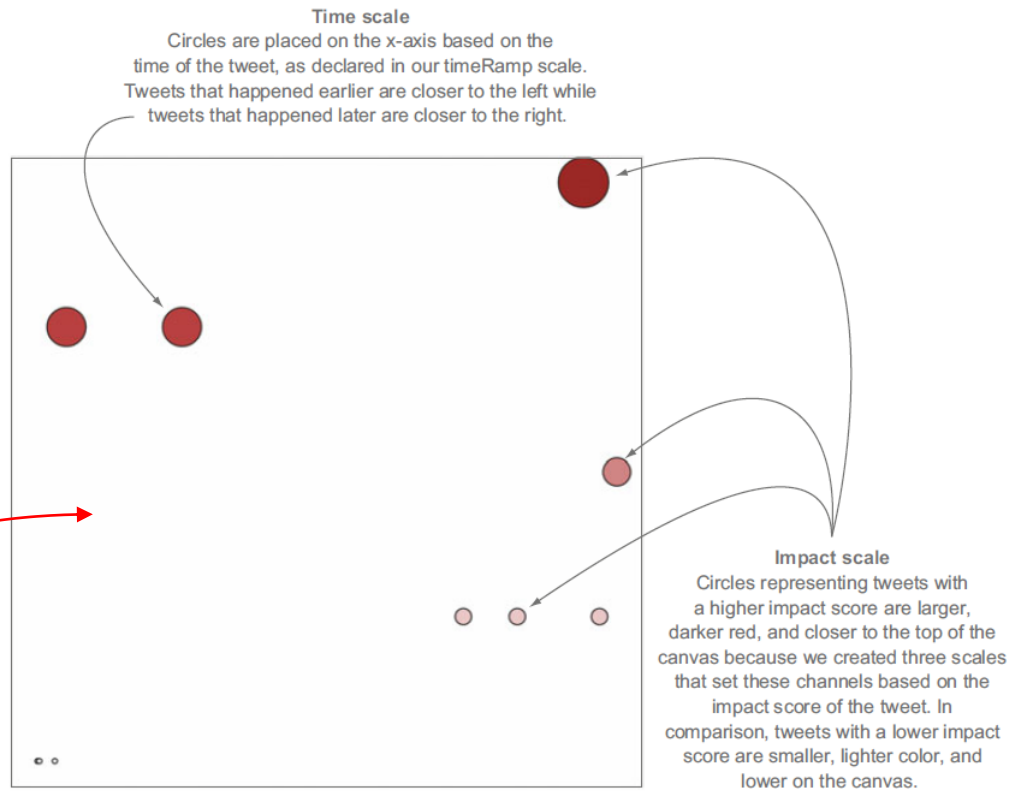
```
"label","population","country","x","y"
"San Francisco", 750000,"USA",122,-37
"Fresno", 500000,"USA",119,-36
"Lahore",12500000,"Pakistan",74,31
"Karachi",13000000,"Pakistan",67,24
"Rome",2500000,"Italy",12,41
```

Loading data

JSON files

```
d3.json("tweets.json", function(data) {doSomethingWithData(data)});
```


```
{
"tweets": [
{"user": "Al", "content": "I really love seafood.",
   "timestamp": " Mon Dec 23 2013 21:30 GMT-0800 (PST)",
   "retweets": ["Raj","Pris","Roy"], "favorites": ["Sam"]},
{"user": "Al", "content": "I take that back, this doesn't taste so good.",
   "timestamp": "Mon Dec 23 2013 21:55 GMT-0800 (PST)",
   "retweets": ["Roy"], "favorites": []},
{"user": "Al",
   "content": "From now on, I'm only eating cheese sandwiches.",
   "timestamp": "Mon Dec 23 2013 22:22 GMT-0800 (PST)",
   "retweets": [],"favorites": ["Roy","Sam"]},
{"user": "Roy", "content": "Great workout!",
   "timestamp": " Mon Dec 23 2013 7:20 GMT-0800 (PST)",
   "retweets": [],"favorites": []},
{"user": "Roy", "content": "Spectacular oatmeal!",
   "timestamp": " Mon Dec 23 2013 7:23 GMT-0800 (PST)",
   "retweets: [],"favorites": []},
{"user": "Roy", "content": "Amazing traffic!",
   "timestamp": " Mon Dec 23 2013 7:47  GMT-0800 (PST)",
   "retweets": [],"favorites": []},
{"user": "Roy", "content": "Just got a ticket for texting and driving!",
   "timestamp": " Mon Dec 23 2013 8:05 GMT-0800 (PST)",
   "retweets": [],"favorites": ["Sam", "Sally", "Pris"]},
{"user": "Pris", "content": "Going to have some boiled eggs.",
   "timestamp": " Mon Dec 23 2013 18:23 GMT-0800 (PST)",
   "retweets": [],"favorites": ["Sally"]},
```

(Meeks, Elijah. "D3. js in Action." (2015))

Loading data

..and also:

Loading External Resources

- d3.csv - request a comma-separated values (CSV) file.
- d3.html - request an HTML document fragment.
- d3.json - request a JSON blob.
- d3.text - request a text file.
- d3.tsv - request a tab-separated values (TSV) file.
- d3.xhr - request a resource using XMLHttpRequest.
- d3.xml - request an XML document fragment.

D3 API Reference
https://github.com/d3/d3/blob/master/API.md

Selecting elements

```
d3.selectAll("body").style("background-color", "black");
```

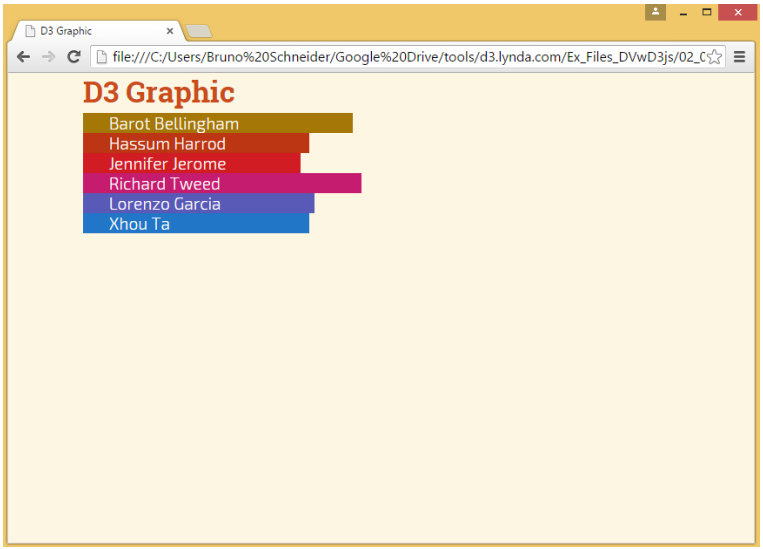
<u>Or by class:</u>

```
d3.selectAll(".container")…
```

<u>Or by ID:</u>

```
d3.select("#dataPoint1“)…
```

D3 selection: example

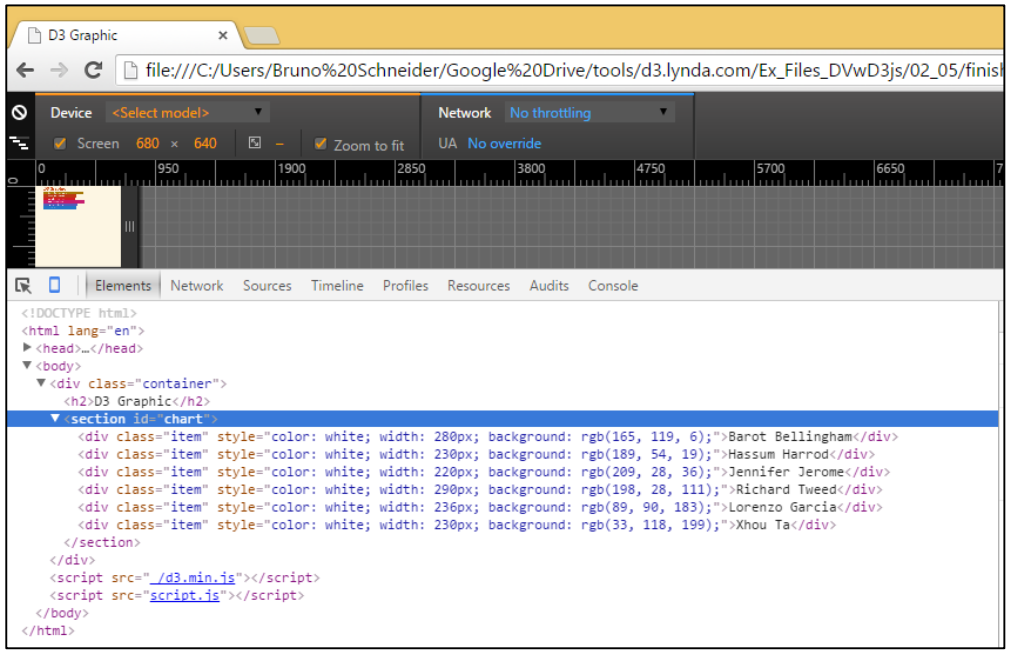


index.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>D3 Graphic</title>
  <link rel="stylesheet" href="_/base.css">
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <div class="container">
    <h2>D3 Graphic</h2>
    <section id="chart">
      <div class="item">Barot Bellingham</div>
      <div class="item">Hassum Harrod</div>
      <div class="item">Jennifer Jerome</div>
      <div class="item">Richard Tweed</div>
      <div class="item">Lorenzo Garcia</div>
      <div class="item">Xhou Ta</div>
    </section>
  </div>
  <script src="_/d3.min.js"></script>
  <script src="script.js"></script>
</body>
</html>
```

script.js

```javascript
var myStyles = [
  { width: 200,
    color: '#A57706'},
  { width: 230,
    color: '#BD3613'},
  { width: 220,
    color: '#D11C24'},
  { width: 290,
    color: '#C61C6F'},
  { width: 236,
    color: '#595AB7'},
  { width: 230,
    color: '#2176C7'}
];

d3.selectAll('.item')
  .data(myStyles)
  .style({
    'color': 'white',
    'background' : function(d) {
      return d.color;
    },
    width : function(d) {
      return d.width + 'px';
    }
  })
```



HTML tag

« Previous Complete HTML Reference Next »

Example

A section in a document that will be displayed in blue:

```
<div style="color:#0000FF">
  <h3>This is a heading</h3>
  <p>This is a paragraph.</p>
</div>
```

Try it yourself »

Definition and Usage

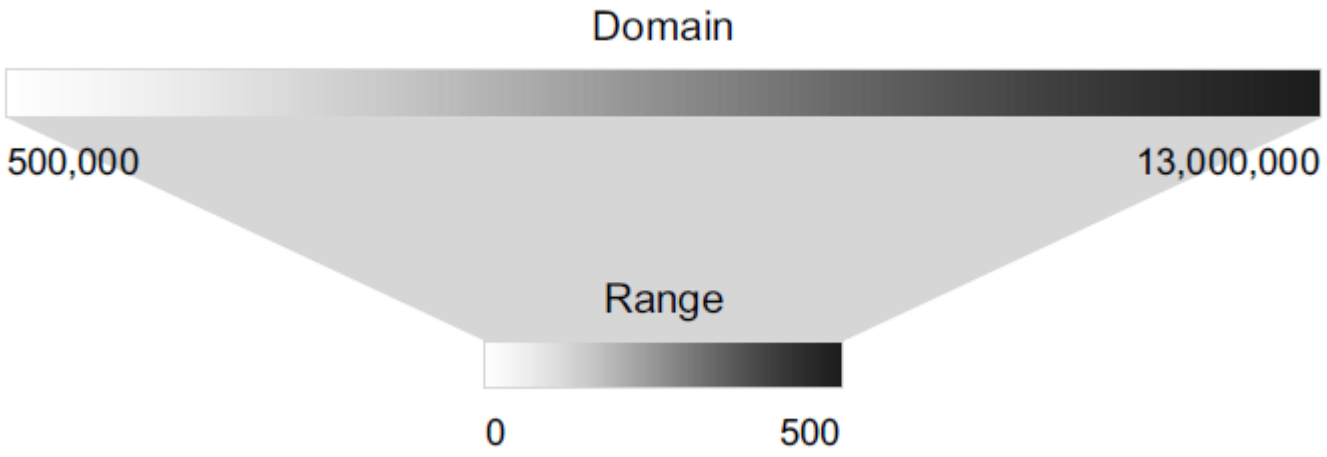
The tag defines a division or a section in an HTML document.

The tag is used to group block-elements to format them with CSS.

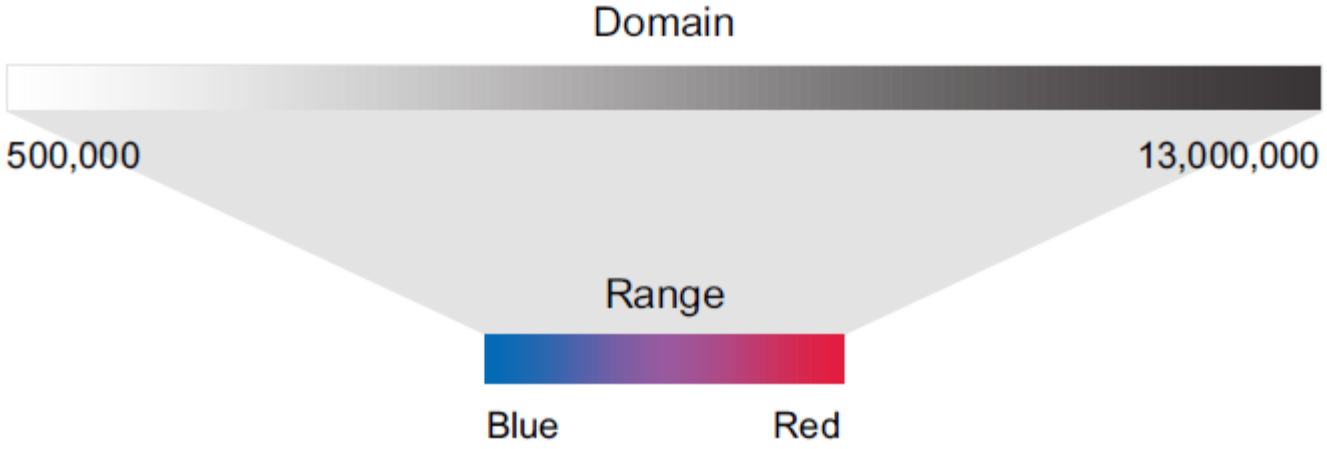
Browser Support

Element						
	Yes	Yes	Yes	Yes	Yes	Yes

D3 scales



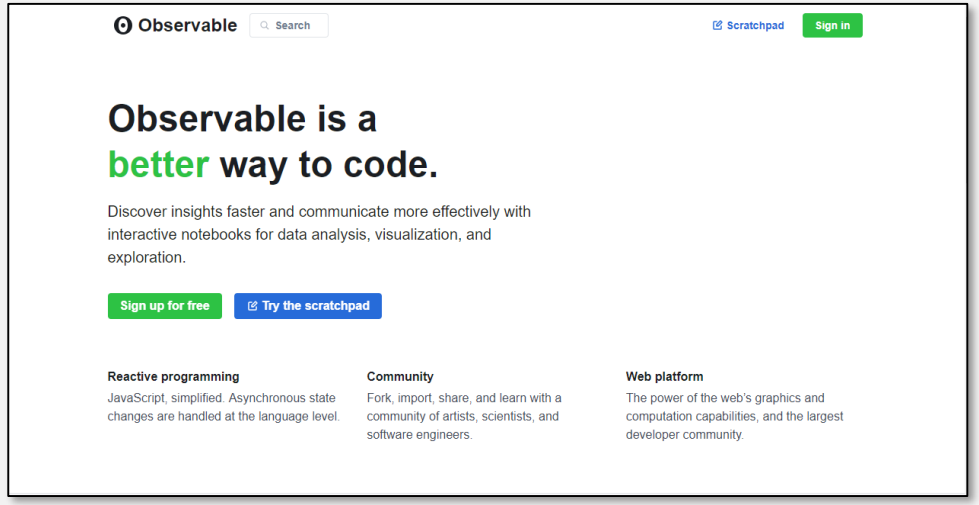

```
var newRamp = d3.scale.linear().domain([500000,13000000]).range([0, 500]);
```

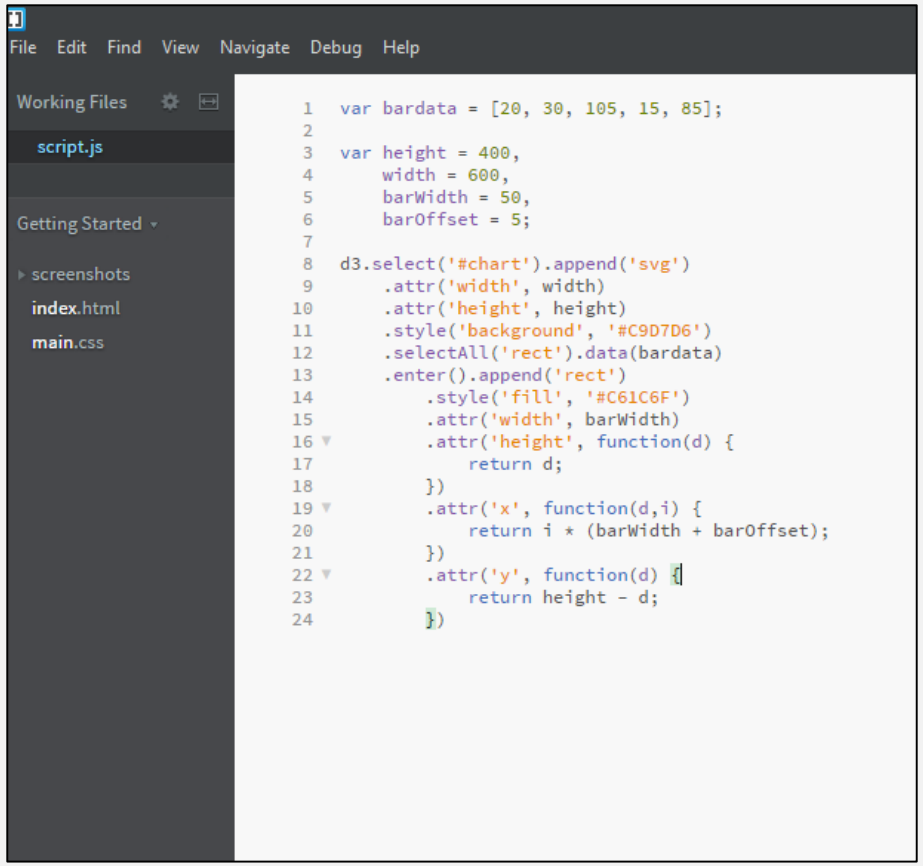



```
var newRamp = d3.scale.linear().domain([500000,13000000]).range(["blue",
"red"]);
```

Content

https://beta.observablehq.com/d/0add62c9618c972f


Observable
Search
Scratchpad
Sign in
Observable is a
better way to code.
Discover insights faster and communicate more effectively with interactive notebooks for data analysis, visualization, and exploration.
Sign up for free
Try the scratchpad
Reactive programming
JavaScript, simplified. Asynchronous state changes are handled at the language level.
Community
Fork, import, share, and learn with a community of artists, scientists, and software engineers.
Web platform
The power of the web's graphics and computation capabilities, and the largest developer community.


File Edit Find View Navigate Debug Help
Working Files
script.js
Getting Started
screenshots
index.html
main.css
var bardata = [20, 30, 105, 15, 85];
var height = 400,
width = 600,
barWidth = 50,
barOffset = 5;
d3.select('#chart').append('svg')
.attr('width', width)
.attr('height', height)
.style('background', '#C9D7D6')
.selectAll('rect').data(bardata)
.enter().append('rect')
.style('fill', '#C61C6F')
.attr('width', barWidth)
.attr('height', function(d) {
return d;
})
.attr('x', function(d,i) {
return i * (barWidth + barOffset);
})
.attr('y', function(d) {
return height - d;
})

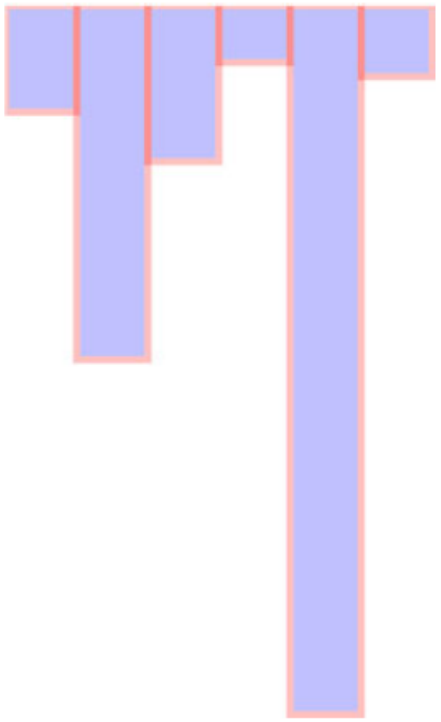
Observable

https://beta.observablehq.com/d/0add62c9618c972f

D3/Observable Tutorial

EMAp FGV

barChart

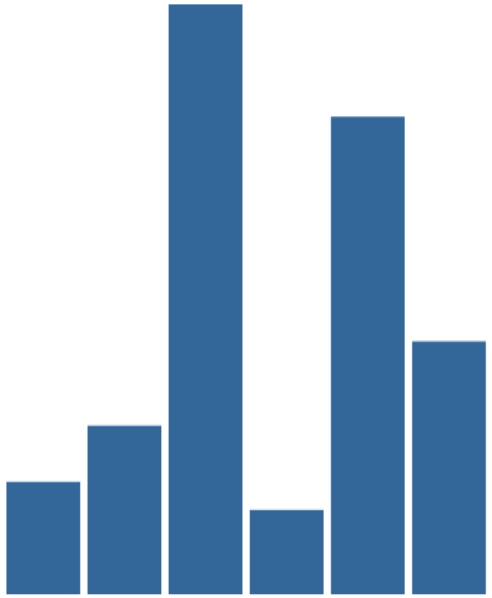


SVG rectangles are drawn from top to bottom.

Observable

https://beta.observablehq.com/d/0add62c9618c972f

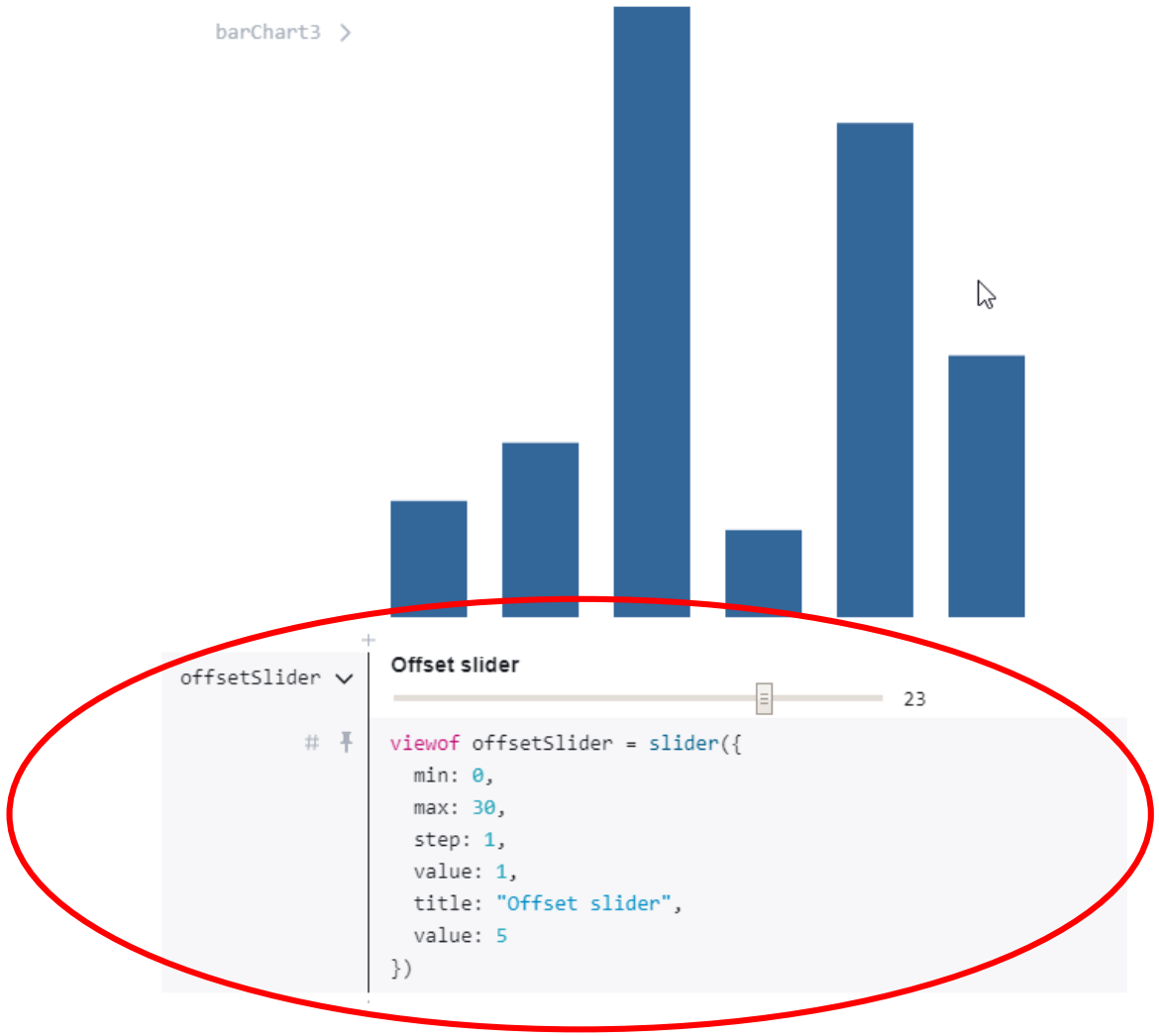
barChart2



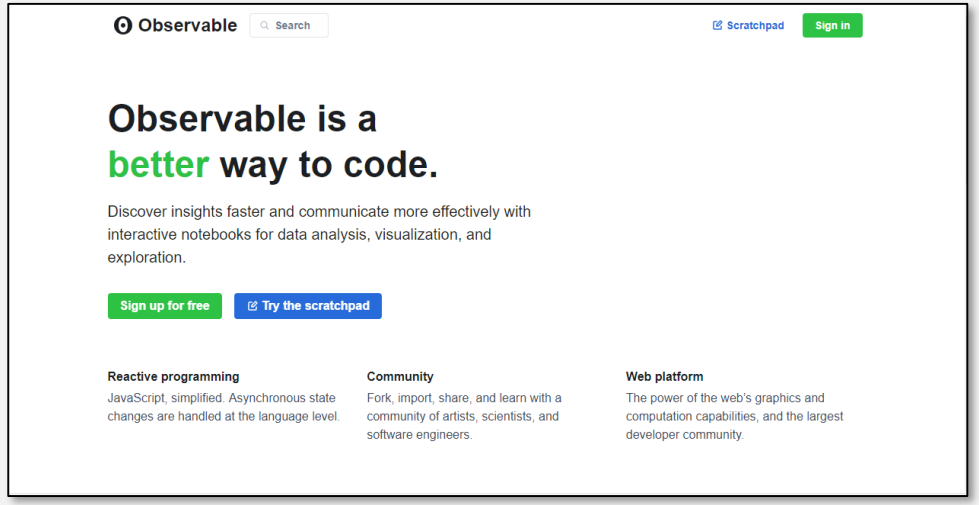
```
barChart2 = {

var bardata = [20, 30, 105, 15, 85, 45];

var height = 400,
    width = 600,
    barWidth = 50,
    barOffset = 5;

const svg = d3.select(DOM.svg(width, height));


var yScale = d3.scaleLinear()
    .domain([0, d3.max(bardata)])
    .range([0, height])
```

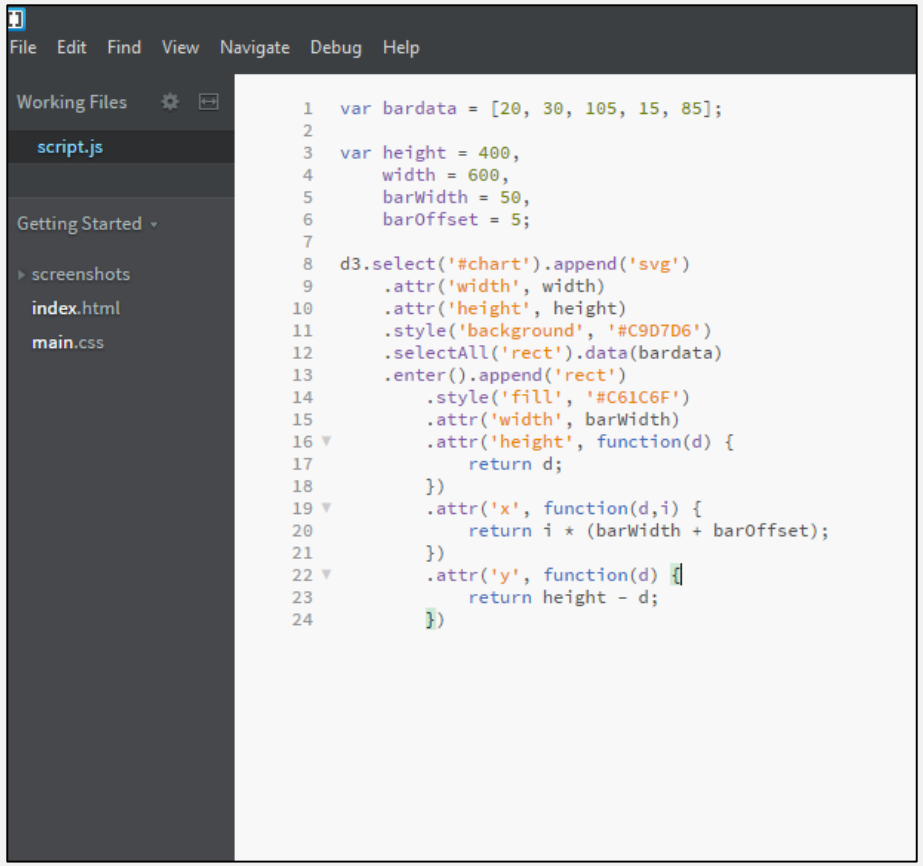
Observable

https://beta.observablehq.com/d/0add62c9618c972f



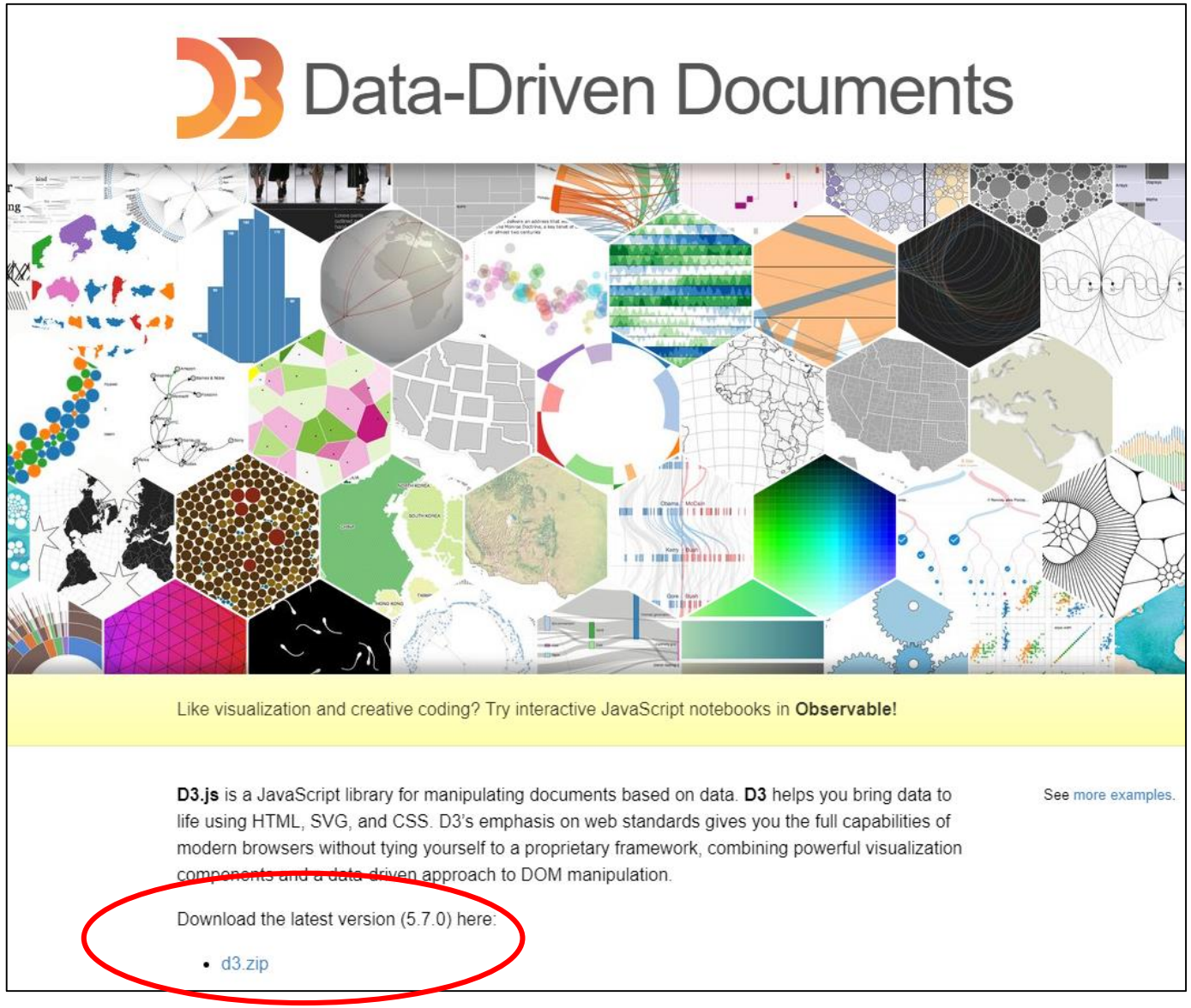
Content


Observable
Search
Scratchpad
Sign in
Observable is a
better way to code.
Discover insights faster and communicate more effectively with interactive notebooks for data analysis, visualization, and exploration.
Sign up for free
Try the scratchpad
Reactive programming
JavaScript, simplified. Asynchronous state changes are handled at the language level.
Community
Fork, import, share, and learn with a community of artists, scientists, and software engineers.
Web platform
The power of the web's graphics and computation capabilities, and the largest developer community.


File Edit Find View Navigate Debug Help
Working Files
script.js
Getting Started
screenshots
index.html
main.css
var bardata = [20, 30, 105, 15, 85];
var height = 400,
width = 600,
barWidth = 50,
barOffset = 5;
d3.select('#chart').append('svg')
.attr('width', width)
.attr('height', height)
.style('background', '#C9D7D6')
.selectAll('rect').data(bardata)
.enter().append('rect')
.style('fill', '#C61C6F')
.attr('width', barWidth)
.attr('height', function(d) {
return d;
})
.attr('x', function(d,i) {
return i * (barWidth + barOffset);
})
.attr('y', function(d) {
return height - d;
})

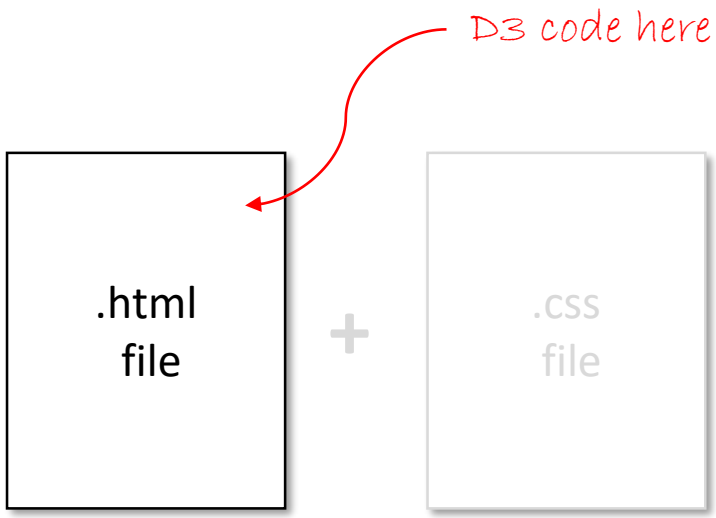
Downloading D3

http://d3js.org/



Organizing the code

1.

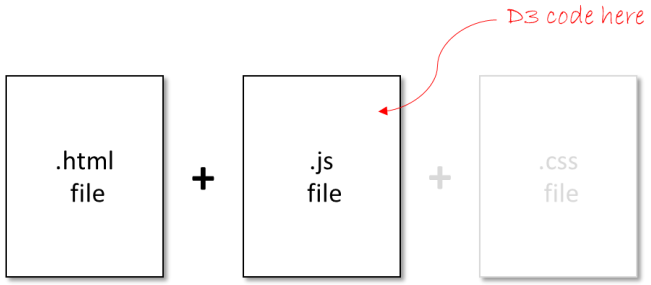


2.

(preferred way)

Example

2.




```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>D3 Graphic</title>
 </head>
<body>
  <div class="container">
    <h2>D3 Graphic</h2>
    <div id="chart"></div>
  </div>
  <script src="_/d3.min.js"></script>
  <script src="script.js"></script>
 </body>
 </html>
```

script.js

```javascript
var bardata = [20, 30, 20, 15, 40, 200];

var height = 400,
    width = 600,
    barWidth = 50,
    barOffset = 5;

var yScale = d3.scale.linear()
        .domain([0, d3.max(bardata)])
        .range([0, height])

d3.select('#chart').append('svg')
    .attr('width', width)
    .attr('height', height)
    .style('background', '#C9D7D6')
    .selectAll('rect').data(bardata)
    .enter().append('rect')
        .style('fill', '#C61C6F')
        .attr('width', barWidth)
        .attr('height', function(d) {
            return yScale(d);
        })
        .attr('x', function(d,i) {
            return i * (barWidth + barOffset);
        })
        .attr('y', function(d) {
            return height - yScale(d);
        })
```

Loading D3.js

Local storage

OR

Online

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>D3 Graphic</title>
 </head>
<body>
  <div class="container">
    <h2>D3 Graphic</h2>
    <div id="chart"></div>
  </div>
  <script src="_/d3.min.js"></script>
  <script src="script.js"></script>
</body>
</html>
```

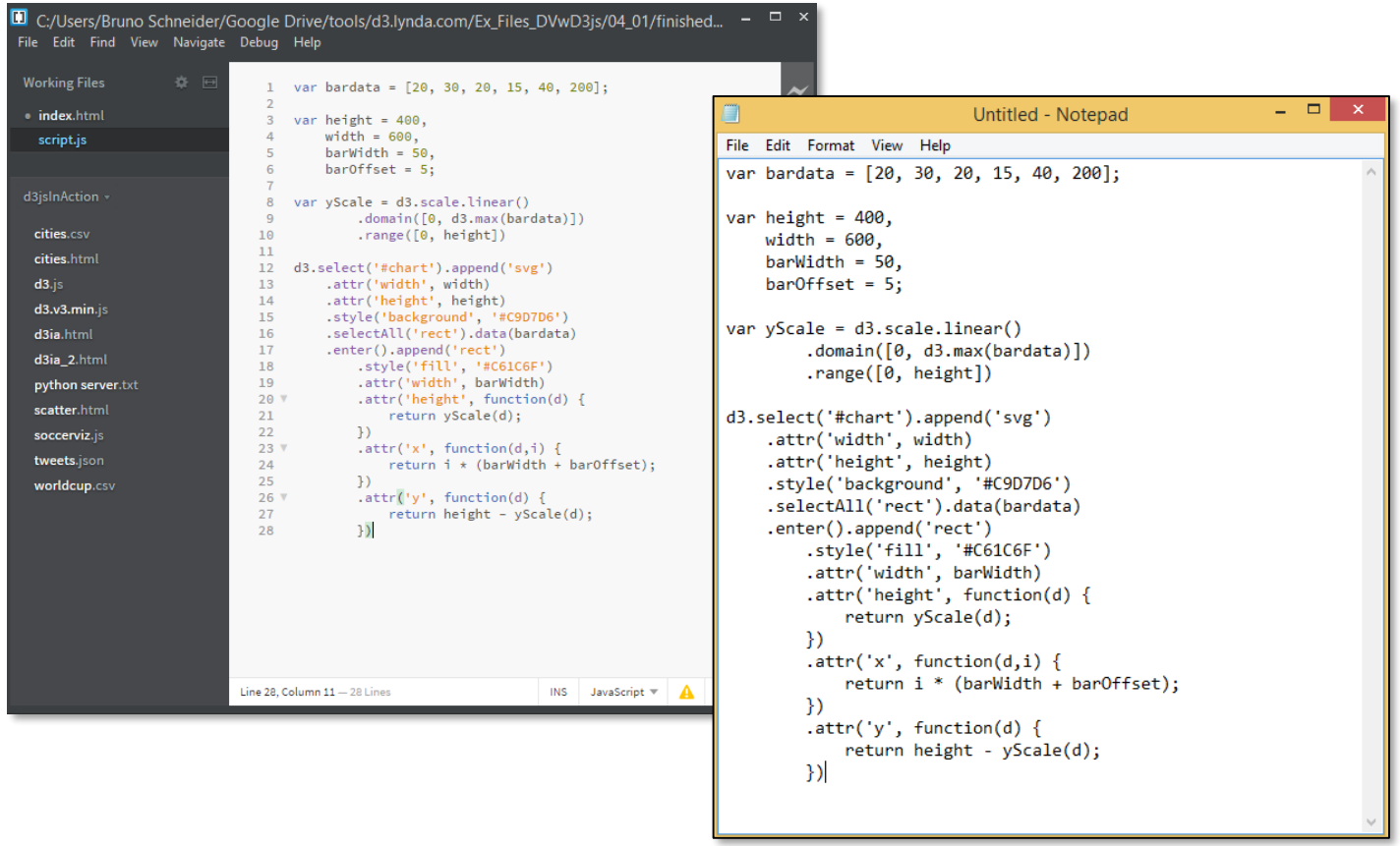
.html file

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>D3 Graphic</title>
 </head>
<body>
  <div class="container">
    <h2>D3 Graphic</h2>
    <div id="chart"></div>
  </div>
  <script src="https://d3js.org/d3.v4.min.js"></script>
  <script src="script.js"></script>
</body>
</html>
```

.html file

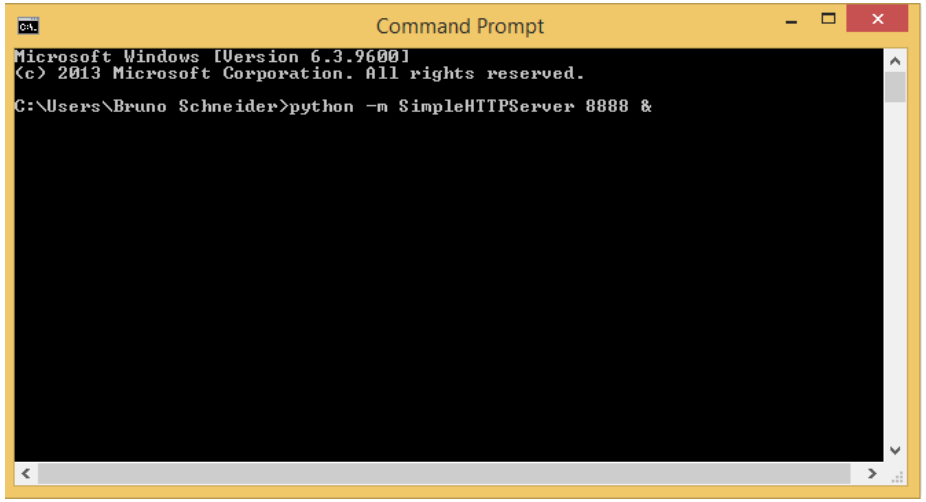
Working with D3

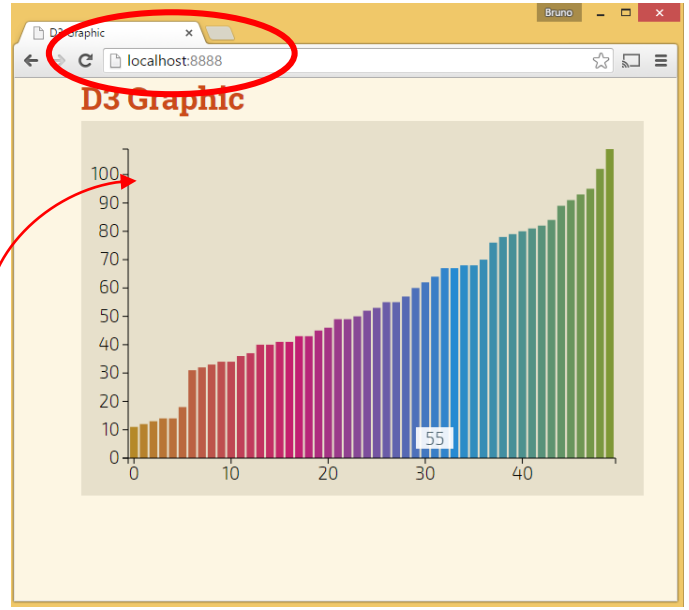
✓Any text editor, HTML/Javascript IDE of your choice



Working with D3

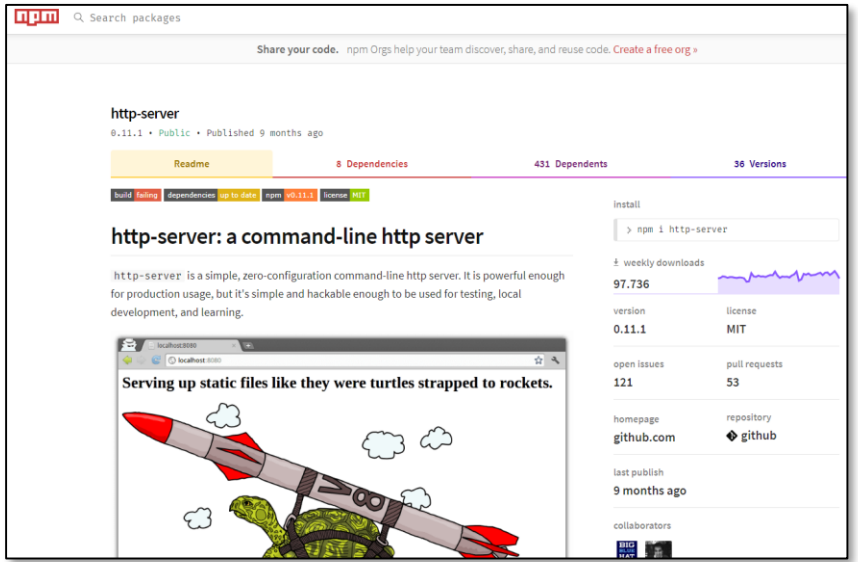
✓ running Python's built-in server





python -m SimpleHTTPServer 8888 &

✓ .. Or npm http-server

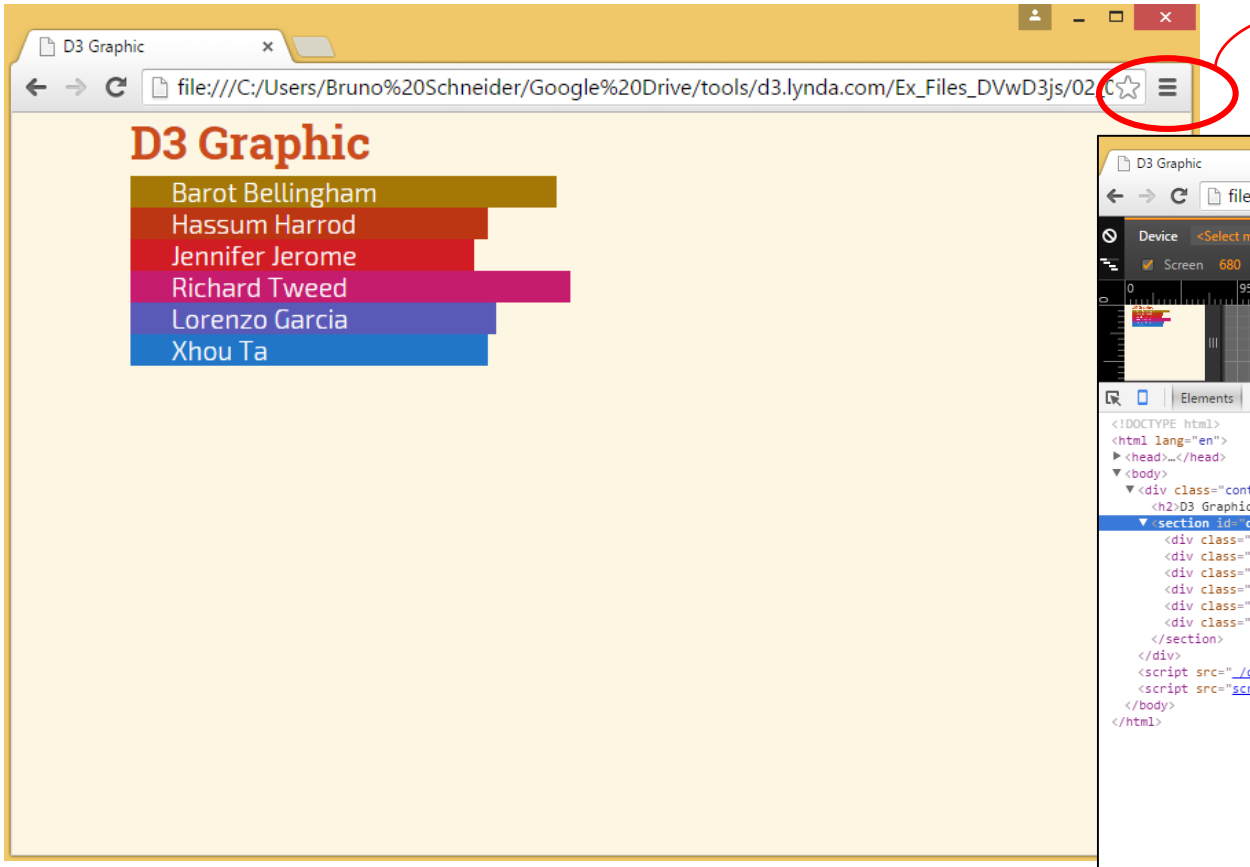


.. due to restrictions of the browser for reading external files

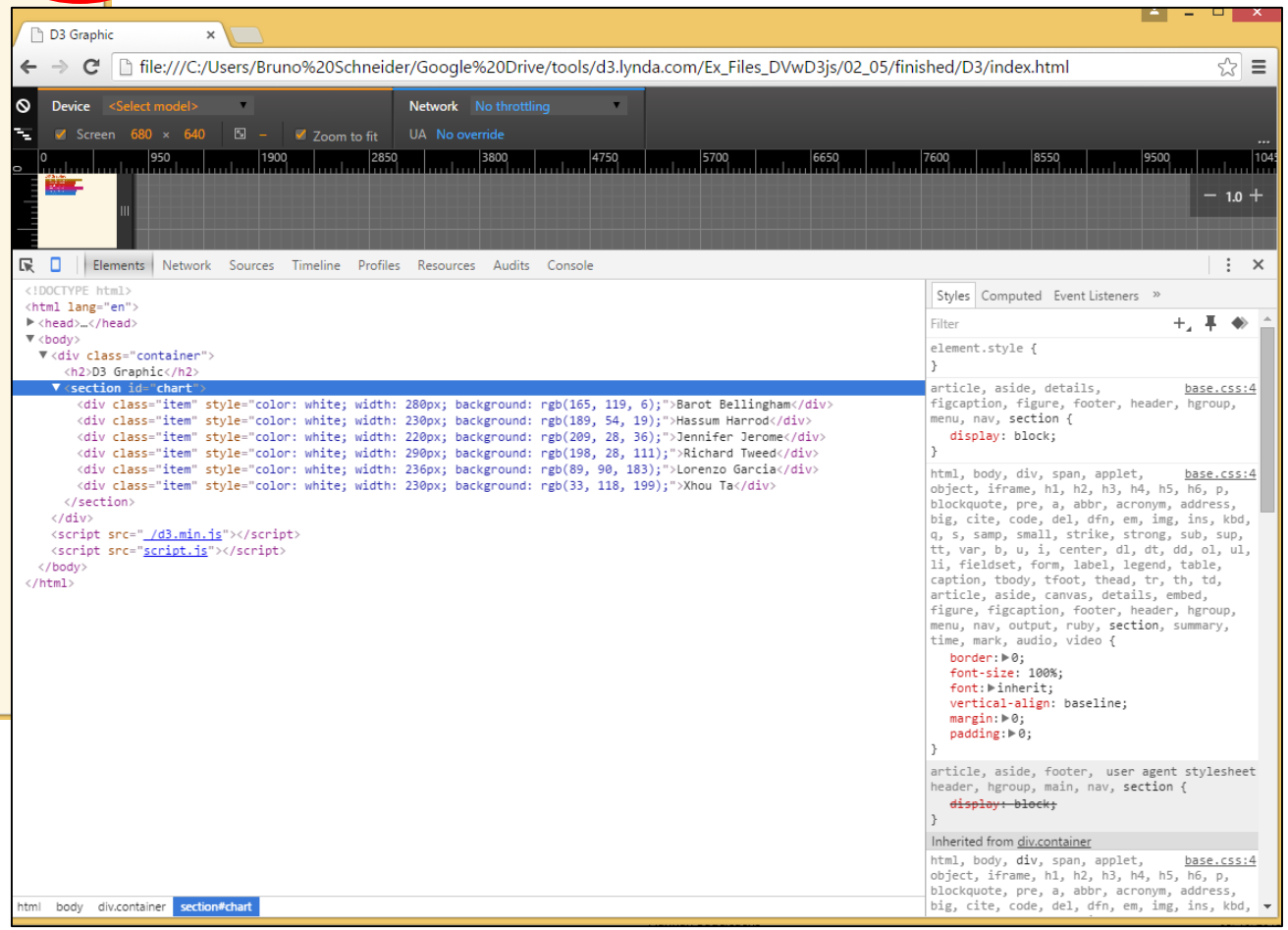
The **developer tools** in Chrome

Show example

- ✓ Shows the current state of the DOM

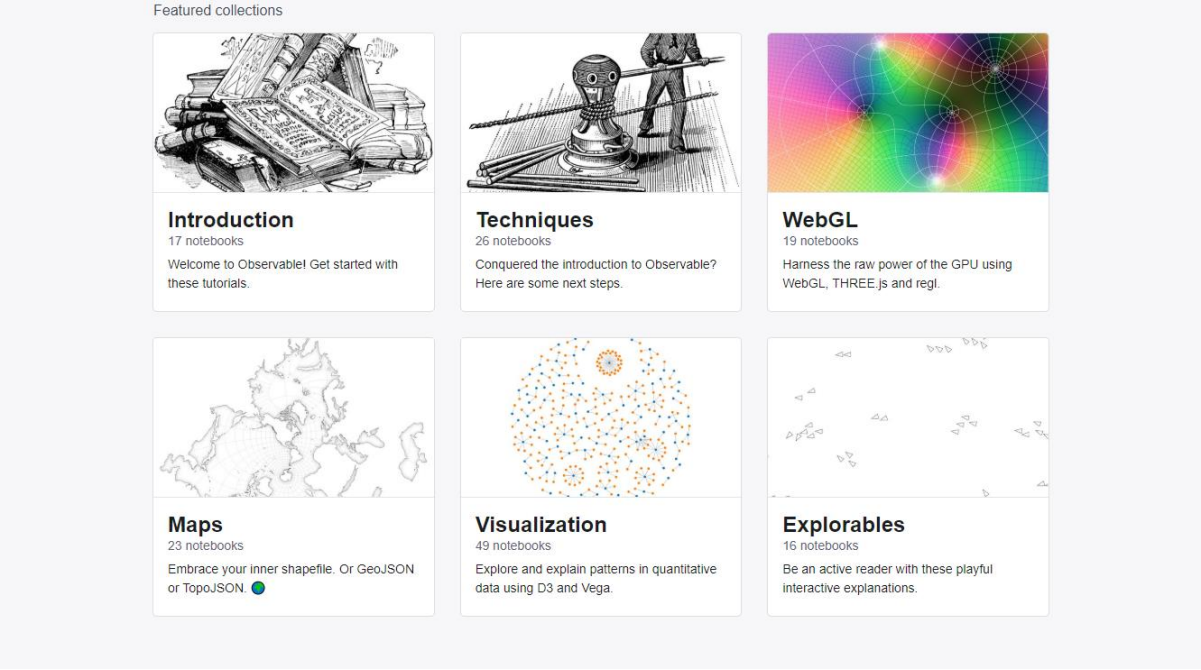


More tools / Developer tools

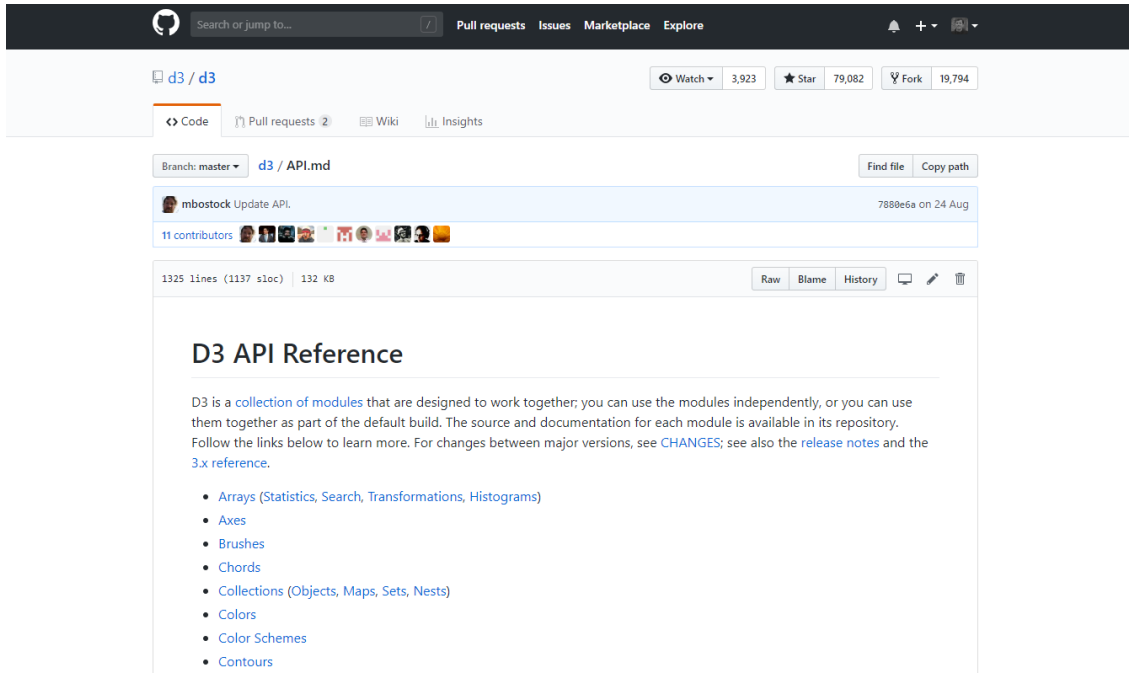


References

beta.observablehq.com

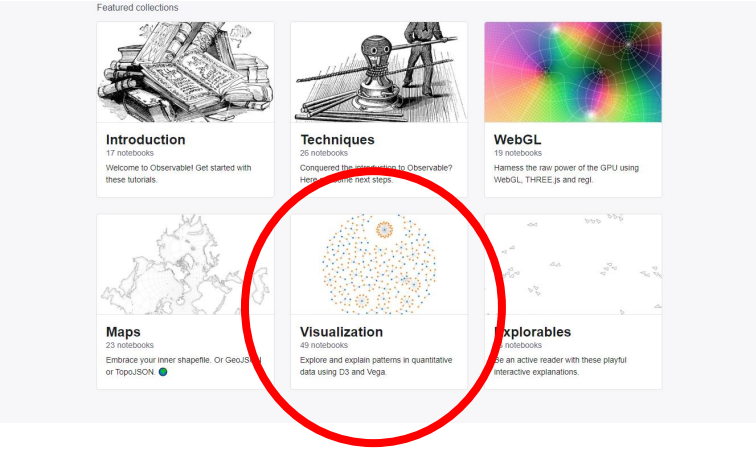


https://github.com/d3/d3/blob/master/API.md



References

https://beta.observablehq.com/collection/@observablehq/visualization



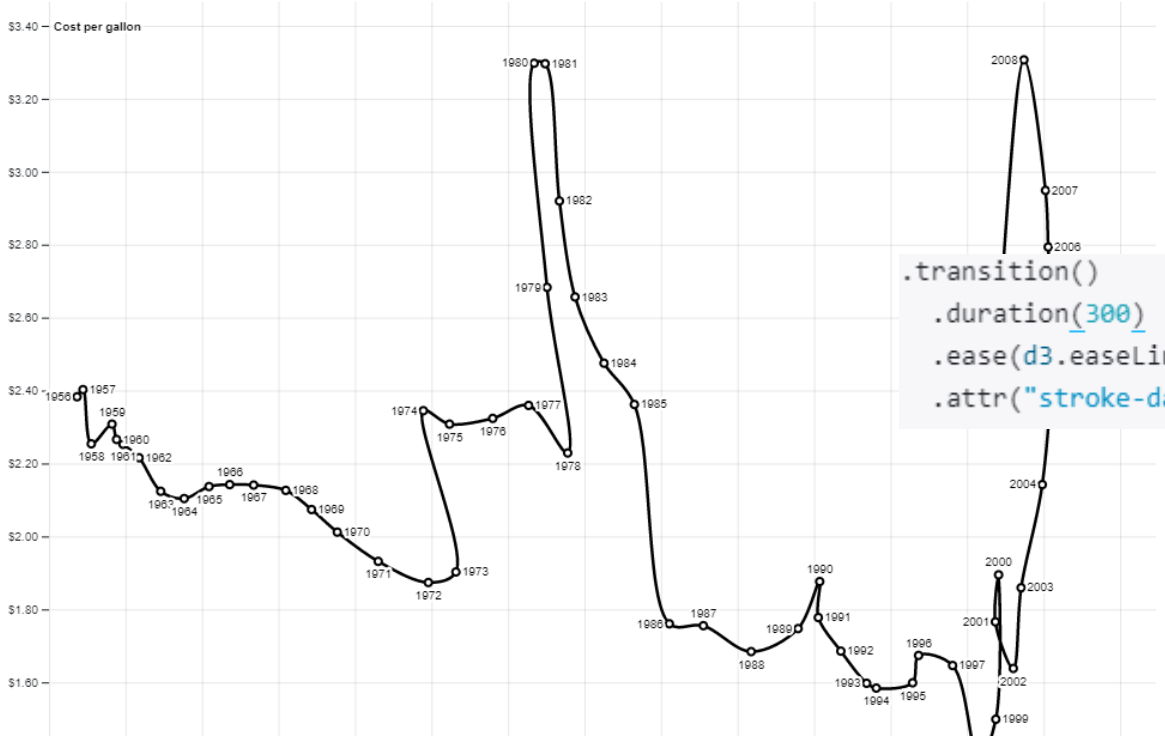
Observable

D3 Connected Scatterplot

This is a recreation of one of my favorite graphics, Hannah Fairfield's *Driving Shifts Into Reverse* from 2010. Be sure to read the annotations of the original! See also Fairfield's *Driving Safety, in Fits and Starts* from 2012, Noah Veltman's variation of this graphic, and a paper on connected scatterplots by Haroz *et al.*

Data: Hannah Fairfield

chart

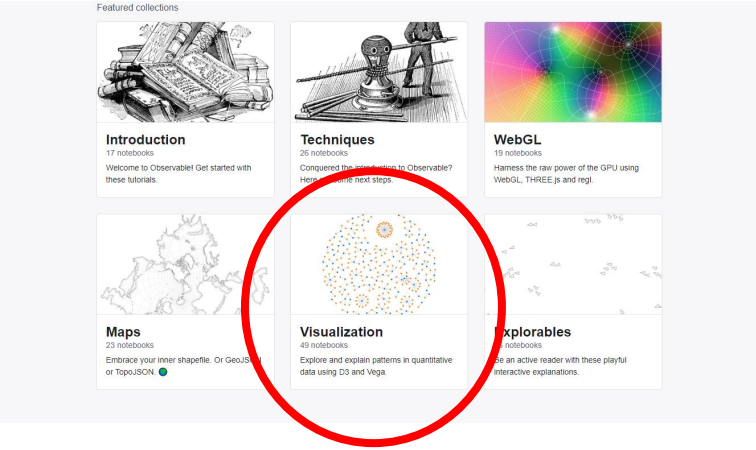


Fork

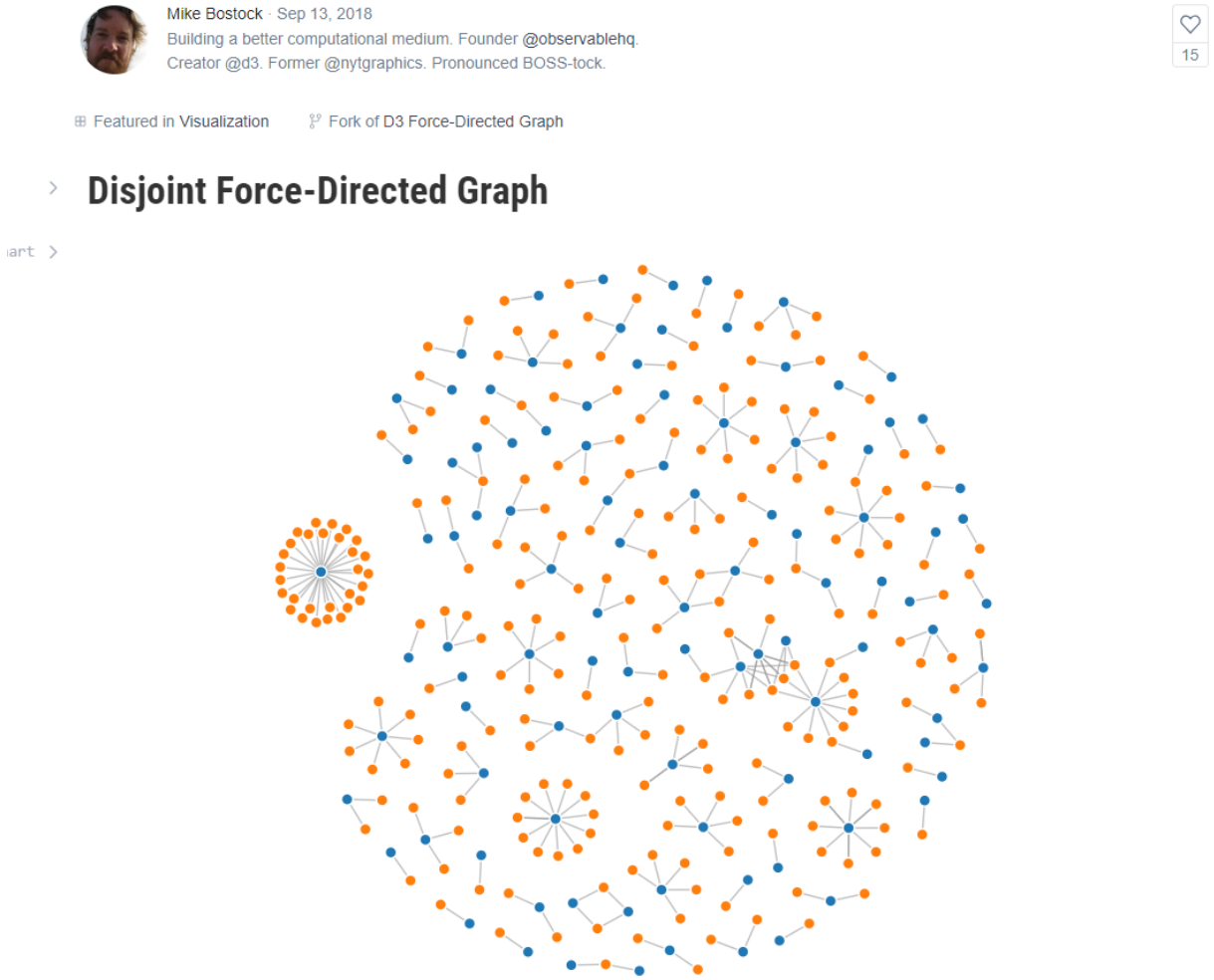
```
.transition()
  .duration(300)
  .ease(d3.easeLinear)
  .attr("stroke-dasharray", `${l},${l}`);
```

References

https://beta.observablehq.com/collection/@observablehq/visualization



Observable



Muito obrigado pela atenção...!

python

Bokeh

08 / 10