Introduction to data visualization for the web

IAT 355

Maha El Meseery

melmesee@sfu.ca

Today

- Review Web standard
  - HTML
  - CSS
  - DOM
  - JavaScript
- Web graphics
  - SVG
  - Canvas
- Introduction to D3
  - Importing files in D3
  - Drawing some shapes
- Exercise

HTML

Hypertext Markup Language is used to structure the content for web

- Add structure to content
- Uses tags to define structure
  - <h1> , <p> <img> ,

```
<!DOCTYPE html>
<html>
  <head>
    <title>TITLE GOES HERE</title>
  </head>
  <body>
    MAIN CONTENT GOES HERE
        <div>
          <p>Normal paragraph</p>
          <p>Red paragraph</p>
        </div>

        <ol>
          <li>Unique element</li>
          <li>Another list element</li>
          <li>
            <p>Paragraph inside list element</p>
            <p>Second paragraph</p>
          </li>
        </ol>

  </body>
</html>
```

Adopted from http://square.github.io/intro-to-d3/web-standards/

CSS

Cascades style sheets. Adds style to the visual representation. Style sheets contains multiple style rules.

Each rules have

- Selectors
  - tag, name, id, class
- properties
  - property name
  - property value

```
/* general syntax  */
Selector {
  property:value;
   property:value;
}
/* Example  " */
.red {
  background: red;
}
```

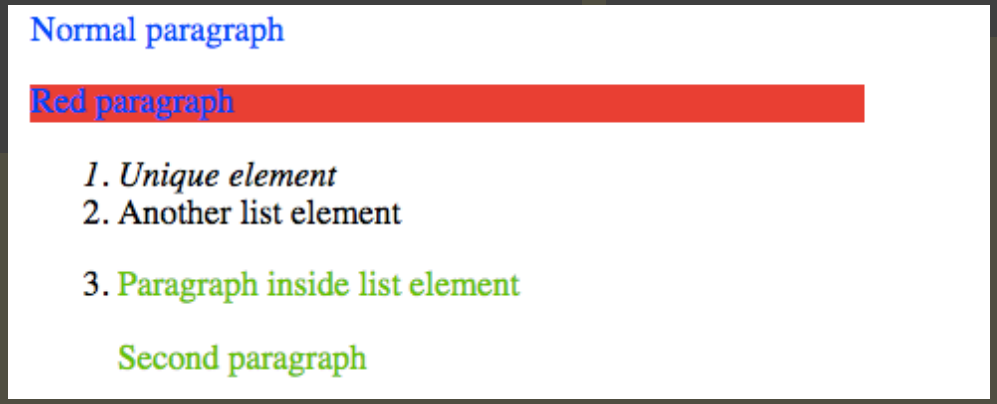
Adopted from http://square.github.io/intro-to-d3/web-standards/

```html
<!DOCTYPE html>
<html>
  <head>
    <title>TITLE GOES HERE</title>
  </head>
  <body>
    MAIN CONTENT GOES HERE
        <div>
          <p>Normal paragraph</p>
          <p>Red paragraph</p>
        </div>

        <ol>
          <li>Unique element</li>
          <li>Another list element</li>
          <li>
            <p>Paragraph inside list element</p>
            <p>Second paragraph</p>
          </li>
        </ol>

  </body>
</html>
```

```css
/* Applied to all <p> tags */
p {
  color: blue;
}
/* Applied to all tags
with the class "red" */
.red {
  background: red;
}
/* Applied to the tag
 with the id "some-id" */
#some-id {
  font-style: italic;
}
/* Applied only to <p> tags
 that are inside <li> tags */
li p {
  color: #0C0;
}
```

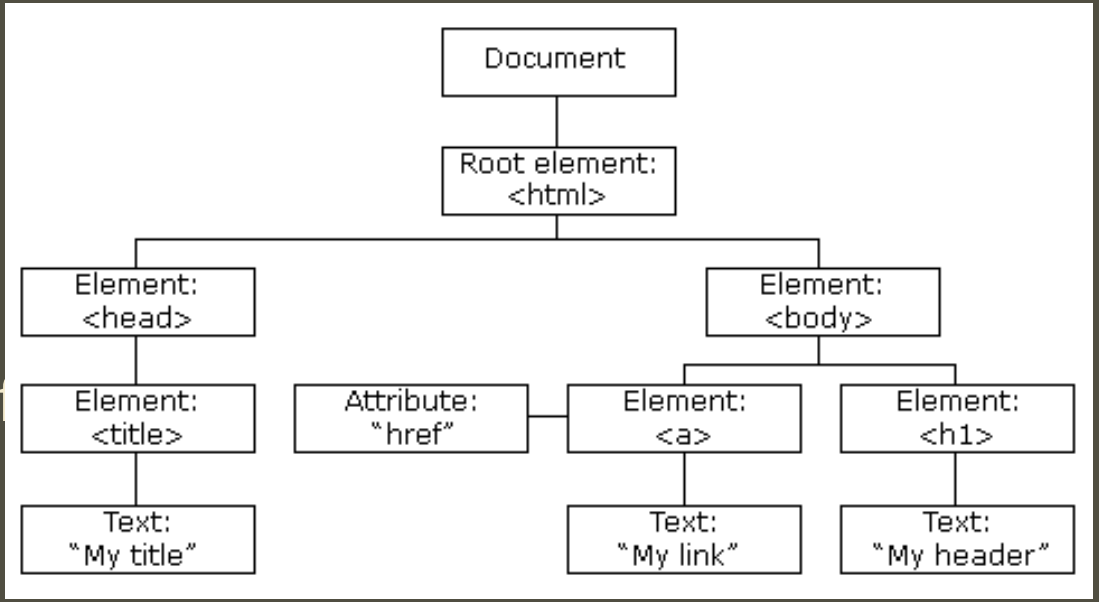


DOM

Document Object Model.

Hierarchical structure of HTML.

W3C gives standard interface for accessing and manipulate the structure of document.



HTML DOM model is constructed as a tree of **Objects** or **Elements**

Image source https://www.w3schools.com/js/js_htmldom.asp

Javascript

Javascript is a programming language,
available in most web browsers, which allows
the developers to programmatically access and manipulate
the DOM of an HTML page.

JavaScript Syntax

- Variables
- Arrays
- Objects
- Comments

```
// variables.
var   x=100;  // var is optional
      y=40;
      x="Rayan"; // Javascript is dynamic type
                 // x could be numeric value then
                 // value of x change to string.
// Arrays
var numbers=[1, 2, 3, 5];
var emptyarr=[];
var arr=[ 21, "another va", "oh boy", true
         , false, 123, 34, 'mixed' ] ;
number[0]  // returns 1
number[1]
number.length // give you length returns 4

// objects
var myObj = {name: “Ryan”, // name is the property - "Rayn" is the valu
             age: 21,
             city: “Vancouver”}

myObj.name // returns "Ryan"

// comments  -- single line
/*
 multi line comments...
*/
```

JavaScript Syntax

- Operation
- Comparison
- Control flow
  - If
  - for
  - while

```
// operations
 x=5+5    //  return 5
 s=5+"5"  //  return '55'
 y=3/4    // return 0.75

//logical operation
== // equal
!= // not equal
> , <  , <=, >= // comparison
&& , || // logical operators

//Functions
function myFunction(arg1, arg2) {  // function declaration .
console.log(arg1);
console.log(arg2);
}
// calling functions...
myFunction("a","b") //a, b
myFunction() //undefined, undefined
myFunction("a") //a, undefined

// If
if (x>y){
// operations...
}

// for (initialization;test ;update)
for (i=0; i< 10; i++){
// repeat this 10 times..
}
```

JavaScript Syntax

- Variable Scope

  https://jsbin.com/simatar/edit?js,console

```
 function f1(){
var myVar = 10; //myVar is local different than
                // the myVar in global..
    myVar = 100; //myVar is STILL local
   console.log("Inside f1 myVar local value "+myVar)
}

function f2(){
 gvar = 1000; //gvar is global
   console.log("Inside f2 gvar ="+gvar)

}

var myVar= 55;  // global value ...
console.log(" The global value myVar="+myVar);
//calling function ...
f1();
f2();
console.log("The global scope after f1 myVar="+myVar);
console.log("Inside the global scope gvar="+gvar);

// https://jsbin.com/simatar/edit?js,console
```

JavaScript Syntax- Objects

- Prototype Objects
  - Prototype of objects
  - **this**
  - **new** operator to create objects
- Classes
  - **Class**
  - **Constructor**

```
//Javascript
function Person(name, age){
this.name = name;
this.age = age;
}
var myPerson = new Person("Ryan", 21);
myPerson.name //Ryan

// ECMAscript6  introduced classes
class Rectangle {
  constructor(height, width) {
    this.height = height;
    this.width = width;
  }
}
const square = new Rectangle(10, 10);
```

JavaScript Syntax- Functions

- Functions can be stored in variables
- Functions can be passed as an argument
- Inline Functions - nameless functions
- Functions can return functions

```
// function variables
var foo = function(){
            console.log("foo"); }
 foo(); // call the function
//  passing function func
function bar(func, count){
    for(var i=0; i<count; i++){ // call func count time
        func();
    }
} // bar(foo, 5);
//  passing function foo to the bar
bar(foo, 5);
bar(function(){
    console.log("foo");
}, 5);
function expo(power){
    var func = function(value){
    var result = 1;
        for(var i=0; i<power; i++){
            result = result * value;
        }
        return result;
    }
    return func; //func is a function!
}
```

https://jsbin.com/sarabav/edit?js,console

JavaScript Syntax
DOM Manipulation

- Object and DOM API to manipulate the HTML document
- DOM API
  - Query
  - Add
  - Manipulate
  - Events
- Libraries like
  - JQuery
  - D3

```
<div>
  <p>Normal paragraph</p>
  <p class="red">Red paragraph</p>
</div>

<ol>
  <li id="some-id">Unique element</li>
  <li>Another list element</li>
  <li>
    <p>Paragraph inside list element</p>
    <p>Second paragraph</p>
  </li>
</ol>
<h1 id="click-me">
  Click on me!
</h1>
```

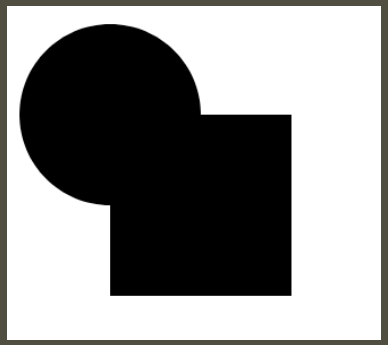
```
// DOM API
document.getElementById('some-id');
// <li id="some-id">Unique element</li>
document.getElementsByTagName('p').length;
// 4
var reds = document.getElementsByClassName('red');
// [<p class="red">Red paragraph</p>]
reds[0].innerText
// "Red paragraph"

/// events ...
var clickMe = document.getElementById('click-me');
clickMe.onclick = function() {
  if (this.style.backgroundColor) {
    this.style.backgroundColor = '';
  } else {
    this.style.backgroundColor = 'red';
  }
}
```

Scalable Vector Graphics (SVG)

- SVG is an XML-based markup to define vector graphics.
- Text based image format
- SVG can be embedded in an HTML document like any other HTML elements such as p, div, h1 etc.
- You have first create SVG element <svg></svg>

```
<body>
  <svg width="500" height="500">
    <circle cx="100" cy="100" r="80" />
    <rect x="100" y="100" width="160" height="160" />
  </svg>
<body>
```



https://jsbin.com/tozene/edit?html,output

Scalable Vector Graphics (SVG) - Elements

- Circle <circle>
- Rectangle <rect>
- Line <line>
- Text <text>
- polygon <polygon>
- path <path>
- Group of elements
  - <g>
- .... others

```
<svg height="400" width="450">
 <path id="lineAB" d="M 100 350 l 150 -300" stroke="red"
 stroke-width="3" fill="none" />
 <path id="lineBC" d="M 250 50 l 150 300" stroke="red"
 stroke-width="3" fill="none" />
 <path d="M 175 200 l 150 0"
  stroke="green" stroke-width="3"
 fill="none" />
 <path d="M 100 350 q 150 -300 300 0" stroke="blue"
 stroke-width="5" fill="none" />
 <!-- Mark relevant points -->
 <g stroke="black" stroke-width="3" fill="black">
   <circle id="pointA" cx="100" cy="350" r="3" />
   <circle id="pointB" cx="250" cy="50" r="3" />
   <circle id="pointC" cx="400" cy="350" r="3" />
 </g>
 <!-- Label the points -->
 <g font-size="30" font-family="sans-serif"
    fill="black" stroke="none"
     text-anchor="middle">
   <text x="100" y="350" dx="-30">A</text>
   <text x="250" y="50" dy="-10">B</text>
   <text x="400" y="350" dx="30">C</text>
 </g>
</svg>
```

https://jsbin.com/dewicig/edit?html,output

Complete Reference

https://developer.mozilla.org/en-US/docs/Web/SVG/Element

Adopted from https://www.w3schools.com/graphics/svg_path.asp

Scalable Vector Graphics (SVG) - Styling and more

- Fill
- Stroke
- Stroke-width
- Opacity
- Font
- Transform
  - Scale
  - Rotate
  - Translate
- Filters

```
<svg width="500" height="500">
<circle cx="100" cy="100" r="80"
              stroke="orange"  stroke-width="5"
               fill="red"/>
<rect x="100" y="100" width="160" height="160" />
</svg>
```

```
<svg width="500" height="500">
<rect x="20" y="20" width="160"
      height="160" class="red"/>
<rect x="100" y="100" width="160"
      height="160" id="mySquare"/>
</svg>
```

```
#mySquare{ fill: green }
.red { fill:red }
```

You can also use CSS to style

https://jsbin.com/gikunez/edit?html,css,output

Adopted from https://www.w3schools.com/graphics/svg_path.asp

CANVAS

<canvas> is an HTML element which can be used to draw graphics using scripting

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8"/>
 </head>
 <body onload="draw();">
   <canvas id="canvas"
           width="150" height="150">
    </canvas>
 </body>
</html>
```

```
function draw() {
    /// get the canvas element
  var canvas = document.getElementById('canvas');
  /// start by getting 2D context
  if (canvas.getContext) {
    var ctx = canvas.getContext('2d');
    // draw each object.
    ctx.fillStyle = 'rgb(200, 0, 0)';
    ctx.fillRect(10, 10, 50, 50);

    ctx.fillStyle = 'rgba(0, 0, 200, 0.5)';
    ctx.fillRect(30, 30, 50, 50);
  }
}
```

Adopted from https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial/Basic_usage

Data Driven Documents - D3

Mike Bostock

- What is D3.js
  - D3 is a javascript library for manipulating the DOM, based on data, and primarily used for making data-driven visualizations.
  - Manipulate web document with Data
    - Load data to browser
    - Bind data to elements
    - Transform elements based on data
    - Transition between states
- D3 main page https://d3js.org/

D3 is a general purpose visualization library

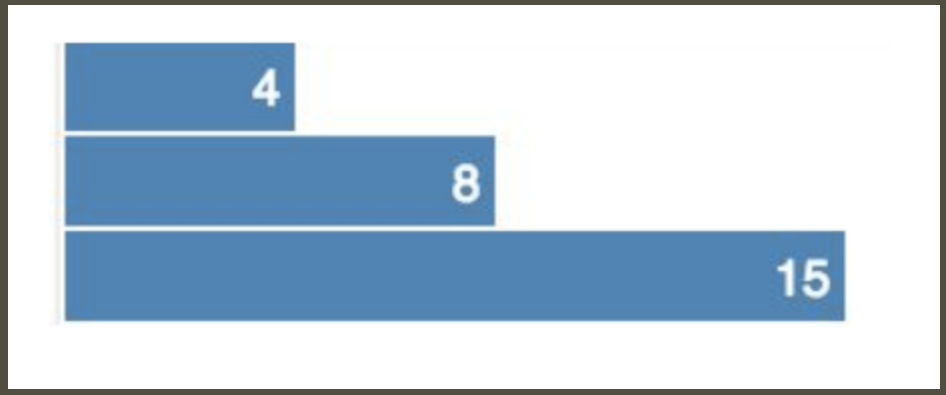
Data Driven Documents - D3

- What is D3 not
  - Not SVG or Canvas layer
  - Not a compatibility layer
    - D3 does not support old browsers
  - Not a charting library
    - D3 does not generate predefined canned visualization
  - Not a map library
    - Does allow you to map data but
    - D3 core functionality does not handle tiles
  - D3 does not hide your data

D3 is a general purpose visualization library

Data Driven Documents – D3

Data are bound to DOM elements to make **D**ata-**D**riven **D**ocuments



DATA

```
var data=[4,8,15]
```

Elements

```
▼<body>
  ▼<div class="chart">
      <div style="width: 40px;">4</div>
      <div style="width: 80px;">8</div>
      <div style="width: 150px;">15</div>
```

Bound

```
> d3.select(".chart").select("div").datum()
< 4
```

Adopted from http://alignedleft.com/tutorials/d3 and https://courses.cs.washington.edu/courses/cse442/17au/lectures/CSE442-D3-

D3 Example

- Generate DOM elements

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width">
  <script src="https://cdnjs.cloudflare.com/ajax/libs/d3/4.2.3/d3.min.js">
    </script>
  <title> D3 page template </title>
</head>
<body>
</body>
</html>
```

Chaining Methods

```
d3.select("body")  //  select element in html DOM
  .append("p")     // create a new element p
  .text("New paragraph");  // edit the text of element p
```

d3	Reference to the D3 object
select	select an element in the DOM that match the given selector
append	creates whatever new DOM element you specify. It return reference of the new element
text	Takes a string and insert it between the tags of the current selected element

https://jsbin.com/bizeyih/edit?html,js,output

Adopted from http://alignedleft.com/tutorials/d3

D3 Example 2

```
var dataset=[5,10,15,29,25];

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

```
var dataset=[5,10,15,29,25];
 d3.select("body")
   .data(dataset)
   .enter()
   .append("p")
   .text(function (d){
   return    d;
 });
```

Lets deconstruct these methods

d3	reference to the D3 object
select	select an element in the DOM that match the given selector
dataset	create and parse our data values
enter	To create new - data bound element, create new bind element for each new element in the data
append	creates whatever new DOM element you specify. It return reference of the new element
text	takes a string and insert it between the tags of the current selected element

Adopted from http://alignedleft.com/tutorials/d3

https://jsbin.com/bizeyih/edit?html,js,output

Drawing with Data

```
var dataset=[5,10,15,29,25];
 d3.select("body")
   .selectAll("div")
   .data(dataset)
   .enter()
   .append("div")
   .attr("class","bar")
   .style("height", function (d){
   return d+"px";

 })
```

```
div.bar{
  display:inline-block;
  width:20px;
  background-color:teal;
  margin-right:2px;
}
```

Adopted from http://alignedleft.com/tutorials/d3

https://jsbin.com/selihaq/edit?html,css,js,output

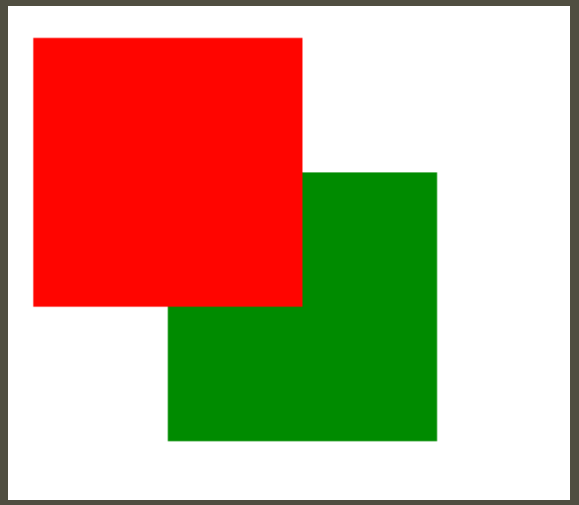
Drawing SVG

```
// first create the SVG

var svg= d3.select("body")
     .append("svg")
    .attr("width",500)
    .attr("height",500);

// then create elements...
  svg.append("rect")
      .attr("id","mySquare")
      .attr("x",100)
      .attr("y",100)
      .attr("width",160)
      .attr("height",160)

    svg.append("rect")
      .attr("class","red")
      .attr("x",20)
      .attr("y",20)
      .attr("width",160)
      .attr("height",160)
```



https://jsbin.com/segirak/edit?html,css,js,output

Adopted from http://alignedleft.com/tutorials/d3

Importing data using D3

D3 provides a set of methods to request data (https://github.com/d3/d3-request/blob/master/README.md#request)

- d3.text()
- d3.csv()
- d3.tsv()
- d3.json()

```
d3.csv("/path/to/file.csv", function(error, data) {
  if (error) throw error;
  console.log(data); // [{"Hello": "world"}, …]
});
```

Importing data using D3

csv

d3.csv(url[, accessor][, callback])

Read from url

row()

Process each row

get(callback)

Run callback once all processing is complete

drawCircle()

- select svg
- bind data
- draw circles

```
var url = "http://www.sfu.ca/~aga53/data/circles.csv";

d3.csv(url)
.row(function(d){
        d.x = +d.x; //convert sales to integer,
                    // default is string
        d.y = +d.y;
        d.r = +d.r;
    return d; //don't forget to return the row
})
.get(function(error, data){
    drawCircles(data);
})


function drawCircles(dataset){
    var svg = d3.select("svg");
    svg.selectAll("circle")
    .data(dataset)
    .enter()
    .append("circle")
    .attr("cx", function(d){return d.x;})
    .attr("cy", function(d){return d.y;})
    .attr("r", function(d){return d.r;})

}
```

Adopted from http://alignedleft.com/tutorials/d3

Exercise Time

Ex1

1. Given a dataset
2. Create a simple bar chart
3. Use svg and rect elements

dataset = [5, 10,13 ,15, 19,21, 25, 22,18, 13, 18 , 15, 13,11, 12, 15]

Hint

```
// you can use index in the array
.attr("attr_name" , function (d,i){
        // d is the data value
        //  i is the index of
        // data in the data array

 });
```

Exercise Time

Ex2

1. Read iris.csv (www.sfu.ca/~aga53/data/iris.csv) using d3
2. Draw circles by using following attributes
    1. Sepal Length as cx
    2. Sepal Width as cy
    3. Petal Length as r (radius)
3. The resulting visualization might look too cluttered because the values are too small. To fix it, when importing data:
    1. Multiply Sepal Length and Sepal Width by 20, and
    2. Multiply radius by 2

D3-Tutorials

- Best tutorials
  - Mike Bostock tutorials
    - https://github.com/d3/d3/wiki/Tutorials
    - https://bost.ocks.org/mike/bar/
  - Scot Murray http://alignedleft.com/tutorials/d3/
  - Square tutorial https://square.github.io/intro-to-d3/
  - Fun introduction https://tmcw.github.io/presentations/dcjq/
  - Other resources
    - D3 Garden https://www.rtfmanual.io/d3garden/
    - D3 official Gallery https://github.com/d3/d3/wiki/Gallery
    - Big list of example of D3 http://christopheviau.com/d3list/
    - Collection of resources https://website.education.wisc.edu/~swu28/d3t/link.html

D3 and more

- Interesting libraries build for D3
  - Cross filter http://square.github.io/crossfilter/
  - DC.js reusable charts - https://dc-js.github.io/dc.js/
  - NVD3 reusable chart - http://nvd3.org/