D3.js Tutorial

Kaitlyn Zhou & Younghoon Kim

Slides by Jane Hoffswell & Kanit "Ham" Wongsuphasawat
(Many thanks to them!)

Resources

Tutorials:

- An older version of UW D3 tutorial: https://uwdata.github.io/d3-tutorials/
- Let's Make a Bar Chart: https://bost.ocks.org/mike/bar/

References:

- JavaScript: The Good Parts: http://bdcampbell.net/javascript/book/javascript_the_good_parts.pdf
- Interactive Visualization for the Web: http://chimera.labs.oreilly.com/books/1230000000345/index.html

Follow Along!

https://uwdata.github.io/d3-tutorials/live/viewer.html

Download your own copy at:

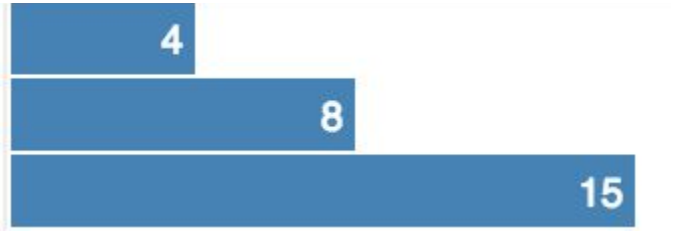
https://github.com/uwdata/d3-tutorials/blob/gh-pages/live.zip

(Click “raw” to download)

Background

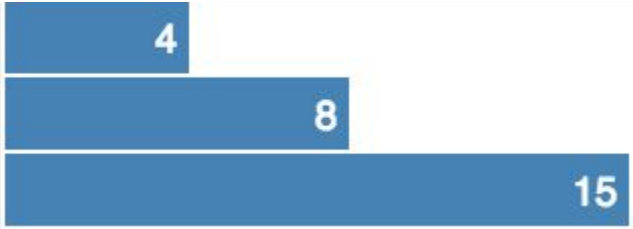
What is D3.js?

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



What is D3.js?

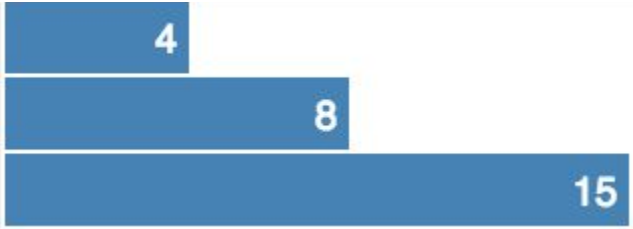
DATA are bound to DOM elements to make Data-Driven Documents




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

What is D3.js?

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

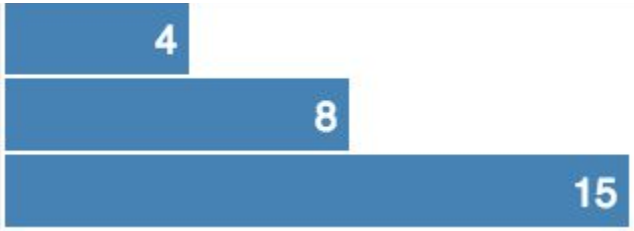



```
var data = [4,8,15];
```

```
▼<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
```

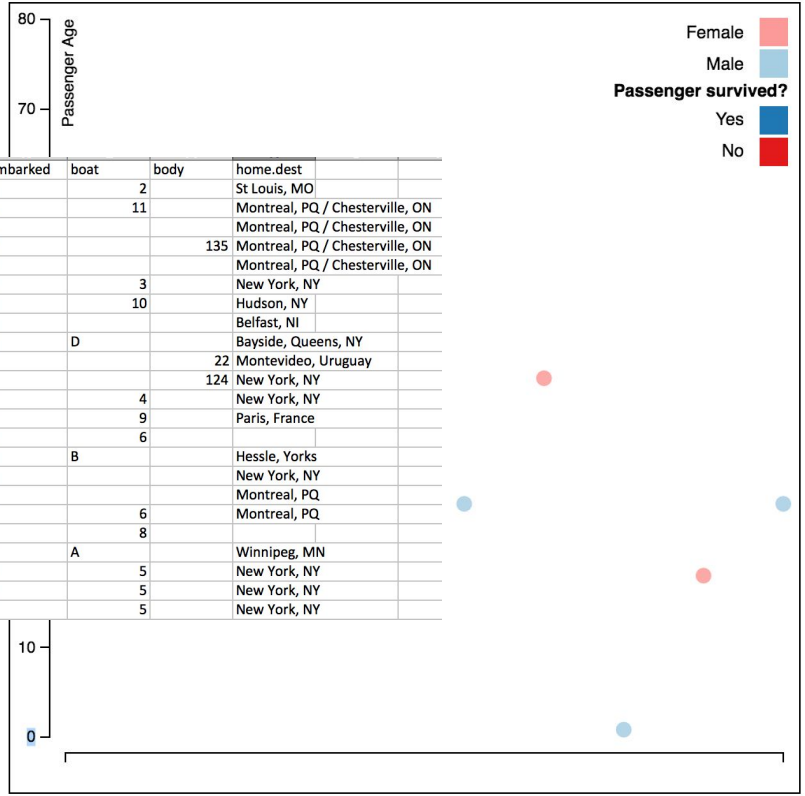
```
var data = [4,8,15];
```

```
▼<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

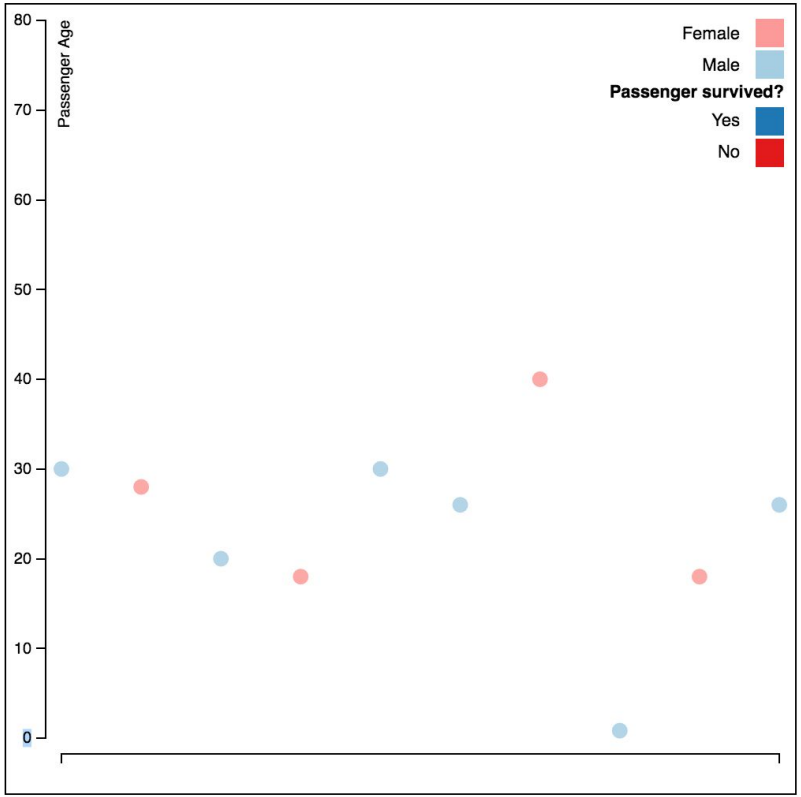
Example

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. J	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. T	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

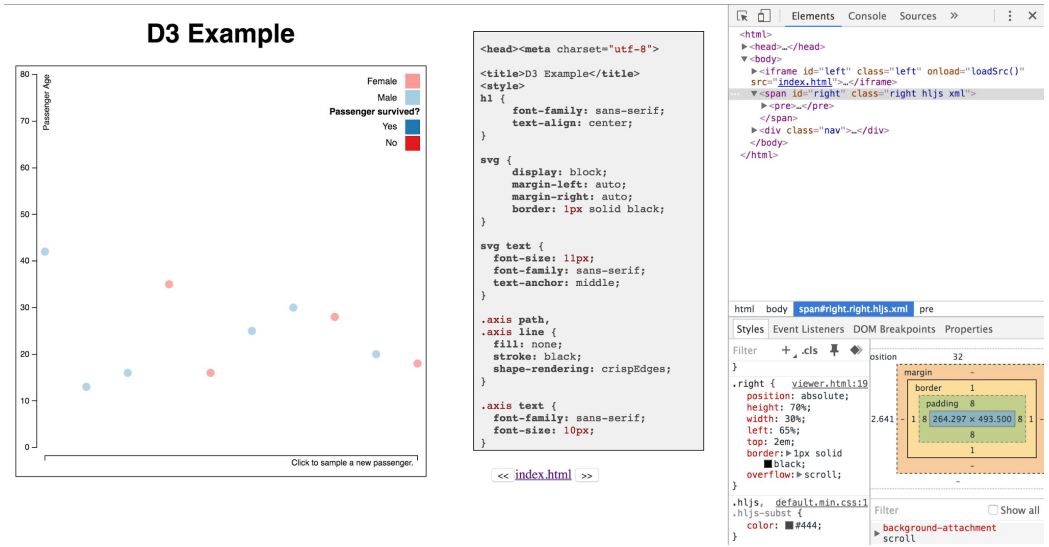
```
> python -m SimpleHTTPServer 8000
```

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.

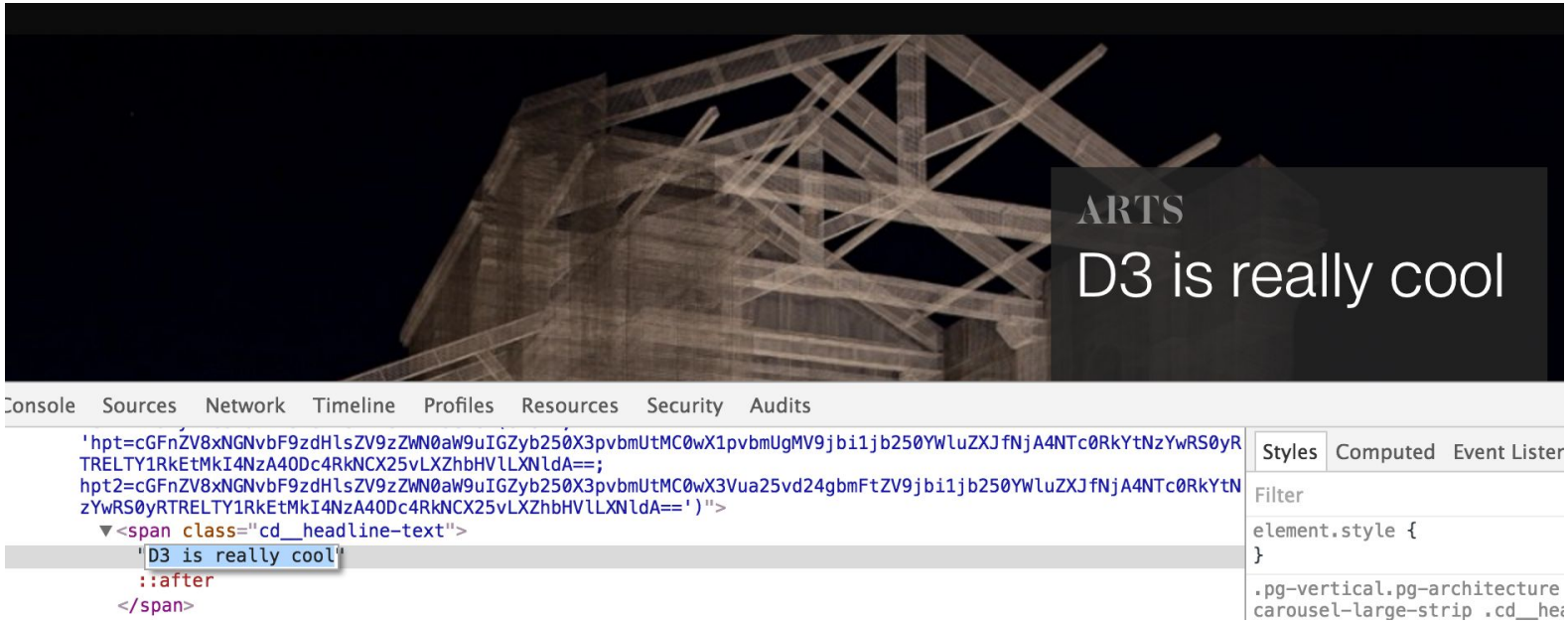
Debugging CSS, HTML, and JavaScript

Use the JavaScript Console in your browser.

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

Link: https://uwdata.github.io/d3-tutorials/live/1-begin.html

index.html

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

```
<head><meta charset="utf-8">

<title>D3 Example</title>
<style>
h1 {
  font-family: sans-serif;
  text-align: center;
}

svg {
  display: block;
  margin-left: auto;
  margin-right: auto;
  border: 1px solid black;
}

</style>
<script src="https://d3js.org/d3.v4.min.js" charset="utf-8"></script>
</head>
```

CSS Style

```
<body>
<h1>D3 Example</h1>

<svg class="chart" width="500px" height="500px">
</svg>



</body>
```

HTML

2. Adding Elements

Link: https://uwdata.github.io/d3-tutorials/live/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 on hover).

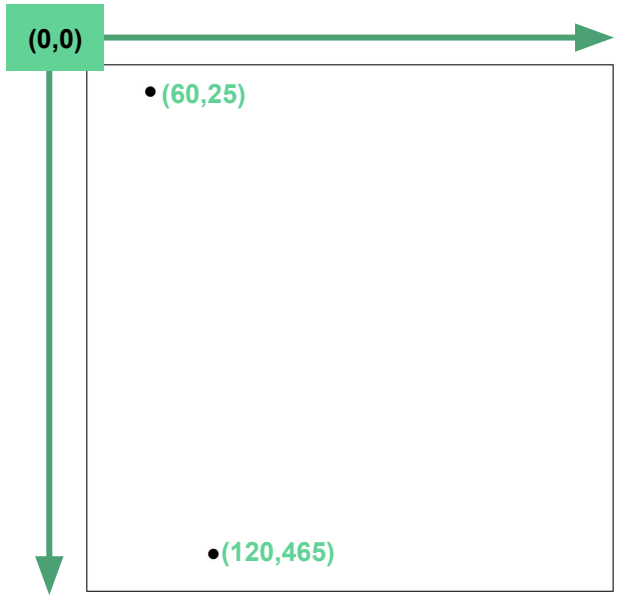
Elements can be added directly to the **<svg></svg>** element, or to **<g></g>** svg groups. They will inherit the properties of their parent group (like location and orientation).

```
<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

Link: https://uwdata.github.io/d3-tutorials/live/3-selection.html

Selecting elements

d3.select() and **d3.selectAll()** can be used to access DOM elements by name, class, id, or many other css selectors. **d3.select()** selects only the first element that matches the css selectors while **d3.selectAll()** selects all matched elements.

```
▼<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]
```

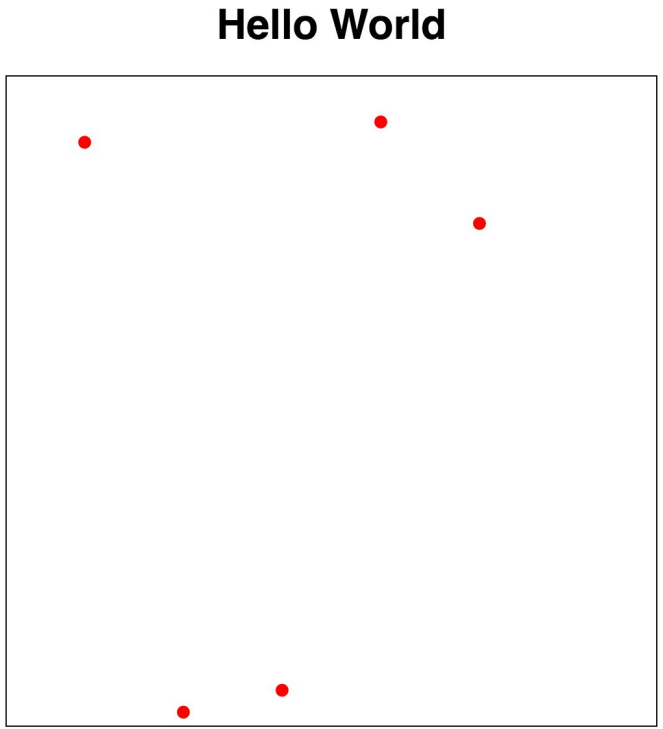
```
> d3.select("svg")
< [▼Array[1] i ]
    ▶0: svg
     length: 1
    ▶parentNode: html
    ▶__proto__: Array[0]
```

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

Modifying selected elements

You can use access and modify the properties of selections with **attr()**, **text()**, **style()**, and other **operators.** Most D3 selection methods return the selection, allowing us to chain the operator calls.

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



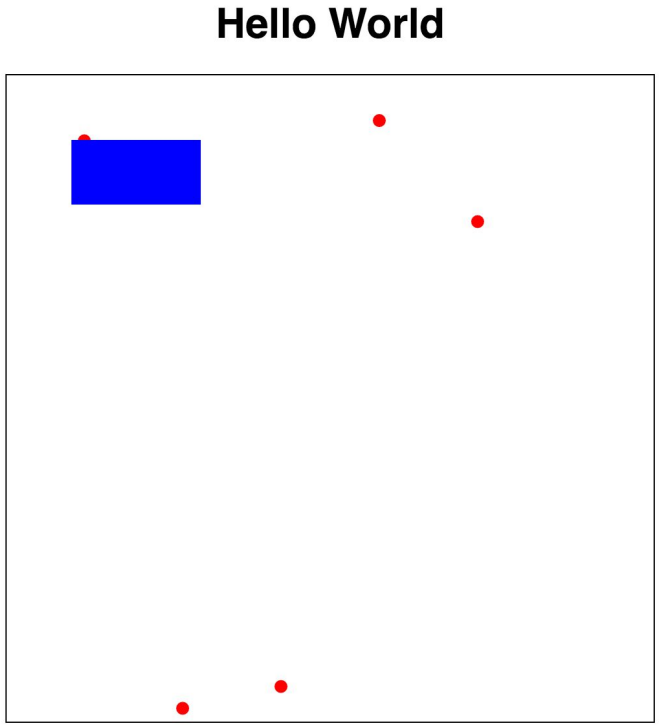
Appending elements

Through **append()**, we can add new elements anywhere in the DOM. We can then use operators or CSS to set the properties of the element.

We can also get rid of elements with **remove()**.

Finally, we can store selections in variables for future use.

D3 selections page extremely helpful!

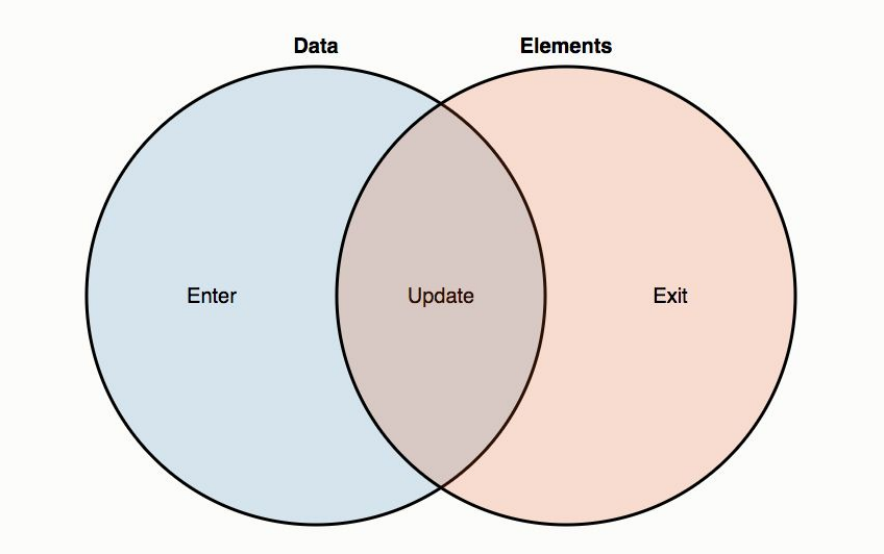



```
> d3.select("svg").append("rect").attr("x",50).attr("y",50).attr("width",100).attr("height",50).attr("fill","#0000ff")
```

4. Data Binding

Link: https://uwdata.github.io/d3-tutorials/live/4-binding.html

“Thinking with Joins” - Mike Bostock



Reference: https://bost.ocks.org/mike/join/

Binding

We can use the **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"
            },
```

We 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
```

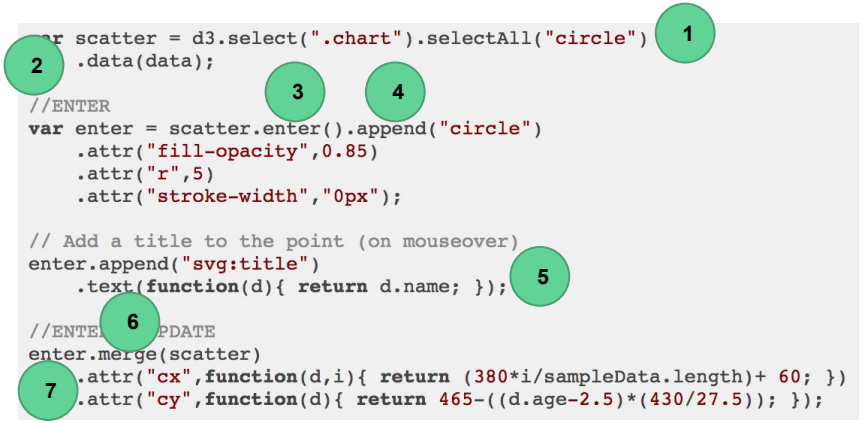
selectAll().data().enter().append()

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 from our selection.
4. **Append** a new DOM element. There are now 5 new elements, each with their own unique data.
5. **Append** titles to the new elements.
6. **Merge** our new elements into our original selections.
7. **Set** attributes with operators, using anonymous functions.

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

//ENTER
var enter = scatter.enter().append("circle")
    .attr("fill-opacity",0.85)
    .attr("r",5)
    .attr("stroke-width","0px");

// Add a title to the point (on mouseover)
enter.append("svg:title")
    .text(function(d){ return d.name; });

//ENTER + UPDATE
enter.merge(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)); });
```

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 from our selection.
4. **Append** a new DOM element. There are now 5 new elements, each with their own unique data.
5. **Append** titles to the new elements.
6. **Merge** our new elements into our original selections.
7. **Set** attributes with operators, using anonymous functions.




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

//ENTER
var enter = scatter.enter().append("circle")
    .attr("fill-opacity",0.85)
    .attr("r",5)
    .attr("stroke-width","0px");

// Add a title to the point (on mouseover)
enter.append("svg:title")
    .text(function(d){ return d.name; });

//ENTER + UPDATE
enter.merge(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)); });
```

5. Scales

Link: https://uwdata.github.io/d3-tutorials/live/5-scales1.html

Link: https://uwdata.github.io/d3-tutorials/live/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:

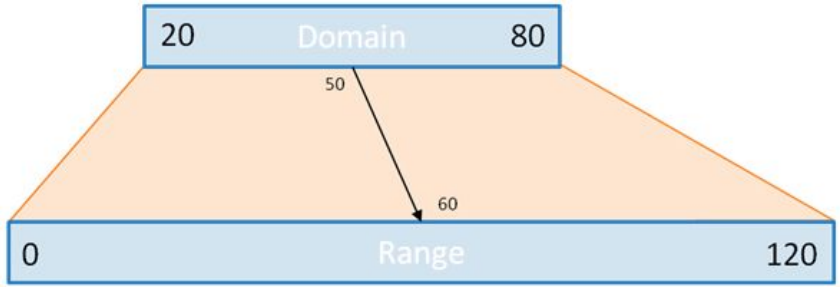
```
enter.merge(scatter)
    .attr("cx",function(d,i){ return (380*i/sampleData.length)+ 60; })
    .attr("cy",function(d){ return 465-((d.age-2.5)*(430/27.5)); });
```

Specifying scales

Scales are functions that map from a domain to a range(a domain of chart).

Anonymous functions can be used to parameterize the element's attributes using the element's data. Anonymous functions can have two parameters **d** (our bound datum) and **i** (the index of our datum).

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



Using a scale:

```
enter.merge(scatter)
    .attr("cx",function(d,i){ return x(i); })
    .attr("cy",function(d){ return y(d.age); });
```

Manual specification:

```
enter.merge(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)); });
```

https://medium.com/@mbostock/introducing-d3-scale-61980c51545f

More scale types

d3.scaleLinear create a linear mapping. You can also have **d3.scaleLog**, **d3.scaleSqrt**, and so on. You can also specify **ordinal** (which include nominal data types) and **temporal** scales. Note that the **range()** does not have to be a set of numbers; it can also be colors or strings.

Check the D3 Scales page for more information.

Note: **d3.scaleLinear** is new to D3v4 and replaces **d3.scale.linear**. This is true for all of these camelCase method names.

```
// 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

6. Axes & Legends

Link: https://uwdata.github.io/d3-tutorials/live/7-axes.html

Link: https://uwdata.github.io/d3-tutorials/live/8-legends.html

Creating axes

Axes can be generated based on the scales in your visualization. Axes are defined based on their position using **d3.axisTop**, **d3.axisBottom**, **d3.axisRight**, or **d3.axisLeft**.

Note: each of these constructors is a function; to create our axis, we create or select the element where we want to place it, and then use **call()** to apply the function to it. For more information on **call()**, see this page.

See the D3 Axes page for more information.

Scale:

```
y = d3.scaleLinear()
    .domain(d3.extent(sampleData,function(d){ return d.age; }))
    .range([465,10]);
```

Specify axis:

```
yAxis = d3.axisLeft()
    .scale(y);
```

Draw axis:

```
var yAxisGroup = canvas.append("g")
    .attr("class","axis")
    .attr("transform","translate(25,0)")
    .call(yAxis);
```

Labeling axes

Labels can be added to your visualization by adding text marks. As with any other mark, you can programmatically specify both HTML attributes and CSS styles.

```
yAxisGroup.append("text")
    .text("Passenger Age")
    .attr("transform", "rotate(-90)")
    .attr("y", 15)
    .attr("dx", -10)
    .style("text-anchor", "end");
```

Legends

Legends can be constructed just like the other elements of your visualization: by creating a new set of marks and using scales to style the attributes.

In addition to the rect for the legend mark, we can append text to create the legend labels.

```
legend.append("rect")
    .attr("x", 475)
    .attr("y", 9)
    .attr("width", 18)
    .attr("height", 18)
    .style("fill", c);

legend.append("text")
    .attr("x", 465)
    .attr("y", 18)
    .attr("dy", ".35em")
    .style("text-anchor", "end")
    .text(function(d) {
      return d.charAt(0).toUpperCase()+d.slice(1);
    });
```

7. Events & Transitions

Link: https://uwdata.github.io/d3-tutorials/live/9-events.html

Link: https://uwdata.github.io/d3-tutorials/live/10-transitions.html

Reacting to events

Event listeners can be added to marks to react to events on the underlying selection using the **on()** method. The **on()** method takes the event name and a callback function that is triggered every time the specified event happens.

An anonymous function can be used as the callback for the event listener. The input to the function **d** represents the underlying data of the mark. The scope, **this**, corresponds to the DOM element.

```
.on("mouseover", function(d){
  d3.select(this)
      .attr("stroke-width", "5px")
      .attr("r", r(d.fare));
})
.on("mouseout", function(){
  d3.select(this)
      .attr("stroke-width","0px")
      .attr("r",5);
});
```

8. Loading Files

Link: https://uwdata.github.io/d3-tutorials/live/11-csv.html

Loading data from external files

Data can be loaded from many types of external files using commands such as **d3.csv**, **d3.json**, **d3.tsv**.

The D3 functions additionally support callback functions for dealing with the resulting data or error cases.

```
d3.csv("titanic passenger list.csv",function(row,i){
  return {
    name: row.name,
    survived: (row.survived==1) ? "Yes": "No",
    sex: row.sex,
    age: +row.age,
    fare: +row.fare,
  };
}, function(error,rows){
  if(error){
    console.log(error);
  }
  rows.sort(function(a,b) { return (a.name).localeCompare(b.name); });
  allData = rows;
  makeChart(rows.slice(index,index+10));
});
```

Loading data from external files

What to do per row:
(Including creating aliases or specifying data type.

Callback function

Error handling

What to do with all returned rows (including sorting, filtering, or

```
d3.csv("titanic passenger list.csv",function(row,i){
  return {
    name: row.name,
    survived: (row.survived==1) ? "Yes": "No",
    sex: row.sex,
    age: +row.age,
    fare: +row.fare,
  };
}, function(error,rows){
  if(error){
    console.log(error);
  }
  rows.sort(function(a,b) { return (a.name).localeCompare(b.name); });
  allData = rows;
  makeChart(rows.slice(index,index+10));
});
```

9. Enter/Update/Exit

Link: https://uwdata.github.io/d3-tutorials/live/12-exit.html

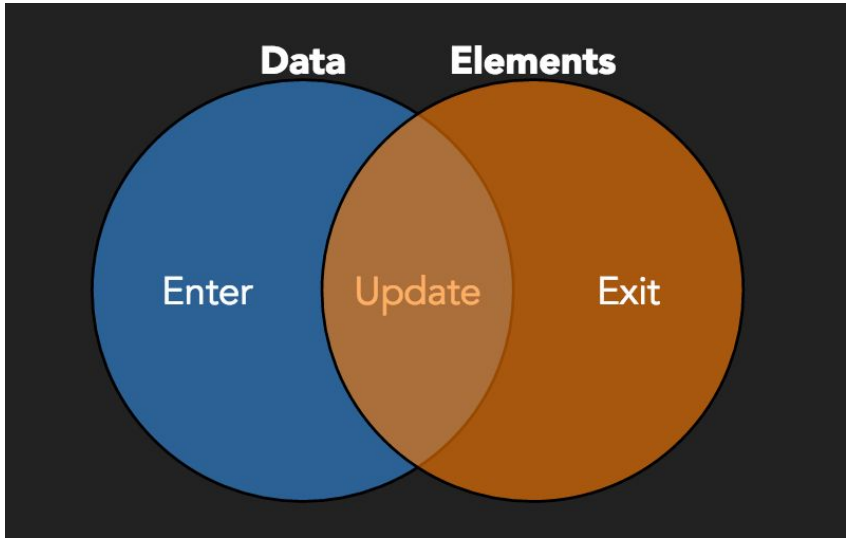
Rebinding

Three things can happen when we call **data()**:

Update: We want to change the elements we already have.

Enter: We have new data.

Exit: We have data that is no longer bound.



Rebinding

Good practice to have an update function.

1. **Bind** or rebind data
2. Perform **update** operations
3. Perform operations on **enter** set
4. Perform operations on **update+enter** sets
5. Perform **exit** operations

```
//BIND DATA
var scatter = d3.select(".chart").selectAll("circle")
    .data(data,key);
```

```
//UPDATE
scatter.attr("stroke-width","5px");
```

```
//ENTER
var enter = scatter.enter().append("circle")
    .attr("fill-opacity",0.85)
    .attr("r",5)
    .attr("cx",function(d,i){ return x(i); })
    .attr("cy",function(d){ return y(d.age); })
    .attr("stroke",function(d){ return s(d.survived); })
  .on("mouseover",function(d){
    d3.select(this).transition()
      .attr("stroke-width","5px")
      .attr("r",r(d.fare));
  })
  .on("mouseout",function(){
    d3.select(this).transition()
      .delay(1000)
      .attr("stroke-width","0px")
      .attr("r",5);
  });

// Add a title to the point (on mouseover)
enter.append("svg:title")
    .text(function(d){ return d.name;});
```

```
//ENTER + UPDATE
enter.merge(scatter).transition().duration(1000)
    .attr("cx",function(d,i){ return x(i); })
    .attr("cy",function(d){ return y(d.age); })
    .attr("fill",function(d){ return c(d.sex); })
    .attr("stroke",function(d){ return s(d.survived); })
    .attr("stroke-width","0px");
```

```
//EXIT
scatter.exit().transition().duration(1000)
    .attr("cx",0)
    .attr("fill-opacity",0)
    .remove();
```

1. Update

Things I want to happen to all of our data, whenever the function is called. Potentially overwritten by later steps.

```
//UPDATE
scatter.attr("stroke-width","5px");
```

2. Enter

Things I want to happen to all new data

Can use **append()** to make new elements for new data.

```
//ENTER
var enter = scatter.enter().append("circle")
    .attr("fill-opacity",0.85)
    .attr("r",5)
    .attr("cx",function(d,i){ return x(i); })
    .attr("cy",function(d){ return y(d.age); })
    .attr("stroke",function(d){ return s(d.survived); })
  .on("mouseover",function(d){
    d3.select(this).transition()
      .attr("stroke-width","5px")
      .attr("r",r(d.fare));
  })
  .on("mouseout",function(){
    d3.select(this).transition()
      .delay(1000)
      .attr("stroke-width","0px")
      .attr("r",5);
  });

// Add a title to the point (on mouseover)
enter.append("svg:title")
    .text(function(d){ return d.name;});
```

3. Enter+Update

Things I want to set initially. Can use transitions to have attributes fade in after creation.

Note: In D3v4 you need to **merge** the enter set into your update selection (scatter) to perform updates on the enter and update set.

```
//ENTER + UPDATE
enter.merge(scatter).transition().duration(1000)
    .attr("cx",function(d,i){ return x(i); })
    .attr("cy",function(d){ return y(d.age); })
    .attr("fill",function(d){ return c(d.sex); })
    .attr("stroke",function(d){ return s(d.survived); })
    .attr("stroke-width","0px");
```

4. Exit

Things I want to happen to old data

Can use transitions to make old data fade away

Can use **remove()** to keep only elements that are bound to our current data.

```
//EXIT
scatter.exit().transition().duration(1000)
    .attr("cx",0)
    .attr("fill-opacity",0)
    .remove();
```

Key binding

With only one argument, binding will only keep track of the amount of data we have.

If we always have the same *amount* of data, then nothing will “exit.”

Can use a argument to specify unique identifiers for data, to define whether data should enter or exit.

Here, our key is the index (row number) of the data in our original csv. Passenger name is not unique, and so would not make a good key.

```
var index = 0;

d3.csv("titanic passenger list.csv",function(row,i){
  return {
    name: row.name,
    survived: (row.survived==1) ? "Yes": "No",
    sex: row.sex,
    age: +row.age,
    fare: +row.fare,
    key: i
  };
},function(error,rows){
  if(error){
    console.log(error);
  }
  rows.sort(function(a,b) { return (a.name).localeCompare(b.name);});
  allData = rows;
  makeChart(rows.slice(index,index+10));
});

var key = function(d){ return d.key; };
```

```
//BIND DATA
var scatter = d3.select(".chart").selectAll("circle")
    .data(data,key);
```

Conclusion

Check out https://bl.ocks.org/ for d3 snippets showing important concepts.

Check out the Resources page for additional tutorials and resources.

Feedback?