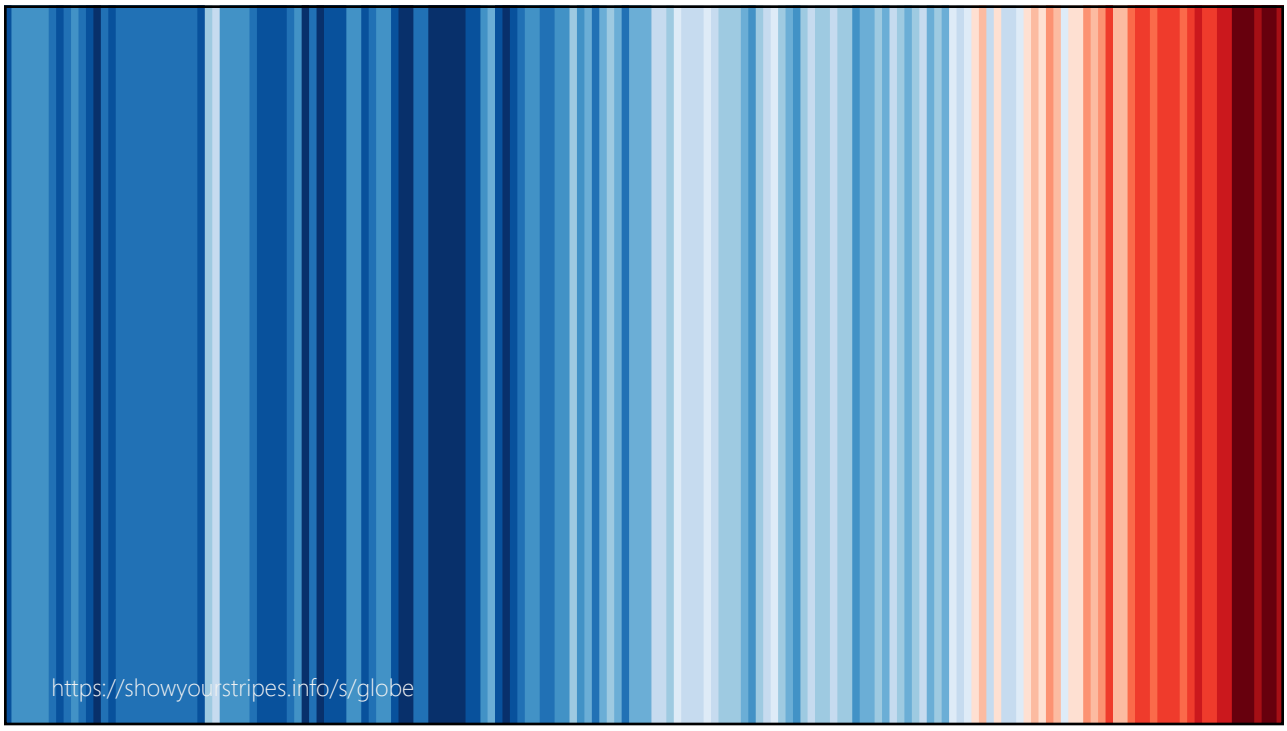
INTRODUCTION TO D3

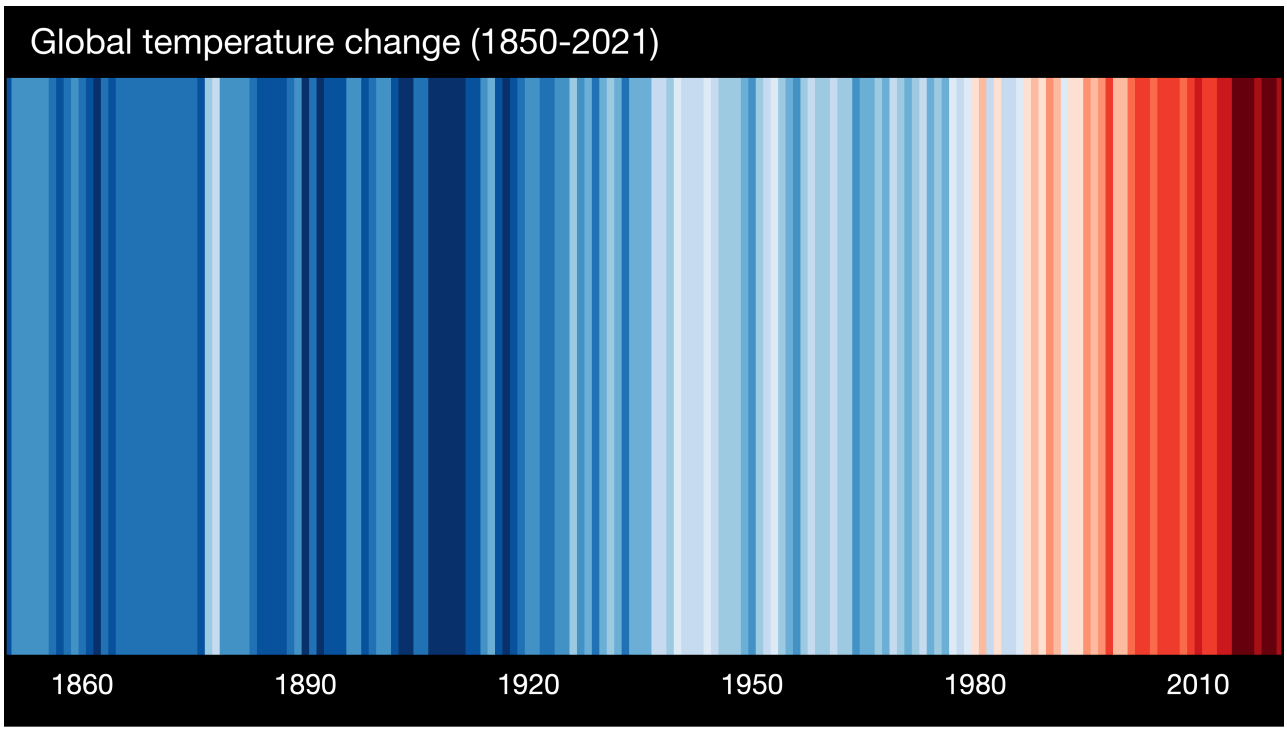
CS 448B | Fall 2023

MANEESH AGRAWALA

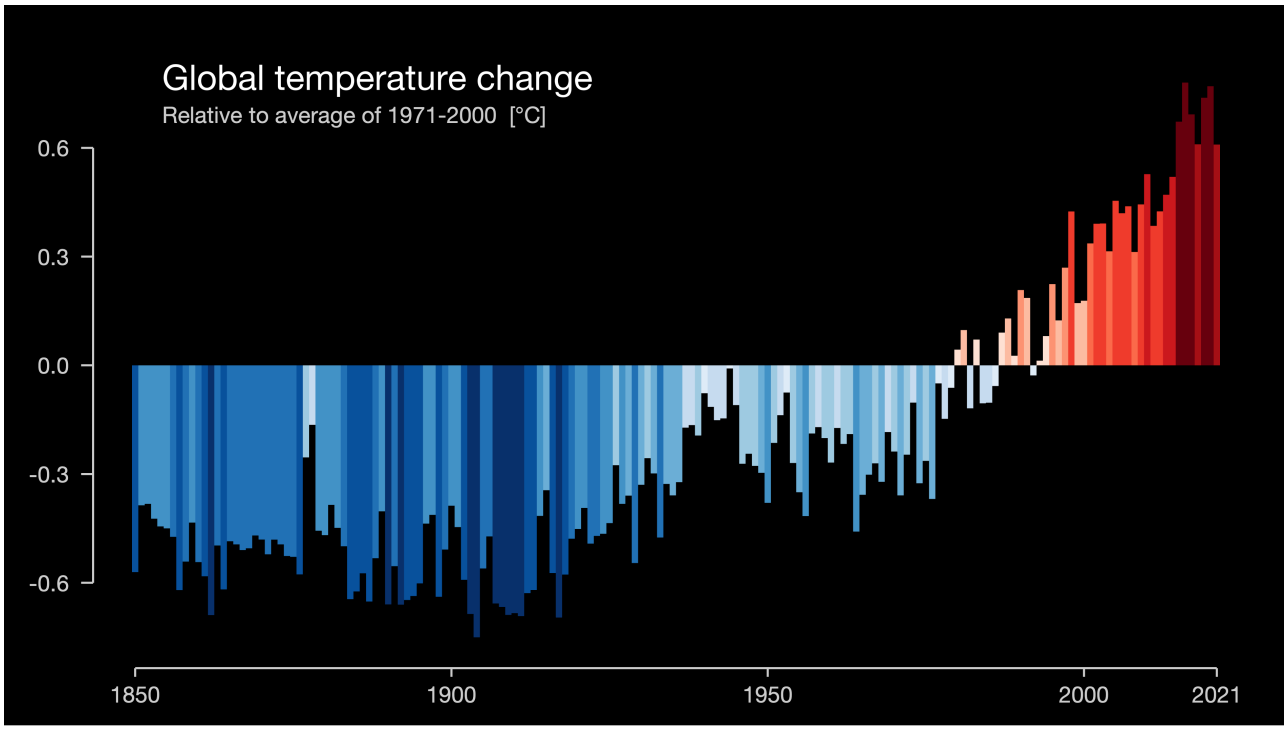
1



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

LAST TIME: INTERACTION

5

LAST TIME

Learning Objectives

1. Conceptual models, system models and the gulfs of execution and evaluation
2. Common interaction techniques: Selection, Brushing and Linking and Dynamic Queries

6

DYNAMIC QUERIES

7

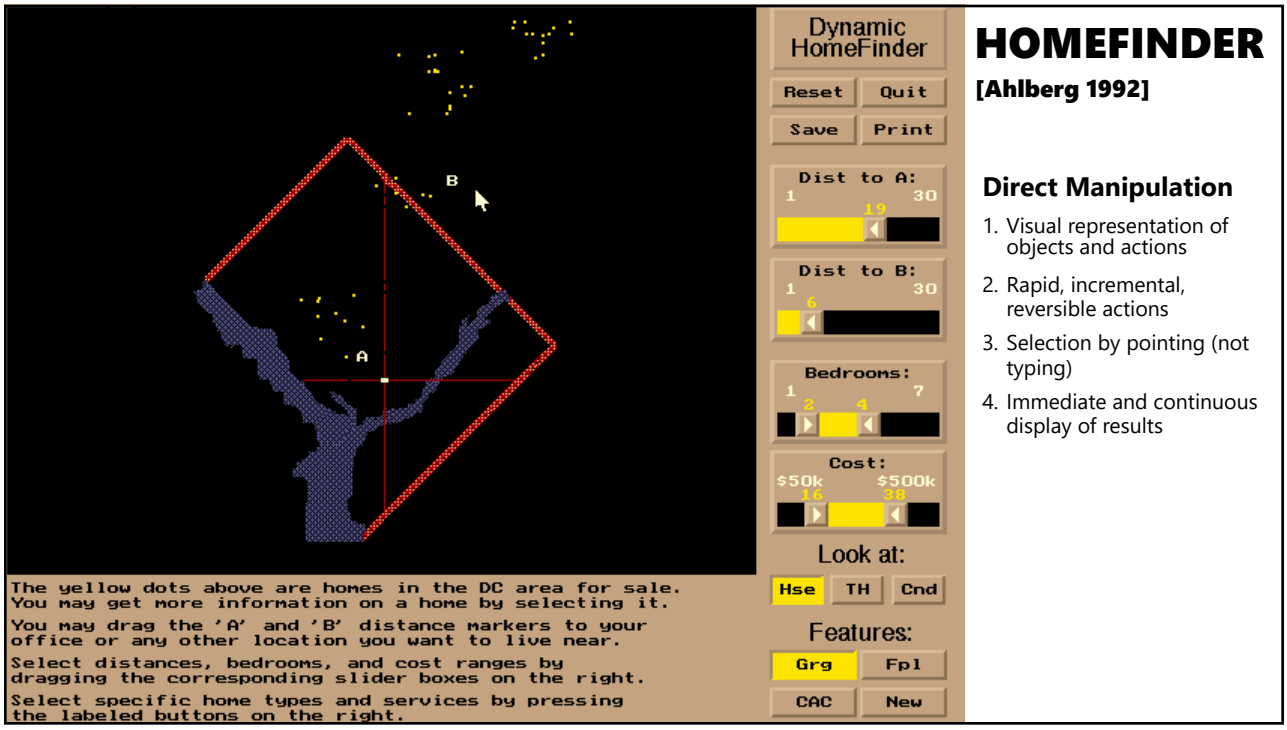
QUERY & RESULTS

SELECT house FROM east bay
WHERE price < 1,000,000 AND bedrooms > 2
ORDER BY price

Issues

1. For programmers
2. Rigid syntax
3. Only shows exact matches
4. Too few or too many hits
5. No hint on how to reformulate the query
6. Slow question-answer loop
7. Results returned as table

8



HOMEFINDER

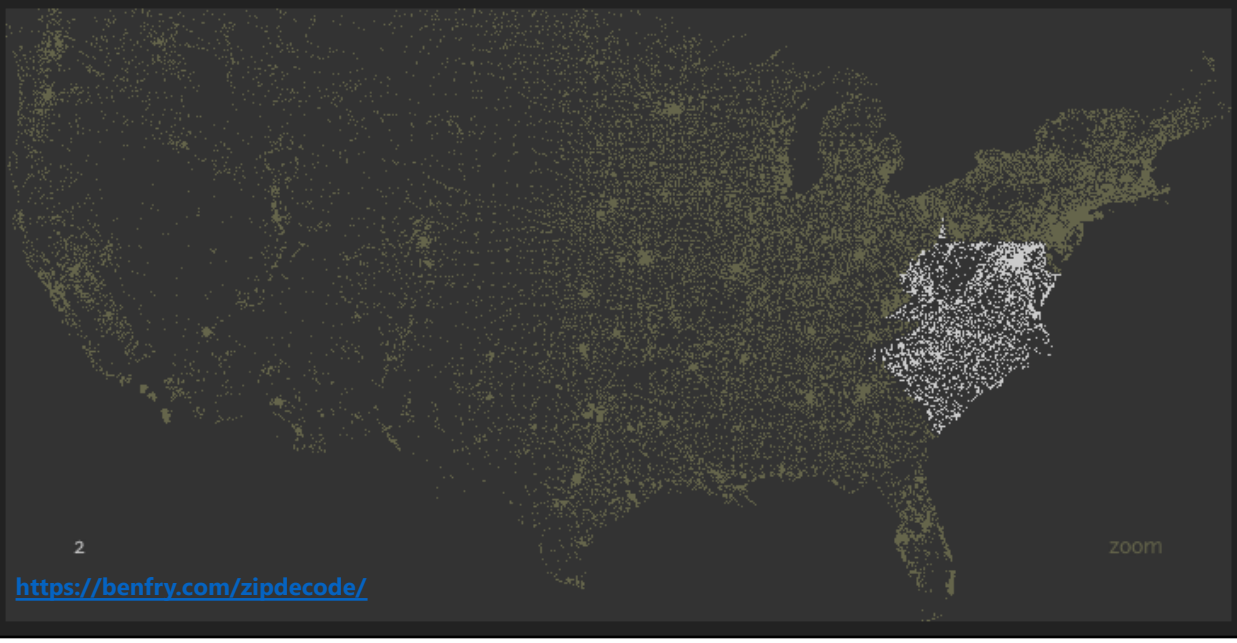
[Ahlberg 1992]

Direct Manipulation

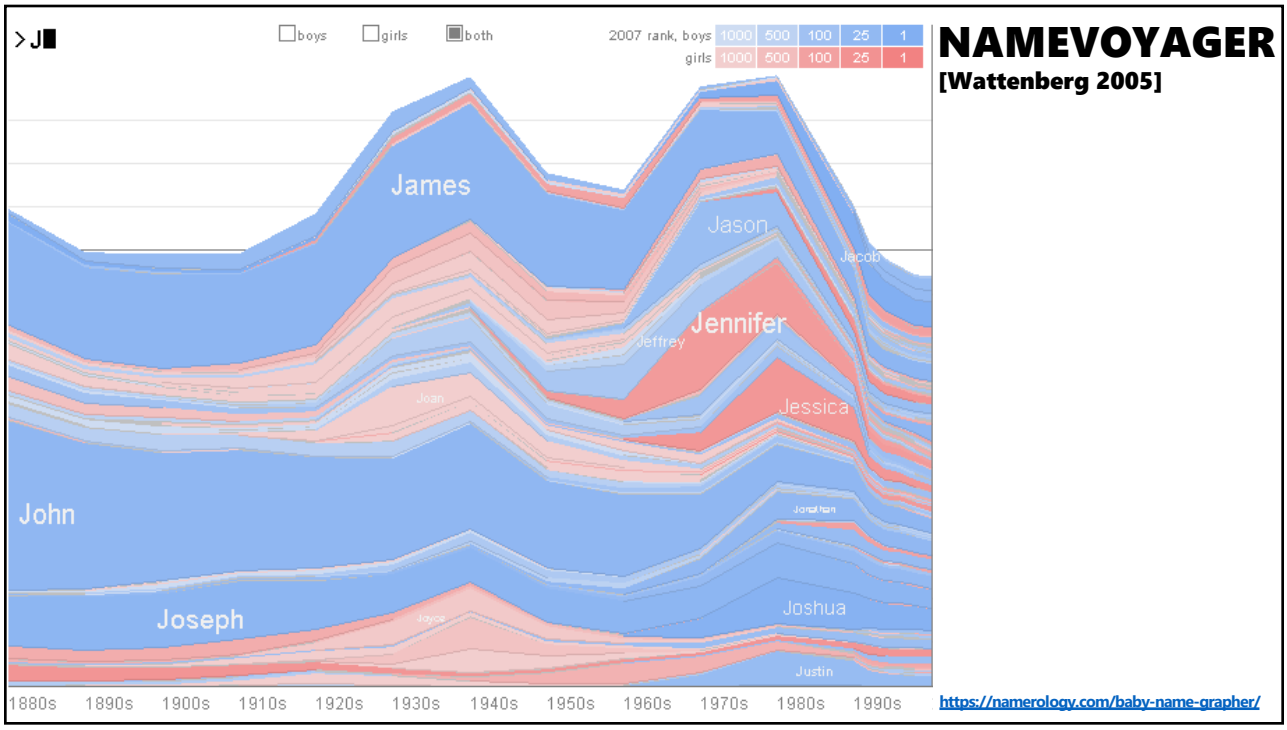
1. Visual representation of objects and actions
2. Rapid, incremental, reversible actions
3. Selection by pointing (not typing)
4. Immediate and continuous display of results

9

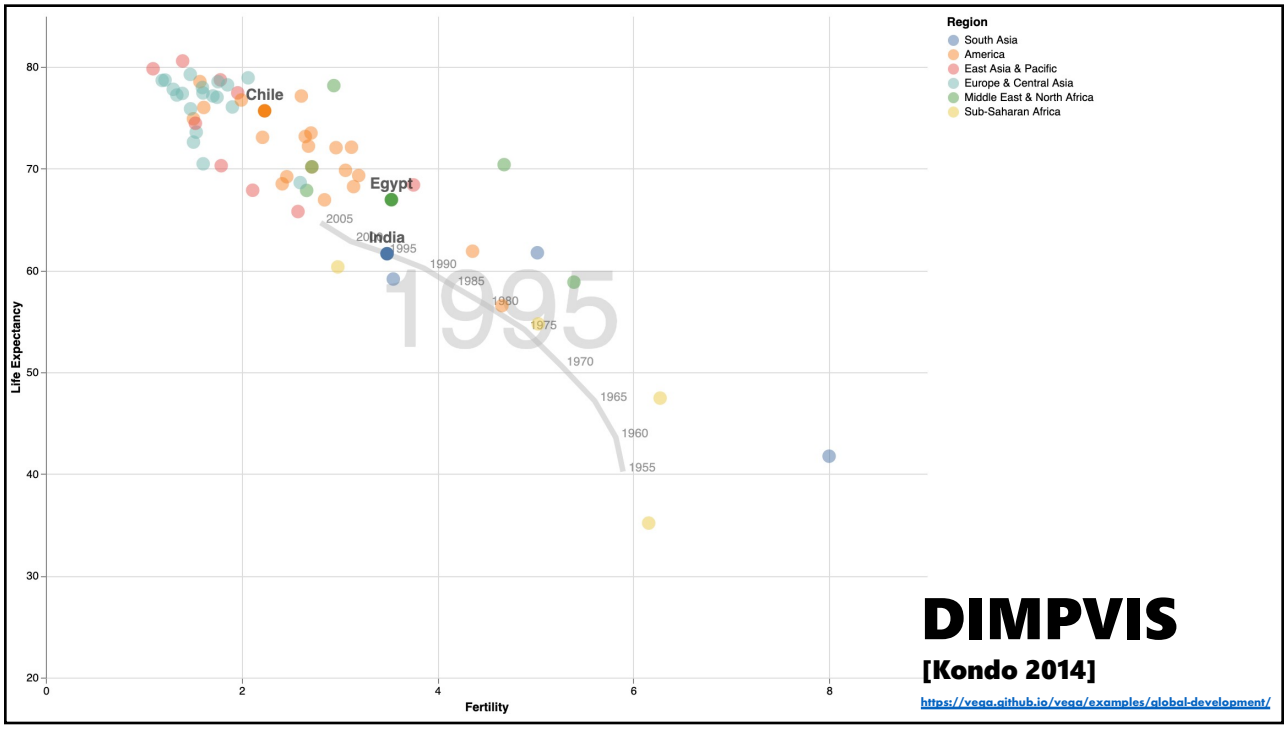
ZIPDECODE [Fry 2004]



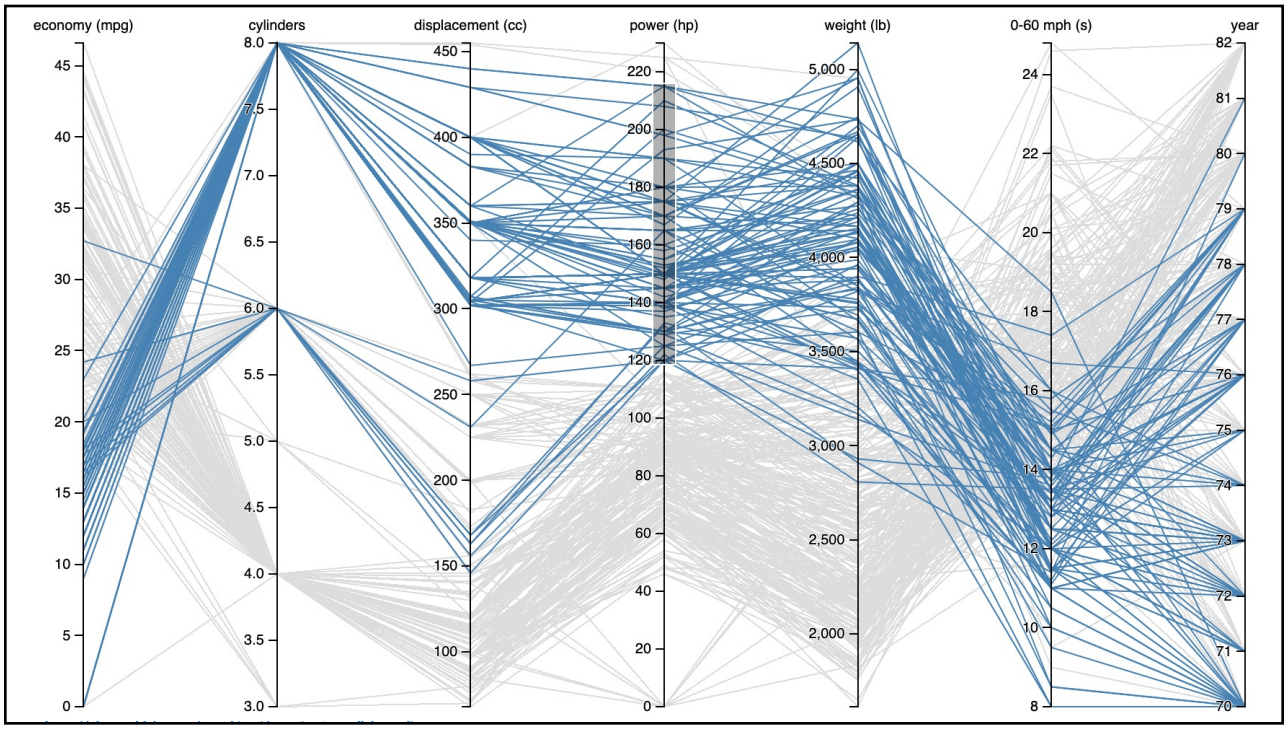
16


boys
girls
both
2007 rank, boys 1000 500 100 25 1
girls 1000 500 100 25 1
James
Jason
Jacob
Jennifer
Jeffrey
Joan
Jessica
John
Joseph
Joyce
Joshua
Justin
1880s 1890s 1900s 1910s 1920s 1930s 1940s 1950s 1960s 1970s 1980s 1990s
NAMEVOYAGER
[Wattenberg 2005]
https://namerology.com/baby-name-grapher/

17

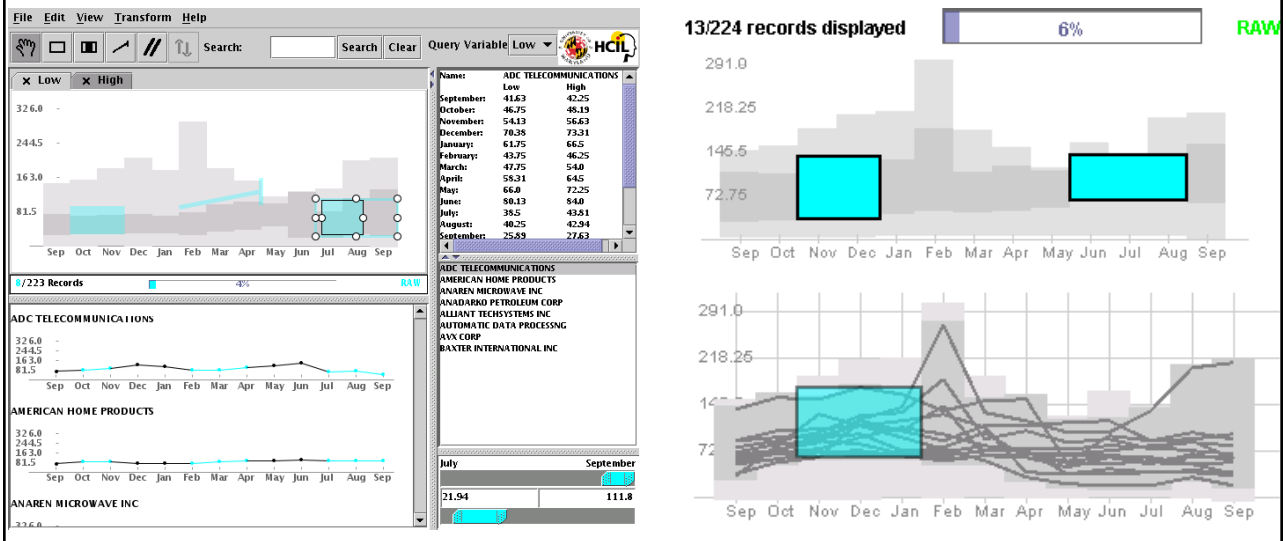

Region
South Asia
America
East Asia & Pacific
Europe & Central Asia
Middle East & North Africa
Sub-Saharan Africa
Chile
Egypt
India
1995
Life Expectancy
Fertility
DIMPVIS
[Kondo 2014]
https://vega.github.io/vega/examples/global-development/

18



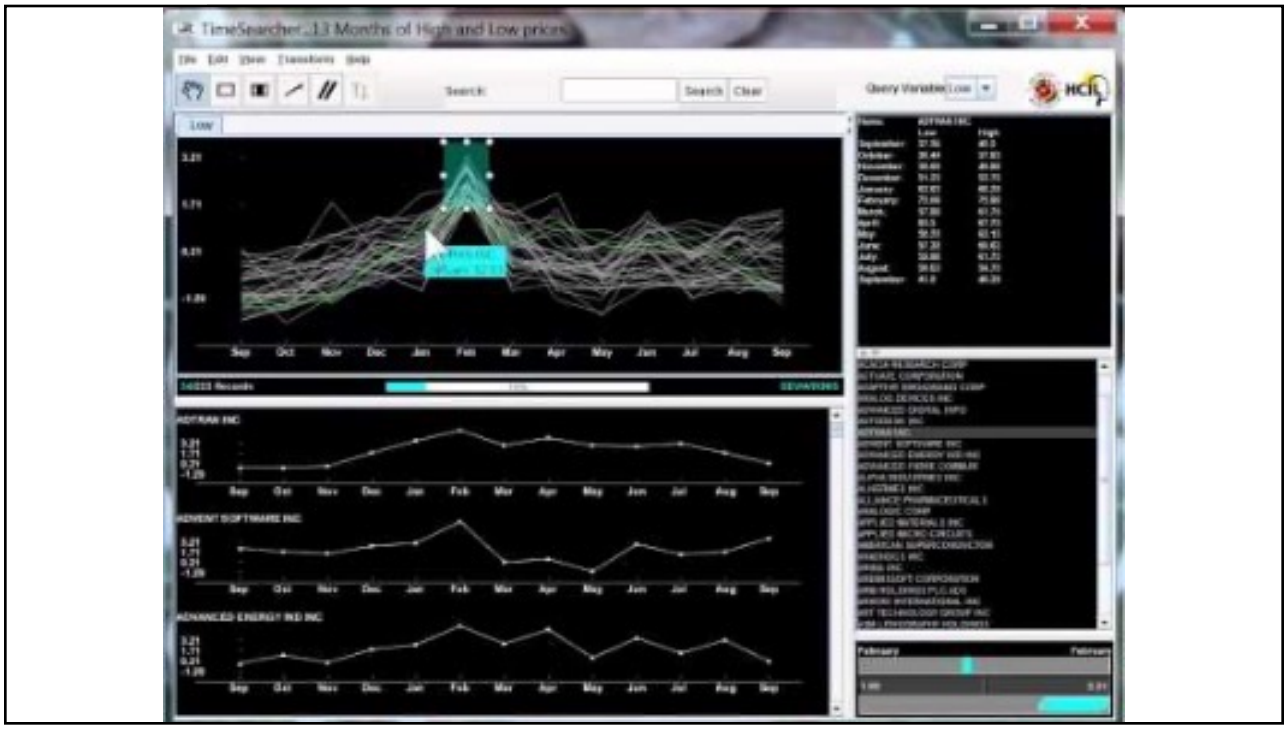
19

TIMESEARCHER [Hocheiser 2002]

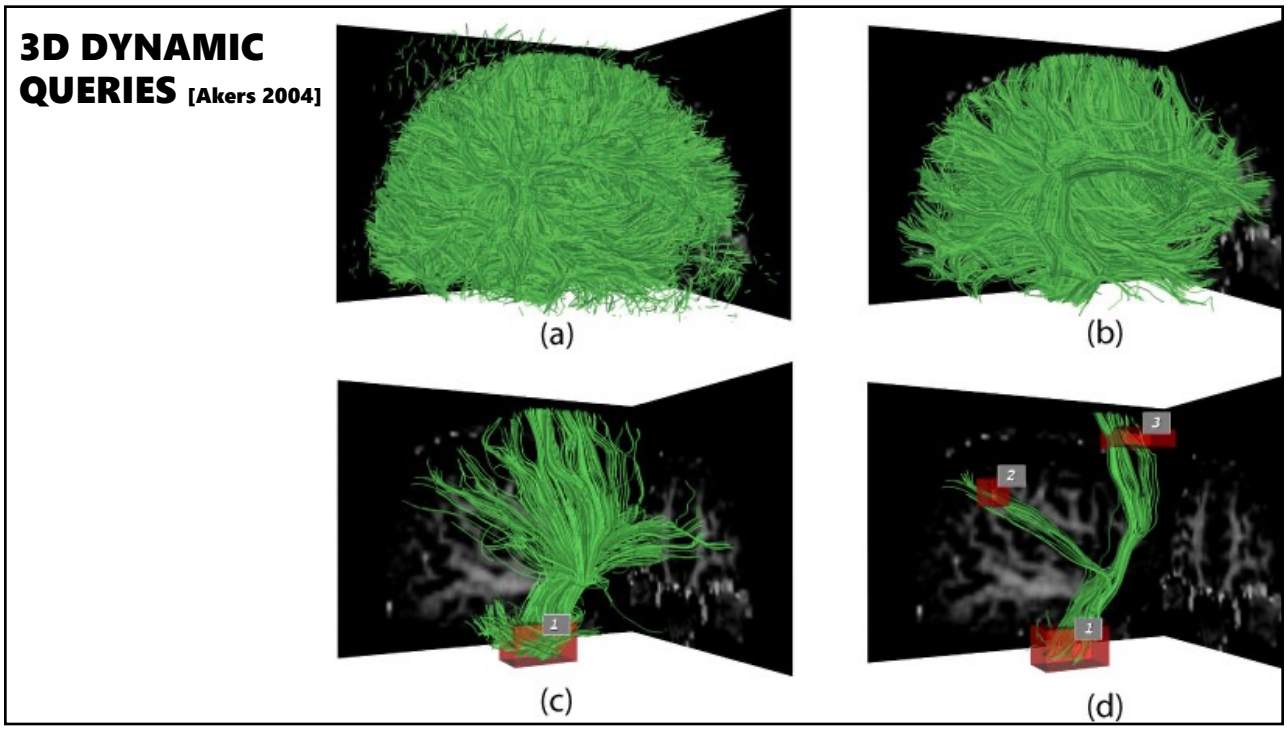


Based on Wattenberg's [2001] idea for sketch-based queries of time-series data

20

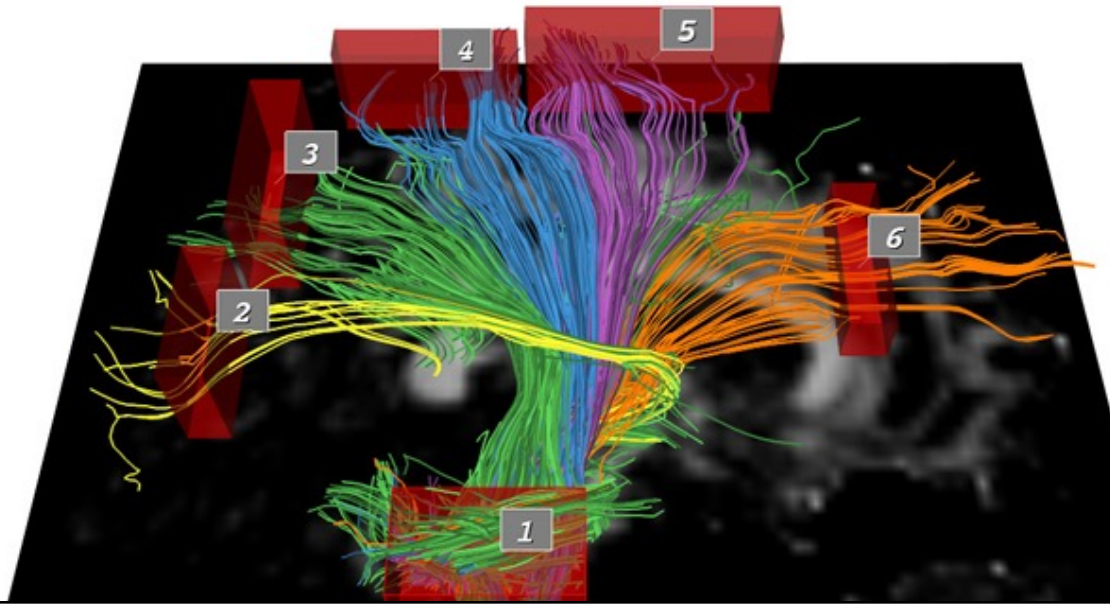

TimeSearcher: 13 Months of High and Low prices

21


3D DYNAMIC QUERIES [Akers 2004]
(a)
(b)
(c)
(d)
1
2
3

22

3D DYNAMIC QUERIES [Akers 2004]



23

DYNAMIC QUERIES PROS & CONS

Pros

- Controls useful for both novices and experts
- Quick way to explore data

Cons

- Simple queries
- Lots of controls
- Amount of data shown limited by screen space

24

SUMMARY

Good visualizations are task dependent

Pick the interaction technique to support the task

Fundamental interaction techniques

Selection
Brushing & Linking
Dynamic Queries

25

ANNOUNCEMENTS

26

ASSIGNMENT 3: INTERACTION

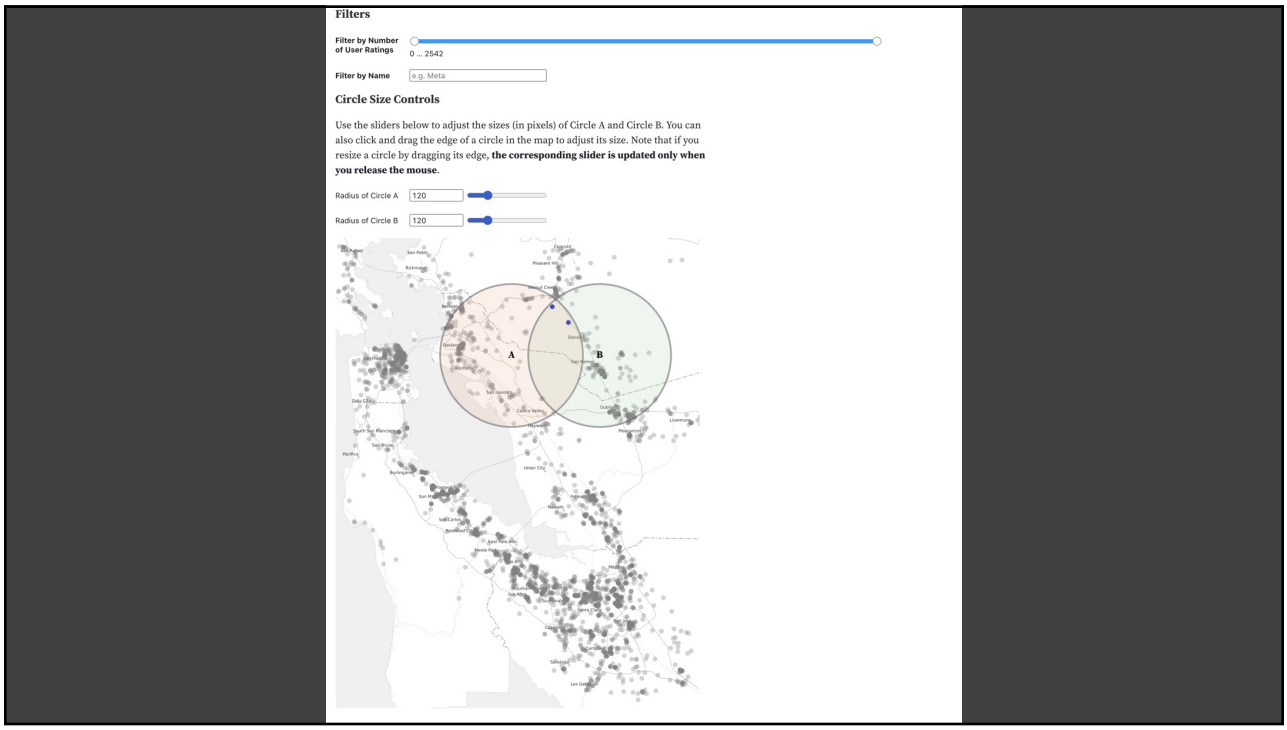
Due 10/30 11:30am

Create a small interactive dynamic query application similar to HomeFinder, but for local software companies data.

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

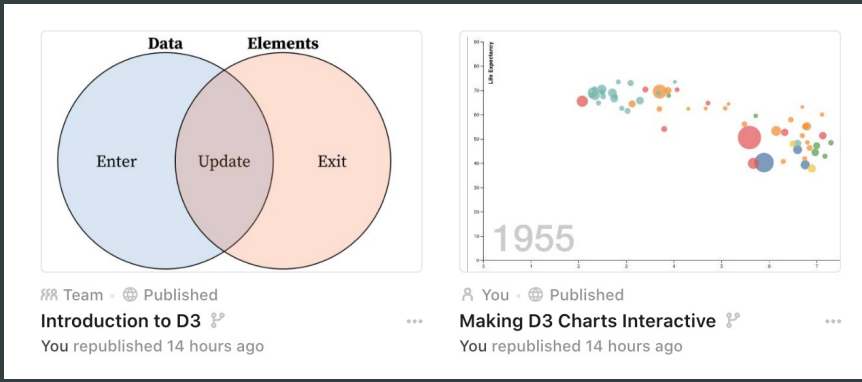
Can work alone or in pairs

27



28

D3 NOTEBOOKS MON



29

TODAY

Learning Objectives

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

30

INTRODUCTION TO D3

31

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*"

32

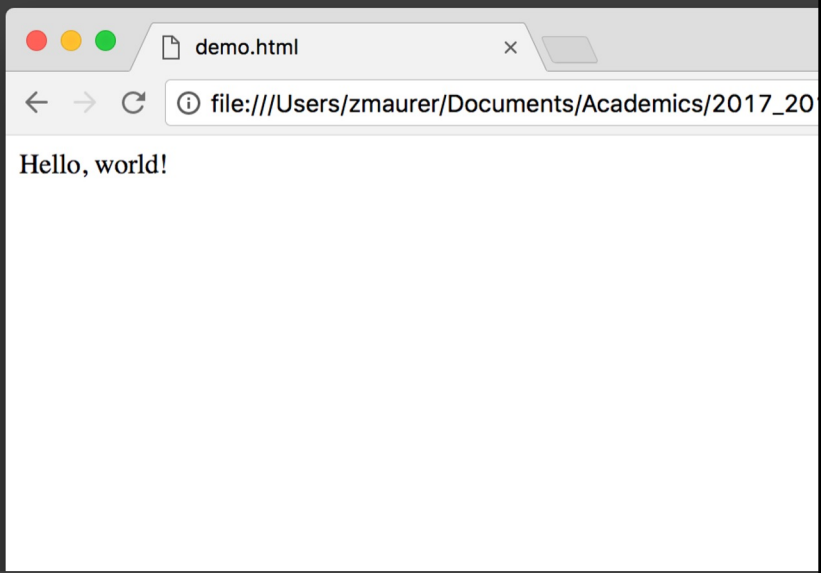
hello-world.html

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

<body>
  Hello, world!
</body>

</html>
```

33

hello-world.html

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

<body>
  Hello, world!
</body>

</html>
```



34

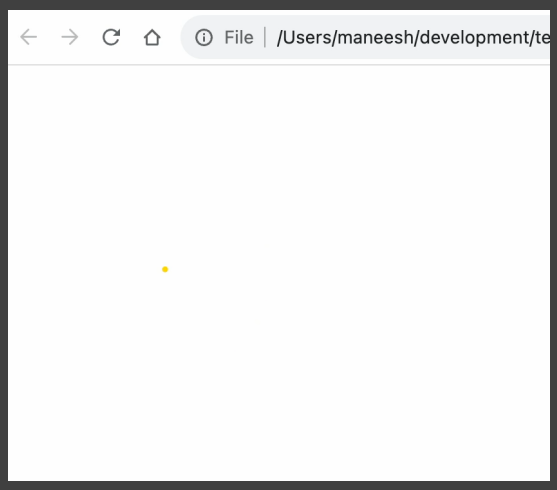
hello-svg.html

```
<!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>
```

37

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>
```



38

DOCUMENT OBJECT MODEL (DOM)

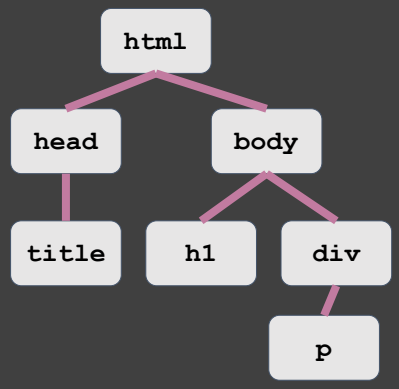
```
<html>
        <head>
                <title></title>
        </head>
        <body>
                <h1></h1>
                <div>
                        <p></p>
                </div>
        </body>
</html>
```

Adapted from Victoria Kirst's cs193x slides

39

DOCUMENT OBJECT MODEL (DOM)

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

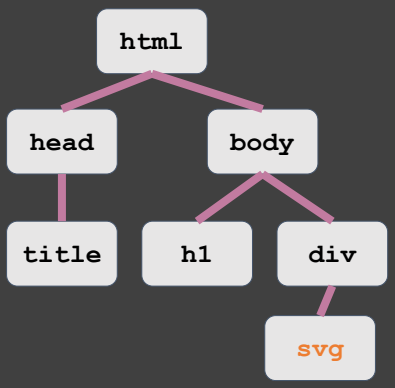


Adapted from Victoria Kirst's cs193x slides

40

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

41

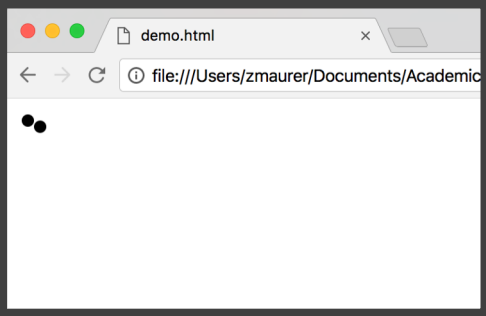
hello-d3.html

```
<!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>
```

44

D3 SELECTION

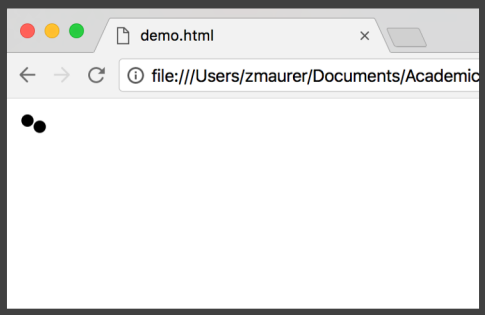
```
<html>
…
<svg width="960" height="500">
  <circle cx="10" cy="10" r="5"></circle>
  <circle cx="20" cy="15" r="5"></circle>
</svg>
…
```



45

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>
```

46

D3 SELECTION AND MANIPULATION

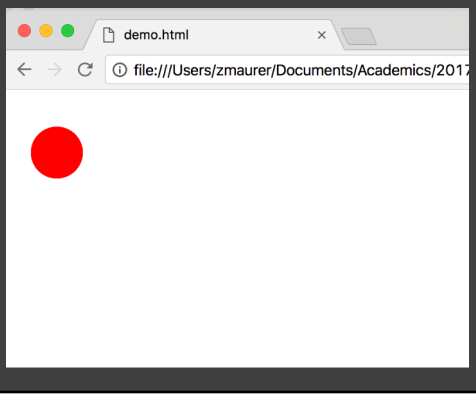
```
<html>
...
<svg width="960" height="500">
  <circle cx="10" cy="10" r="5"></circle>
  <circle cx="20" cy="15" r="5"></circle>
</svg>
...
```

```
<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>
```

47

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>
```

48

D3 SELECTION AND MANIPULATION

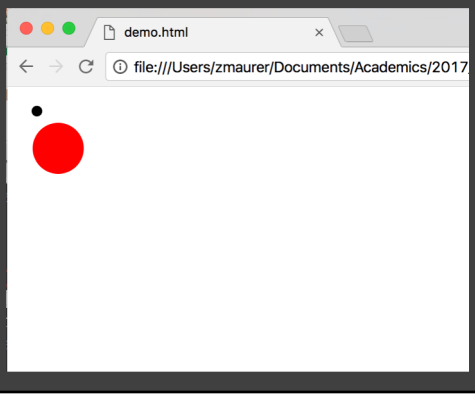
```
<html>
…
<svg width="960" height="500">
  <circle cx="10" cy="10" r="5"></circle>
  <circle cx="20" cy="15" r="5"></circle>
</svg>
…
```

```
<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>
```

49

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>
…
```

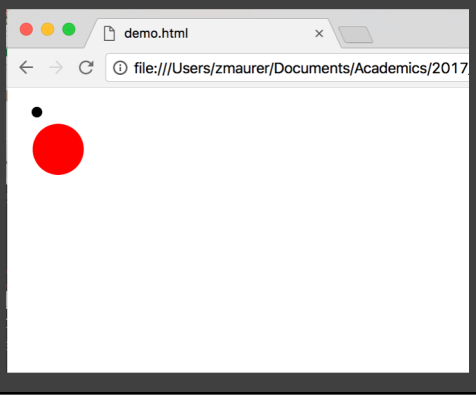



```
<script>

// 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>
```

50

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>
```

51

D3 BINDING DATA & JOIING DOM ELEMENTS

```
gapminder = ▸ Array(693) [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)
```

52

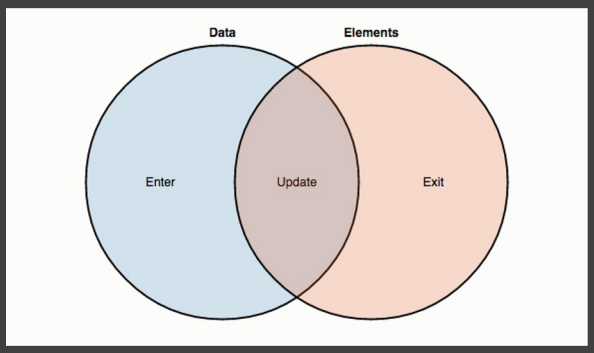
D3 BINDING DATA & JOIING DOM ELEMENTS

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();
}
```

53

D3 BINDING DATA & JOIING DOM ELEMENTS



A *join* creates three sub-selections:

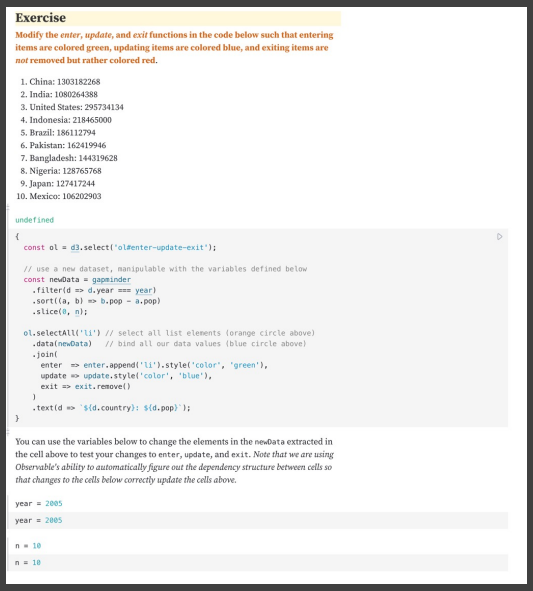
***Enter*:** selection containing placeholders for every data value that did not have a corresponding DOM element in the original selection

***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

54

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
```

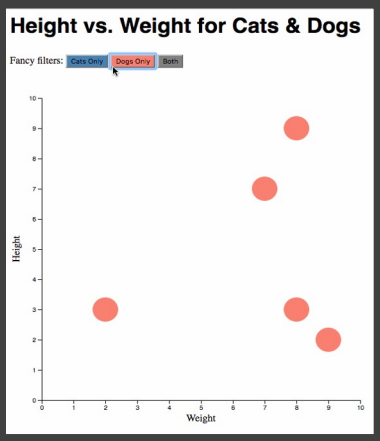
```
n = 10
```

```
n = 10
```

56

LET'S MAKE A SCATTERPLOT 🐱🐶

```
id,animal,weight,height,name
1,cat,10,3,phyllis
2,cat,3,3,oreo
3,cat,9,9,sam
4,cat,3,5,dog
5,cat,6,5,fred
6,cat,5,6,jane
7,cat,1,8,esmerelda
8,dog,9,2,garfield
9,dog,8,9,alpha
10,dog,7,7,omega
11,dog,2,3,zeta
12,dog,8,3,cupcake
```



https://observablehq.com/@stanfordvis/lets-make-a-scatterplot

57