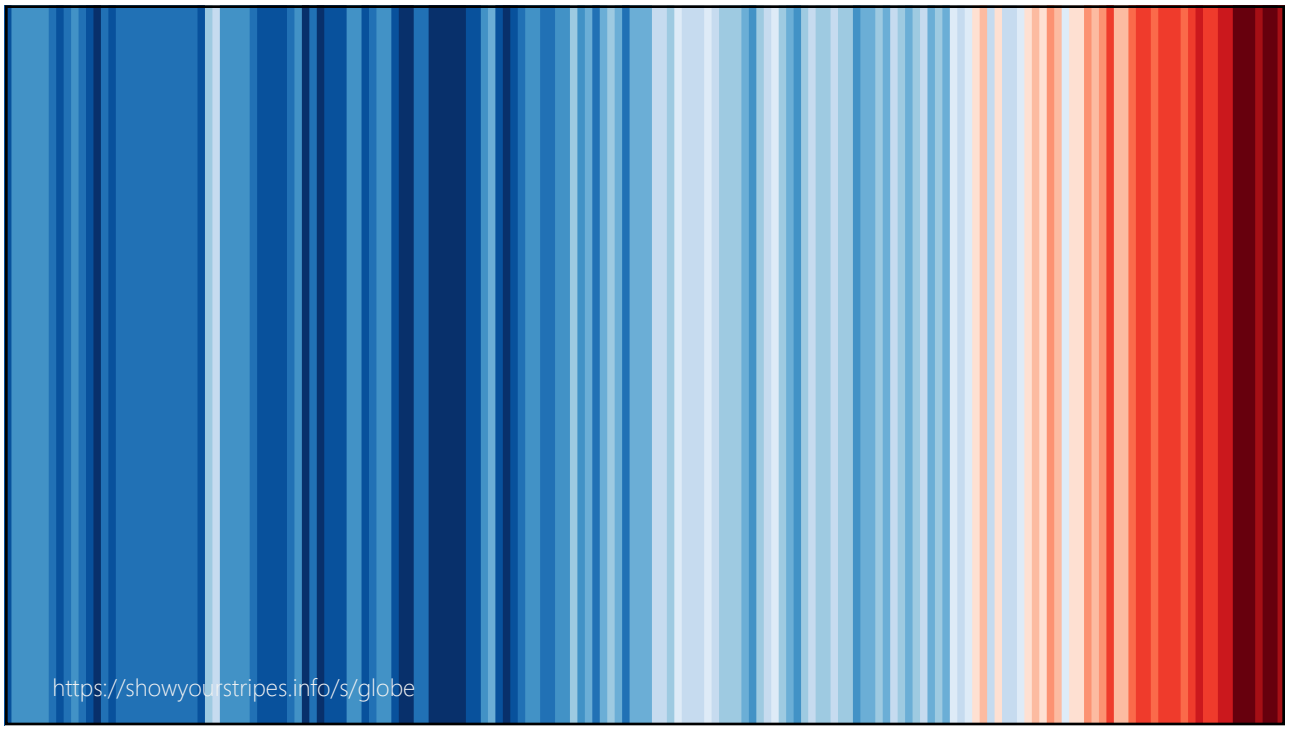
INTRODUCTION TO D3

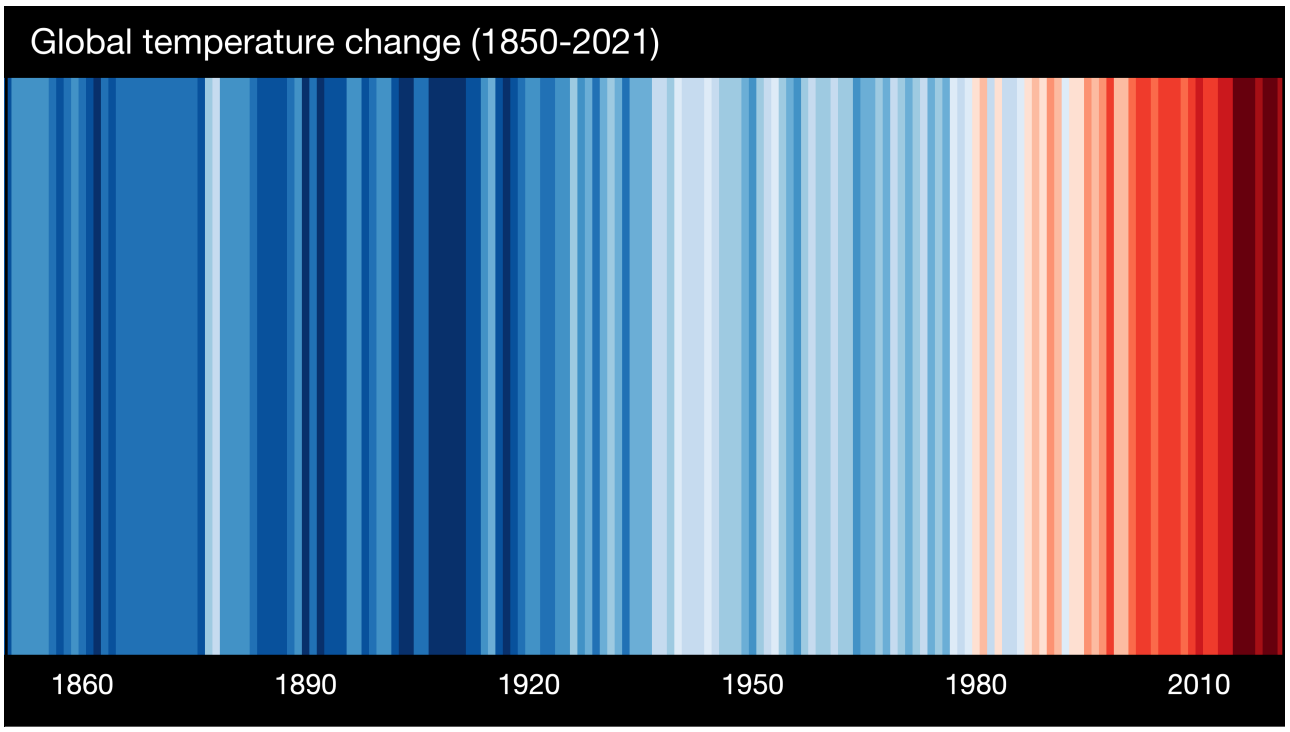
CS 448B | Fall 2025

MANEESH AGRAWALA

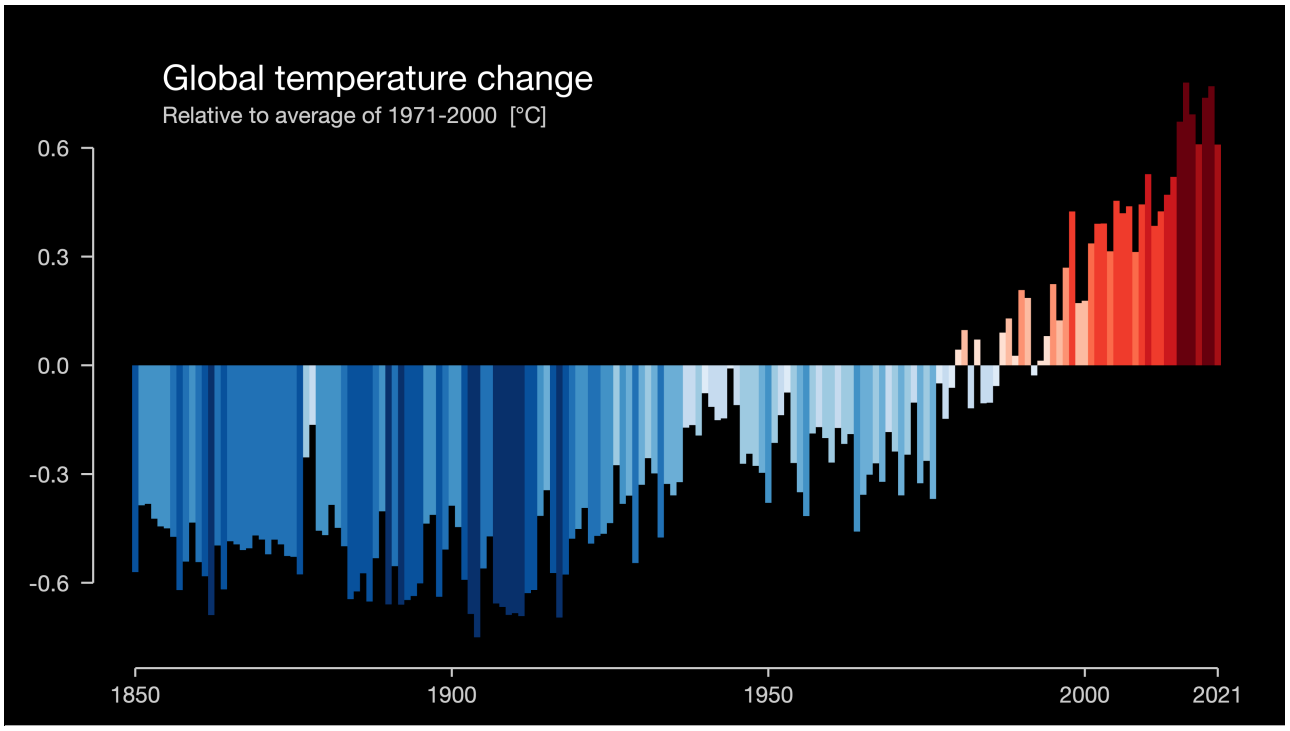
1



2


Global temperature change (1850-2021)
1860
1890
1920
1950
1980
2010

3


Global temperature change
Relative to average of 1971-2000 [°C]
0.6
0.3
0.0
-0.3
-0.6
1850
1900
1950
2000
2021

4

READING RESPONSE: QUESTIONS/THOUGHTS

- I was shocked to learn from Baur's Medium piece (quoting Aisch) that ***85% of people don't take advantage of the interactivity of interactive graphics***, at least on the New York Times website; … More broadly, it surprises me because ***users are used to interactivity these days***. Most people are used to flipping around phone operating systems, playing little games, … ***How can a visualization increase the likelihood of someone taking advantage of its interactivity?***

- Reading a bit more about interactive visualizations, it made me wonder whether ***if the purpose of interactivity is more for perceptual purposes or for better analysis (or both)***. How do you know which component of the data the viewer is going to interact with?

6

ANNOUNCEMENTS

7

ASSIGNMENT 2: EXP. DATA ANALYSIS

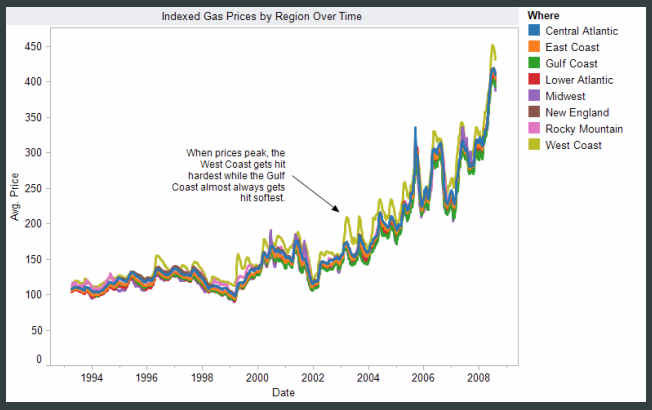
Due NOW

Use Tableau or Vega-Lite to formulate & answer data questions

First steps
- Step 1: Pick domain & data
- Step 2: Pose questions
- Step 3: Profile data
- Iterate as needed

Create visualizations
- See different views of data
- Refine questions

Author a report
- Screenshots of most insightful views (8+)
- Include titles and captions for each view



8

ASSIGNMENT 3: INTERACTION

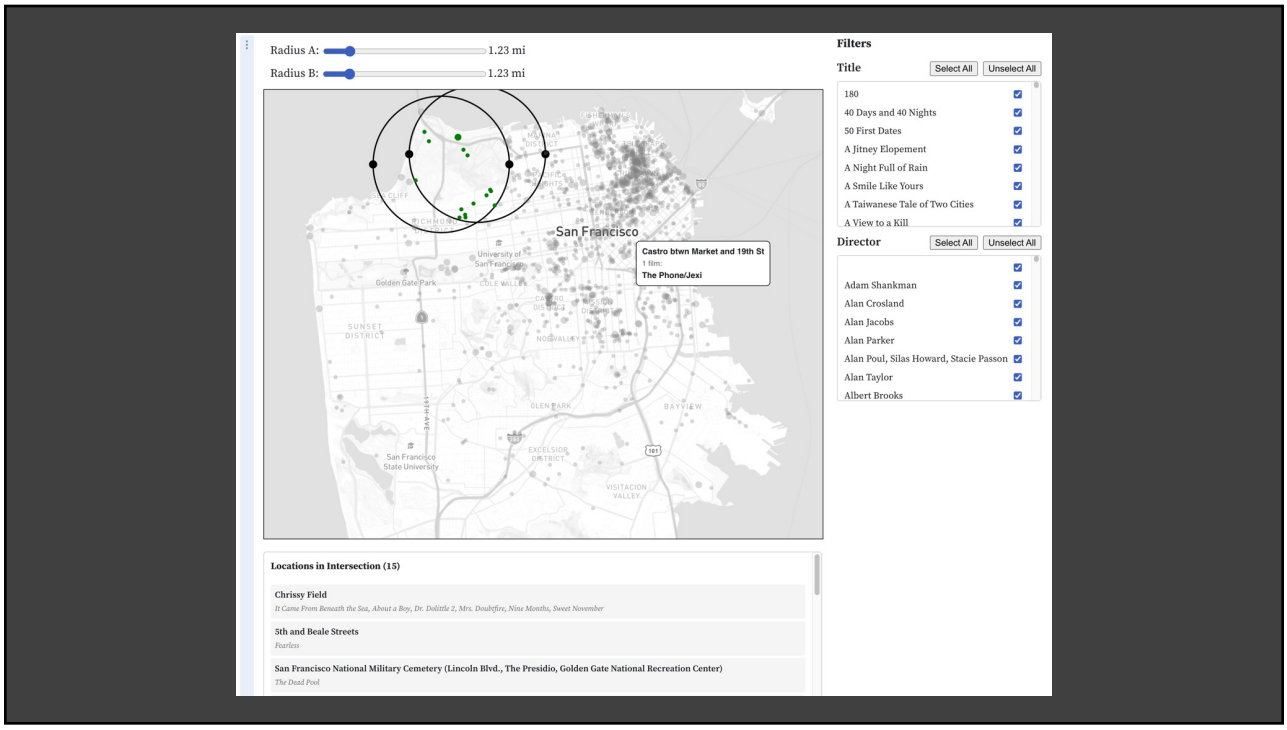
Due 10/27 10:30am

Create a small interactive dynamic query application similar to HomeFinder, but for San Francisco film locations.

1. Implement interface
2. Submit the application as a website and a short write-up on canvas

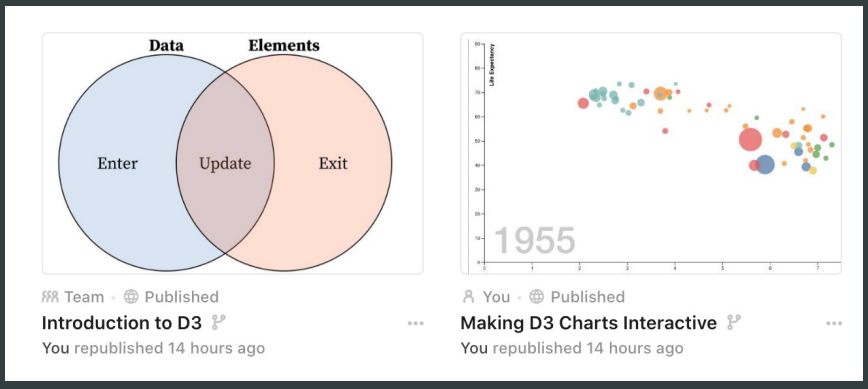
Can work alone or in pairs

9



10

D3 NOTEBOOKS TODAY & WED



11

TODAY

Learning Objectives

1. Get started with D3 and web technologies it is based on.
2. D3 binding data and joining it with DOM elements.

12

INTRODUCTION TO D3

13

WHAT IS D3?

D3: "Data-Driven Documents"
Data visualization API built **on top of HTML, CSS, JavaScript, & SVG**

Pros:
Highly-customizable
- Development and debugging tools
- Good documentation, many resources, large community

Integrates with the web

Cons:
- Very "*low-level*"

14

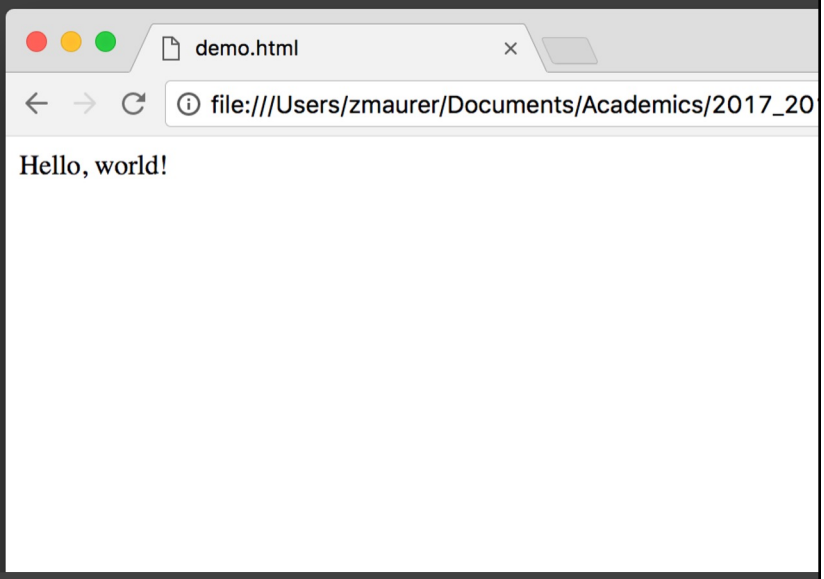
hello-world.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
</head>

<body>
  Hello, world!
</body>

</html>
```

15

hello-world.html

```
<!DOCTYPE html>
<html>
<head>
 <meta charset="utf-8">
</head>

<body>
 Hello, world!
</body>

</html>
```



16

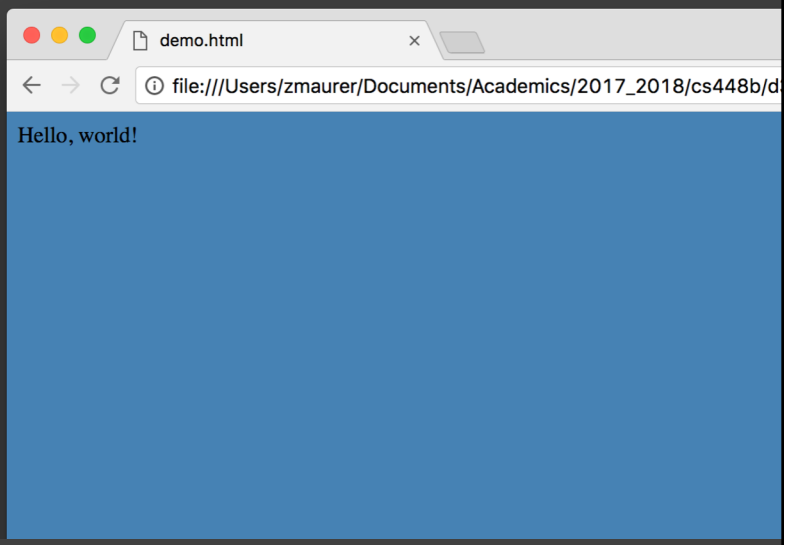
hello-css.html

```
<!DOCTYPE html>
<html>
<head>
 <meta charset="utf-8">

 <style>
  body { background: steelblue; }
 </style>

</head>

<body>
 Hello, world!
</body>

</html>
```

17

hello-css.html

```
<!DOCTYPE html>
<html>
<head>
 <meta charset="utf-8">

 <style>
  body { background: steelblue; }
 </style>

</head>

<body>
 Hello, world!
</body>

</html>
```



18

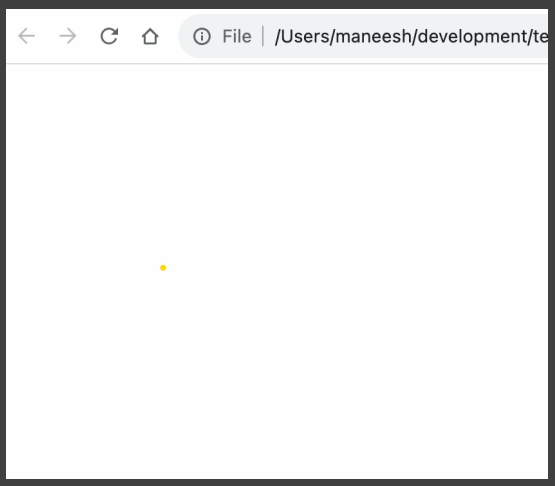
hello-svg.html

```
<!DOCTYPE html>
<html>
<head>
 <meta charset="utf-8">
 <style> /* CSS */ </style>
</head>

<body>
    <svg width="960" height="500">
      <circle cx='120' cy='150' r='60' style='fill: gold;'>
          <animate
             attributeName='r'
             from='2' to='80' begin='0' dur='3'
             repeatCount='indefinite' />
      </circle>
    </svg>
</body>
</html>
```

19

hello-svg.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <style> /* CSS */ </style>
</head>

<body>
    <svg width="960" height="500">
      <circle cx='120' cy='150' r='60' style='fill: gold;'>
          <animate
            attributeName='r'
            from='2' to='80' begin='0' dur='3'
            repeatCount='indefinite' />
      </circle>
    </svg>
</body>
</html>
```



20

DOCUMENT OBJECT MODEL (DOM)

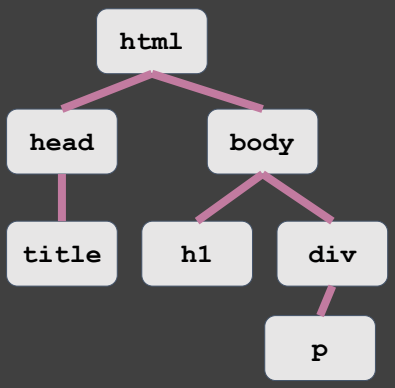
```
<html>
        <head>
                <title></title>
        </head>
        <body>
                <h1></h1>
                <div>
                        <p></p>
                </div>
        </body>
</html>
```

Adapted from Victoria Kirst's cs193x slides

21

DOCUMENT OBJECT MODEL (DOM)

```
<html>
        <head>
                <title></title>
        </head>
        <body>
                <h1></h1>
                <div>
                        <p></p>
                </div>
        </body>
</html>
```

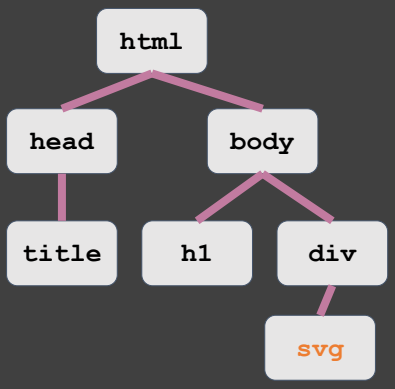


Adapted from Victoria Kirst's cs193x slides

22

DOCUMENT OBJECT MODEL (DOM)

```
<html>
        <head>
                <title></title>
        </head>
        <body>
                <h1></h1>
                <div>
                        <svg></svg>
                </div>
        </body>
</html>
```



Adapted from Victoria Kirst's cs193x slides

23

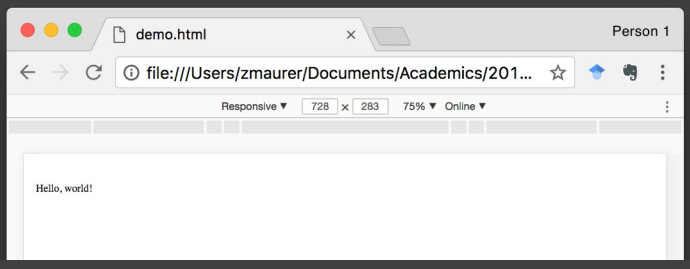
hello-javascript.html

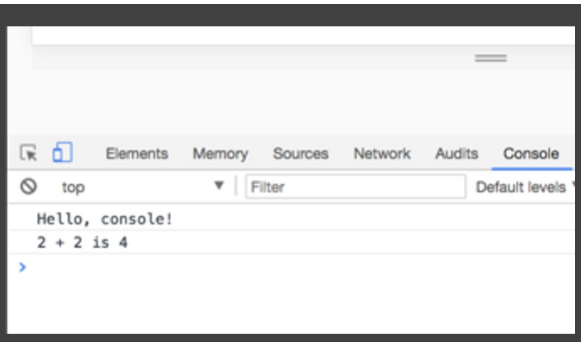
```
<!DOCTYPE html>
<html>
<head>
 <meta charset="utf-8">
 <style> /* CSS */ </style>
</head>

<body>
Hello, world!
 <script>
  console.log("Hello, console!");
   function add2(x) {
        return x + 2;
     }
  console.log("2 + 2 is " + add2(2));
 </script>
</body>
</html>
```

24

hello-javascript.html

```
<!DOCTYPE html>
<html>
<head>
 <meta charset="utf-8">
 <style> /* CSS */ </style>
</head>

<body>
Hello, world!
 <script>
  console.log("Hello, console!");
   function add2(x) {
        return x + 2;
     }
  console.log("2 + 2 is " + add2(2));
 </script>
</body>
</html>
```





25

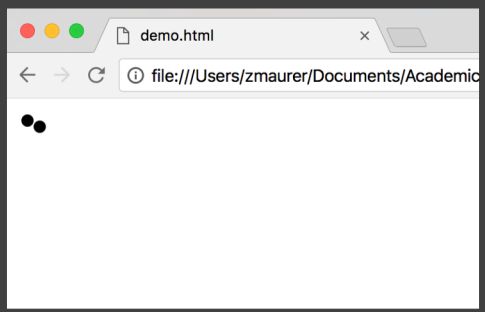
hello-d3.html

```
<!DOCTYPE html>
<html>
<head>
 <meta charset="utf-8">
 <style> /* CSS */ </style>
</head>

<body>
 <script src="https://d3js.org/d3.v7.min.js"></script>
 <script>

  // JavaScript code that handles the logic of adding SVG elements
  // that make up the visual building blocks of your data visualization

 </script>
</body>
</html>
```

26

D3 SELECTION

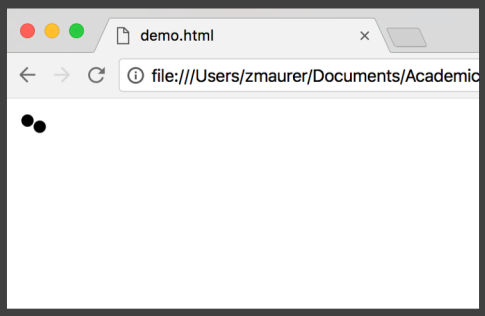
```
<html>
...
<svg width="960" height="500">
  <circle cx="10" cy="10" r="5"></circle>
  <circle cx="20" cy="15" r="5"></circle>
</svg>
...
```



27

D3 SELECTION

```
<html>
…
<svg width="960" height="500">
  <circle cx="10" cy="10" r="5"></circle>
  <circle cx="20" cy="15" r="5"></circle>
</svg>
…
```




```
<script>

// select all SVG circle elements
var circles = d3.selectAll("circle");

</script>
```

28

D3 SELECTION AND MANIPULATION

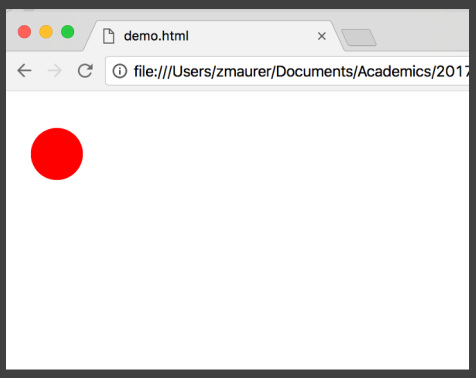
```
<html>
…
<svg width="960" height="500">
  <circle cx="10" cy="10" r="5"></circle>
  <circle cx="20" cy="15" r="5"></circle>
</svg>
…
```

```
<script>

// select all SVG circle elements
var circles = d3.selectAll("circle");


// set attributes and styles
circles.attr("cx", 40);
circles.attr("cy", 50);
circles.attr("r", 24);
circles.style("fill", "red");

</script>
```

29

D3 SELECTION AND MANIPULATION

```
<html>
…
<svg width="960" height="500">
  <circle cx="10" cy="10" r="5"></circle>
  <circle cx="20" cy="15" r="5"></circle>
</svg>
…
```




```
<script>

// select all SVG circle elements
var circles = d3.selectAll("circle");


// set attributes and styles
circles.attr("cx", 40);
circles.attr("cy", 50);
circles.attr("r", 24);
circles.style("fill", "red");

</script>
```

30

D3 SELECTION AND MANIPULATION

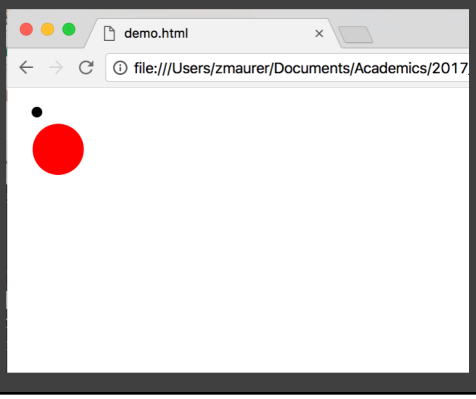
```
<html>
…
<svg width="960" height="500">
  <circle cx="10" cy="10" r="5"></circle>
  <circle cx="20" cy="15" r="5"></circle>
</svg>
…
```

```
<script>

// select SVG circle element
var circles = d3.select("circle");


// set attributes and styles
circles.attr("cx", 40);
circles.attr("cy", 50);
circles.attr("r", 24);
circles.style("fill", "red");

</script>
```

31

D3 SELECTION AND MANIPULATION

```
<html>
…
<svg width="960" height="500">
  <circle cx="10" cy="10" r="5"></circle>
  <circle cx="20" cy="15" r="5"></circle>
</svg>
…
```

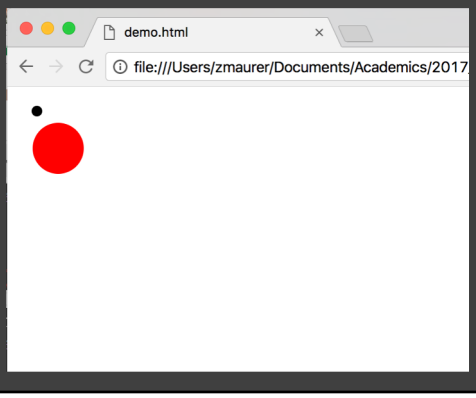



```
<script>

// select SVG circle element
var circles = d3.select("circle");


// set attributes and styles
circles.attr("cx", 40);
circles.attr("cy", 50);
circles.attr("r", 24);
circles.style("fill", "red");

</script>
```

32

D3 SELECTION AND MANIPULATION

```
<html>
…
<svg width="960" height="500">
  <circle cx="10" cy="10" r="5"></circle>
  <circle cx="20" cy="15" r="5"></circle>
</svg>
…
```




```
<script>

// all together!!
d3.select("circle")
      .attr("cx", 40)
      .attr("cy", 50)
      .attr("r", 24)
      .style("fill", "red");

</script>
```

33

D3 BINDING DATA & JOIING DOM ELEMENTS

```
gapminder = ▸ Array(693) [Object, Object, Object, Object, Object, Object, Object, Object, Object, Object, Object, Object, Object, Object,
```

year	country	cluster	pop	life_expect	fertility
1955	"Afghanistan"	0	8891209	30.332	7.7
1960	"Afghanistan"	0	9829450	31.997	7.7
1965	"Afghanistan"	0	10997885	34.02	7.7
1970	"Afghanistan"	0	12430623	36.088	7.7
1975	"Afghanistan"	0	14132019	38.438	7.7
1980	"Afghanistan"	0	15112149	39.854	7.8
1985	"Afghanistan"	0	13796928	40.822	7.9
1990	"Afghanistan"	0	14669339	41.674	8
1995	"Afghanistan"	0	20881480	41.763	8
2000	"Afghanistan"	0	23898198	42.129	7.4792

*Note that we have put a **Imports** section at the end of this document where we import various utility functions such as the `printTable()` function.*

Let's also extract a subset of this data for the year 2005, sort it by population and slice off the top 10 countries. Expand the cell below to see how we obtain this subset. We will use this subset in our first few D3 examples.

```
listData = ▸ Array(10) [Object, Object, Object, Object, Object, Object, Object, Object, Object, Object]
```

```
// extract the 10 most populous countries in 2005
listData = gapminder
  .filter(d => d.year === 2005)
  .sort((a, b) => b.pop - a.pop)
  .slice(0, 10)
```

34

D3 BINDING DATA & JOIING DOM ELEMENTS

```
listData = ▾ Array(10) [
  0: ▸ Object {year: 2005, country: "China", cluster: 4, pop: 1304887562, life_expect: 72.98, fertility: 1.62}
  1: ▸ Object {year: 2005, country: "India", cluster: 0, pop: 1154638713, life_expect: 65.39, fertility: 2.96}
  2: ▸ Object {year: 2005, country: "United States", cluster: 3, pop: 296842670, life_expect: 77.73, fertility: 2.06}
  3: ▸ Object {year: 2005, country: "Indonesia", cluster: 4, pop: 228805144, life_expect: 68.26, fertility: 2.43}
  4: ▸ Object {year: 2005, country: "Brazil", cluster: 3, pop: 186797334, life_expect: 72.81, fertility: 1.97}
  5: ▸ Object {year: 2005, country: "Pakistan", cluster: 0, pop: 174372098, life_expect: 61.05, fertility: 4.64}
  6: ▸ Object {year: 2005, country: "Bangladesh", cluster: 0, pop: 140912590, life_expect: 67.44, fertility: 2.81}
  7: ▸ Object {year: 2005, country: "Nigeria", cluster: 2, pop: 140490722, life_expect: 56.63, fertility: 6.07}
  8: ▸ Object {year: 2005, country: "Japan", cluster: 4, pop: 127798373, life_expect: 82.5, fertility: 1.27}
  9: ▸ Object {year: 2005, country: "Mexico", cluster: 3, pop: 105442402, life_expect: 75.01, fertility: 2.5}
]
```

```
// extract the 10 most populous countries in 2005
listData = gapminder
  .filter(d => d.year === 2005)
  .sort((a, b) => b.pop - a.pop)
  .slice(0, 10)
```

35

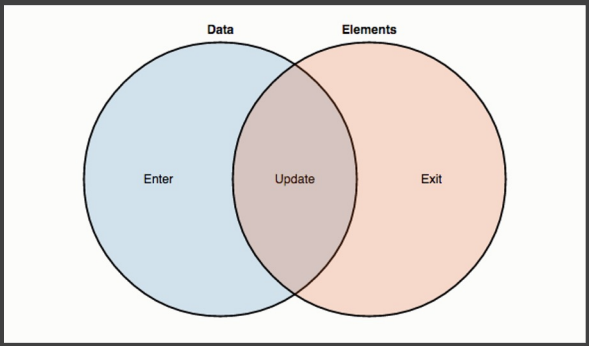
D3 BINDING DATA & JOIING DOM ELEMENTS

1. China: 1303182268
2. India: 1080264388
3. United States: 295734134
4. Indonesia: 218465000
5. Brazil: 186112794
6. Pakistan: 162419946
7. Bangladesh: 144319628
8. Nigeria: 128765768
9. Japan: 127417244
10. Mexico: 106202903

```
{
  const ol = d3.create('ol');

  ol.selectAll('li') // select all list elements (orange circle above)
    .data(listData)  // bind all our data values (blue circle above)
    .join(
      enter  => enter.append('li'), // append an li element for each entering item
      update => update,             // do nothing with items that match an existing element
      exit   => exit.remove()       // remove li elements whose backing data is now gone
    )
      .text(d => `${d.country}: ${d.pop}`)

  return ol.node();
}
```

36

D3 BINDING DATA & JOIING DOM ELEMENTS



A *join* creates three sub-selections:

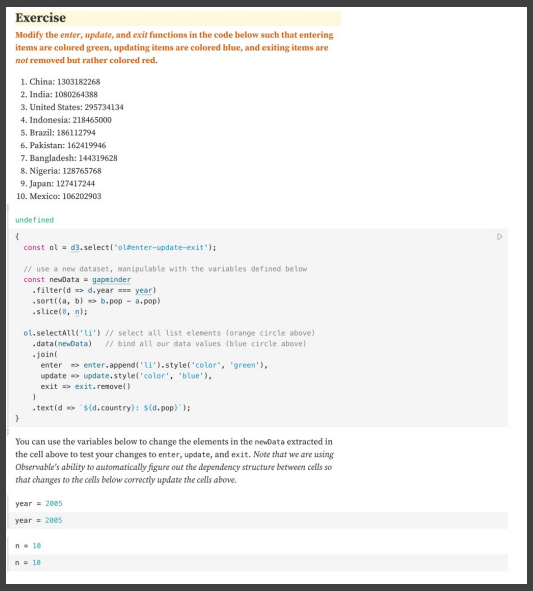
***Enter*:** selection containing placeholders for every data value that did not have a corresponding DOM element in the original selection

***Update*:** selection containing *existing* DOM elements that match a bound data value

***Exit*:** selection that also contains *existing* DOM elements, but for which a matching data value was not found

37

D3 BINDING DATA & JOIING DOM ELEMENTS



Exercise

Modify the *enter*, *update*, and *exit* functions in the code below such that entering items are colored green, updating items are colored blue, and exiting items are *not* removed but rather colored red.

1. China: 1303182268
2. India: 1080264388
3. United States: 295734134
4. Indonesia: 218465000
5. Brazil: 186112794
6. Pakistan: 162419946
7. Bangladesh: 144319628
8. Nigeria: 128765768
9. Japan: 127417244
10. Mexico: 106202903

undefined

```
{
  const ol = d3.select('ol#enter-update-exit');

  // use a new dataset, manipulable with the variables defined below
  const newData = gapminder
    .filter(d => d.year === year)
    .sort((a, b) => b.pop - a.pop)
    .slice(0, n);

  ol.selectAll('li') // select all list elements (orange circle above)
    .data(newData)   // bind all our data values (blue circle above)
    .join(
      enter  => enter.append('li').style('color', 'green'),
      update => update.style('color', 'blue'),
      exit => exit.remove()
    )
    .text(d => `${d.country}: ${d.pop}`);
}
```

You can use the variables below to change the elements in the `newData` extracted in the cell above to test your changes to `enter`, `update`, and `exit`. *Note that we are using Observable's ability to automatically figure out the dependency structure between cells so that changes to the cells below correctly update the cells above.*

```
year = 2005
```

```
year = 2005
```

```
n = 10
```

```
n = 10
```

39