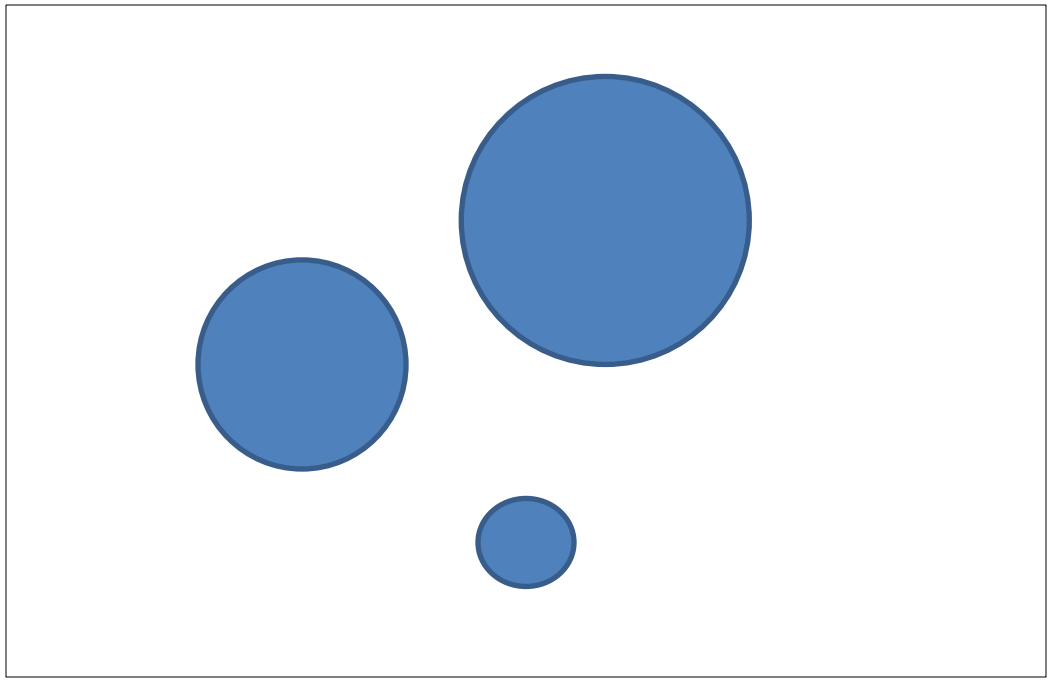
Side note: Bootstrap

```
<body>


    <div class="container-fluid">
        <div class="row">
            <div class="col-md-12">
                <div class="row">
                    <div class="col-md-6">
                        <!-- Your left column content goes here-->
                    </div>
                    <div class="col-md-6">
                        <!-- Your right column content goes here -->
                    </div>
                </div>
            </div>
        </div>
    </div>
```

http://www.layoutit.com/

Drawing many things automatically



Last week, fully manual – gets tedious for 100 circles!

This week, use Javascript and d3 to automate this process

Basic data storage: Objects and arrays

Arrays are the simplest: lists of information that Javascript can read

```
var myArray = [ item1, item2, item3, item4 ]

myArray[1] = item2
```

Objects store groups of information in properties, which can be accessed anytime you need it.

```
var myObject = {
    name: "object1",
    xPos: 100,
    yPos: 300,
    radius: 100
}

myObject.radius = 100
```

Can have arrays of objects, and objects that contain arrays!

Array of objects (each with its own properties):

```
var myArray = [ object1, object2, object3, object4 ]

myArray[1] = object2

myArray[0].yPos = 300;
```

Object containing an array:

```
var myObject = {
    name: "object1",
    xPos: 100,
    yPos: 300,
    array: [ 2, 10, 3, 15, 7, 9 ]
}

myObject.array[2] = 3
```

We can use loops to make objects and arrays automatically

What's a loop?

Envelope example #1:

Envelope contains three cards

- Draw a check mark on each card.

Convert that to programming language:

```
for (card = 0; card < 3; card++ ) {                    //increment/decrement
    card.check = true;
}
```

Envelope example #2:

Envelope contains three cards

- Look at each card.
- If it is green, write 100 on it
- If it isn't green, write 10 on it

```
for ( card = 0; card < 3; card++ ) {

    if ( card.color  == "green" ){                    //operator logic!
          card.value = 100;
     }
    else {
          card.value = 10;
    }

}
```

Make an array of objects containing information about a circle

Option one:

Create array of empty objects, iterate through and populate properties

```
var array = [ { }, { }, { }, { }, { }, { }, { }, { }, { }, { } ];

for ( i =0;  i <10;  i++){
    array [ i ].value = Math.random ( );

}
```

Make an array of objects containing information about a circle

Option two:

Create array of empty objects, iterate through using array.forEach to populate properties

```
var array = [ {}, {}, {}, {}, {}, {}, {}, {}, {}, {} ];

array.forEach ( function ( arrayElement ) {

    arrayElement.value = Math.random (  );

});
```

Make an array of objects containing information about a circle

Option three:

Initialize array, for loop to make objects with desired value, push into the array

```
var array = [ ];

for ( i = 0;  i < 10;  i++ ) {
    //create an object
    var object = { value: Math.random ( ) };

    //put it inside the array
    array.push ( object );
}
```

D3 data bind

1) Make an empty selection
2) Bind that selection to data
3) Use Enter to make DOM elements for each data point
4) Use data stored in DOM elements to draw things

```
svg.selectAll  ( '.circles' )                    //#1
    .data ( myData )                             //#2
    .enter ( )                                   //#3
    .append ( 'circle' )
    .attr ( 'cx', function ( d ) {
              return d.x;                        //#4
     })...
```

Loading data from .csv

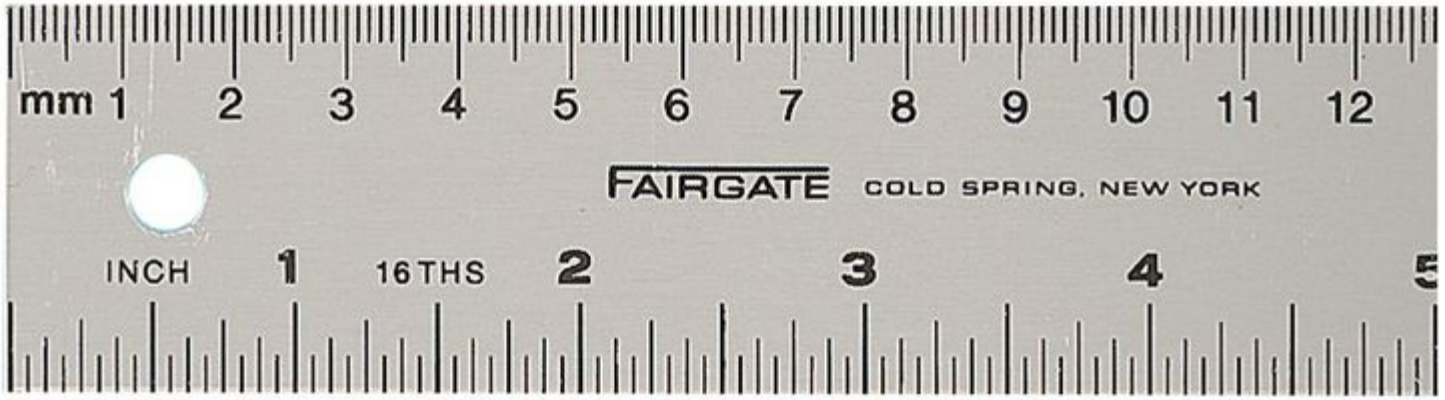
CSV file in folder. In Javascript:

```
d3.csv ( ‘ ./data.csv’ , function ( dataIn ) {
    console.log ( dataIn );
})
```

d3 scales

Scales in d3 help us convert values from one unit to another.

cm

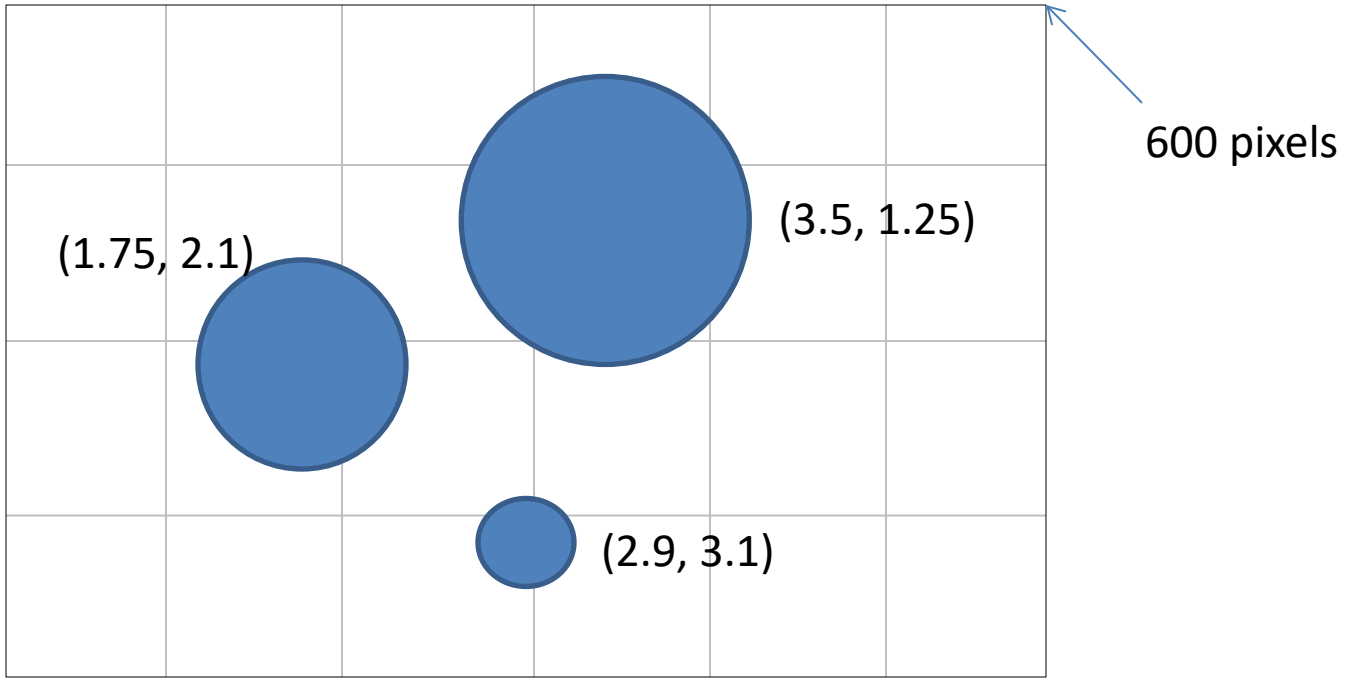


inches

```
d3.scaleLinear
        .domain ( [ values to convert from ] )
        .range ( [ values to convert to ] );

d3.scaleLinear
        .domain ( [ 0, 4 ] )                    //measurement in inches
        .range ( [ 0, 10 ] );                   //measurement in cm
```

For our purposes:




```
scaleX = d3.scaleLinear
         .domain ( [ 0, 6 ] )                    //measurement in gridlines
         .range ( [ 0, 600 ] );                  //measurement in pixels

scaleY = d3.scaleLinear .domain ( [ 0, 4 ] ) .range ( [ 0, 400 ] );
```

Matching names to values using scales

d3.scaleLinear( ) takes one number and turns it into another number. Once you set up the ends of the scale, it works for all numbers in between:

scaleX (3.5) = 350

Sometimes, you want to match one specific value to another value, without any options in between. For that, you use scaleOrdinal:

```
scaleR = d3.scaleOrdinal ( )
            .domain ( [ "small", "medium", "large"] )
            .range ( [ 5, 10, 15 ] );
```

scaleR (“medium”) = 10;