What is D3.js

• D3.js (also known as D3, short for Data-Driven Documents) is a JavaScript library for producing dynamic, interactive data visualizations in web browsers;
• D3 is a lightweight library, it works with web standards, you don’t need any other technology or plugin other than a browser to make use of D3;
• D3 gives you complete control over your visualization to customize it the way you want;
• Book for reference
• Tutorials and Examples
HTML basics - DOM

• A Web page is a document. This document can be either displayed in the browser window or as the HTML source.
• The Document Object Model (DOM) represents that same document.
• The DOM is the data representation of the objects that comprise the structure and content of the document.
• The DOM is a programming interface for HTML documents.
• The DOM objects can be modified with a scripting language such as JavaScript
HTML basics - DOM

Listing 1.1  A simple web page demonstrating the DOM

```html
<!doctype html>
<html>
<head>
    <script src="d3.v4.min.js"></script>
</head>
<body>
    <div id="someDiv" style="width:200px;height:100px;border:black 1px solid;">  
        <input id="someCheckbox" type="checkbox" />
    </div>
</body>
</html>
```
HTML basics - DOM

• Three categories of information about each element determine its behavior and appearance:

  • Styles can determine transparency, color, size, borders, and so on.
  
  • Attributes include classes, IDs etc.
  
  • Properties typically refer to states, such as the “checked” property of a check box, which is true if the box is checked and false if the box is unchecked.
Use D3 modify DOM element

- d3.select("#someDiv").style("border", "5px darkgray dashed");
- d3.select("#someDiv").attr("id", "newID");
- d3.select("#someCheckbox").property("checked", true);
SVG

• SVG stands for Scalable Vector Graphics, it defines vector-based graphics.

• Unlike bitmap images, vector-based images provide a set of instructions about how the image should be rendered.

• <path d="M 10,60 40,30 50,50 60,30 70,80"/>
SVG

- An SVG image begins with an <svg> element
- The width and height attributes of the <svg> element define the width and height of the SVG canvas
- You can use CSS to style your SVG canvas or use D3 to add inline styles
The `<path>` element is the most basic element in the SVG library.

The shape of a `<path>` element is defined by one parameter: `d`, which contains a series of commands and parameters.

Other SVG elements include `<CIRCLE>`, `<RECT>`, `<LINE>`, `<POLYGON>`, each of them has attributes that determine their size and position to make them easier to deal with than the generic `d` attribute.

D3 provides an abstraction layer for drawing SVG.
<!doctype html>
<html>
    <script src="d3.v4.min.js"></script>
</html>
<body>
    <div id="infovizDiv">
        <svg style="width:500px;height:500px;border:1px lightgray solid;">
            <path d="M 10,60 40,30 50,50 60,30 70,80"
                style="fill:black;stroke:gray;stroke-width:4px;" />
            <polygon style="fill:gray;"
                points="80,400 120,400 160,440 120,480 60,460" />
            <g>
                <line x1="200" y1="100" x2="450" y2="225"
                style="stroke:black;stroke-width:2px;"/>
                <circle cy="100" cx="200" r="30"/>
                <rect x="410" y="200" width="100" height="50"
                    style="fill:pink;stroke:black;stroke-width:1px;" />
            </g>
        </svg>
    </div>
</body>
</html>
SVG

- The `<g>` SVG element is a container used to group other SVG elements.
- Adjust the `transform` attribute of the `<g>` element to move a `<g>` around the canvas.
- `transform` attribute accepts a structured description in text of how you want to transform a shape.
  - `transform = "translate (100,50)"
  - `transform = "scale(2.5)"`
CSS

• CSS (Cascading Style Sheets) is used to style and lay out web pages — for example, to alter the font, color, size, and spacing of your content.

• CSS is a rule-based language — you define rules specifying groups of styles that should be applied to particular elements or groups of elements on your web page.

• tutorial
CSS

- **Selector:** target the HTML element that we are going to style. It can target element by type, ID, class etc.
- **Declarations:** Inside a set of curly braces `{ }` including one or more declarations, which take the form of property and value pairs.

```css
h1 {
  color: red;
  font-size: 5em;
}
```

- Rules occurs later overwrite the rules occurs earlier
- more specific rules overwrite more general rules.
CSS

• To link css file to html document, add the following line inside the <head> of the HTML document:

```html
<link rel="stylesheet" href="filename.css">
```

• The rel stands for "relationship"
• The href specify the location of the css file
CSS

• Use the helper function d3.classed() to add or remove a class from the classes in an element.
  
  d3.select("circle").classed("active", true);

• By using .classed(), you don’t overwrite the existing attribute, but rather append or remove the named class from the list.
JavaScript

• JavaScript is a programming language initially designed to interact with elements of web pages.

• Tutorial
JavaScript

• Use var to declare a variable

```javascript
var a
var length = 16; // Number
var lastName = "Johnson"; // String
var x = {firstName: "John", lastName: "Doe"}; // Object
```

• JavaScript types are Dynamic

```javascript
var x; // Now x is undefined
x = 5; // Now x is a Number
x = "John"; // Now x is a String
```
Object: In JavaScript, an object is a collection of properties, where each property is defined as a key-value pair.

```javascript
var emptyObject = {};
var person = {
    firstName: "John",
    lastName: "Doe"
};
```

• The key of the object can by any string.
• Access the properties of an object by using two notations: the dot notation (.) and array-like notation ([]).

```javascript
person.firstName
person["firstName"]
```
JavaScript

• Unlike objects in other programming languages such as Java and C++, you can add a property to an object after object creation.

• `person.age = 25; // add the new property age to the person`
JavaScript

• Array: an array is an ordered list of values
• An array can hold values of different types.
• The length of an array is dynamically sized and auto-growing. In other words, you don’t need to specify the array size upfront.
• Create array:
  • Use the array literal notation:
    var arrayName = [element1, element2, …]
  • With array constructor:
    var scores = new Array(9,10,8,7,6);
JavaScript

• Get element in array with index: `arrayName[index]`
• Use the `push()` method to add an element to the end of an array:
  `arrayName.push(newElement)`
• Use the `unshift()` method to add an element to the beginning of an array:
  `arrayName.unshift(newElement)`
• Use the `pop()` method to remove an element from the end of an array:
  `arrayName.pop()`
• Use the `shift()` method to remove an element from the beginning of an array:
  `arrayName.shift()`
JavaScript

• Define a function:
  
  ```javascript
  function functionname(parameter-list) {
    statements
  }
  ```

• Hoisting: JavaScript's default behavior of moving declarations to the top of the current scope.

  ```javascript
  var x = myFunction(4, 3); // Function is called, return value will end up in x

  function myFunction(a, b) {
    return a * b; // Function returns the product of a and b
  }
  ```
Anonymous function

• An anonymous function is a function without a name.
• A function expression can be stored in a variable

• `var x = function (a, b) {return a * b};`
• `var z = x(4, 3);`
Arrow function

• An **arrow function** expression is a compact alternative to a traditional function expression

```javascript
function myFunction(a, b) {
  return a * b;  // Function returns the product of a and b
}
```

• `(a, b) => a * b;`  // the resulting value will be implicitly returned when the function is called
Arrow function

// Traditional Function
function (a, b){
  let chuck = 42;
  return a + b + chuck;
}

// Arrow Function
(a, b) => {
  let chuck = 42;
  return a + b + chuck;
}
Method Chaining, also known as Cascading, refers to repeatedly calling one method after another on an object, in one continuous line of code.

Method chaining is used a lot in D3 examples, which means you’ll see something like this written on one line or formatted (but functionally identical) to something written on multiple lines:

```javascript
  d3.selectAll("div").html("Even More Wow").style("font-weight", "900");
```

Equal to:
```
  var s = d3.selectAll("div")
  s.html ("Even More Wow");
  s.style("font-weight", "900");
```
Hello World

- Example: add a cycle on html page and customize its attribute.
- d3_helloworld.html

```javascript
let svg = d3.select("svg")
  .style("width", 50)
  .style("height", 50)
  .append("circle")
  .attr("cx", 25)
  .attr("cy", 25)
  .attr("r", 25)
  .style("fill", "purple");
```
Selection

• A selection is a group of one or more web page elements that may be associated with a set of data
• `select()` group the elements by matching the given CSS selector.
• The real power of D3 comes from using selections to combine data and web page elements
• Perform actions on the elements in the group, such as moving them or changing their color.
• `d3.select()`: selects the first single element found
• `d3.selectAll()`: select all matched elements
Loading Data

- `d3.csv(url [, callback])`
- The first parameter is the url of .csv file
- The second optional parameter is a callback function which will be executed once the .csv file is loaded.
- `d3.csv("cities.csv", d => console.log(d));`
- The data file is loaded as an array of JSON objects
- Other functions to load data include: `d3.json()`, `d3.text()`, `d3.xml()`, `d3.html()`
- The loading data functions are **asynchronous** function, and will return after the request to open the file and not after processing the file.
Loading Data

Because of cross origin resource sharing restrict, you need to set up a local testing server to load the local files.

https://developer.mozilla.org/en-US/docs/Learn/Common_questions/set_up_a_local_testing_server

In your project directory, run:

```
python3 -m http.server
```

Example:

```
d3_dataloading.html
```
Data Binding

- d3_databinding.html

```javascript
// Load data from CSV file
d3.csv("cities.csv", data => {
  dataViz(data)
});

// Function to visualize data
function dataViz(incomingData) {
  console.log(incomingData)

  d3.select("body").selectAll("div")
    .data(incomingData)
    .enter()
    .append("div")
    .attr("class", "cities")
    .html(d=>d.label);
}
```
Data Binding

• The first part is `d3.select()` or `d3.selectAll()` with a CSS identifier that corresponds to a part of the DOM.
• Often no elements match the identifier, which is referred to as an empty selection, because you want to create new elements on the page based on the data using the `.enter()` function.

• `.data()`
• Bind the data with the DOM elements you selected.
• Each entry of the data is associated with a DOM element in the selection, and that associated data is stored in a data attribute of the element.
Data Binding

• When binding data to selections, there will be either more, less, or the same number of DOM elements as there are data values.
• .enter()
• When you have more data values than DOM elements in the selection, the .enter() function will allow you to define behavior to perform for every value that doesn’t have a corresponding DOM element in the selection.
• .exit()
• When fewer data elements exist, then .exit() is triggered to define the behavior to perform for the DOM element that doesn’t have a corresponding data
Data Binding

• `d3.selectAll("p").data([1,2,3,4]).exit().remove();`

• This code deleted all but four of our `<p>` elements, because we have only four values in our array.

• `D3_exit.html`

• When data values and DOM elements are equal in a selection, then neither `.exit()` nor `.enter()` is fired.
Data Binding

• `.APPEND()` and `.INSERT()`

• The `.append()` function used to add more elements and define which elements to add.

• The `.insert()` method used to create a new element and insert it into the selected element before its ending tag.
Data Binding

• .ATTR()
  • Used to change styles and attributes
  • The only thing to note is that each of the functions you define here will be applied to each new element added to the page.

• .HTML()
  • Set the content with a .html() function
Data Binding

• Any anonymous function called when setting the .style(), .attr(), .property(), .html(), or other function of a selection can provide you with the data bound to that selection

• The first variable (typically represented with the letter d, but you can declare it as whatever you want) contains the data value bound to that element

• The second variable returns the array position (known as an index, hence the variable name i) of the value bound to that element

• .style("background", function(d) {return d})
• .attr("cx", function(d,i) {return i})
• .html(function(d) {return d})
Data Binding

d3.csv("cities.csv", (error, data) => {
  if (error) {
    console.error(error)
  } else {
    dataViz(data)
  }
});

function dataViz(incomingData) {
  d3.select("body").selectAll("div.cities")
    .data(incomingData)
    .enter()
    .append("div")
    .attr("class", "cities")
    .html(d => d.label);
}

An empty selection because there are no <div> elements in <body> with class of “cities”

Binds the data to your selection

Defines how to respond when there’s more data than DOM elements in a selection

Creates an element in the current selection

Sets the class of each newly created element

Sets the content of the created <div>
More Example: Histogram

• If we want to draw a histogram based on an array of data, how can we do that?
• D3_hist.html
Data Scale

• Use D3’s scaling functions to normalize the values for display
  [14, 68, 24500, 430, 19, 1000, 5555]
• D3 scale has two primary functions: .domain() and .range(), both of which expect arrays and must have arrays of the same length to get the right results
• The array in .domain() indicates the series of values being mapped to .range()

```javascript
var yScale = d3.scaleLinear().domain([0,24500]).range([0,100]);
yScale(0); ← Returns 0
yScale(100); ← Returns 0.40816326530612246
yScale(24000); ← Returns 97.95918367346938
```

• yScale now allows us to map the values in a way suitable for display
Axes

• D3 provides d3.axisLeft(), d3.axisRight(), d3.axisBottom(), and d3.axisTop() method
• Provide axis with pre-defined scale to determine the axis range
• Use a <g> element to hold the axis, by invoking the selection’s call() method draw the axis
• Left, right, bottom, top is the orientation, axes are always rendered at the origin (0, 0)
• d3_axis.html
  ```javascript
  var xAxis = d3.axisBottom().scale(xScale)
  d3.select("svg").append("g").call(xAxis)
  ```
Style Axis

• Adjust the .attr("translate") of the axis’s parent <g> elements to adjust axis’s position

• d3.selectAll("#xAxisG").attr("transform","translate(0,500)")

• Uses the .tickSize() function to change the ticks to lines
• and manually sets the number of ticks using the ticks() function
Line chart example

• D3 line generator, d3.line(), this line generator expects an array of points as data, used to generate d attribute of the SVG <path> element
  
  ```javascript
  var line = d3.line()([[10, 60], [40, 90], [60, 10], [190, 10]])
  ```

  Use accessor to set the x and y
  
  ```javascript
  var tweetLine = d3.line()
  .x(d => xScale(d.day))
  .y(d => yScale(d.tweets))
  ```
Line chart example

- Steps to draw a line:
  - Write a generator function and define how it accesses the data it uses to draw the line
  - Append a <path> element to SVG canvas, set it’s d attribute to equal to the generator function we defined

D3_linechart.html
Line interpolation

- D3 provides a number of interpolation methods with which to draw these lines (as well as areas and diagonals and radial lines), exposed as the .curve method.
- Orange: curveCardinal interpolation provides a curve that touches each sample point.
- Green: curveStep interpolation changes the position of the line at right angles.
D3 generators

• commonly used generators include:
  • line, area, stack, arc, symbol
layout

- A layout encapsulates a strategy for laying out data elements visually, relative to each other.
- Layouts take a set of input data, apply an algorithm, and output the resulting positions/shapes for the display of the data.
- Currently, Bundle, Chord, Cluster, Force, Histogram, Pack, Partition, Pie, Stack, Tree, Treemap layouts are available.
Layout example - piechart

• pieChart layout used to transform the data value to a radius, it will generate a new array of objects

```javascript
var pieChart = d3.pie();
var yourPie = pieChart([1, 1, 2]);
```

• The generated radius data will be passed to arc generator, the arc generators with produce path data from angle and radius values.

```javascript
var newArc = d3.arc();
newArc.innerRadius(0)
  .outerRadius(100)
```
Pie chart

• The pie layout is centered on the 0,0 point by default.

• Need to create a new `<g>` element to hold the `<path>` elements to move it.
Cycle packing

• One of the most popular way to visualize hierarchical data is circle packing
• Each object is placed graphically inside the hierarchical parent of that object.
• D3 provide pack() layout to help implement cycle packing
• pack() expects a JavaScript object array where the child elements in a hierarchy are stored in a children attribute that points to an array.
Cycle packing

- `d3.pack()` Creates a new pack layout
- `pack(root)` Lays out the specified root hierarchy, assigning the following properties on `root` and its descendants:
  - `node.x` - the x-coordinate of the circle’s center
  - `node.y` - the y-coordinate of the circle’s center
  - `node.r` - the radius of the circle
Cycle packing

- d3. nest(), can be used to create hierarchical datasets out of flat data.
- d3.pack() want a hierarchical object and d3.nest returns an array, so we need to put the results of the array inside an object

- d3.hierarchy(data[, children])
  - Constructs a root node from the specified hierarchical data. The specified data must be an object representing the root node.
  - The specified children accessor function is invoked for each datum, starting with the root data, and must return an iterable of data representing the children
- d3_cyclepack.html
Event

• Use .on function to listen to mouse event and visualize data according to that
• The mouse events including "click", “dblclick”, "mouseover", "mouseout", etc.
• d3 sends the bound data to the function automatically and in the same format as the anonymous inline functions.
• We can use .on() to tie events to any object

• Example: d3_event.html
• Use cycle to represent the world cup dataset, one cycle represent a country
• Create buttons based on the attributes of the data
• resize the circles representing each team to reflect the teams with the highest and lowest values in each category
Transitions

• Transitions are defined for a selection and can be set to occur after a certain delay using `delay()` or to occur over a set period (milliseconds) using `duration()`.

  • `transition.delay([value])`
  • For each selected element, sets the transition delay to the specified value in milliseconds.

  • `transition.duration([value])`
  • For each selected element, sets the transition duration to the specified value in milliseconds.

• D3_transition.html