

# **Lab 7**

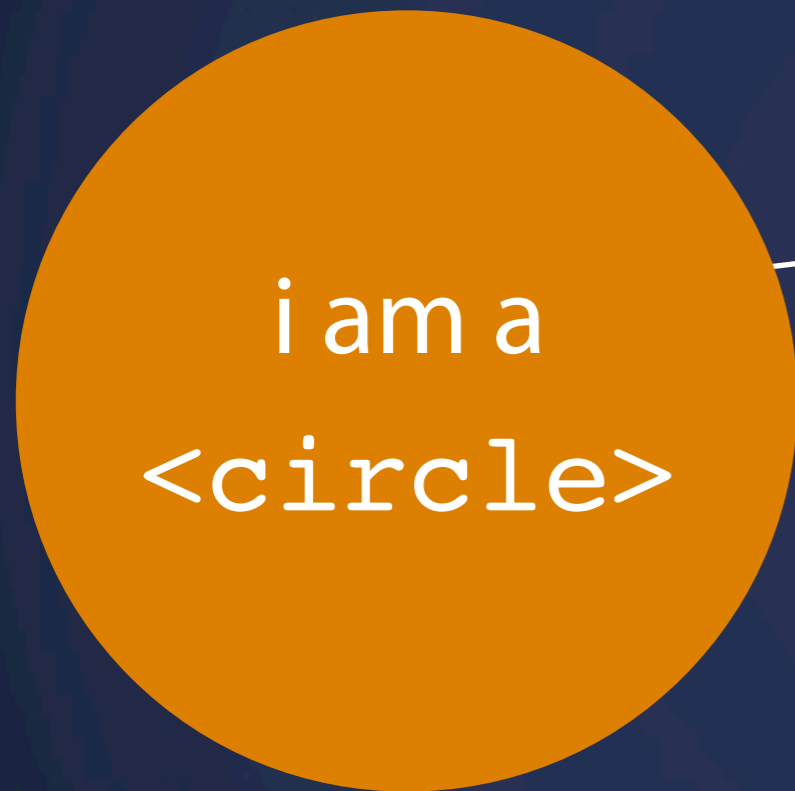
## **D3.js (Part II)**

March 7, 2013 – Michael Porath (@poezn)

# Assignment 4

- get started EARLY!
- include data at least through **March 12**.
- leave time to iterate. Show iterations
- No files in a proprietary format (including Word, Pages, Illustrator, etc.)
- Get feedback!
- We're here to help
- Have a look at updated A4 guidelines for more info

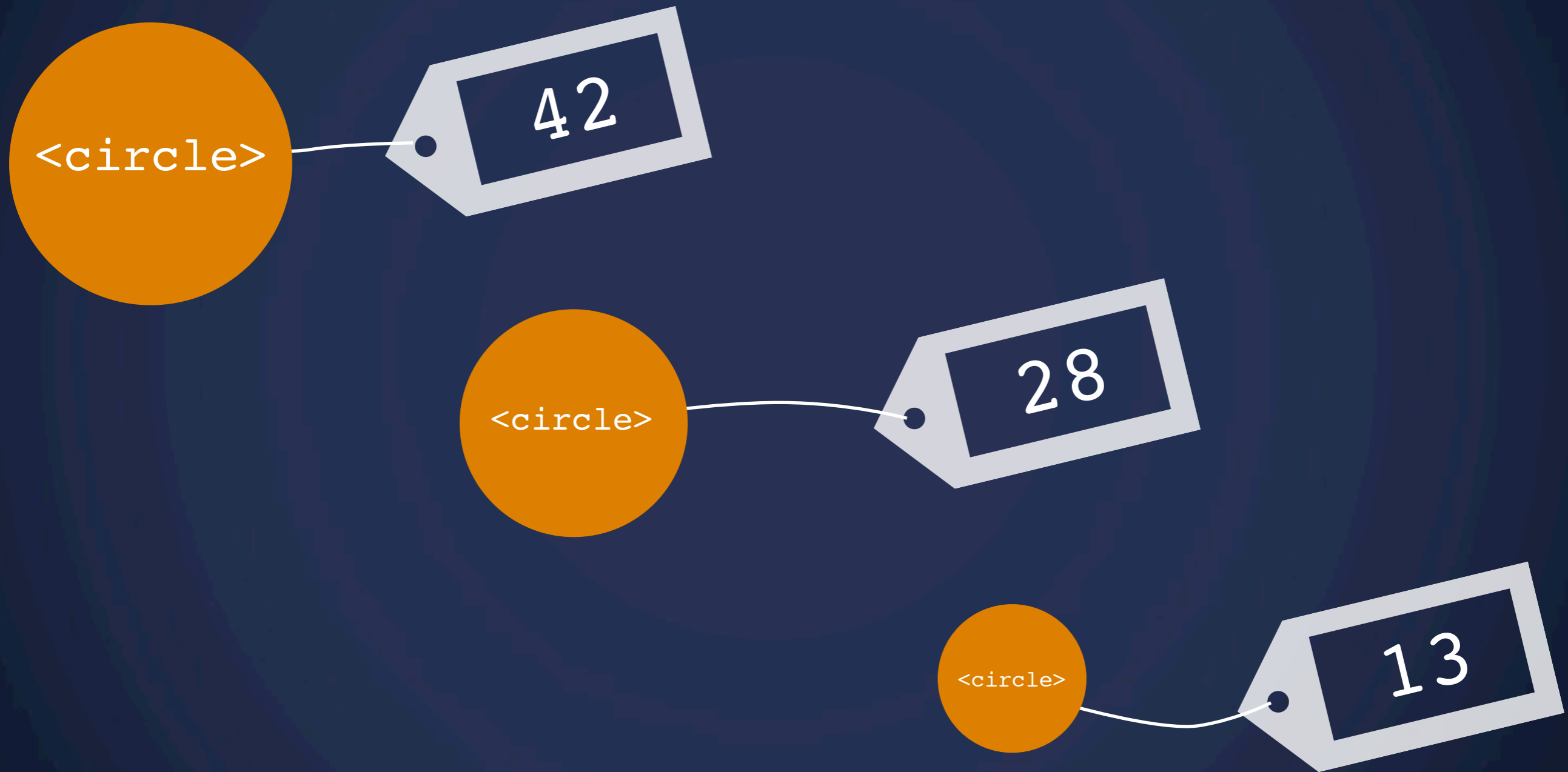
# Data Binding



each `<element>` has  
a datum "attached"

# Data Binding

Data Driven Attributes. Here: radius



# **Your Turn**

**Let's try this**

# First example

Re-creating the OK Cupid color matrix with D3.js

**Reply Rate By Race**  
*male sender*

	Asian - Female	Black - Female	Hispanic/Latin - Female	Indian - Female	Middle Eastern - Female	Native American - Female	Other - Female	Pacific Islander - Female	White - Female	
Asian - Male	22	34	22	18	27	31	30	23	21	22.2
Black - Male	17	28	19	21	21	27	24	25	21	21.7
Hispanic/Latin - Male	20	31	24	25	25	29	27	24	22	23.1
Indian - Male	20	34	19	18	23	27	28	24	20	20.8
Middle Eastern - Male	26	37	23	24	23	36	29	28	25	25.7
Native American - Male	26	34	26	27	32	35	32	29	27	27.8
Other - Male	25	32	26	21	23	34	32	22	26	26.8
Pacific Islander - Male	21	39	21	33	32	32	35	30	23	24.6
White - Male	29	38	30	30	28	34	33	29	29	29.2
	25.7	34.1	27.1	25.1	27.0	33.0	31.4	27.0	27.3	27.6

<http://blog.okcupid.com/index.php/your-race-affects-whether-people-write-you-back/>



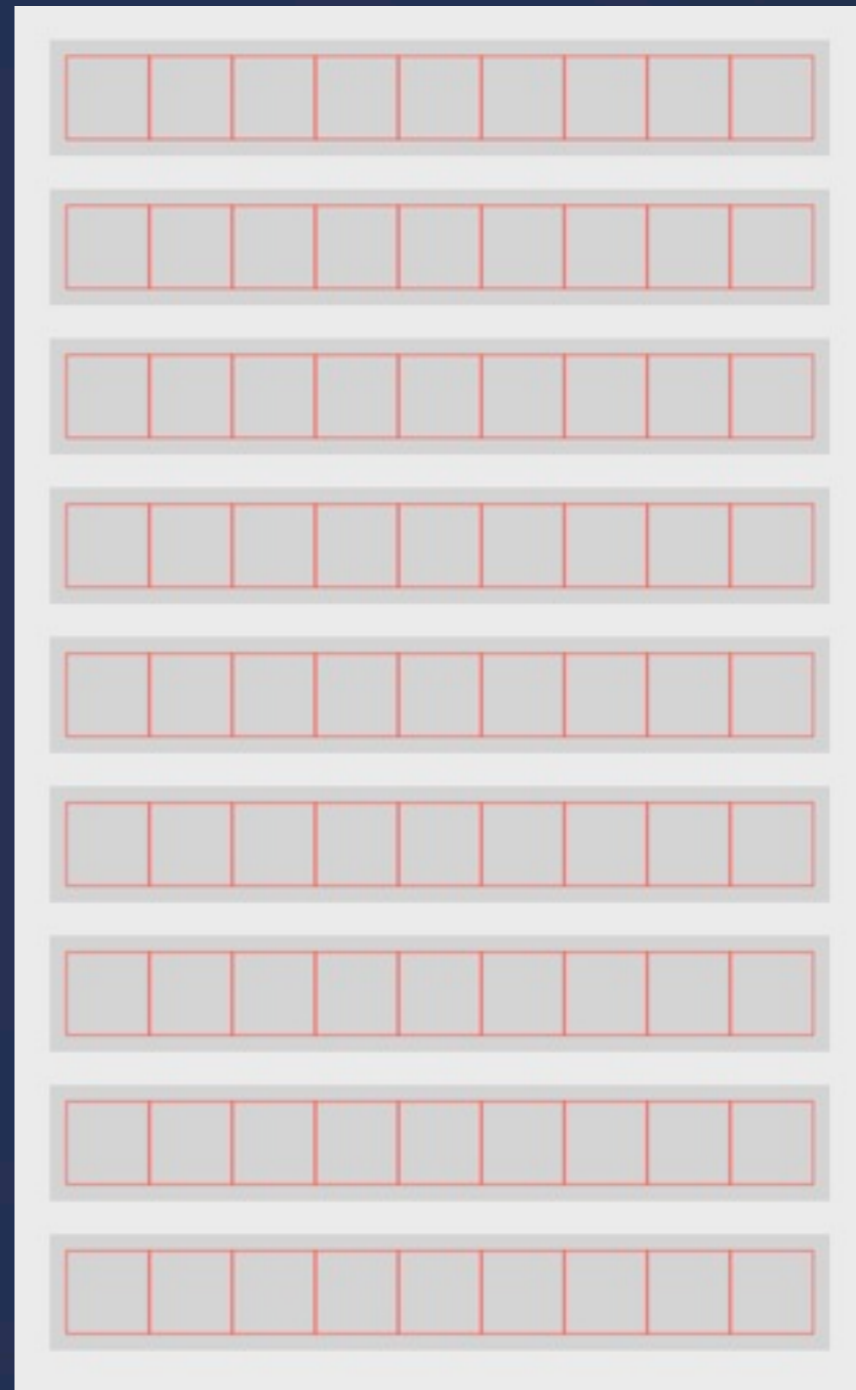




# Make a Plan

Let's make a concept up front

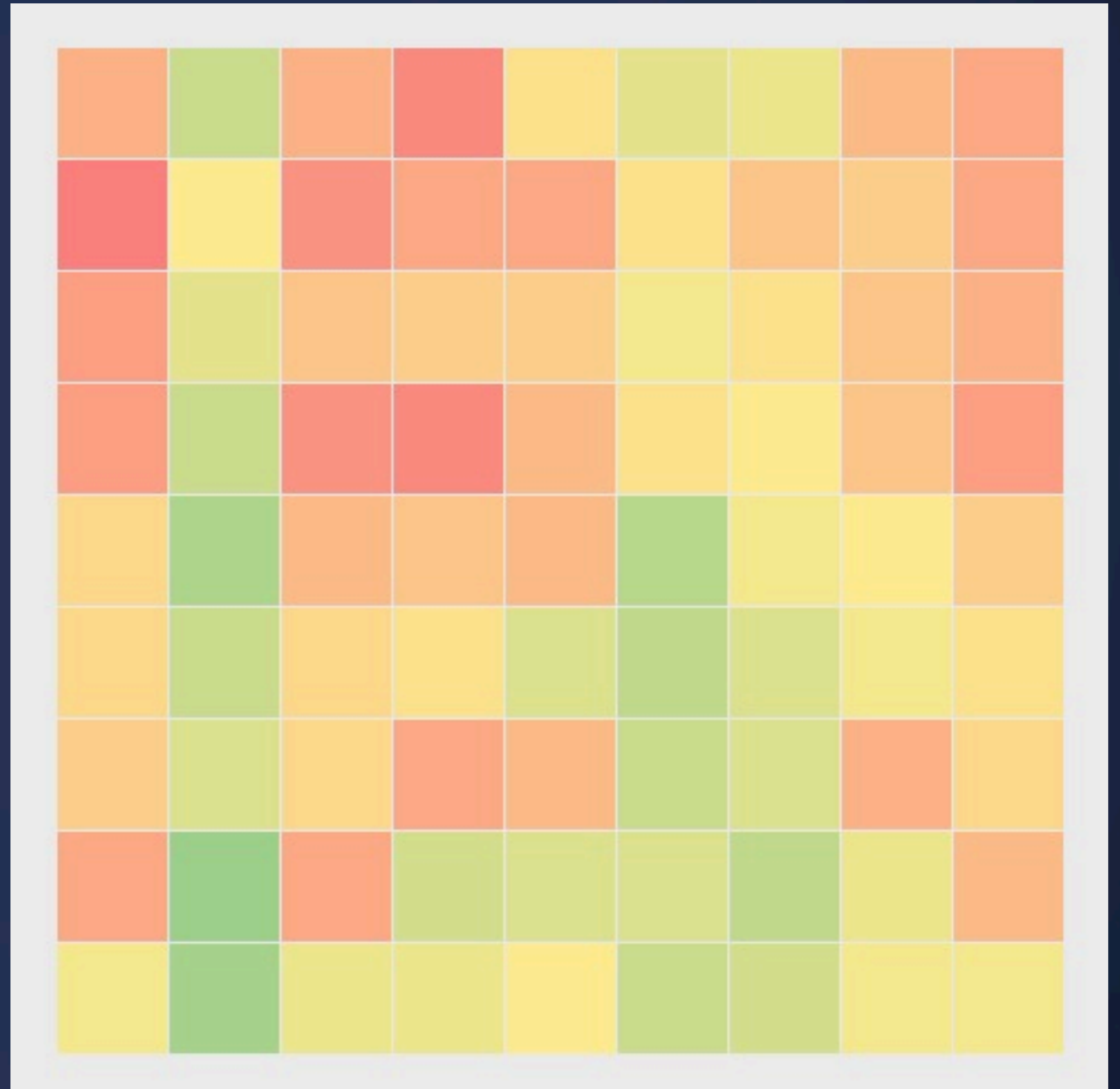
9 squares  
per row



# Make a Plan

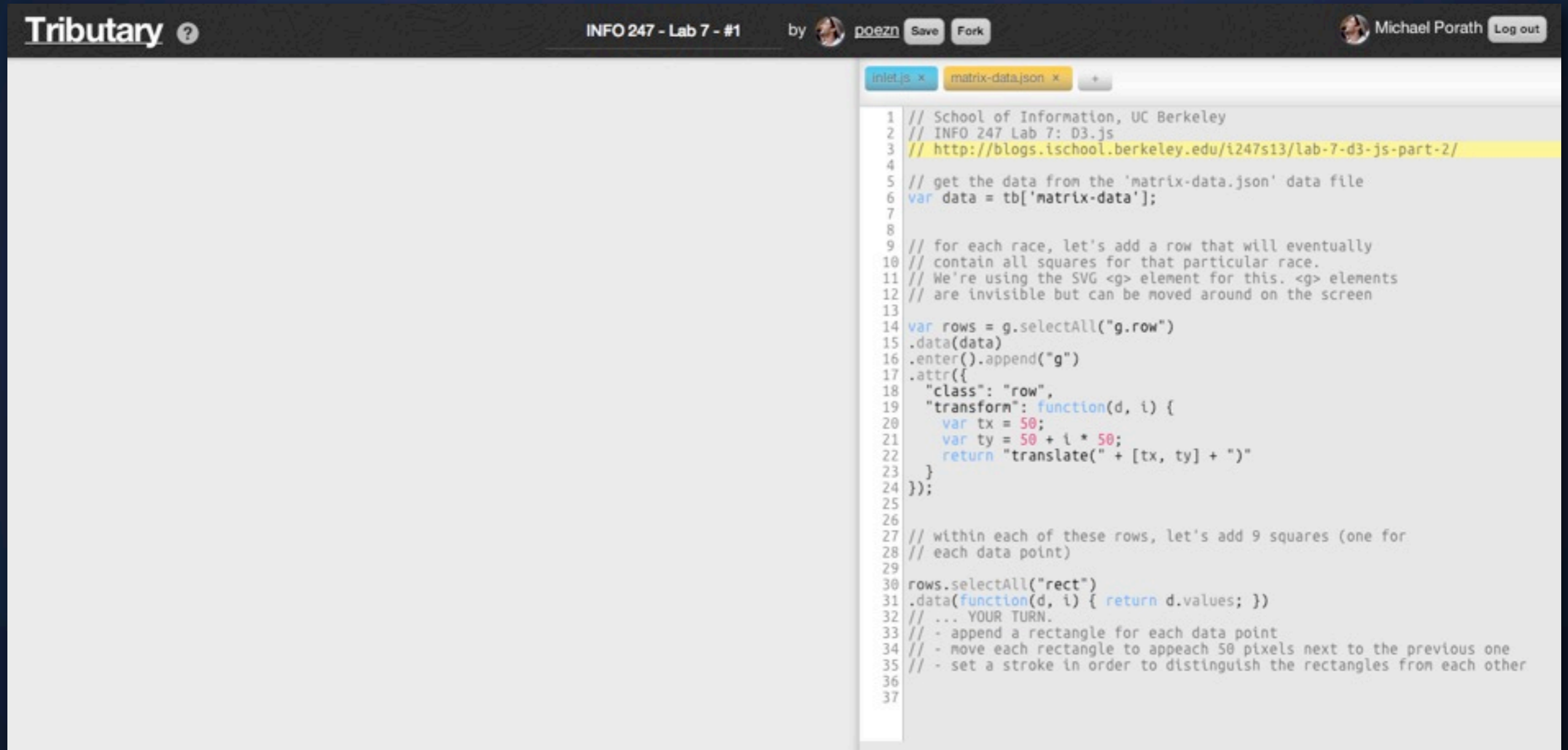
Let's make a concept up front

fill the squares  
with the right  
color



# Exercise 1

Draw the squares

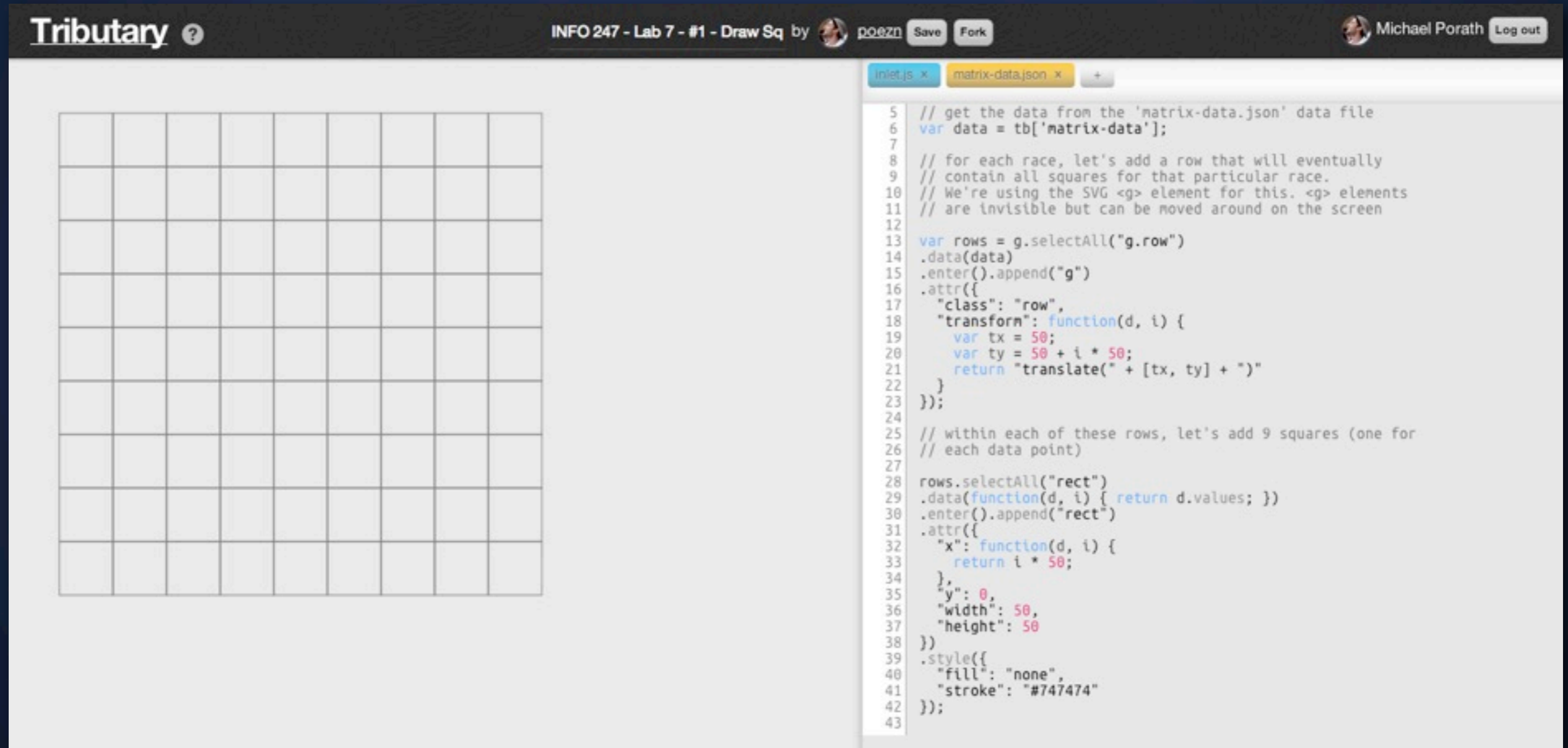


```
1 // School of Information, UC Berkeley
2 // INFO 247 Lab 7: D3.js
3 // http://blogs.ischool.berkeley.edu/i247s13/lab-7-d3-js-part-2/
4
5 // get the data from the 'matrix-data.json' data file
6 var data = tb['matrix-data'];
7
8
9 // for each race, let's add a row that will eventually
10 // contain all squares for that particular race.
11 // We're using the SVG <g> element for this. <g> elements
12 // are invisible but can be moved around on the screen
13
14 var rows = g.selectAll("g.row")
15 .data(data)
16 .enter().append("g")
17 .attr({
18   "class": "row",
19   "transform": function(d, i) {
20     var tx = 50;
21     var ty = 50 + i * 50;
22     return "translate(" + [tx, ty] + ")"
23   }
24 });
25
26
27 // within each of these rows, let's add 9 squares (one for
28 // each data point)
29
30 rows.selectAll("rect")
31 .data(function(d, i) { return d.values; })
32 // ... YOUR TURN.
33 // - append a rectangle for each data point
34 // - move each rectangle to appeach 50 pixels next to the previous one
35 // - set a stroke in order to distinguish the rectangles from each other
36
37
```

<http://tributary.io/inlet/5054734>

# Exercise 1

Draw the squares (full code)



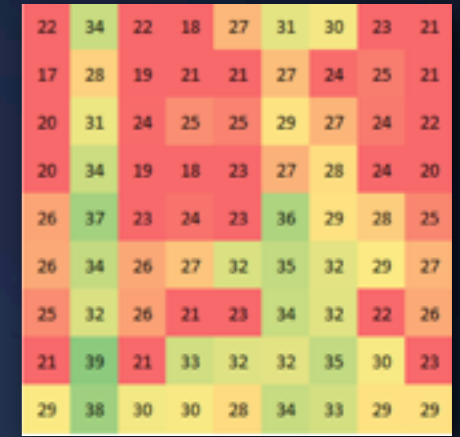
The screenshot shows a Tributary.io workspace. On the left, there is a 10x10 grid of squares. On the right, there is a code editor with the following JavaScript code:

```
5 // get the data from the 'matrix-data.json' data file
6 var data = tb['matrix-data'];
7
8 // for each race, let's add a row that will eventually
9 // contain all squares for that particular race.
10 // We're using the SVG <g> element for this. <g> elements
11 // are invisible but can be moved around on the screen
12
13 var rows = g.selectAll("g.row")
14 .data(data)
15 .enter().append("g")
16 .attr({
17   "class": "row",
18   "transform": function(d, i) {
19     var tx = 50;
20     var ty = 50 + i * 50;
21     return "translate(" + [tx, ty] + ")"
22   }
23 });
24
25 // within each of these rows, let's add 9 squares (one for
26 // each data point)
27
28 rows.selectAll("rect")
29 .data(function(d, i) { return d.values; })
30 .enter().append("rect")
31 .attr({
32   "x": function(d, i) {
33     return i * 50;
34   },
35   "y": 0,
36   "width": 50,
37   "height": 50
38 })
39 .style({
40   "fill": "none",
41   "stroke": "#747474"
42 });
43
```

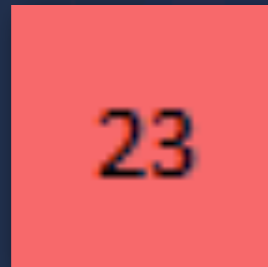
<http://tributary.io/inlet/5054769>

# Color Scales

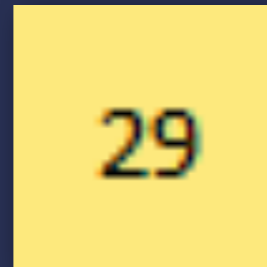
How?



...



...



...



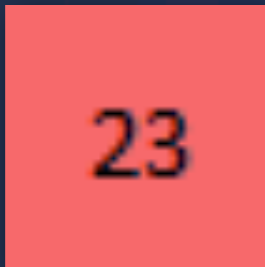
...

# Color Scales

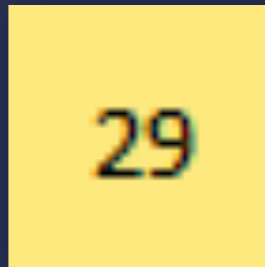
How?

22	34	22	18	27	31	30	23	21
17	28	19	21	21	27	24	25	21
20	31	24	25	25	29	27	24	22
20	34	19	18	23	27	28	24	20
26	37	23	24	23	36	29	28	25
26	34	26	27	32	35	32	29	27
25	32	26	21	23	34	32	22	26
21	39	21	33	32	32	35	30	23
29	38	30	30	28	34	33	29	29

...



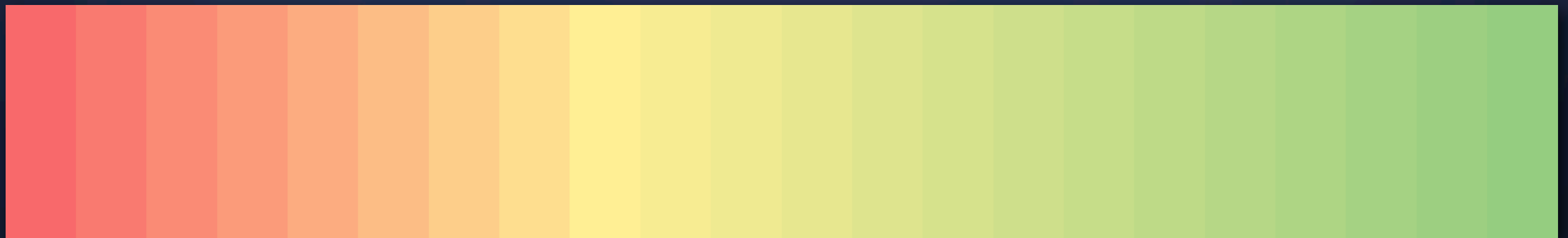
...



...



...



19

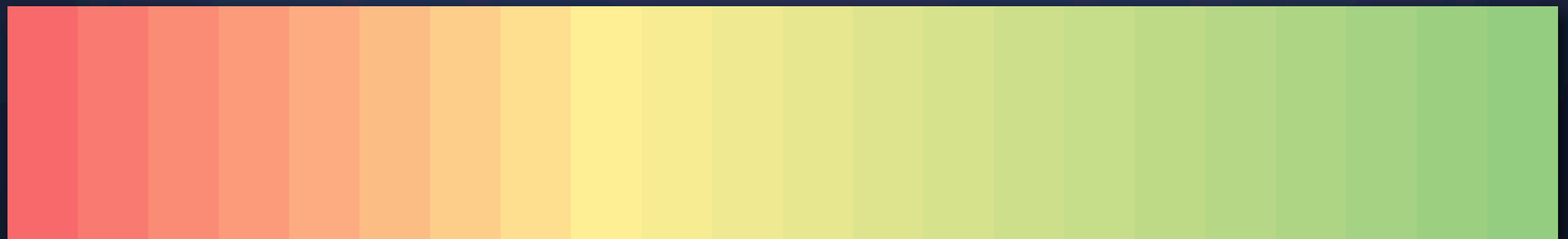
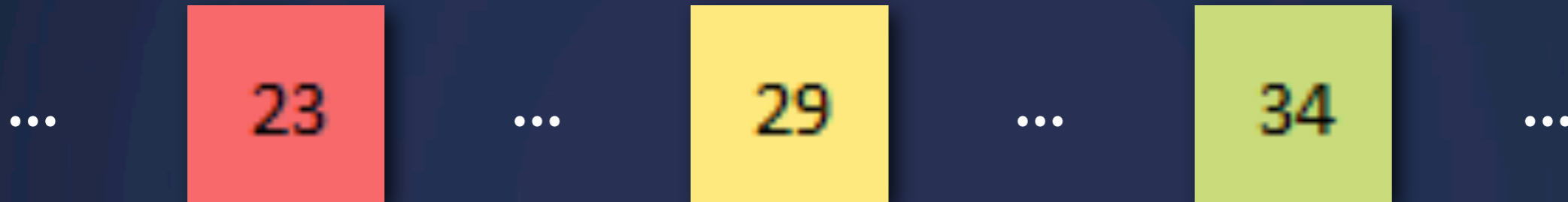


37

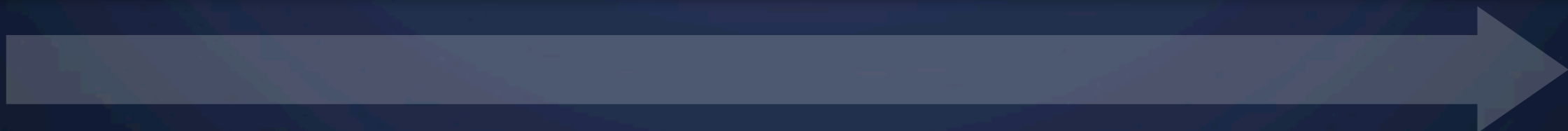
# Color Scales

How?

22	34	22	18	27	31	30	23	21
17	28	19	21	21	27	24	25	21
20	31	24	25	25	29	27	24	22
20	34	19	18	23	27	28	24	20
26	37	23	24	23	36	29	28	25
26	34	26	27	32	35	32	29	27
25	32	26	21	23	34	32	22	26
21	39	21	33	32	32	35	30	23
29	38	30	30	28	34	33	29	29



19

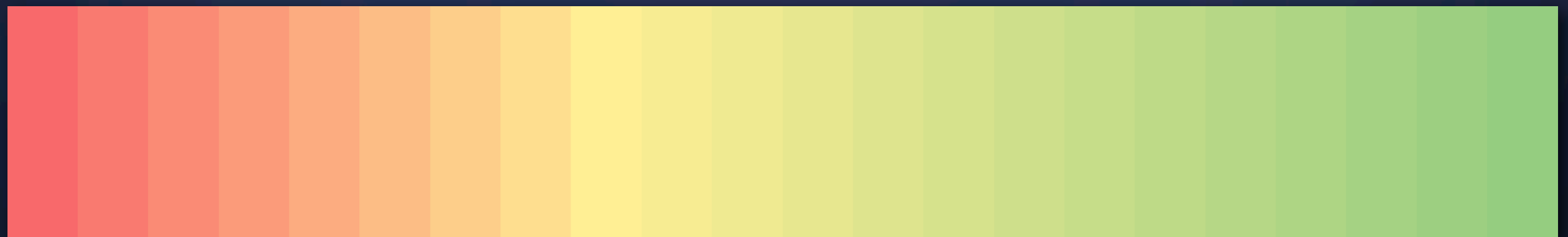
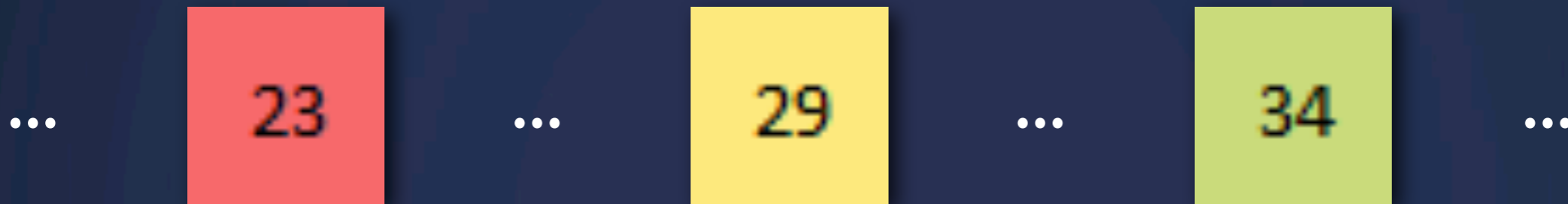


37

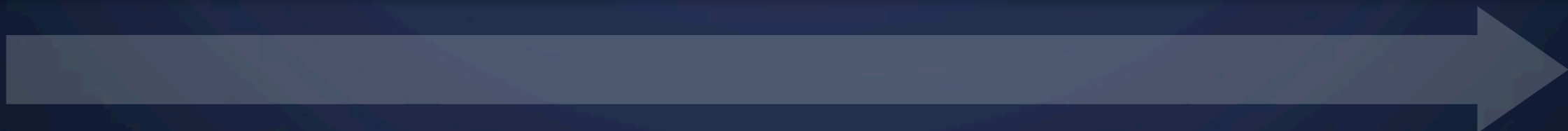
# Color Scales

How?

22	34	22	18	27	31	30	23	21
17	28	19	21	21	27	24	25	21
20	31	24	25	25	29	27	24	22
20	34	19	18	23	27	28	24	20
26	37	23	24	23	36	29	28	25
26	34	26	27	32	35	32	29	27
25	32	26	21	23	34	32	22	26
21	39	21	33	32	32	35	30	23
29	38	30	30	28	34	33	29	29



19



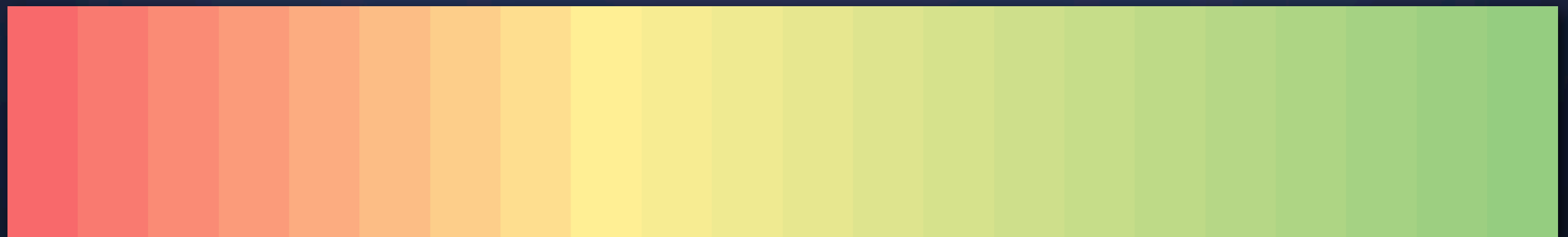
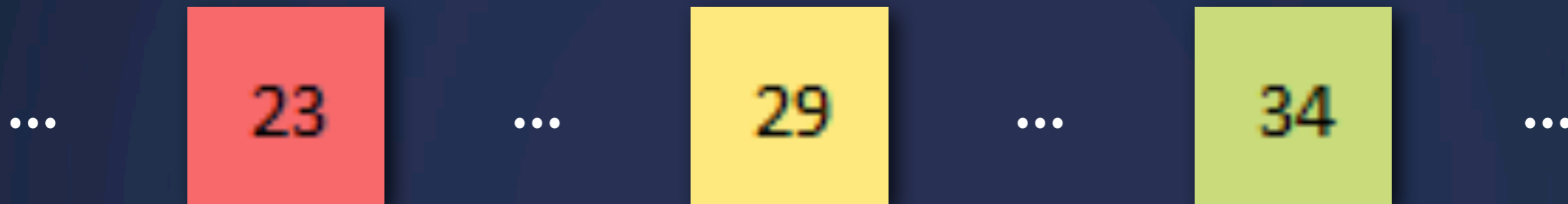
37



# Color Scales

How?

22	34	22	18	27	31	30	23	21
17	28	19	21	21	27	24	25	21
20	31	24	25	25	29	27	24	22
20	34	19	18	23	27	28	24	20
26	37	23	24	23	36	29	28	25
26	34	26	27	32	35	32	29	27
25	32	26	21	23	34	32	22	26
21	39	21	33	32	32	35	30	23
29	38	30	30	28	34	33	29	29




19

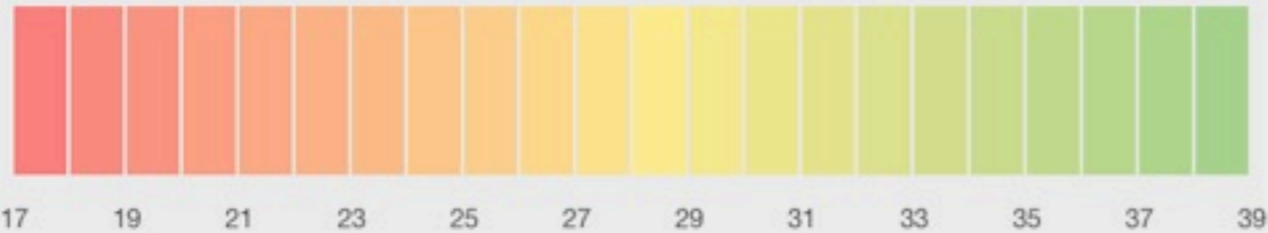


37


# Exercise 2

## Color Scale

Tributary ? INFO 247 - Lab 7 - #2 - Color Sc by  poezn Save Fork  Michael Porath Log out



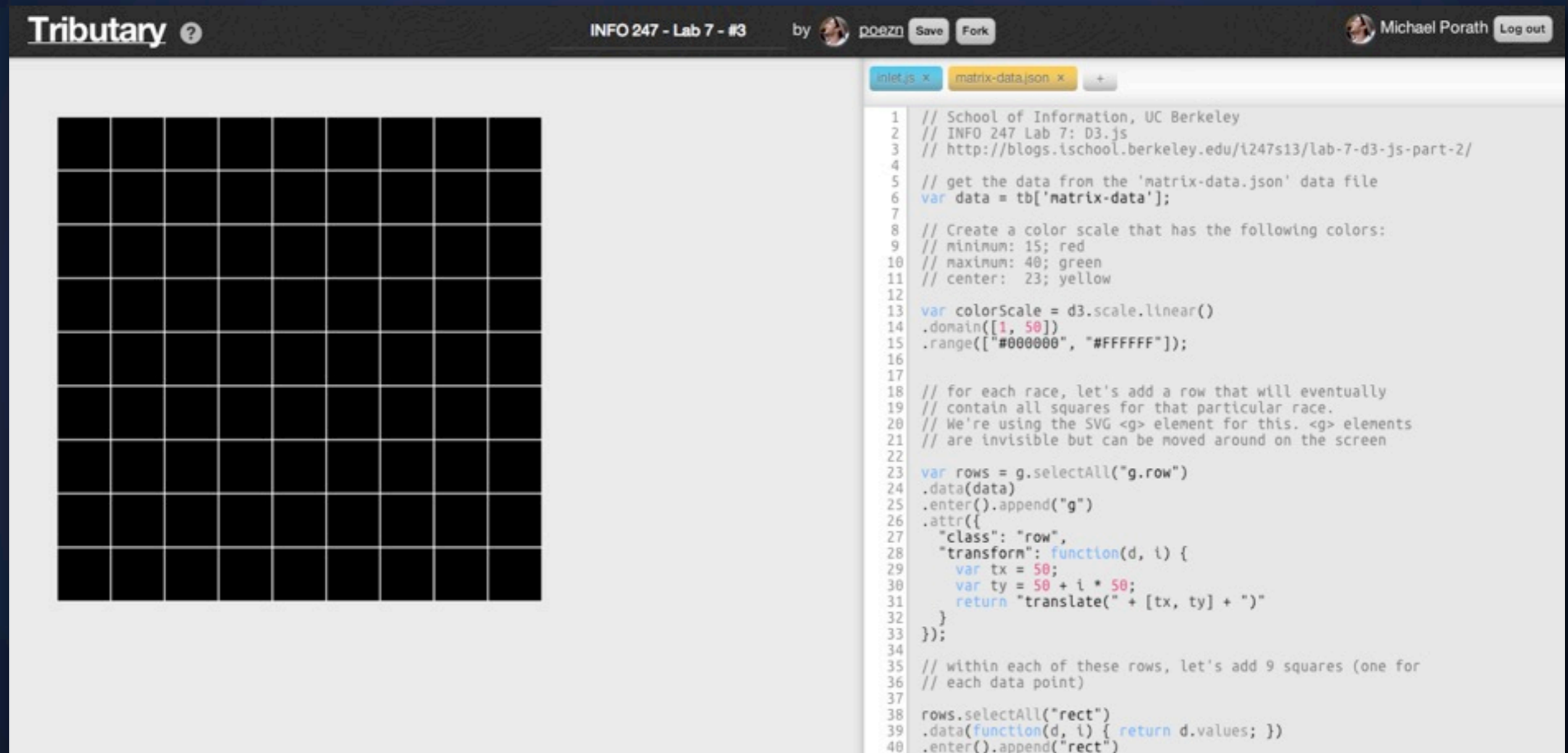
```
1 // School of Information, UC Berkeley
2 // INFO 247 Lab 7: D3.js
3 // http://blogs.ischool.berkeley.edu/i247s13/lab-7-d3-js-part-2/
4
5 var start = 17;
6 var middle = 28;
7 var end = 39;
8 var legendSquareWidth = 32;
9
10 // YOUR TURN
11 // Experiment with the color scale
12
13 // this maps a start, middle, and end point to a color each
14 var colorScale = d3.scale.linear()
15 .domain([start, middle, end])
16 .range(['#F8696B', '#FCEA84', '#8dca7e']);
17
18 // draw big rectangle
19 g.append("rect")
20 .attr({
21   "width": 150,
22   "height": 150,
23   "x": 350,
24   "y": 330
25 })
26 .style({
27   "fill": function(d, i) {
28     return colorScale(28); // <== CHANGE THIS
29   }
30 });
31
32
33
34
35
36
37 // This draws the legend
38 g.selectAll("rect.legend")
39 .data(d3.range(start, end))
40 .enter().append("rect")
```



<http://tributary.io/inlet/5111589>

# Exercise 3

Putting it together



The screenshot displays a Tributary.io project interface. On the left, a 10x10 grid of black squares is shown. On the right, the code editor shows the following JavaScript code:

```
1 // School of Information, UC Berkeley
2 // INFO 247 Lab 7: D3.js
3 // http://blogs.ischool.berkeley.edu/~t247s13/lab-7-d3-js-part-2/
4
5 // get the data from the 'matrix-data.json' data file
6 var data = tb['matrix-data'];
7
8 // Create a color scale that has the following colors:
9 // minimum: 15; red
10 // maximum: 40; green
11 // center: 23; yellow
12
13 var colorScale = d3.scale.linear()
14 .domain([1, 50])
15 .range(["#000000", "#FFFFFF"]);
16
17
18 // for each race, let's add a row that will eventually
19 // contain all squares for that particular race.
20 // We're using the SVG <g> element for this. <g> elements
21 // are invisible but can be moved around on the screen
22
23 var rows = g.selectAll("g.row")
24 .data(data)
25 .enter().append("g")
26 .attr({
27   "class": "row",
28   "transform": function(d, i) {
29     var tx = 50;
30     var ty = 50 + i * 50;
31     return "translate(" + [tx, ty] + ")";
32   }
33 });
34
35 // within each of these rows, let's add 9 squares (one for
36 // each data point)
37
38 rows.selectAll("rect")
39 .data(function(d, i) { return d.values; })
40 .enter().append("rect")
```

<http://tributary.io/inlet/5054877>

# Exercise 3

## Putting it together - Full Code



<http://tributary.io/inlet/5054889>

# Transformations

SVG attribute "transform"

# Transformations

SVG attribute "transform"

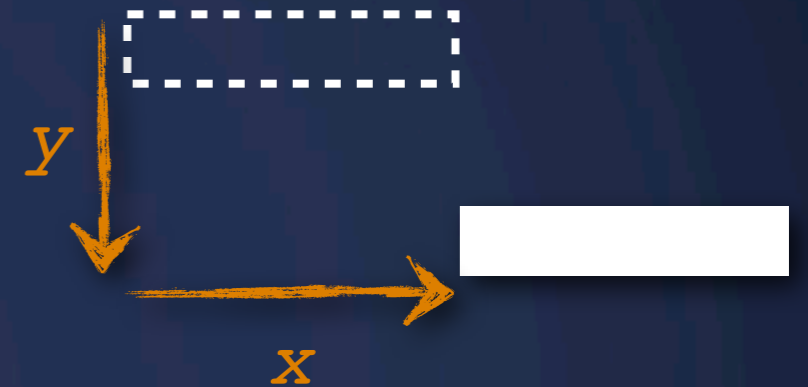
move      `translate(x, y)`



# Transformations

SVG attribute "transform"

move      `translate(x, y)`



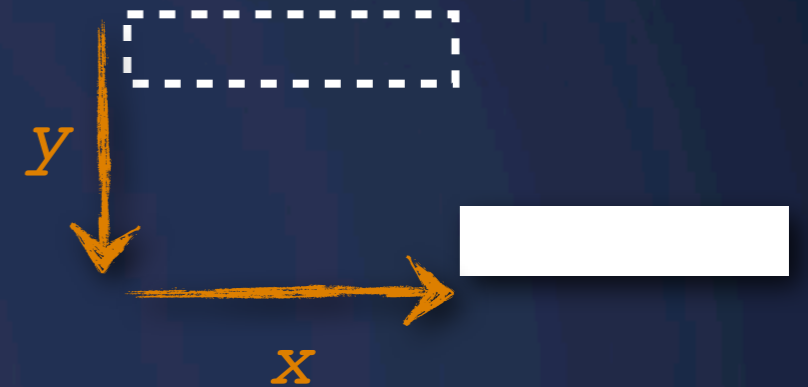
rotate      `rotate(degrees)`



# Transformations

SVG attribute "transform"

move      `translate(x, y)`



rotate      `rotate(degrees)`



scale      `scale(factor)`





# Transformations

You can chain them

```
<rect  
  width="200"  
  height="30"  
  transform="translate(20, 50) rotate(-45) scale(3)"  
></rect>
```



# Exercise 4

Play with transformations

The screenshot shows a Tributary.io workspace. The top navigation bar includes the Tributary logo, the workspace title "INFO 247 - Lab 7 - #4 - Transform" by user "poezn", and buttons for "Save" and "Fork". The user "Michael Porath" is logged in. The main canvas area contains three text elements: "Move Me!", "Rotate Me!" (which is rotated), and "Scale Me!" (which is scaled). On the right, a code editor shows the JavaScript code for these elements:

```
1 // School of Information, UC Berkeley
2 // INFO 247 Lab 7: D3.js
3 // http://blogs.ischool.berkeley.edu/1247s13/lab-7-d3-js-part-2/
4
5 g.append("text")
6 .attr({
7   "transform": function(d, i) {
8     var tx = 21;
9     var ty = 50;
10    return "translate(" + [tx, ty] + ")";
11  }
12 })
13 .text("Move Me!");
14
15
16 g.append("text")
17 .attr({
18   "transform": function(d, i) {
19     var tx = 21;
20     var ty = 200;
21     var rotation = -20; // degrees
22     return "rotate(" + rotation + ") translate(" + [tx, ty] + ")";
23   }
24 })
25 .text("Rotate Me!");
26
27
28 g.append("text")
29 .attr({
30   "transform": function(d, i) {
31     var tx = 21;
32     var ty = 200;
33     var scale = 2; // factor
34     // What happens if you change the order of scale and translate?
35     return "scale(" + scale + ") translate(" + [tx, ty] + ")";
36   }
37 })
38 .text("Scale Me!");
39
```

<http://tributary.io/inlet/5111933>

# **Next Lecture**

**Storytelling**

# **Next Lab**

**Storytelling**