<table>
<thead>
<tr>
<th>Lecture</th>
<th>Topic</th>
<th>Projects</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Intro, schedule, and logistics</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Basic tasks and data types</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Data sources and what to look out for</td>
<td>Project 1 out</td>
</tr>
<tr>
<td>4</td>
<td>Data preparation and representation</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Data reduction, notion of similarity and distance</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Dimension reduction</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Bias, introduction to plotly and plotly Dash</td>
<td>Project 2 out</td>
</tr>
<tr>
<td>8</td>
<td>Visual perception and cognition</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Visual design and aesthetic</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td><strong>Introduction to D3</strong></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Cluster analysis</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>High-dimensional data, dimensionality reduction</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Visualization of spatial data: volume visualization intro</td>
<td>Project 3 out</td>
</tr>
<tr>
<td>14</td>
<td>Introduction to GPU programming</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Visualization of spatial data: raycasting, transfer functions</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Illumination and isosurface rendering</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Midterm</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Scientific visualization</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Non-photorealistic and illustrative rendering</td>
<td>Project 4 out</td>
</tr>
<tr>
<td>20</td>
<td>Midterm discussion</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>Principles of interaction</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>Visual analytics and the visual sense making process</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>Visualization of graphs and hierarchies</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>Visualization of time-varying and streaming data</td>
<td>Project 5 out</td>
</tr>
<tr>
<td>25</td>
<td>Maps</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>Memorable visualizations, visual embellishments</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>Evaluation and user studies</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>Narrative visualization, storytelling, data journalism, XAI</td>
<td></td>
</tr>
</tbody>
</table>
The material presented in these slides is derived from this book:

Also available online
**What is D3.js?**

D3 = Data Driven Documents

JavaScript library for manipulating documents based on data
  - frequent tool to support *data journalism* ([New York Times](https://www.nytimes.com))

D3 helps you bring data to life using HTML, SVG, and CSS
  - great library to construct animated visualizations ([D3 website](http://d3js.org))

Runs in any modern web browser (Chrome, Firefox, IE)
  - no need to download any software
  - independent of OS (Linux, Windows, Mac)
Makes Use Of

HTML  Hypertext Markup Language
CSS   Cascading Style Sheets
JS    JavaScript
DOM   The Document Object Model
      • tree structured organization of HTML objects
SVG   Scalable Vector Graphics

Raster .jpeg .gif .png
Vector .svg
**WHAT YOU NEED**

A text editor
- textMate, eclipse/aptana, sublime text 2, or your browser
- need an editor with syntax highlighting. else it’s easy to get lost

The d3 library
- from [http://d3js.org](http://d3js.org)

Data files for your code

A web server (recommended)
- if your visualization is reading data from files or a database (XMLHttpRequest)
- many options: EasyPHP (windows), Mac OS X Server, MAMP, Python
- else need to specify the data in the code

A browser
- to run the code
Your folder structure should look like this:

```
project-folder/  
  d3/  
   d3.v3.js // D3 library  
   d3.v3.min.js (optional) // minified D3 library  
index.html
```
Your initial webpage (index.html) should look like this:

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>D3 Page Template</title>
  <script type="text/javascript" src="d3/d3.v3.js"></script>
</head>
<body>
  <script type="text/javascript">
    // Your beautiful D3 code will go here
  </script>
</body>
</html>
```
MAMP = My Apache, MySQL, PHP
- really only need Apache for now
- MS Windows = WampServer and XAMPP for Windows
- Mac = MAMP or XAMPP for Mac

Procedure
- install package (Linux has it already installed)
- find webserver folder (only files residing there will be served)
- put project files there
- open browser and point to http://localhost/ or http://localhost:8888/ or http://localhost:8888/project-folder/
Let’s Use Some Simple Data

var dataset = [ 5, 10, 15, 20, 25 ];
Consider the following js code ... all methods are chained:

d3.select("body").selectAll("p")
  .data(dataset)
  .enter()
  .append("p")
  .text("New paragraph!");

which gives this output
  - how did this happen?
Consider the following js code ... all methods are chained:

d3.select("body").selectAll("p") // selects all paragraphs in the DOM (none so far ...)
  .data(dataset) // counts and parses the data values
  .enter() // creates new, data-bound elements (placeholders) for the data values
  .append("p") // takes the empty placeholder and adds a p-element
  .text("New paragraph!"); // takes the p-element and inserts a text value

which gives this output
  - how did this happen?
Change the last line to:

```javascript
d3.select("body").selectAll("p")
  .data(dataset)
  .enter()
  .append("p")
  .text(function(d) { return d; });
```

which gives this output
- how did this happen?
Change the last line to:

```javascript
d3.select("body").selectAll("p")
  .data(dataset)
  .enter()
  .append("p")
  .text(function(d) { return d; }); // used the data to populate the contents of each paragraph of the data-driven document
```

which gives this output

- how did this happen?
Change the last line to:

d3.select("body").selectAll("p")
  .data(dataset)
  .enter()
  .append("p")
  .text(function(d) { return "I can count up to " + d; });

which gives this output
  - how did this happen?
Change the last line to:

```javascript
d3.select("body").selectAll("p")
  .data(dataset)
  .enter()
  .append("p")
  .text(function(d) { return "I can count up to " + d; })
  .style("color", "red");
```

which gives this output

- how did this happen?
Replace the last line with:

d3.select("body").selectAll("p")
  .data(dataset)
  .enter()
  .append("p")
  .text(function(d) { return "I can count up to " + d; })
  .style("color", function(d) { if (d > 15) {return "red";} else { return "black";} });

which gives this output
  - how did this happen?
Let’s draw some bar charts

For this, put this embedded style in the document head

div.bar {
    display: inline-block;
    width: 20px;
    height: 75px; /* We'll override height later */
    background-color: teal;
}

Run this code:

```javascript
var dataset = [ 5, 10, 15, 20, 25 ];

d3.select("body").selectAll("div")
 .data(dataset)
 .enter()
 .append("div")
 .attr("class", "bar");
```

which gives this output
- five bars with no space between them
- how did this happen?
Run this code:

```javascript
var dataset = [ 5, 10, 15, 20, 25 ];

d3.select("body").selectAll("div")
  .data(dataset)
  .enter()
  .append("div")
  .attr("class", "bar")
  .style("height", function(d) { return d + "px"; });

which gives this output
  - how did this happen?
Run this code:

```javascript
var dataset = [ 5, 10, 15, 20, 25 ];

d3.select("body").selectAll("div")
 .data(dataset)
 .enter()
 .append("div")
 .attr("class", "bar")
 .style("height", function(d) { return d + "px"; }); // adds text "px" to specify that the units are pixels → heights are 5px, 10px, 15px, 20px, and 25px
```

which gives this output

- how did this happen?
Run this code: 
(also add margin-right: 2px; to the css style)

var dataset = [ 5, 10, 15, 20, 25 ];

d3.select("body").selectAll("div")
  .data(dataset)
  .enter()
  .append("div")
  .attr("class", "bar")
  .style("height", function(d) { var barHeight = d * 5; return barHeight + "px"; });

which gives this output
  ▪ how did this happen?
Optionally define some variable beforehand, e.g.:

```javascript
// width and height
var w = 500;
var h = 50;
```

Define the svg object:

```javascript
var svg = d3.select("body")
  .append("svg")
  .attr("width", w)
  .attr("height", h);
```
Define the circles as variables for ease of reference:

```javascript
var circles = svg.selectAll("circle")
  .data(dataset)
  .enter()
  .append("circle");
```

But could so this just as well:

```javascript
svg.selectAll("circle")
  .data(dataset)
  .enter()
  .append("circle"); // now circles are appended to the end of the SVG element
```
Run this code (still using var dataset = [ 5, 10, 15, 20, 25 ];)

circles.attr("cx", function(d, i) {return (i * 50) + 25;})
    .attr("cy", h/2)
    .attr("r", function(d) {return d;});

or append it to the .append("circle") method

This gives this output
  ▪  how did this happen?
Now Draw The Circles

Run this code (still using var dataset = [ 5, 10, 15, 20, 25 ]);)

circles.attr("cx", function(d, i) {return (i * 50) + 25;}) // i increments by 1 each time, starting at 0
  .attr("cy", h/2)
  .attr("r", function(d) {return d;});

or append it to the .append("circle") method

This gives this output
  - how did this happen?
Run this code (still using var dataset = [ 5, 10, 15, 20, 25 ];)

circles.attr("cx", function(d, i) {return (i * 50) + 25;})
  .attr("cy", h/2)
  .attr("r", function(d) {return d;})
  .attr("fill", "yellow")
  .attr("stroke", "orange")
  .attr("stroke-width", function(d) {return d/2;});

This gives this output
  ▪  how did this happen?
Bar Charts

Code
This will update the bar chart on a mouse click:

d3.select("p")
  .on("click", function() {

    // New values for dataset
    dataset = [ 11, 12, 15, 20, 18, 17, 16, 18, 23, 25, 5, 10, 13, 19, 21, 25, 22, 18, 15, 13 ];

    // Update all rects
    svg.selectAll("rect")
      .data(dataset)
      .attr("y", function(d) {
        return h - yScale(d);  
      })
      .attr("height", function(d) {
        return yScale(d);  
      });

  });
**Smooth animations** are desirable:

```javascript
svg.selectAll("rect")
  .data(dataset)
  .transition()
  .attr("y", function(d) {
    return h - yScale(d);
  })
  .attr("height", function(d) {
    return yScale(d);
  })
  .attr("fill", function(d) {
    return "rgb(0, 0, " + (d * 10) + ")";
  });
```
Now run **this code:**

```javascript
svg.selectAll("rect")
  .data(dataset)
  .transition()
  .duration(1000)  // <-- Now this is new!
  .attr("y", function(d) {
    return h - yScale(d);
  })
  .attr("height", function(d) {
    return yScale(d);
  })
  .attr("fill", function(d) {
    return "rgb(0, 0, " + (d * 10) + ")";
  });
```

Facilitated by event handlers (listeners), e.g.:

```javascript
d3.select("p")
  .on("click", function() {
    //Do something on click
  });
```

others react on

- mouse hovering
- mouse over
- mouse out
- and others

**Example**
Directing Action to a Specific Item

Assume you selected a certain item by mouseover

```javascript
.on("mouseover", function() {
    //Do something on mouseover of any bar
});
```

Keyword “this” maps the action to the selected item

```javascript
.on("mouseover", function() {
    d3.select(this)
        .attr("fill", "orange");
});
```
D3 layouts take data that you provide
- remap or otherwise transform it
- and so generating new data that is more convenient for a specific visual task

The supported layouts are:
- Bundle and Chord
- Cluster
- Force
- Histogram
- Pack, Partition, and Pie
- Stack
- Tree and Treemap
```javascript
var dataset = {
  nodes: [
    { name: "Adam" },
    { name: "Bob" },
    { name: "Carrie" },
    { name: "Donovan" },
    { name: "Edward" },
    { name: "Felicity" },
    { name: "George" },
    { name: "Hannah" },
    { name: "Iris" },
    { name: "Jerry" }
  ],
  edges: [
    { source: 0, target: 1 },
    { source: 0, target: 2 },
    { source: 0, target: 3 },
    { source: 0, target: 4 },
    { source: 1, target: 5 },
    { source: 2, target: 5 },
    { source: 2, target: 5 },
    { source: 3, target: 4 },
    { source: 5, target: 8 },
    { source: 5, target: 9 },
    { source: 6, target: 7 },
    { source: 7, target: 8 },
    { source: 8, target: 9 }
  ]
};
```
Next, we create an SVG line for each edge:

```javascript
var edges = svg.selectAll("line")
  .data(dataset.edges)
  .enter()
  .append("line")
  .style("stroke", 
    "ccc")
  .style("stroke-width", 1);
```

Note that I set all the lines to have the same stroke color and weight, but of course you could set this dynamically based on data (say, thicker or darker lines for “stronger” connections, or some other value).

Then, we create an SVG circle for each node:

```javascript
var nodes = svg.selectAll("circle")
  .data(dataset.nodes)
  .enter()
  .append("circle")
  .attr("r", 10)
  .style("fill", function(d, i) {
    return colors(i);
  })
  .call(force.drag);
```