Tutorial 5: D3.js Part 1

WANG Yong
College of Computing and Data Science
Nanyang Technological University

Resources

Highly-recommended Tutorials:

- Let's Make a Bar Chart (Parts 1 to 3): https://bost.ocks.org/mike/bar/
- D3 official tutorial: https://d3js.org/

Online References:

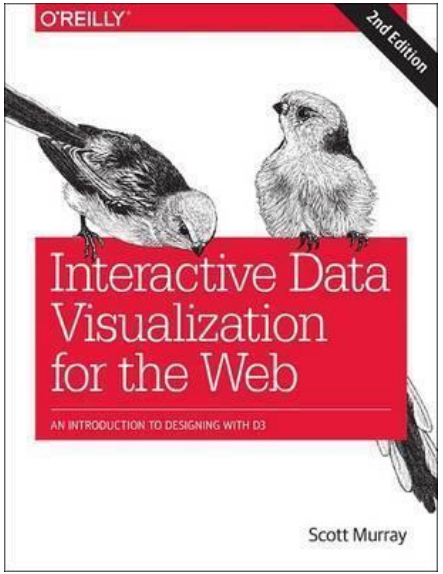
- D3 Gallery and official documents:
  https://observablehq.com/@d3/gallery
  https://github.com/d3/d3/wiki
  https://github.com/d3/d3/blob/main/API.md

Resources

Books:

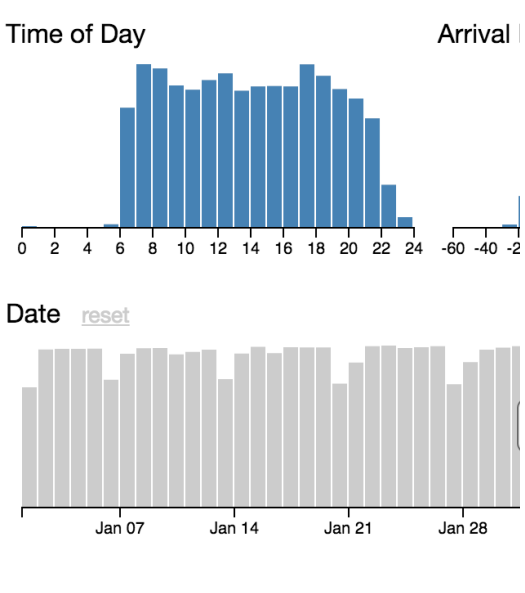
- Interactive Visualization for the Web: An Introduction to Designing with D3, by Murray Scott
- JavaScript: The Good Parts, by Crockford Douglas

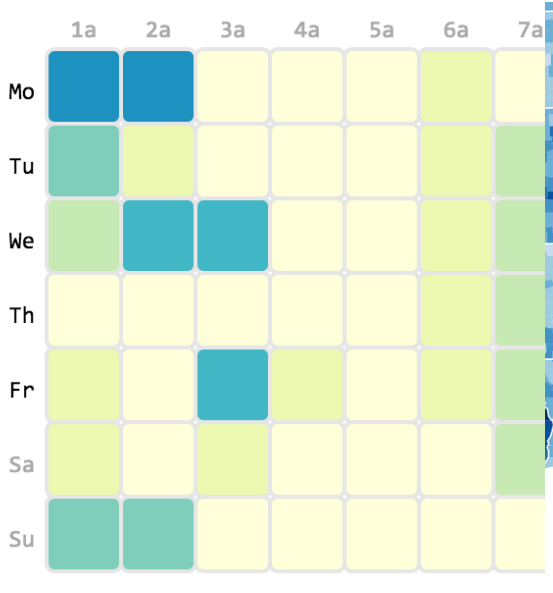
Both books are available online through SMU library!

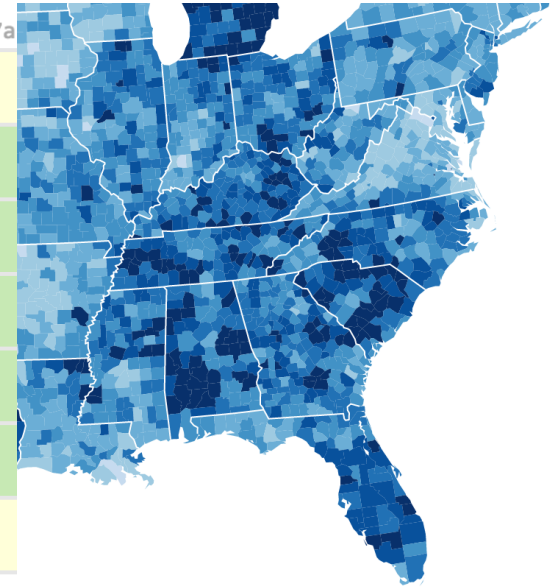


Background

What is D3.js?

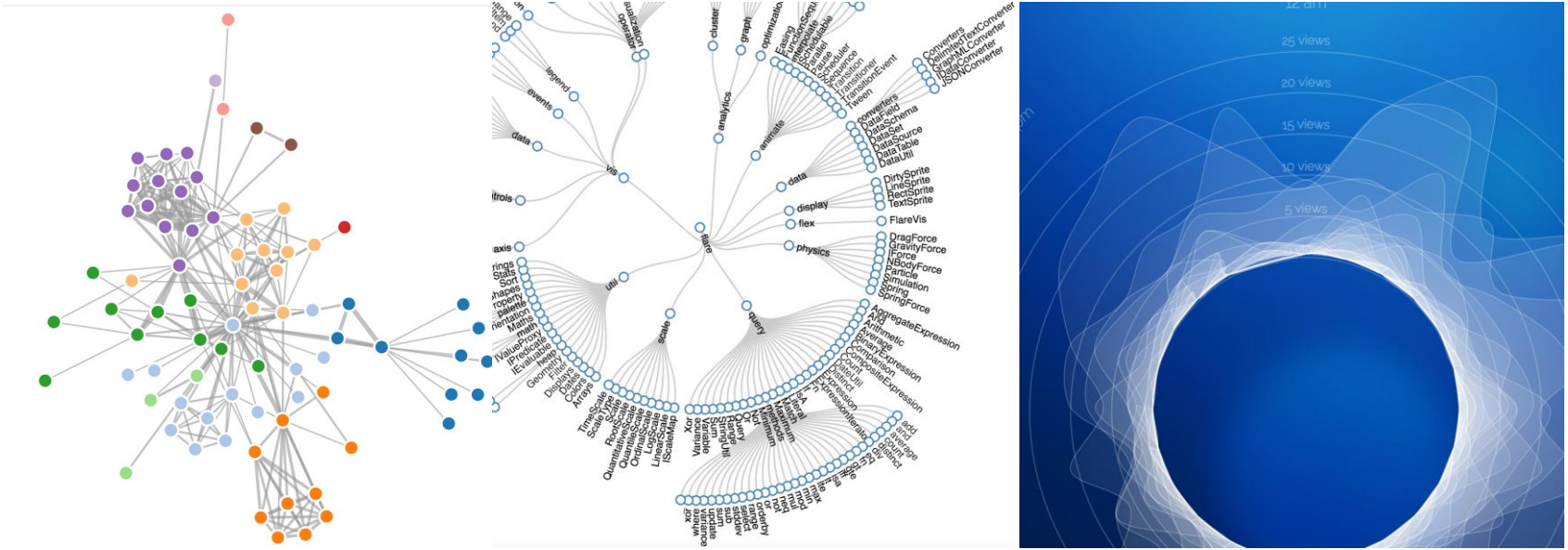






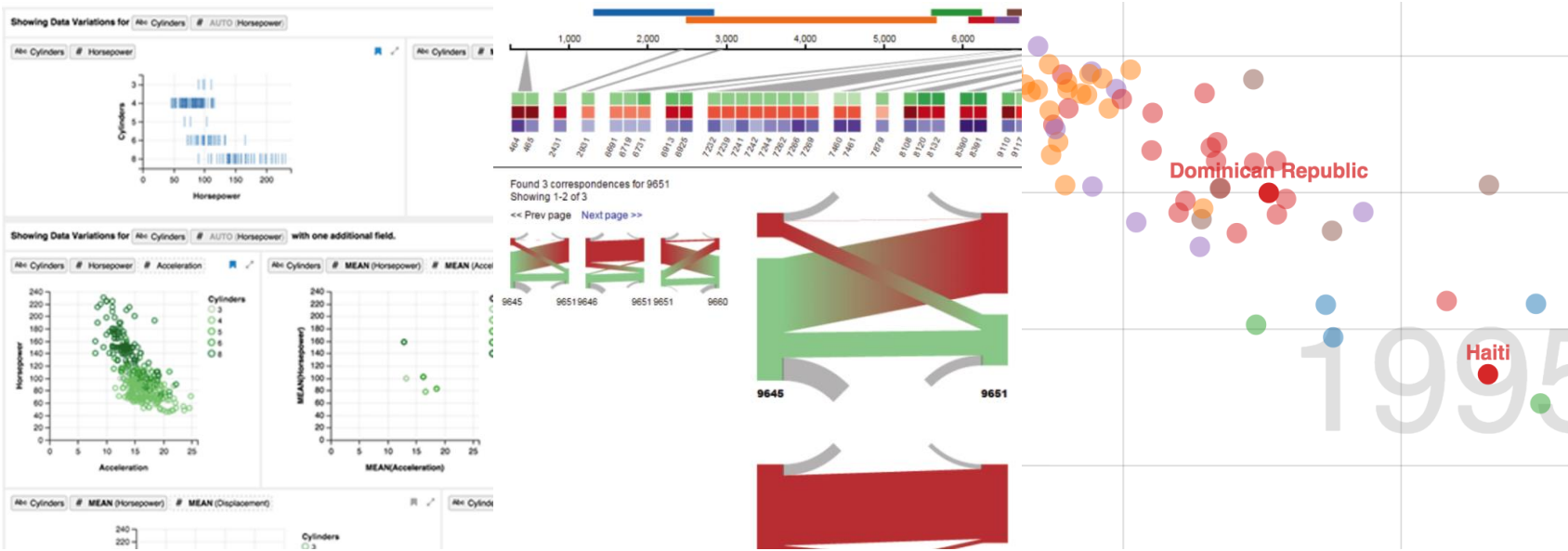
Check out the Gallery

What is D3.js?



Check out the Gallery

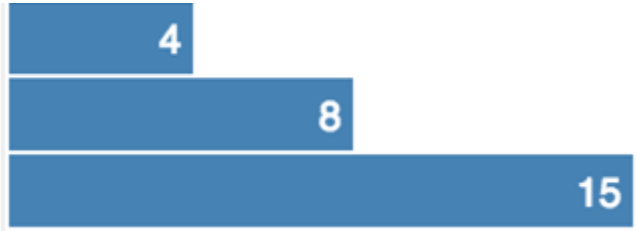
What is D3.js?



Check out the Gallery

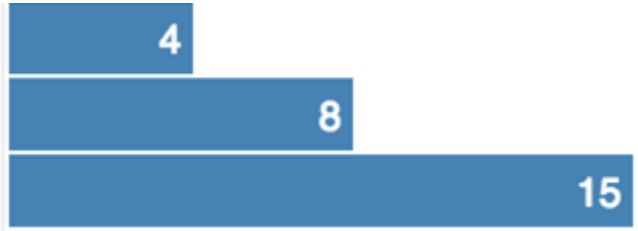
What is D3.js?

Data are bound to DOM elements to make **D**ata-**D**riven **D**ocuments



What is D3.js?

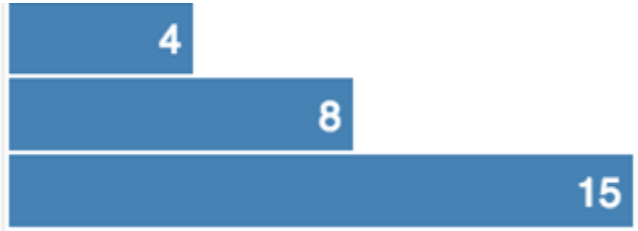
DATA are bound to DOM elements to make Data-Driven Documents




```
var data = [4,8,15];
```

What is D3.js?

Data are bound to **DOM ELEMENTS** to make Data-Driven Documents

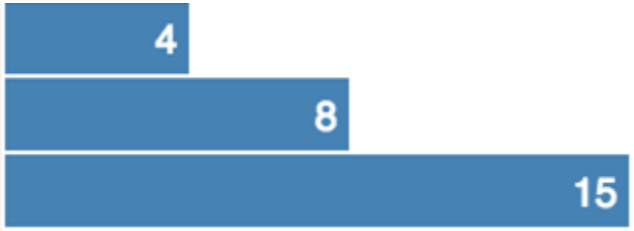



```
var data = [4,8,15];
```

```
▼<body>
  ▼<div class="chart">
      <div style="width: 40px;">4</div>
      <div style="width: 80px;">8</div>
      <div style="width: 150px;">15</div>
```

What is D3.js?

Data are **BOUND** to DOM elements to make Data-Driven Documents




```
> d3.select(".chart").select("div").datum()
< 4
```

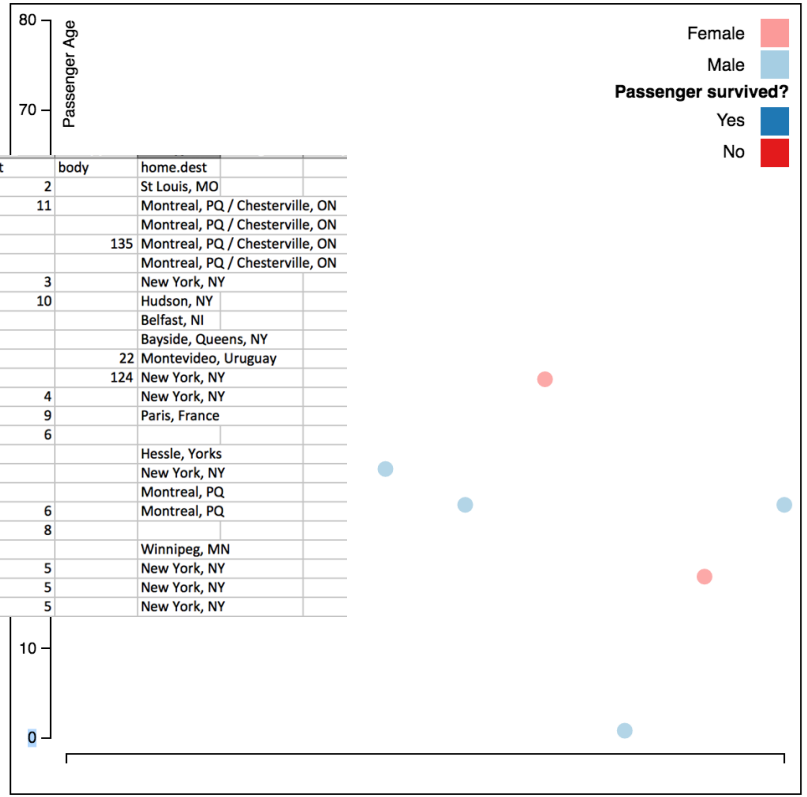
```
var data = [4,8,15];
```

```
▼<body>
  ▼<div class="chart">
      <div style="width: 40px;">4</div>
      <div style="width: 80px;">8</div>
      <div style="width: 150px;">15</div>
```

Example:
Titanic Passengers

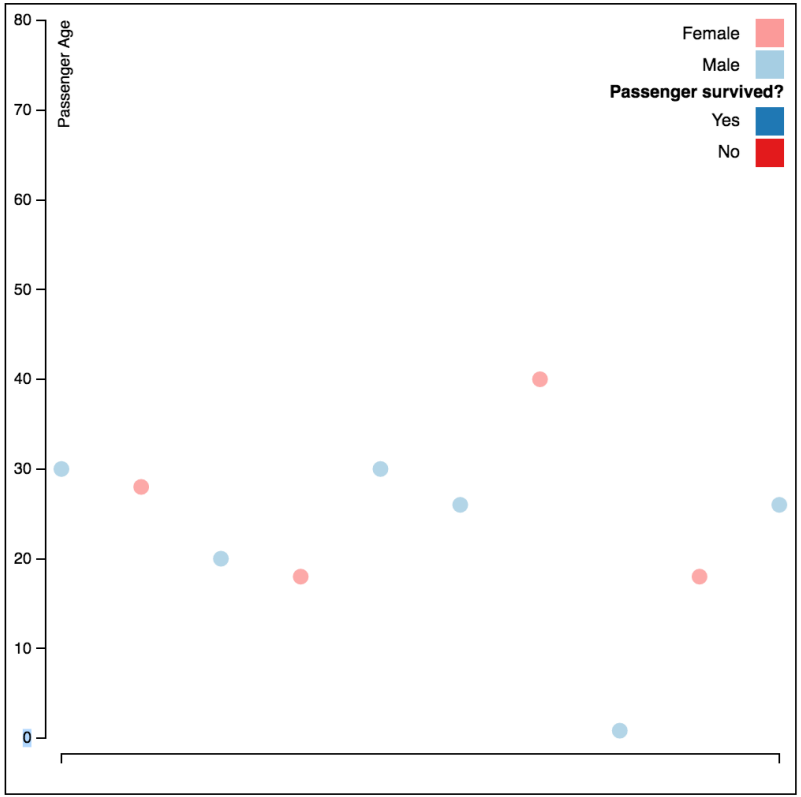
Example

pclass	survived	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	home.dest
1	1	Allen, Miss. E	female	29	0	0	24160	211.3375	B5	S	2		St Louis, MO
1	1	Allison, Mast	male	0.92	1	2	113781	151.55	C22 C26	S	11		Montreal, PQ / Chesterville, ON
1	0	Allison, Miss	female	2	1	2	113781	151.55	C22 C26	S			Montreal, PQ / Chesterville, ON
1	0	Allison, Mr. H	male	30	1	2	113781	151.55	C22 C26	S		135	Montreal, PQ / Chesterville, ON
1	0	Allison, Mrs.	female	25	1	2	113781	151.55	C22 C26	S			Montreal, PQ / Chesterville, ON
1	1	Anderson, M	male	48	0	0	19952	26.55	E12	S	3		New York, NY
1	1	Andrews, Mi	female	63	1	0	13502	77.9583	D7	S	10		Hudson, NY
1	0	Andrews, Mr	male	39	0	0	112050	0	A36	S			Belfast, NI
1	1	Appleton, M	female	53	2	0	11769	51.4792	C101	S	D		Bayside, Queens, NY
1	0	Artagaveytia	male	71	0	0	PC 17609	49.5042		C		22	Montevideo, Uruguay
1	0	Astor, Col. Jo	male	47	1	0	PC 17757	227.525	C62 C64	C		124	New York, NY
1	1	Astor, Mrs. Jo	female	18	1	0	PC 17757	227.525	C62 C64	C	4		New York, NY
1	1	Aubart, Mme	female	24	0	0	PC 17477	69.3	B35	C	9		Paris, France
1	1	Barber, Miss	female	26	0	0	19877	78.85		S	6		
1	1	Barkworth, M	male	80	0	0	27042	30	A23	S	B		Hessle, Yorks
1	0	Baumann, M	male		0	0	PC 17318	25.925		S			New York, NY
1	0	Baxter, Mr. C	male	24	0	1	PC 17558	247.5208	B58 B60	C			Montreal, PQ
1	1	Baxter, Mrs.	female	50	0	1	PC 17558	247.5208	B58 B60	C	6		Montreal, PQ
1	1	Bazzani, Miss	female	32	0	0	11813	76.2917	D15	C	8		
1	0	Beattie, Mr.	male	36	0	0	13050	75.2417	C6	C	A		Winnipeg, MN
1	1	Beckwith, Mr	male	37	1	1	11751	52.5542	D35	S	5		New York, NY
1	1	Beckwith, Mr	female	47	1	1	11751	52.5542	D35	S	5		New York, NY
1	1	Behr, Mr. Ka	male	26	0	0	111369	30	C148	C	5		New York, NY



Example

pclass	survived	name	sex	age	sibsp	parch	ticket	fare	cabin	e
1	1	Allen, Miss. E	female	29	0	0	24160	211.3375	B5	S
1	1	Allison, Mast	male	0.92	1	2	113781	151.55	C22 C26	S
1	0	Allison, Miss	female	2	1	2	113781	151.55	C22 C26	S
1	0	Allison, Mr. H	male	30	1	2	113781	151.55	C22 C26	S
1	0	Allison, Mrs.	female	25	1	2	113781	151.55	C22 C26	S
1	1	Anderson, M	male	48	0	0	19952	26.55	E12	S
1	1	Andrews, Mi	female	63	1	0	13502	77.9583	D7	S
1	0	Andrews, Mr	male	39	0	0	112050	0	A36	S
1	1	Appleton, M	female	53	2	0	11769	51.4792	C101	S
1	0	Artagaveytia	male	71	0	0	PC 17609	49.5042		C
1	0	Astor, Col. Jo	male	47	1	0	PC 17757	227.525	C62 C64	C
1	1	Astor, Mrs. J	female	18	1	0	PC 17757	227.525	C62 C64	C
1	1	Aubart, Mme	female	24	0	0	PC 17477	69.3	B35	C
1	1	Barber, Miss	female	26	0	0	19877	78.85		S
1	1	Barkworth, M	male	80	0	0	27042	30	A23	S
1	0	Baumann, M	male		0	0	PC 17318	25.925		S
1	0	Baxter, Mr. C	male	24	0	1	PC 17558	247.5208	B58 B60	C
1	1	Baxter, Mrs.	female	50	0	1	PC 17558	247.5208	B58 B60	C
1	1	Bazzani, Miss	female	32	0	0	11813	76.2917	D15	C
1	0	Beattie, Mr. T	male	36	0	0	13050	75.2417	C6	C
1	1	Beckwith, Mr	male	37	1	1	11751	52.5542	D35	S
1	1	Beckwith, Mr	female	47	1	1	11751	52.5542	D35	S
1	1	Behr, Mr. Ka	male	26	0	0	111369	30	C148	C



Testing

Make a webpage

```
> python -m SimpleHTTPServer 8000  (For Python <= 2.7 users)

> python -m http.server 8000 (For Python 3.x users)
```

To test your web page, run the above command from the folder in which your project is located. If your page has an `index.html` file, it will appear automatically. Otherwise, add the desired html file to the end of the web address or navigate to it in the browser.

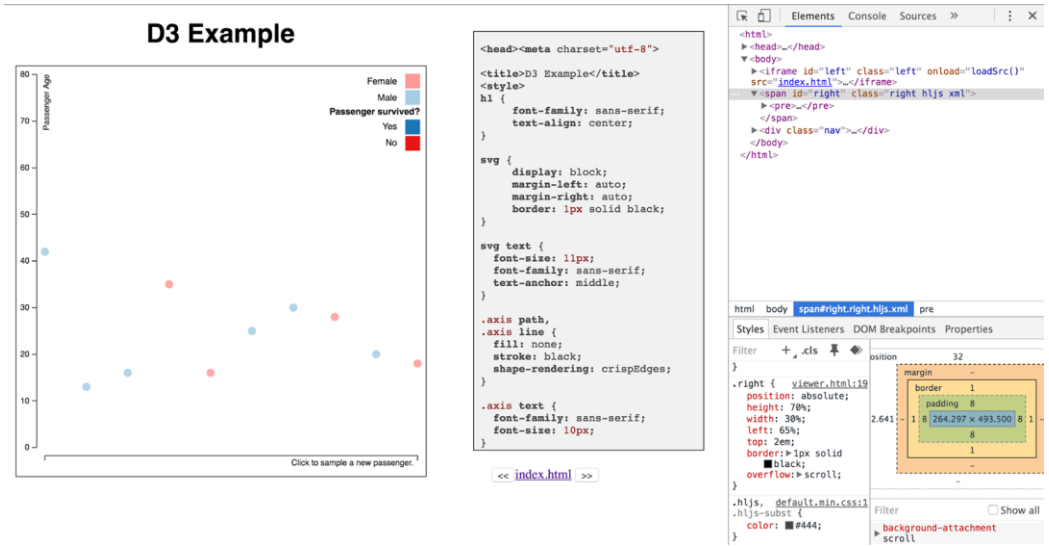
Debugging CSS, HTML, and JavaScript

Option 1: Use the JavaScript Console in your browser.

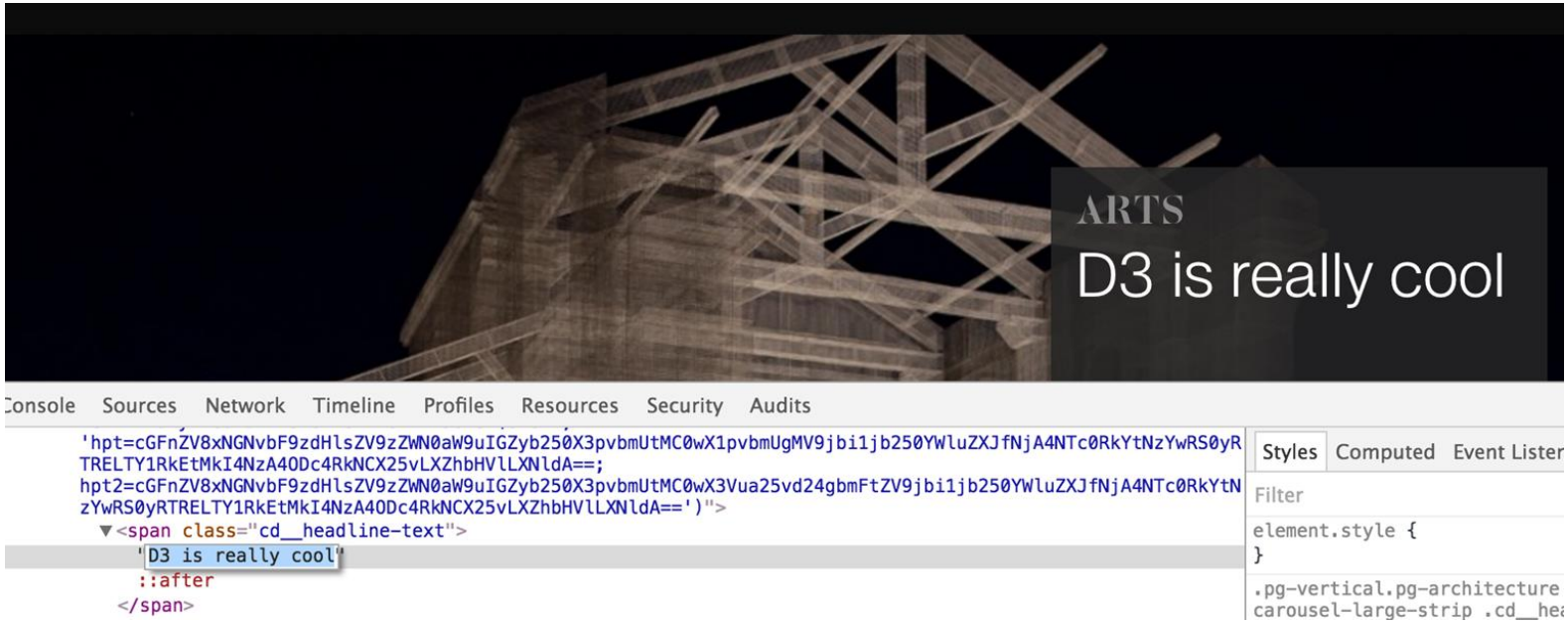
Option 2: Use Firebug or a similar debugging plugin.

Option 3: Use a Webkit Inspector.

In the JavaScript console, you can view the DOM including assigned properties and the underlying style of elements. When you select an element, you can change the properties to prototype changes before adding them to your program.



Debugging CSS, HTML, and JavaScript



1.Getting Started

Code: 1-begin.html

1-begin.html

We can start by defining the web page we want which has a header (h1) and an svg element that will hold our visualization (svg). In this style tags, we can add CSS styling for both elements defined in the HTML.

```
1  <html>
2
3  <meta charset="utf-8" />
4  <head>
5  <title>D3 Example</title>
6  <style>
7  h1 {
8      font-family: sans-serif;
9      text-align: center;
10 }
11
12 svg {
13     display: block;
14     margin-left: auto;
15     margin-right: auto;
16     border: 1px solid black;
17 }
18 </style>
19 <script src="https://d3js.org/d3.v3.min.js" charset="utf-8"></script>
20 </head>
21
22 <body>
23 <h1>D3 Example</h1>
24
25 <svg class="chart" width="500px" height="500px">
26 </svg>
27
28 </body>
29 </html>
30
```

CSS Style

HTML

2.Adding Elements

Code: 2-svg.html

Manually specifying elements

We can manually add elements to the DOM, and specify properties such as the x and y position, and the title (which appears as a tooltip).

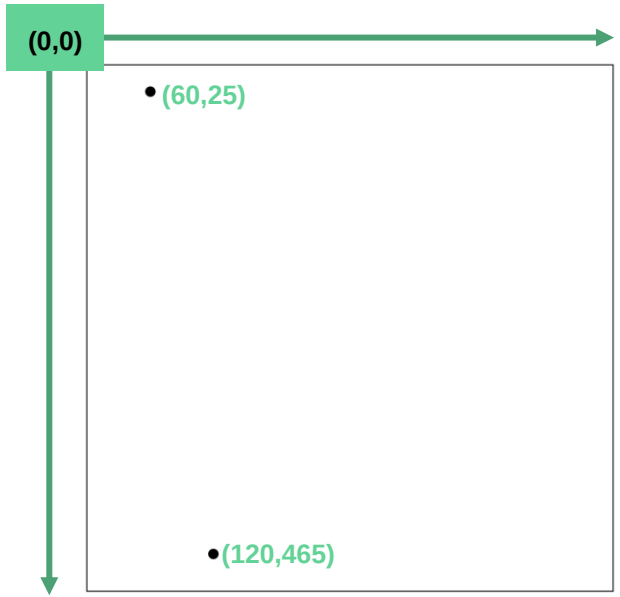
Elements can be added directly to the **<svg></svg>** element, or to **<g></g>** svg groups. They will inherit the properties of their parent group (like location and orientation).

```
<circle cx="60px" cy="25" r="5">
<title>Allen, Miss. Elisabeth Walton</title>
</circle>

<circle cx="120px" cy="465" r="5">
<title>Allison, Master. Hudson Trevor</title>
</circle>
```

Positioning Elements

Keep in mind that the origin for positioning elements is the upper left corner.

```
<circle cx="60px" cy="25" r="5">
<title>Allen, Miss. Elisabeth Walton</title>
</circle>

<circle cx="120px" cy="465" r="5">
<title>Allison, Master. Hudson Trevor</title>
</circle>
```



3.Selections

Code: 3-selection.html

Method Chaining in D3

- D3 uses a convention called method chaining. Basically, what it means is that when you call a method on an object, it performs the method and returns an object. Since the method on an object returns an object, another method can be called without having to explicitly reference an object again.

```
d3.select("body").append("h1").text("Hello");
```

```
<body>
   <h1>Hello</h1>
</body>
```

https://www.carlosrendon.me/unfinished_d3_book/d3_method_chaining.html

Selecting elements

d3.select() and **d3.selectAll()** can be used to access DOM elements by name, class, id, or many other css selectors.

```
▼<svg class="chart" width="500" height="500">
  ▶<circle r="5" cx="60" cy="50.636363636363626">…</circle>
  ▶<circle r="5" cx="136" cy="489.70545454545453">…</circle>
  ▶<circle r="5" cx="212" cy="472.8181818181818">…</circle>
  ▶<circle r="5" cx="288" cy="35">…</circle>
  ▶<circle r="5" cx="364" cy="113.18181818181819">…</circle>
  </svg>
```

```
> d3.select(".chart")
< [▼Array[1] i ]
    ▶0: svg
     length: 1
    ▶parentNode: html
    ▶__proto__: Array[0]
```

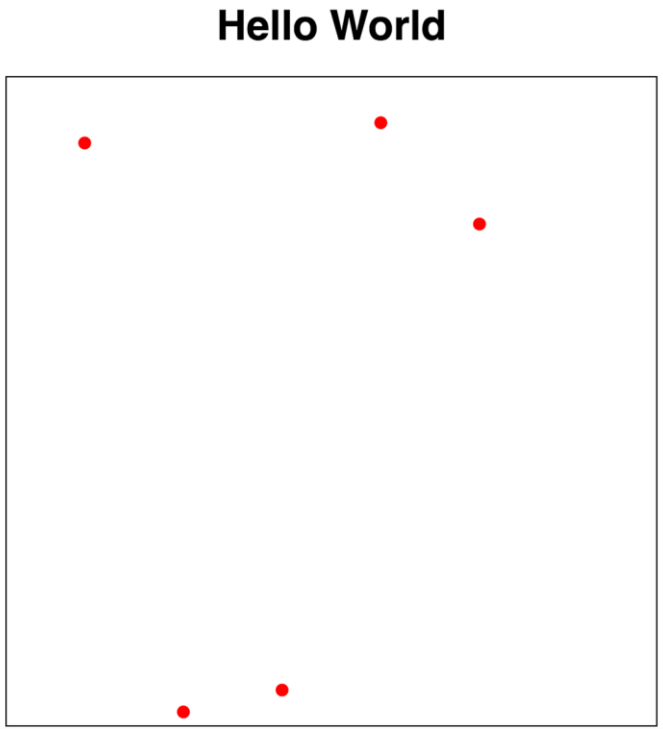
```
> d3.selectAll("circle")
< [▼Array[5] i ]
    ▶0: circle
    ▶1: circle
    ▶2: circle
    ▶3: circle
    ▶4: circle
     length: 5
    ▶parentNode: html
    ▶__proto__: Array[0]
```

```
> d3.select("svg")
< [▼Array[1] i ]
    ▶0: svg
     length: 1
    ▶parentNode: html
    ▶__proto__: Array[0]
```

Modifying selected elements

You can use access and modify the properties of selections with **attr()**, **text()**, **style()**, and other **operators.**

```
> d3.selectAll("circle").attr("fill","#ff0000")
> d3.select("h1").text("Hello World")
```

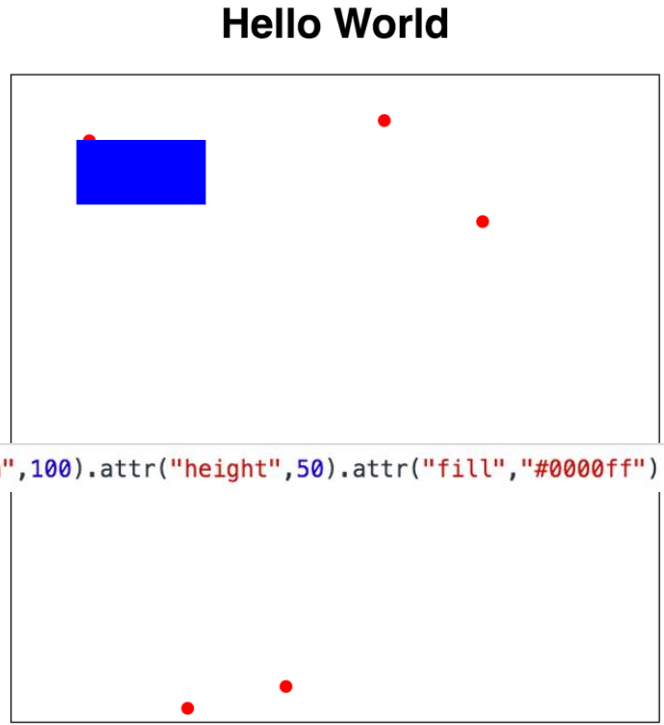


Appending elements

Through **append()**, we can add new elements anywhere in the DOM. Can combine with operators or CSS to set initial properties.

Can also get rid of elements with **remove()**.

Can store selections in variables for future use.




```
> d3.select("svg").append("rect").attr("x",50).attr("y",50).attr("width",100).attr("height",50).attr("fill","#0000ff")
```

The D3 selections page is extremely helpful!

4. Data Binding

Code: 4-binding.html

Binding

Can use **data()** function to **bind** data to a **selection**.

```
var sampleData = [
            {
            "pclass": 1,
            "survived": 1,
            "name": "Allen, Miss. Elisabeth Walton",
            "sex": "female",
            "age": 29,
            "sibsp": 0,
            "parch": 0,
            "ticket": 24160,
            "fare": 211.3375,
            "cabin": "B5",
            "embarked": "S",
            "boat": 2,
            "body": 0,
            "home.dest": "St Louis, MO"
            },
            {
            "pclass": 1,
            "survived": 1,
            "name": "Allison, Master. Hudson Trevor",
            "sex": "male",
            "age": 0.92,
            "sibsp": 1,
            "parch": 2,
            "ticket": 113781,
            "fare": 151.55,
            "cabin": "C22 C26",
            "embarked": "S",
            "boat": 11,
            "body": 0,
            "home.dest": "Montreal, PQ / Chesterville, ON"
            },
```

Can also use **data()** or **datum()** to access the data that belong to a selection.

```
> d3.selectAll("circle").data()
< [▼Object i , ▶Object, ▶Object, ▶Object, ▶Object]
    age: 29
    boat: 2
    body: 0
    cabin: "B5"
    embarked: "S"
    fare: 211.3375
    home.dest: "St Louis, MO"
    name: "Allen, Miss. Elisabeth Walton"
    parch: 0
    pclass: 1
    sex: "female"
    sibsp: 0
    survived: 1
    ticket: 24160
  ▶__proto__: Object
```

selectAll().data().enter().append()

1. **Select** all of our circles (currently we don't have any).
2. **Bind** our data (in this case, 5 rows worth)
3. **Enter** each new datum into our selection.
4. **Append** a new DOM element. There are now 5 new elements, each with their own unique data.
5. **Set** attributes with operators, using anonymous functions.

```
var scatter = d3.select(".chart").selectAll("circle")
    .data(data);

// make marks
  scatter.enter().append("circle")
    .attr("fill-opacity",0.85)
    .attr("r",function(d){ return 5;})
    .attr("stroke-width","0px")
    .append("svg:title")
      .text(function(d){ return d.name;});

   scatter
   .attr("cx",function(d,i){ return (380*i/sampleData.length)+ 60; })
   .attr("cy",function(d){ return 465-((d.age-2.5)*(430/27.5));});
```

Questions & Experiments

- What is **the major difference** between Demo 4 and Demo 3 in terms of visualizing the same dataset as the same scatterplot?
- Can we use d3.selectAll('circle').datum() to trace the data associated with the circles in Demo 3?
- Try to change the circle attributes (e.g., fill-opacity, radius, stroke-width, cx, cy) and see what happens
- By revising the code of Demo 4, try to **draw rectangles** instead of circles!

5.Scales

Code: 5-scales1.html

Code: 6-scales2.html

Specifying scales

To position the dots, we can manually specify the x and y position attributes, but this process can be tedious and error prone for complex attributes:

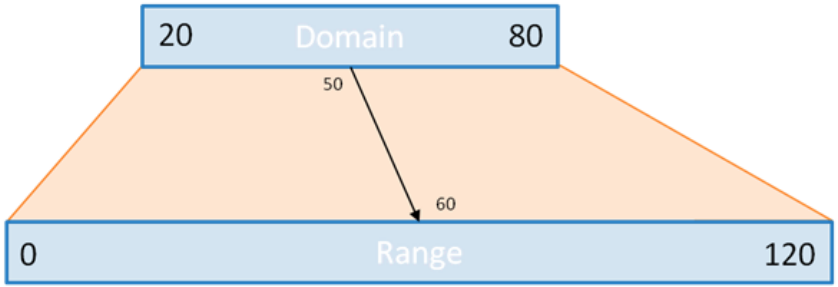
```
scatter
.attr("cx",function(d,i){ return (380*i/sampleData.length)+ 60; })
.attr("cy",function(d){ return 465-((d.age-2.5)*(430/27.5));});
```

Specifying scales

D3 **scales** are functions that map from a domain to a range (for instance, from the data domain, to the domain of the chart).

Anonymous functions can have two parameters **d** (our bound datum) and **i** (the index of our datum).

```
// x position
x = d3.scaleLinear()
    .domain([0,data.length-1])
    .range([60,440]);
```



Using a scale:

```
scatter
  .attr("cx",function(d,i){ return x(i); })
```

Manual specification:

```
scatter
.attr("cx",function(d,i){ return (380*i/sampleData.length)+ 60; })
.attr("cy",function(d){ return 465-((d.age-2.5)*(430/27.5));});
```

More scale types

d3.scaleLinear assumes linear mapping. Can also have **d3.scaleLog**, **d3.scaleSqrt**, and so on.

Can also specify **ordinal** (which include nominal data types) and **temporal** scales.

range() doesn't have to be numbers. It can also be colors or strings.

Check the D3 Scales page for more information.

```
// color
c = d3.scaleOrdinal()
    .domain(["male","female"])
    .range(["#a6cee3","#fb9a99"]);
```

D3 also has built in scales for categorical colors:

d3.schemeCategory10

#1f77b4 #ff7f0e #2ca02c #d62728 #9467bd #8c564b #e377c2 #7f7f7f #bcbd22 #17becf

Experiments

- Try to change the stroke-width to a non-zero value!
- Try to change the scale of x position, y position, fill color, stroke color
- Set the radius to a value that are proportional to “fare”, instead of being always 5px!

Questions?

Optional Exercises After Class

- Introduction to D3 (Part 1): https://observablehq.com/@jiayouwyhit/introduction-to-d3-part-1

- You are recommended to fork the above exercises before you start to work the three TO-DO tasks

Acknowledgement

The source code of this tutorial is adapted from the data visualization course taught by Dr. Jeffrey Heer at the University of Washington.