INTRODUCTION TO D3

CS 448B | Fall 2023

MANEESH AGRAWALA
Global temperature change (1850-2021)

Global temperature change
Relative to average of 1971-2000 [°C]
LAST TIME:
INTERACTION

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
DYNAMIC QUERIES

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
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

ZIP DECODE
[Fry 2004]

The yellow dots above are homes in the DC area for sale. You may get more information on a home by selecting it. You may drop the ‘A’ and ‘B’ distance markers to your office or any other location you want to live near. Select distances, bedrooms, and cost ranges by dragging the corresponding slider boxes on the right. Select specific home types and services by pressing the labeled buttons on the right.

https://benfry.com/zipcode/
NAMEVOYAGER
[Wattenberg 2005]

17

DIMPVIS
[Kondo 2014]

18
PARALLEL COORDINATES

https://observablehq.com/@sophiegri/exercise-3-parallel-coordinates

TimeSearcher
Based on Wattenberg’s [2001] idea for sketch-based queries of time-series data
3D DYNAMIC QUERIES [Akers 2004]
**3D DYNAMIC QUERIES** [Akers 2004]

---

**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
SUMMARY

Good visualizations are task dependent
Pick the interaction technique to support the task

Fundamental interaction techniques
Selection
Brushing & Linking
Dynamic Queries

ANNOUNCEMENTS
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
Learning Objectives

1. Getting started with D3 and web technologies it is based on

2. D3 binding data and joining it with DOM elements
INTRODUCTION TO D3

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”
hello-world.html

<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
</head>
<body>
Hello, world!
</body>
</html>
hello-svg.html

```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
DOCUMENT OBJECT MODEL (DOM)

Adapted from Victoria Kirst’s cs193x slides
**DOCUMENT OBJECT MODEL (DOM)**

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

Adapted from Victoria Kirst's cs193x slides.

---

**hello-d3.html**

```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>
```
D3 SELECTION

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

D3 SELECTION

```html
<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>
D3 SELECTION AND MANIPULATION

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

```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>

---

D3 SELECTION AND MANIPULATION

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

```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>
D3 SELECTION AND MANIPULATION

```html
<html>
...</html>
```

```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>

D3 SELECTION AND MANIPULATION

```html
<html>
...</html>
```

```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>
D3 SELECTION AND MANIPULATION

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

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

D3 BINDING DATA & JOINING DOM ELEMENTS

```javascript
// extract the 10 most populous countries in 2005
listData = ggpfinder
  .filter(d => d.year === 2005)
  .sort((a, b) => b.pop - a.pop)
  .slice(0, 10)
```
D3 BINDING DATA & JOINING DOM ELEMENTS

1. China: 1303182268
2. India: 1080264388
3. United States: 295734134
4. Indonesia: 218845999
5. Brazil: 186112794
6. Pakistan: 162419946
7. Bangladesh: 144319628
8. Nigeria: 128765768
9. Japan: 127417244
10. Mexico: 106202903

```javascript
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`) // text
return ol.node();
```

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
D3 BINDING DATA & JOINING DOM ELEMENTS

Exercise
Modify the code above, and test 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.

```javascript
let data = [{
  id: 1,
  animal: "cat",
  weight: 10,
  height: 3,
  name: "Phyllis"
}, {
  id: 2,
  animal: "cat",
  weight: 3,
  height: 3,
  name: "Oreo"
}, {
  id: 3,
  animal: "cat",
  weight: 9,
  height: 9,
  name: "Sam"
}, ...
]
```

LET'S MAKE A SCATTERPLOT

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