

# Tutoriais | Intro




01 / 10 (segunda-feira)



08 / 10

beta.observablehq.com

 **Observable**

[Scratchpad](#)

[Sign in](#)

[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.

# Content

- 1 D3 Intro
- 2 Observable notebooks
- 3 D3 Basics
- 4 Working with D3 in Observable
- 5 Running D3 locally

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

# Content

**1 D3 Intro**

2 Observable notebooks


3 D3 Basics

4 Working with D3 in Observable

5 Running D3 locally

# Why D3?

[Overview](#) [Examples](#) [Documentation](#) [Source](#)



## Data-Driven Documents

[Fork me on GitHub](#)



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 life using **HTML, SVG, and CSS**. **D3's** emphasis on web standards gives you the full capabilities of

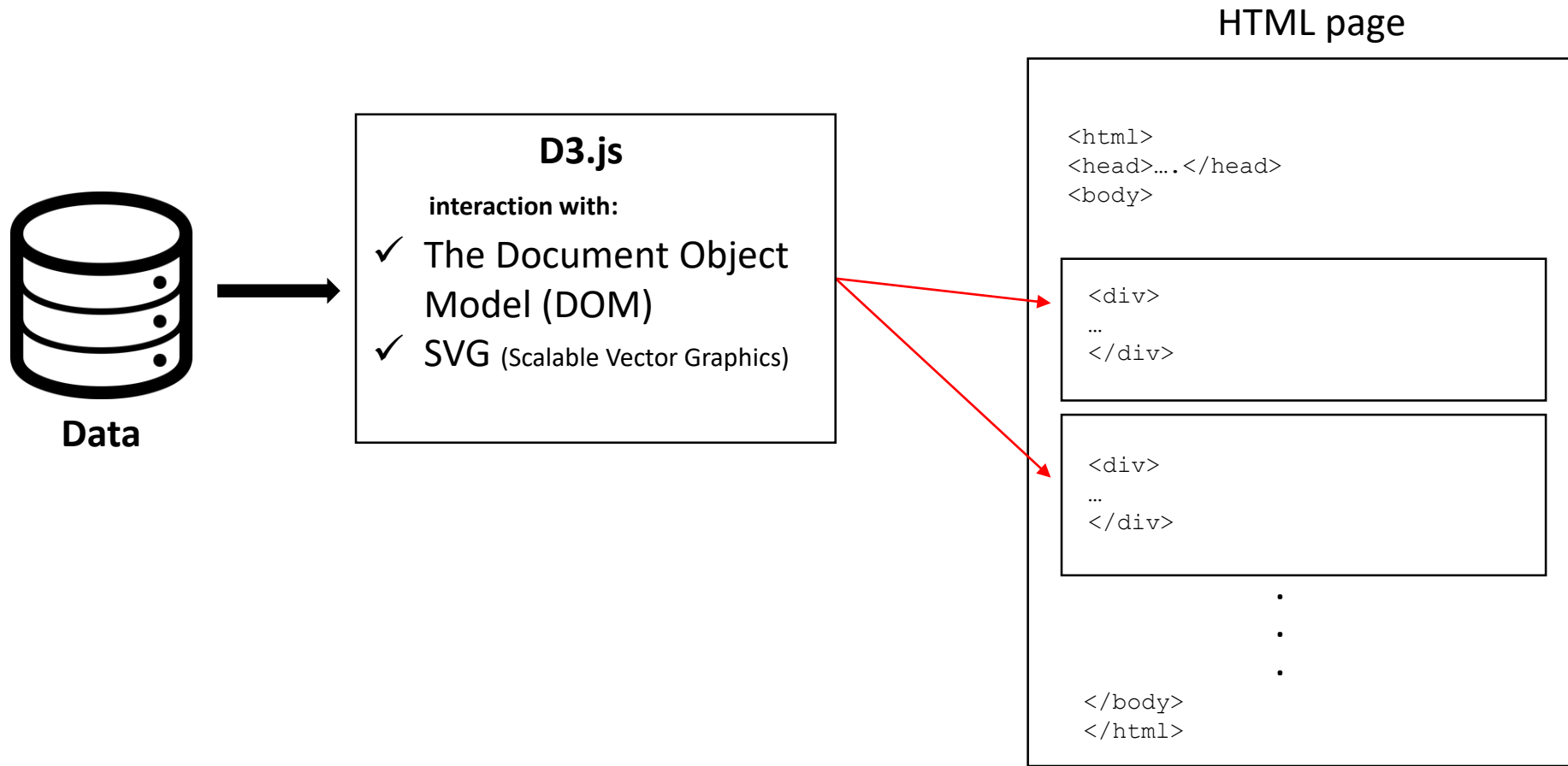
[See more examples.](#)

# 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...?





# History

Rich internet applications



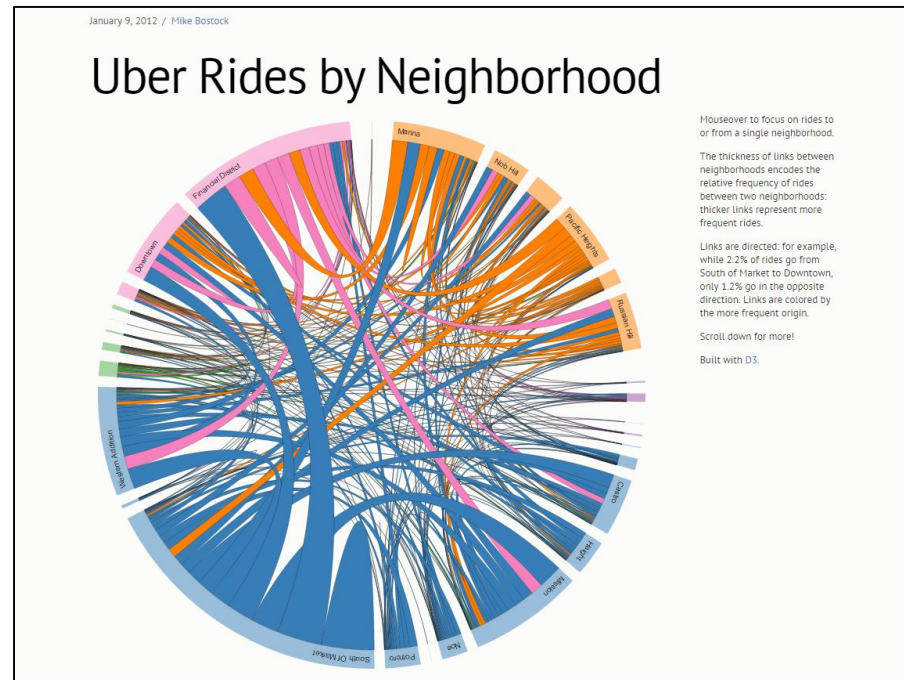
**Java applet**



**Flash**

# HTML5 + JavaScript (D3 also..) + CSS

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



# Content


1 D3 Intro

**2 Observable notebooks**

3 D3 Basics

4 Working with D3 in Observable


5 Running D3 locally





...


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 >





Mike Bostock · Jun 8, 2018

Building a better computational medium. Founder @observablehq.  
Creator @d3. Former @nytgraphics. Pronounced BOSS-lock.

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:

$$\begin{aligned}\frac{d\text{🐭}}{dt} &= \alpha \text{🐭} - \beta \text{🐭} \text{🐱} \\ \frac{d\text{🐱}}{dt} &= \delta \text{🐭} \text{🐱} - \gamma \text{🐱}\end{aligned}$$

In this model, the prey have unlimited food and grow exponentially ( $\alpha$ ); however, they are sometimes eaten ( $\beta$ ) by predators. The predators have unlimited appetite, but eat only prey, limiting their growth ( $\delta$ ); their decay is exponential ( $\gamma$ ). The parameters  $\alpha$ ,  $\beta$ ,  $\gamma$ , and  $\delta$  thus describe how the two populations behave, while  $\frac{d\text{🐭}}{dt}$  and  $\frac{d\text{🐱}}{dt}$  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.)



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

1 D3 Intro

2 Observable notebooks

**3 D3 Basics**

4 Working with D3 in Observable

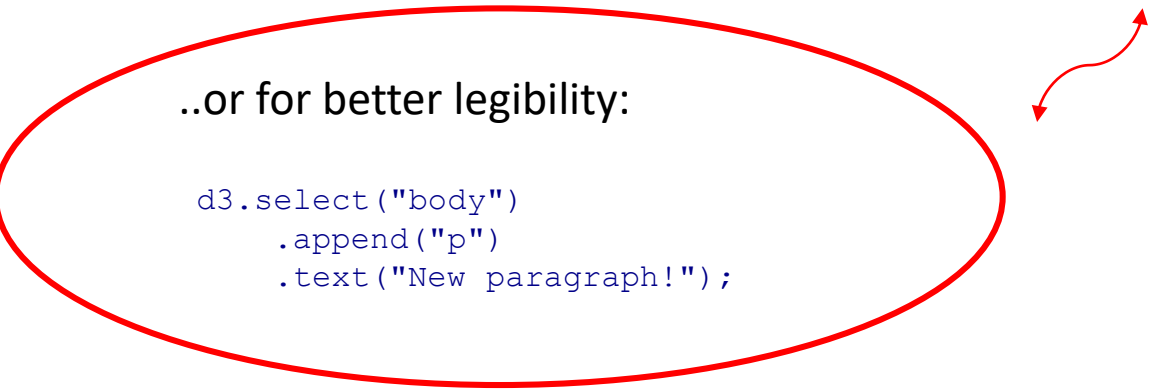
5 Running D3 locally

# 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!");
```



# Loading data

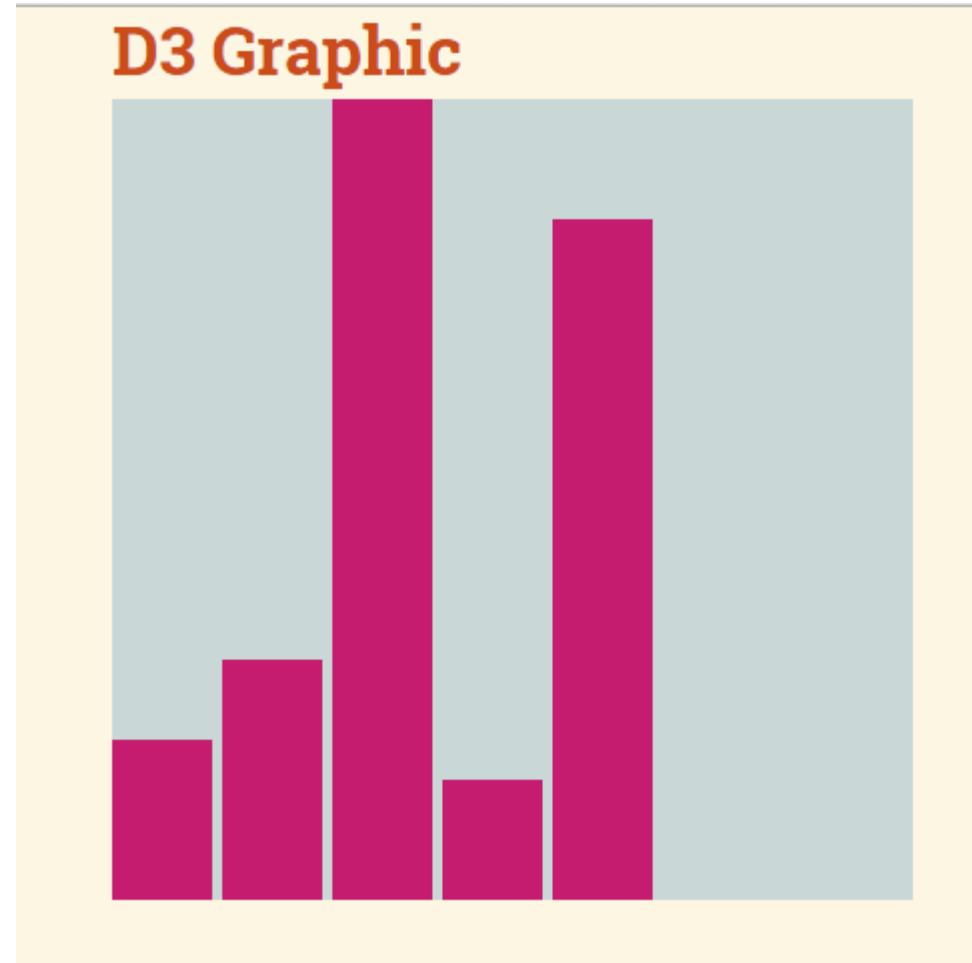
## Arrays

```
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)});
```

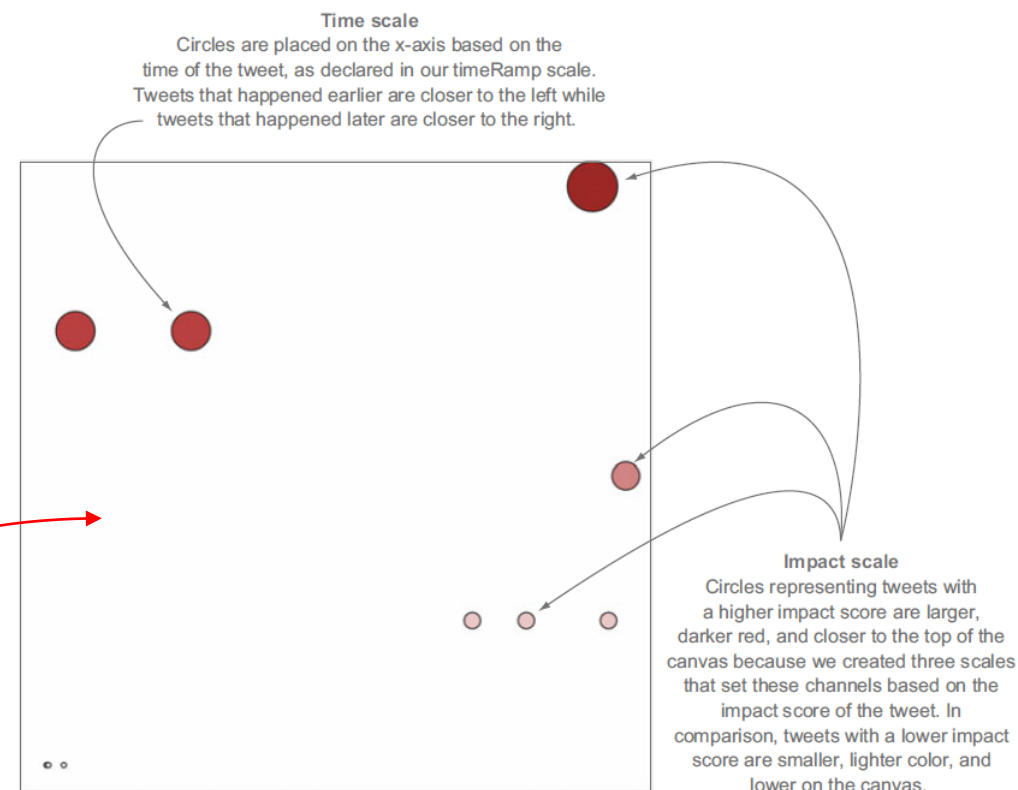
```
"label","population","country","x","y"
"San Francisco", 750000,"USA",122,-37
"Fresno", 500000,"USA",119,-36
"Lahore",12500000,"Pakistan",74,31
"Karachi",13000000,"Pakistan",67,24
"Rome",2500000,"Italy",12,41
```

# 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");
```

Or by class:

```
d3.selectAll(".container")...
```

Or by ID:

```
d3.select("#dataPoint1")...
```

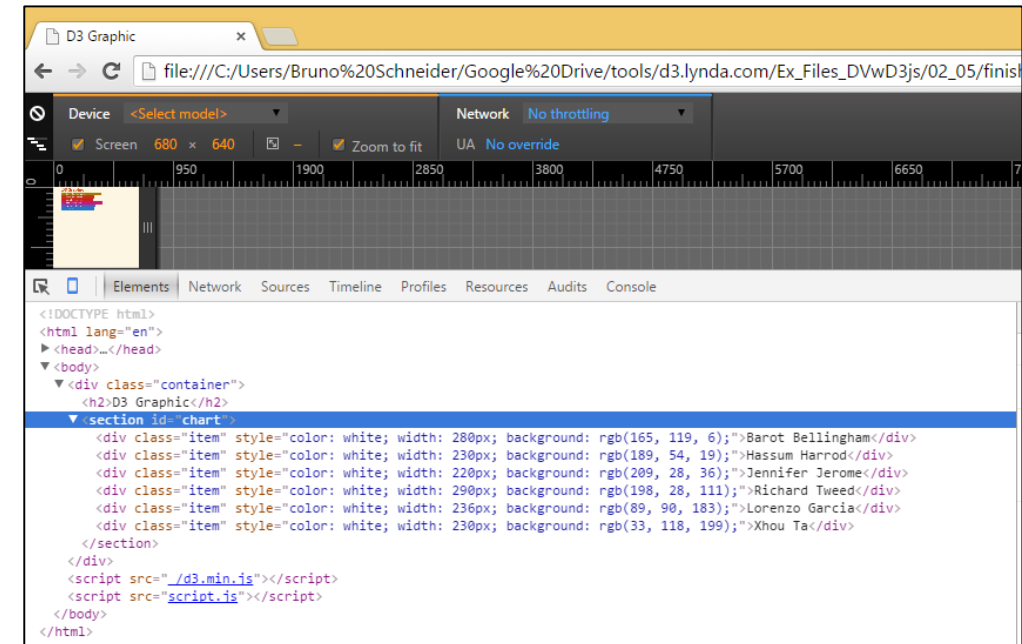
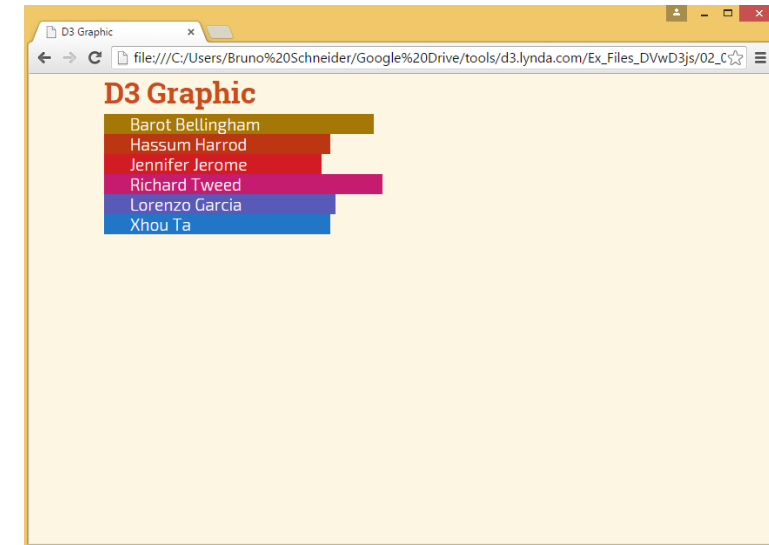
# D3 selection: example

index.html

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>D3 Graphic</title>
6   <link rel="stylesheet" href="_/base.css">
7   <link rel="stylesheet" href="style.css">
8 </head>
9 <body>
10 <div class="container">
11   <h2>D3 Graphic</h2>
12   <section id="chart">
13     <div class="item">Barot Bellingham</div>
14     <div class="item">Hassum Harrod</div>
15     <div class="item">Jennifer Jerome</div>
16     <div class="item">Richard Tweed</div>
17     <div class="item">Lorenzo Garcia</div>
18     <div class="item">Xhou Ta</div>
19   </section>
20 </div>
21 <script src="_/d3.min.js"></script>
22 <script src="script.js"></script>
23 </body>
24 </html>
```

script.js

```
1 var myStyles = [
2   { width: 200,
3     color: '#A57706'},
4   { width: 230,
5     color: '#BD3613'},
6   { width: 220,
7     color: '#D11C24'},
8   { width: 290,
9     color: '#C61C6F'},
10  { width: 236,
11    color: '#595AB7'},
12  { width: 230,
13    color: '#2176C7'}
14 ];
15
16 d3.selectAll('.item')
17   .data(myStyles)
18   .style({
19     'color': 'white',
20     'background' : function(d) {
21       return d.color;
22     },
23     width : function(d) {
24       return d.width + 'px';
25     }
26   })
```



# HTML <div> 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 <div> tag defines a division or a section in an HTML document.

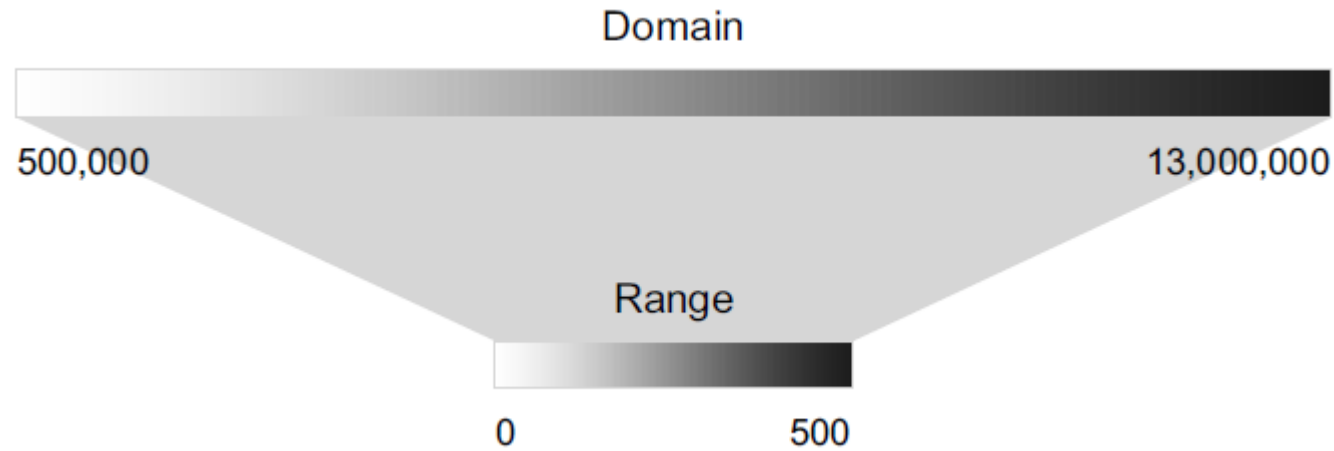
The <div> tag is used to group block-elements to format them with CSS.

## Browser Support

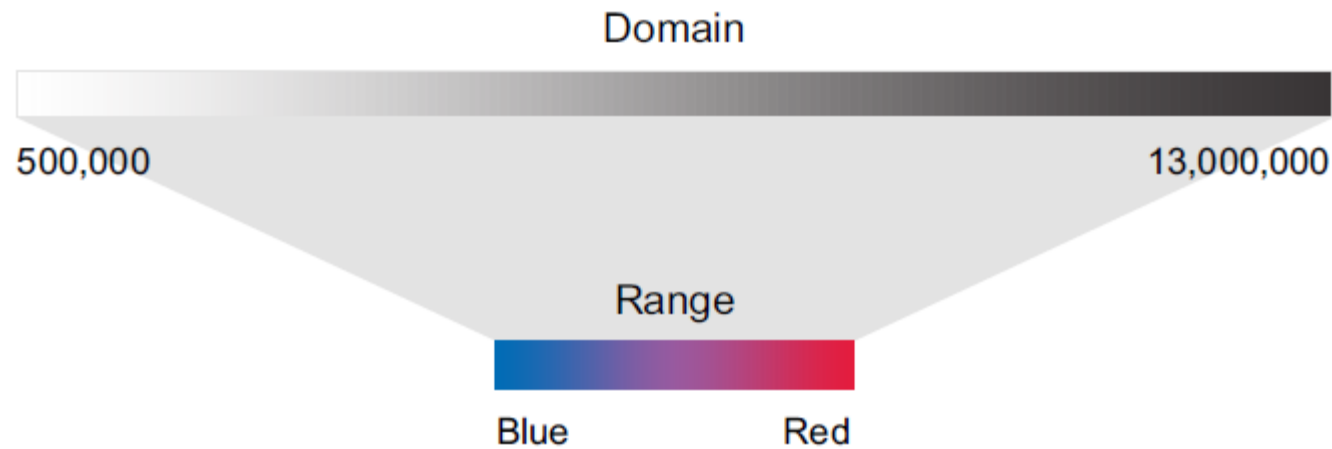
Element						
<div>	Yes	Yes	Yes	Yes	Yes	Yes



# 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

1 D3 Intro

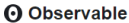
2 Observable notebooks

3 D3 Basics

**4 Working with D3 in Observable**

5 Running D3 locally

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



[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

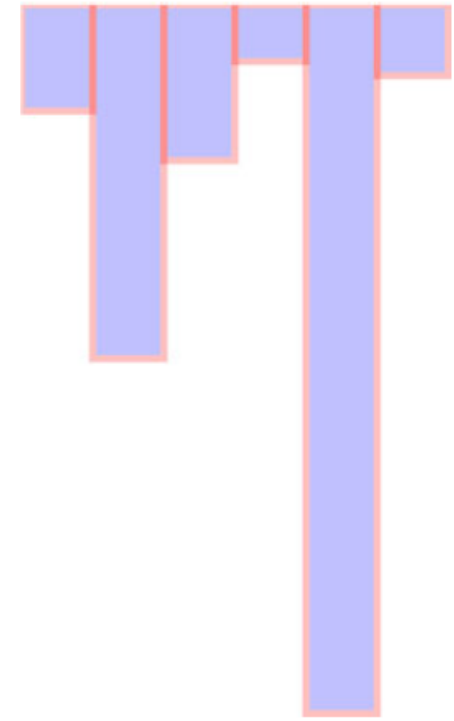
```
1  var bardata = [20, 30, 105, 15, 85];
2
3  var height = 400,
4      width = 600,
5      barWidth = 50,
6      barOffset = 5;
7
8  d3.select('#chart').append('svg')
9      .attr('width', width)
10     .attr('height', height)
11     .style('background', '#C9D7D6')
12     .selectAll('rect').data(bardata)
13     .enter().append('rect')
14         .style('fill', '#C61C6F')
15         .attr('width', barWidth)
16         .attr('height', function(d) {
17             return d;
18         })
19         .attr('x', function(d,i) {
20             return i * (barWidth + barOffset);
21         })
22         .attr('y', function(d) {
23             return height - d;
24         })
```

# Observable

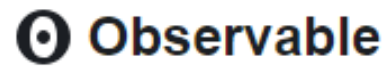
<https://beta.observablehq.com/d/0add62c9618c972f>

## > D3/Observable Tutorial EMAp FGV

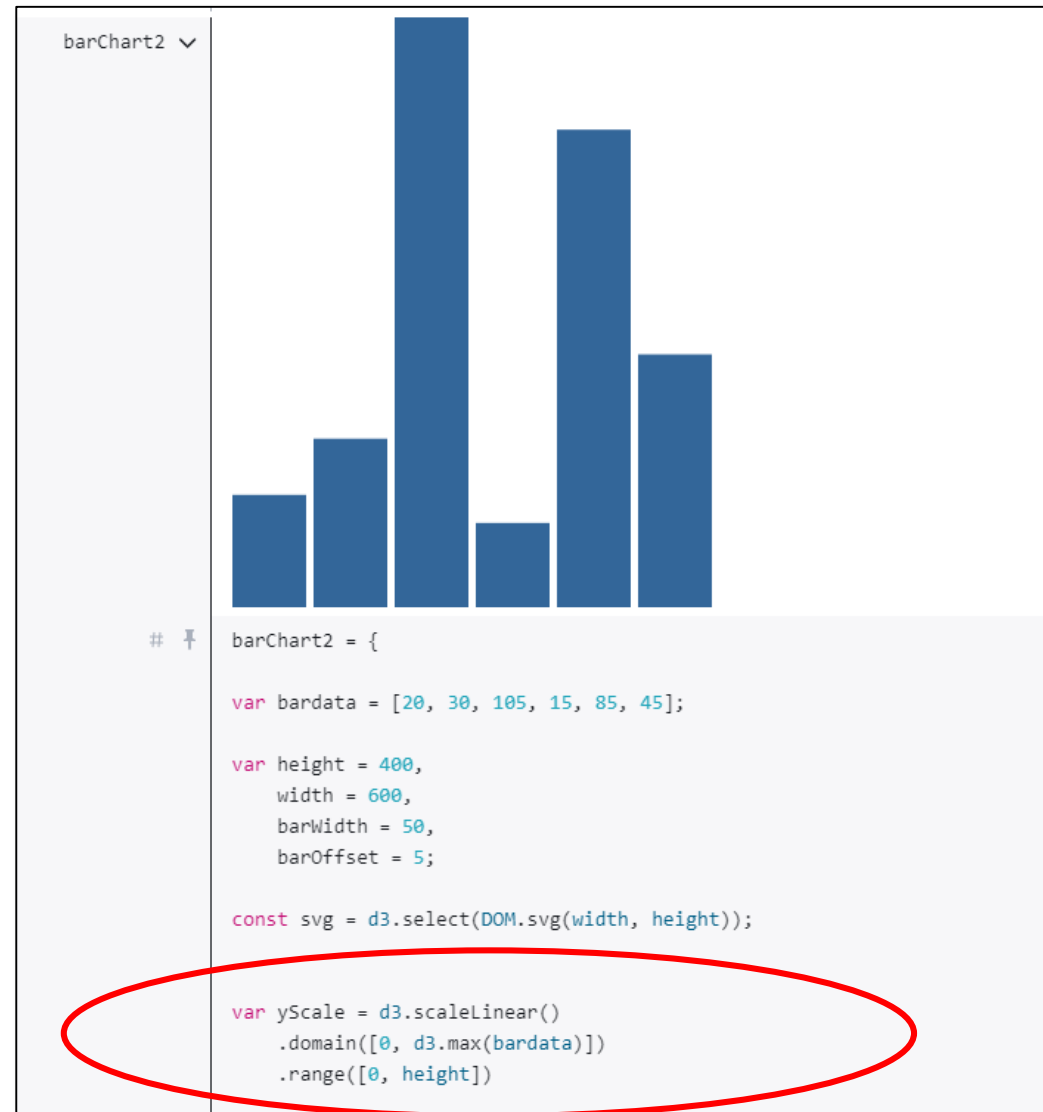
barChart >



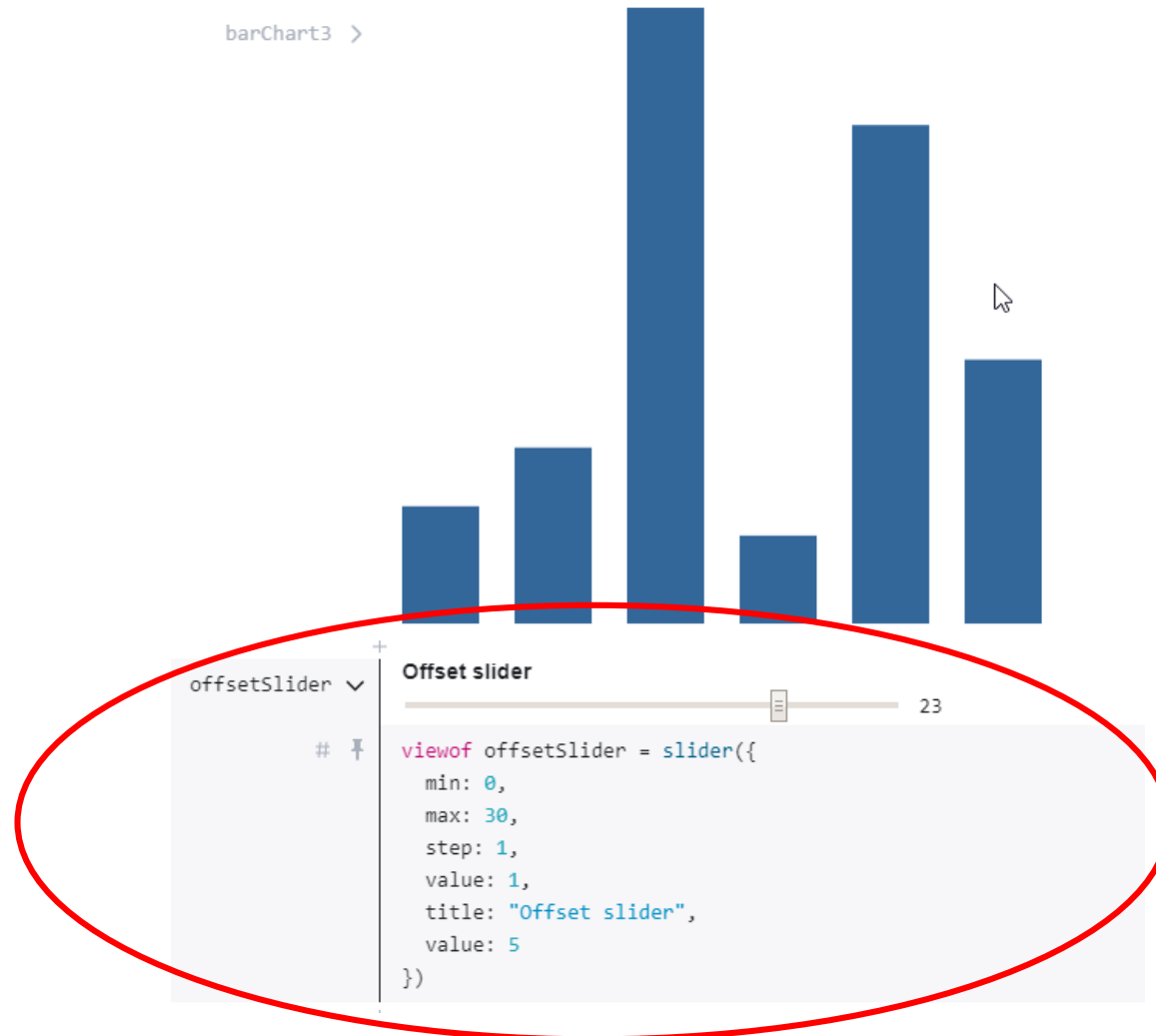
SVG rectangles  
are drawn from top to  
bottom. 



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




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



# Content

- 1 D3 Intro
- 2 Observable notebooks
- 3 D3 Basics
- 4 Working with D3 in Observable
- 5 Running D3 locally**



## 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

```
1  var bardata = [20, 30, 105, 15, 85];
2
3  var height = 400,
4      width = 600,
5      barWidth = 50,
6      barOffset = 5;
7
8  d3.select('#chart').append('svg')
9      .attr('width', width)
10     .attr('height', height)
11     .style('background', '#C9D7D6')
12     .selectAll('rect').data(bardata)
13     .enter().append('rect')
14         .style('fill', '#C61C6F')
15         .attr('width', barWidth)
16         .attr('height', function(d) {
17             return d;
18         })
19         .attr('x', function(d,i) {
20             return i * (barWidth + barOffset);
21         })
22         .attr('y', function(d) {
23             return height - d;
24         })
```




# Downloading D3

<http://d3js.org/>



## Data-Driven Documents



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 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.

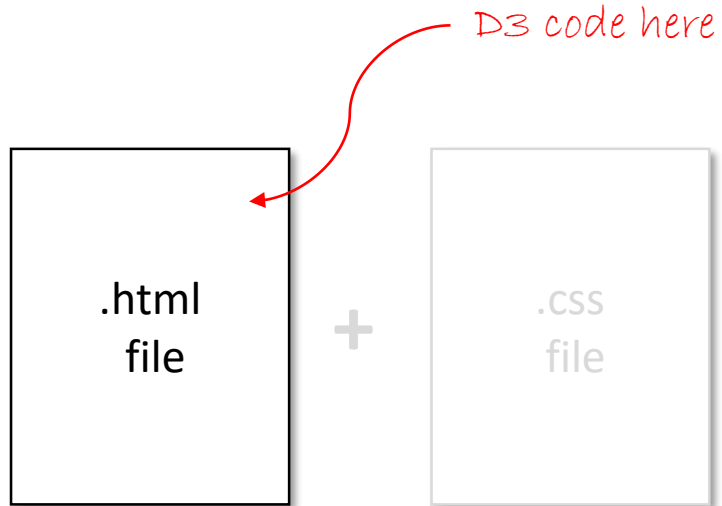
[See more examples.](#)

Download the latest version (5.7.0) here:

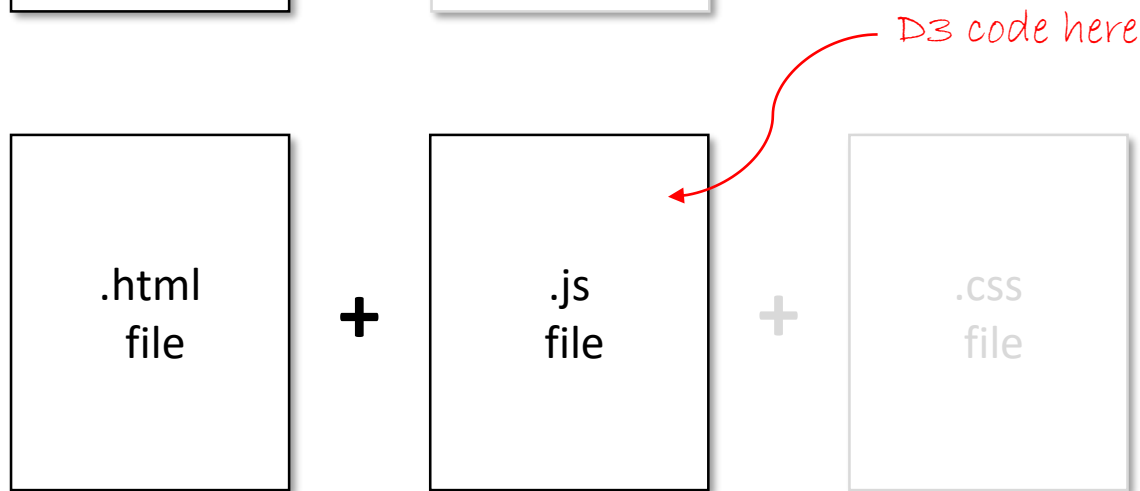
- [d3.zip](#)

# Organizing the code

1.



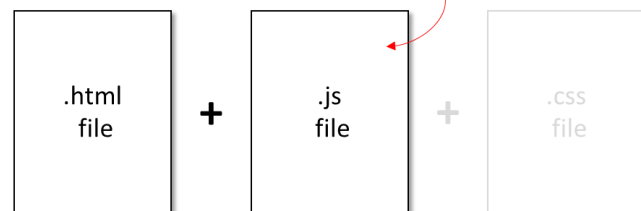
2.



(preferred way)

# Example

2.



script.js

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>D3 Graphic</title>
6 </head>
7 <body>
8   <div class="container">
9     <h2>D3 Graphic</h2>
10    <div id="chart"></div>
11  </div>
12  <script src="_/d3.min.js"></script>
13  <script src="script.js"></script>
14 </body>
15 </html>

```

```

1 var bardata = [20, 30, 20, 15, 40, 200];
2
3 var height = 400,
4     width = 600,
5     barWidth = 50,
6     barOffset = 5;
7
8 var yScale = d3.scale.linear()
9     .domain([0, d3.max(bardata)])
10    .range([0, height])
11
12 d3.select('#chart').append('svg')
13   .attr('width', width)
14   .attr('height', height)
15   .style('background', '#C9D7D6')
16   .selectAll('rect').data(bardata)
17   .enter().append('rect')
18   .style('fill', '#C61C6F')
19   .attr('width', barWidth)
20   .attr('height', function(d) {
21     return yScale(d);
22   })
23   .attr('x', function(d,i) {
24     return i * (barWidth + barOffset);
25   })
26   .attr('y', function(d) {
27     return height - yScale(d);
28   })

```

# Loading D3.js

Local storage

OR

Online

```

1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8">
5    <title>D3 Graphic</title>
6  </head>
7  <body>
8    <div class="container">
9      <h2>D3 Graphic</h2>
10     <div id="chart"></div>
11   </div>
12   <script src="_/d3.min.js"></script>
13   <script src="script.js"></script>
14 </body>
15 </html>

```

.html file

```

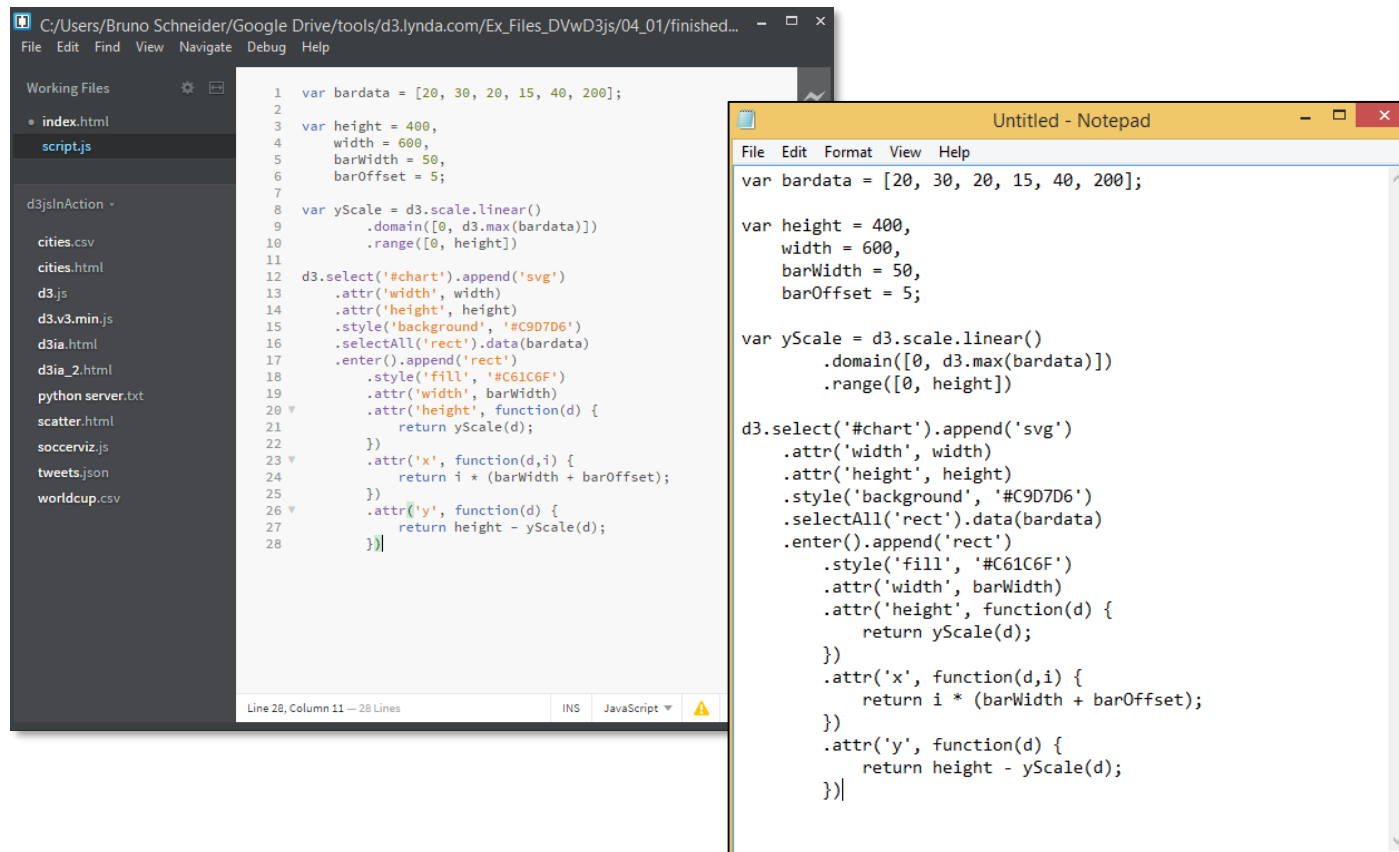
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8">
5    <title>D3 Graphic</title>
6  </head>
7  <body>
8    <div class="container">
9      <h2>D3 Graphic</h2>
10     <div id="chart"></div>
11   </div>
12   <script src="https://d3js.org/d3.v4.min.js"></script>
13   <script src="script.js"></script>
14 </body>
15 </html>

```

.html file

# Working with D3

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



The image shows two windows side-by-side. The left window is a code editor with a dark theme, showing a file explorer on the left with files like 'index.html', 'script.js', 'cities.csv', etc. The main editor area shows a JavaScript file named 'script.js' with D3.js code for a bar chart. The right window is a standard Windows Notepad application with a light theme, showing the same JavaScript code as the left window.

```

1  var bardata = [20, 30, 20, 15, 40, 200];
2
3  var height = 400,
4      width = 600,
5      barWidth = 50,
6      barOffset = 5;
7
8  var yScale = d3.scale.linear()
9      .domain([0, d3.max(bardata)])
10     .range([0, height]);
11
12  d3.select('#chart').append('svg')
13     .attr('width', width)
14     .attr('height', height)
15     .style('background', '#C9D7D6')
16     .selectAll('rect').data(bardata)
17     .enter().append('rect')
18     .style('fill', '#C61C6F')
19     .attr('width', barWidth)
20     .attr('height', function(d) {
21         return yScale(d);
22     })
23     .attr('x', function(d,i) {
24         return i * (barWidth + barOffset);
25     })
26     .attr('y', function(d) {
27         return height - yScale(d);
28     })
  
```

# Working with D3

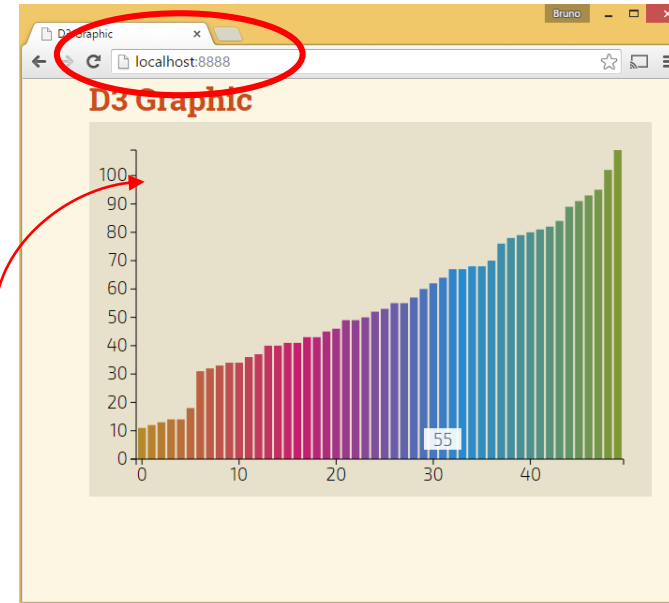
✓ running Python's built-in server

```

Command Prompt
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.
C:\Users\Bruno Schneider>python -m SimpleHTTPServer 8888 &
  
```

**python -m SimpleHTTPServer 8888 &**

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



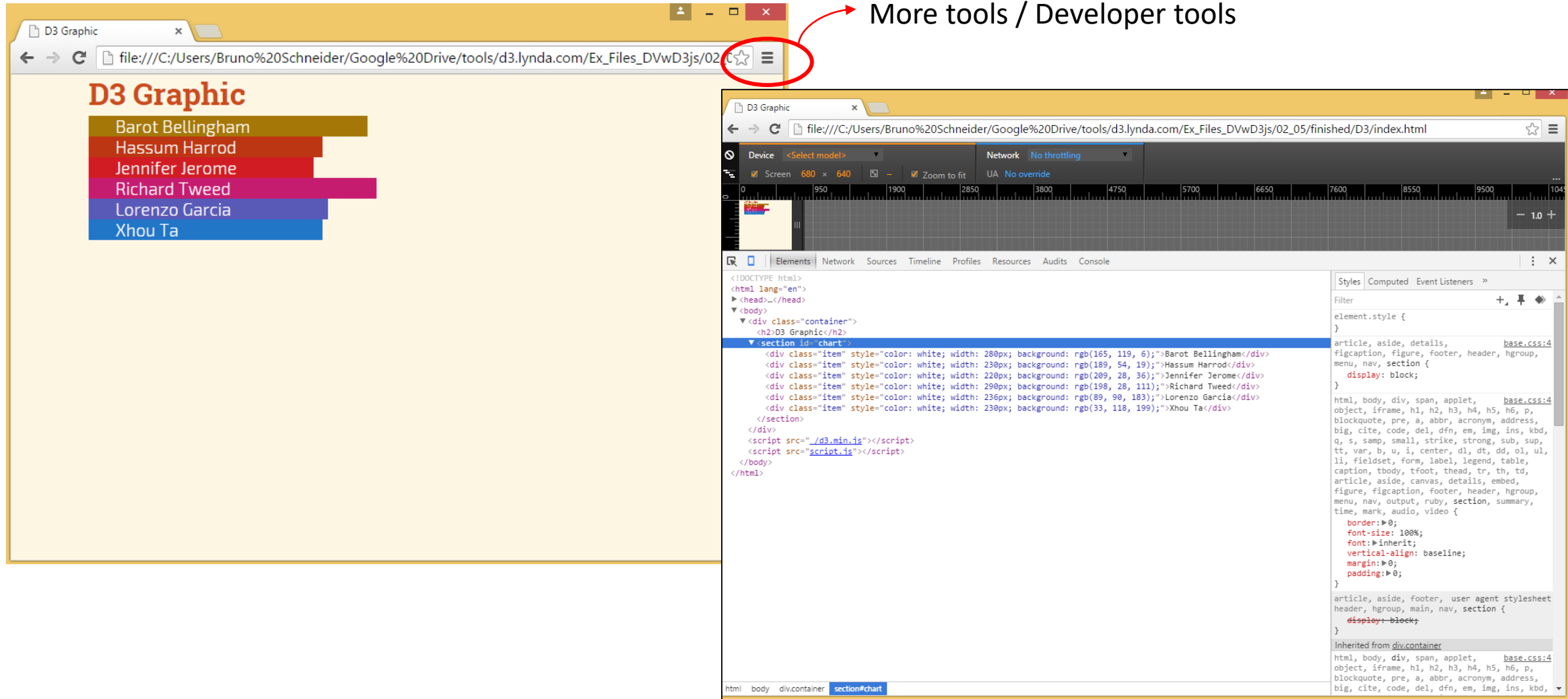
✓ .. Or npm http-server



# The developer tools in Chrome

- ✓ Shows the current state of the DOM

Show example



The image shows a web browser window displaying a D3.js chart titled "D3 Graphic". The chart is a horizontal bar chart with the following data:

Name	Value (approximate)
Barot Bellingham	100
Hassum Harrod	150
Jennifer Jerome	120
Richard Tweed	180
Lorenzo Garcia	140
Xhou Ta	160

Below the chart, the Chrome Developer Tools interface is open, showing the "Elements" panel. The DOM tree is expanded to show the following structure:

```

<!DOCTYPE html>
<html lang="en">
  <head>...</head>
  <body>
    <div class="container">
      <h2>D3 Graphic</h2>
      <div class="item" style="color: white; width: 280px; background: rgb(165, 119, 6);">Barot Bellingham</div>
      <div class="item" style="color: white; width: 230px; background: rgb(189, 54, 19);">Hassum Harrod</div>
      <div class="item" style="color: white; width: 220px; background: rgb(209, 28, 36);">Jennifer Jerome</div>
      <div class="item" style="color: white; width: 290px; background: rgb(198, 28, 111);">Richard Tweed</div>
      <div class="item" style="color: white; width: 236px; background: rgb(89, 90, 183);">Lorenzo Garcia</div>
      <div class="item" style="color: white; width: 230px; background: rgb(33, 118, 199);">Xhou Ta</div>
    </div>
    <script src="/d3.min.js"></script>
    <script src="script.js"></script>
  </body>
</html>
  
```

The "Styles" panel on the right shows the default styles for the selected element, including:

```

element.style {
}

article, aside, details, figcaption, figure, footer, header, hgroup, menu, nav, section {
  display: block;
}

html, body, div, span, applet, object, iframe, h1, h2, h3, h4, h5, h6, p, blockquote, pre, a, abbr, acronym, address, big, cite, code, del, dfn, em, img, ins, kbd, q, s, samp, small, strike, strong, sub, sup, tt, var, b, u, i, center, dl, dt, dd, ol, ul, li, fieldset, form, label, legend, table, caption, tbody, tfoot, thead, tr, th, td, article, aside, canvas, details, embed, figure, figcaption, footer, header, hgroup, menu, nav, output, ruby, section, summary, time, mark, audio, video {
  border: 0;
  font-size: 100%;
  font: inherit;
  vertical-align: baseline;
  margin: 0;
  padding: 0;
}

article, aside, footer, user agent stylesheet, header, hgroup, main, nav, section {
  display: block;
}

Inherited from div.container
html, body, div, span, applet, object, iframe, h1, h2, h3, h4, h5, h6, p, blockquote, pre, a, abbr, acronym, address, big, cite, code, del, dfn, em, img, ins, kbd,
  
```


A red arrow points to the "More tools / Developer tools" button in the browser's top right corner.




# References

**beta.observablehq.com**


Featured collections




**Introduction**  
17 notebooks  
Welcome to Observable! Get started with these tutorials.



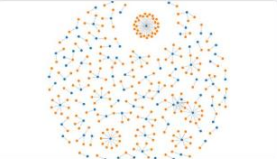
**Techniques**  
26 notebooks  
Conquered the Introduction to Observable? Here are some next steps.




**WebGL**  
19 notebooks  
Harness the raw power of the GPU using WebGL, THREE.js and regl.



**Maps**  
23 notebooks  
Embrace your inner shapefile. Or GeoJSON or TopoJSON.

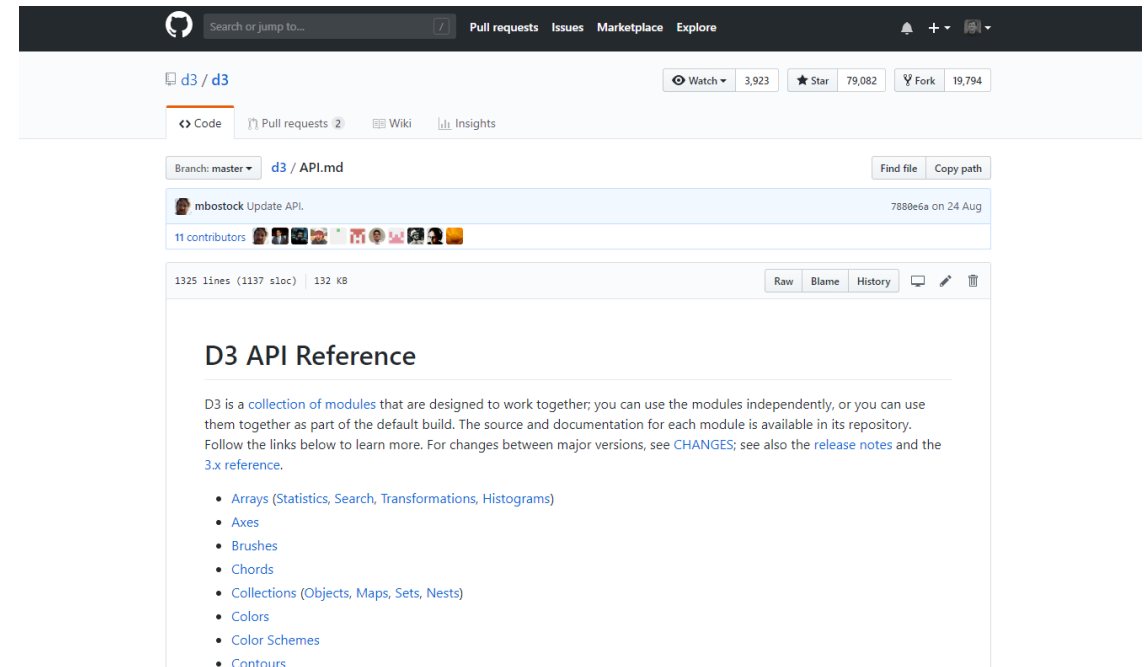


**Visualization**  
49 notebooks  
Explore and explain patterns in quantitative data using D3 and Vega.



**Explorables**  
16 notebooks  
Be an active reader with these playful interactive explanations.

**<https://github.com/d3/d3/blob/master/API.md>**

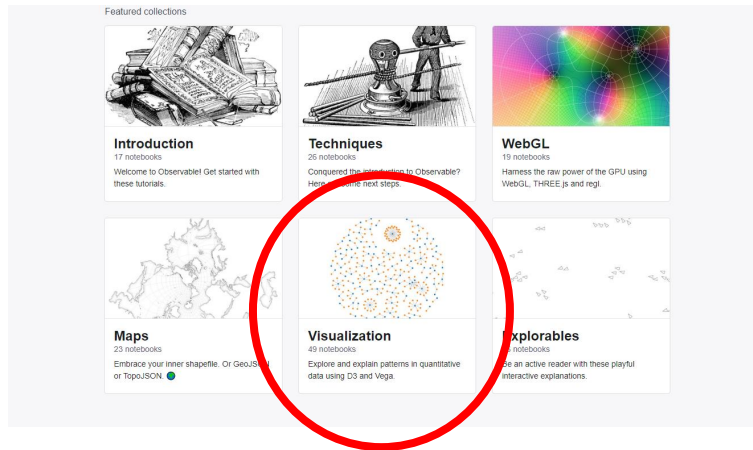



The screenshot shows the GitHub repository for `d3/d3`. The file `API.md` is selected, showing the `Branch: master` and the file path `d3 / API.md`. The file is 1325 lines (1137 sloc) and 132 KB. The content of the file is the D3 API Reference, which includes a list of modules: `Arrays (Statistics, Search, Transformations, Histograms)`, `Axes`, `Brushes`, `Chords`, `Collections (Objects, Maps, Sets, Nests)`, `Colors`, `Color Schemes`, and `Contours`.



# References

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



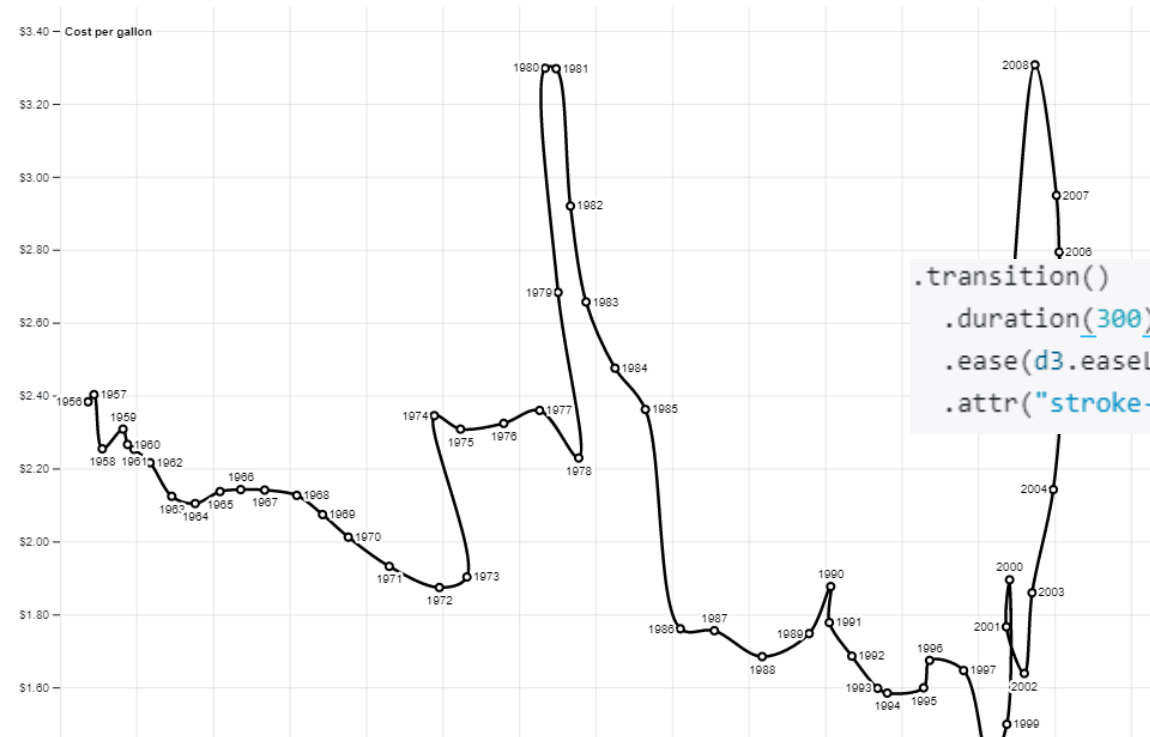
 Observable

## > 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 >

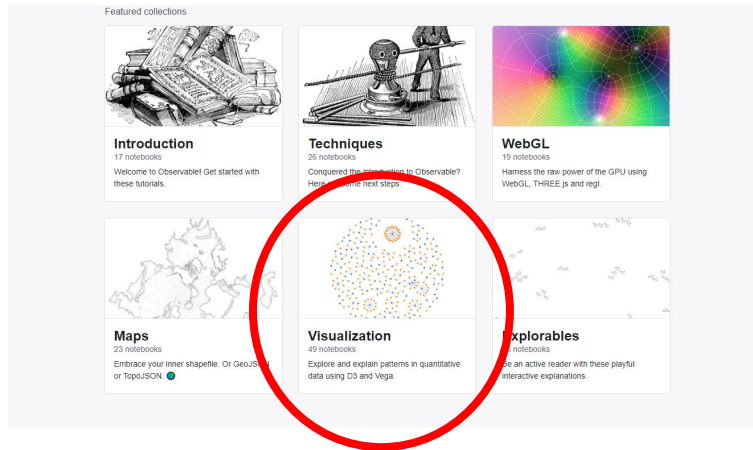



```
.transition()
  .duration(300)
  .ease(d3.easeLinear)
  .attr("stroke-dasharray", `${1},${1}`);
```


 Fork

# References

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



 **Observable**

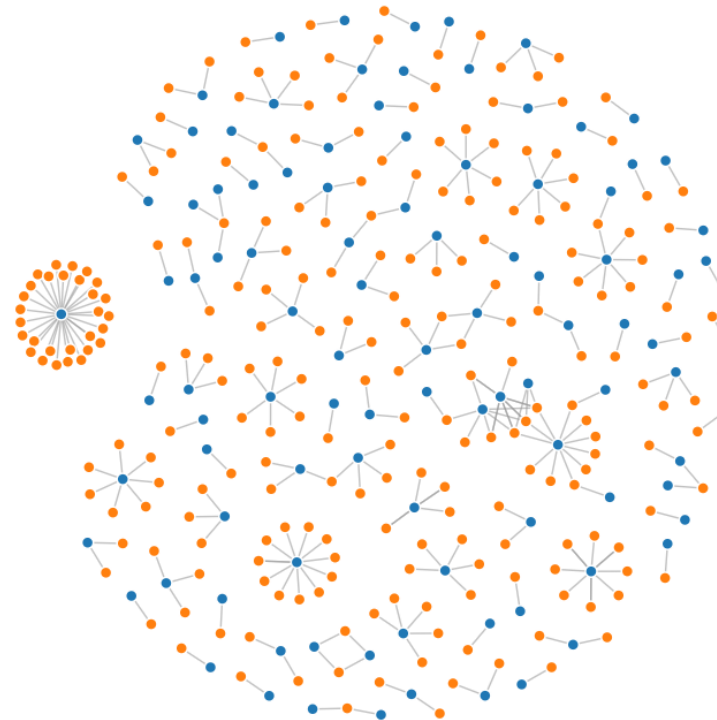
 Mike Bostock · Sep 13, 2018  
Building a better computational medium. Founder @observablehq.  
Creator @d3. Former @nytgraphics. Pronounced BOSS-tock.

  
15

Featured in Visualization Fork of D3 Force-Directed Graph

## > Disjoint Force-Directed Graph

part >



# Muito obrigado pela atenção...!

