D3 (v5) Calisthenics

CS448B FALL 2018

Warm-Up: Run a local server

Create a file named *index.html* with the text "Hello, world!"

Warm-Up: Run a local server

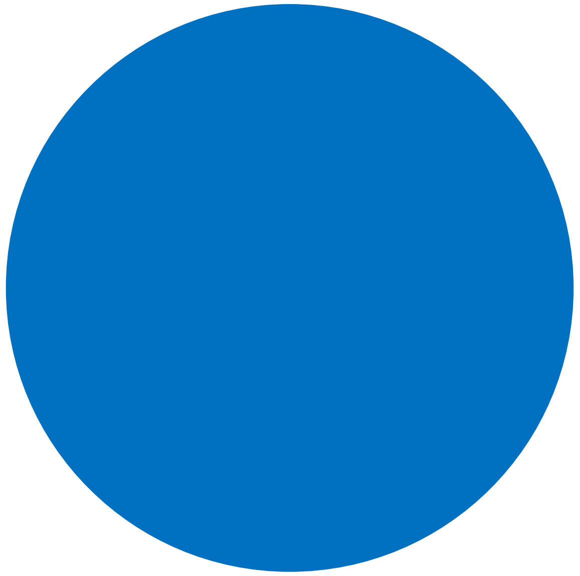
```
> python -m SimpleHTTPServer
> python3 -m http.server 8000 (try this one if you are using Python 3)
```

Type “localhost:<port_number>” in a web browser.

You should see a webpage displaying “Hello, world!”

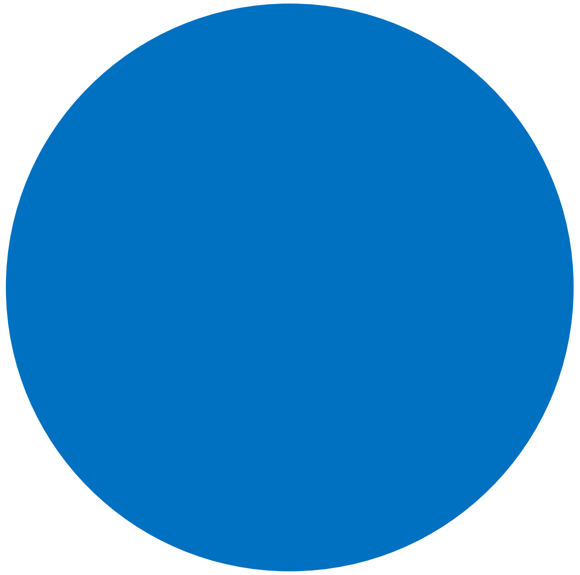
Exercise 1: Draw a blue circle using SVG (Don’t use D3/Javascript!)

```
<html>
<head>
</head>
<body>
   <svg width=”1000" height=”1000">
        Fill in the blank.
   </svg>
</body>
</html>
```



Exercise 1: Solution

```
<html>
<head>
</head>
<body>
   <svg width=”1000" height=”1000">
      <circle style="fill: blue;" r="55" cx="60" cy="60” />
   </svg>
</body>
</html>
```



Exercise 2: Draw a blue circle using D3

```
…
<head>
   <script src="http://d3js.org/d3.v5.min.js"></script>
</head>
<body>
   <div id="visualization_area"></div>
   <script>
       …Put your D3 code here...
   </script>
   …
```

Exercise 2: Solution

```
var svg = d3.select("#visualization_area")
        .append("svg")
        .attr("width", 1000)
        .attr("height", 1000);

svg.append("circle")
        .style("fill", "blue")
        .attr("r", 55)
        .attr("cx", 60)
        .attr("cy", 60);
```

Exercise 3: Bind circles and place on page according to (x,y) data.

Use the below data, which creates an Array of objects containing a random X and Y position within a 1000x1000 pixel canvas.

```
// Assuming SVG width and height = 1000, circle radius is 55
var circle_position_data = d3.range(10).map(function() {
        return {
                x: Math.round(Math.random() * (1000 - 55 * 2) + 55), // Random x-pixel on the page
                y: Math.round(Math.random() * (1000 - 55 * 2) + 55) // Random y-pixel on the page
        };
});
```

Note: When refreshing the page, the 3 blue circles should appear in a new random position.

Exercise 3: Solution

```
svg.selectAll("circle")
        .data(circle_position_data)
        .enter()
        .append("circle")
        .style("fill", "blue")
        .attr("r", 55)
        .attr("cx", function(d) { return d.x; })
        .attr("cy", function(d) { return d.y; });
```

Exercise 4: Drag circles around the screen

Hint: Check out the D3 documentation at https://github.com/d3/d3-drag.

Read the section on **d3.drag()**.

Also, read the sections on **drag.on()** and **event.on()**.

Use the D3 **call()** method to “call“ the D3 drag behavior on your selection of circles.

In general, https://github.com/d3/d3/blob/master/API.md is your friend!

Exercise 4: Solution

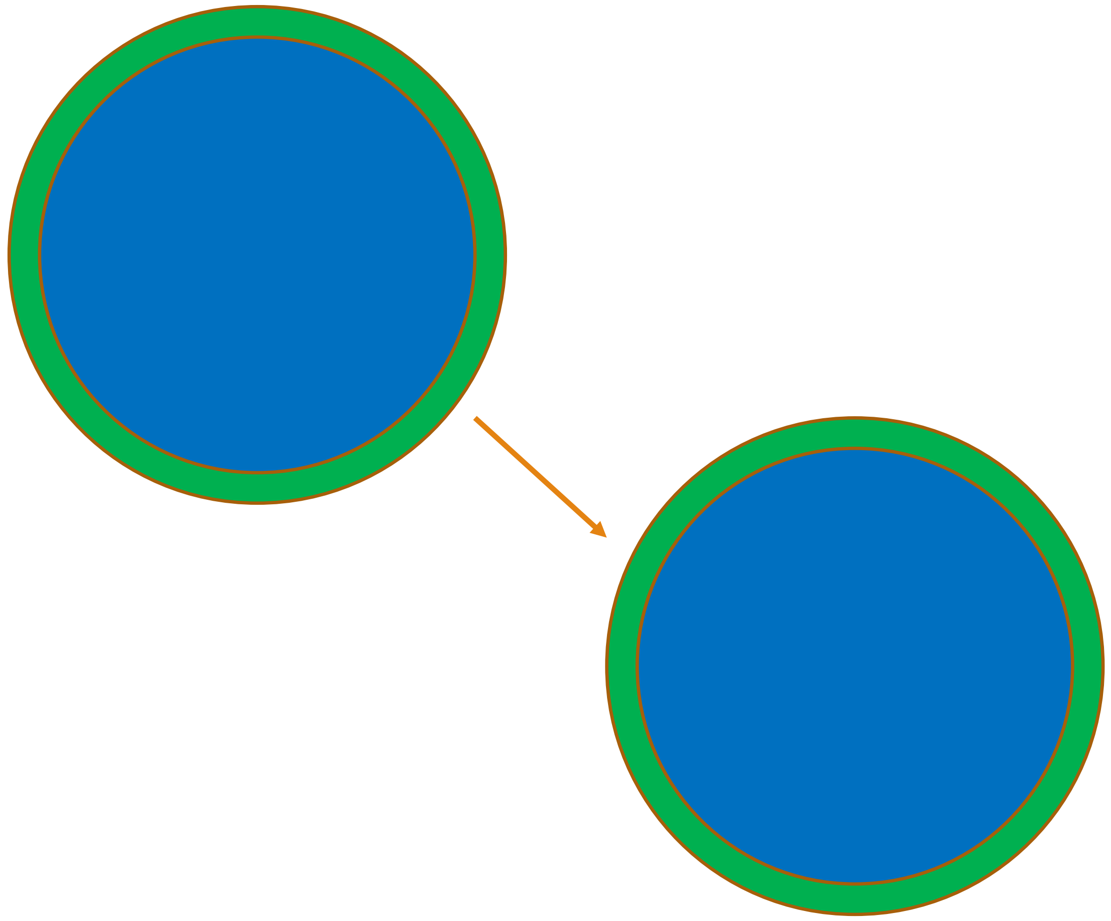
```
svg.selectAll("circle")
	.data(circle_position_data)
	.enter()
	.append("circle")
	.style("fill", "blue")
	.attr("r", 55)
	.attr("cx", function(d) { return d.x; })
	.attr("cy", function(d) { return d.y; })
	.call(d3.drag().on("drag", on_circle_drag));

function on_circle_drag(d) {
	d3.select(this)
		.attr("cx", d.x = d3.event.x)
		.attr("cy", d.y = d3.event.y);
 }
```

Exercise 5: Create a Second Draggable Green Circle Behind the Blue Circles

One possible implementation strategy:

1) Add an id variable to all of the data.
2) Use this id when naming the id attribute of the circles.
3) In the drag function for the inner circle, drag the corresponding outer circle (based on id) .



Exercise 5: Solution

```
circle_position_data.forEach(function(d, i) {
    d.i = i;
});

var g = svg.selectAll("circle")
    .data(circle_position_data)
    .enter()
    .append("g");
```

Exercise 5: Solution (Continued)

```
g.append("circle")
    .style("fill", "green")
    .attr("id", function(d) { return "circle_border" + d.i; })
    .attr("r", 60)
    .attr("cx", function(d) { return d.x; })
    .attr("cy", function(d) { return d.y; });

g.append("circle")
    .style("fill", "blue")
    .attr("r", 55)
    .attr("cx", function(d) { return d.x; })
    .attr("cy", function(d) { return d.y; })
    .call(d3.drag().on("drag", on_circle_drag));
```

Exercise 5: Solution (Continued)

```
function on_circle_drag(d, i) {
    d3.select(this)
        .attr("cx", d.x = d3.event.x)
        .attr("cy", d.y = d3.event.y);
    d3.select("#circle_border" + d.i)
        .attr("cx", d.x = d3.event.x)
        .attr("cy", d.y = d3.event.y);
}
```

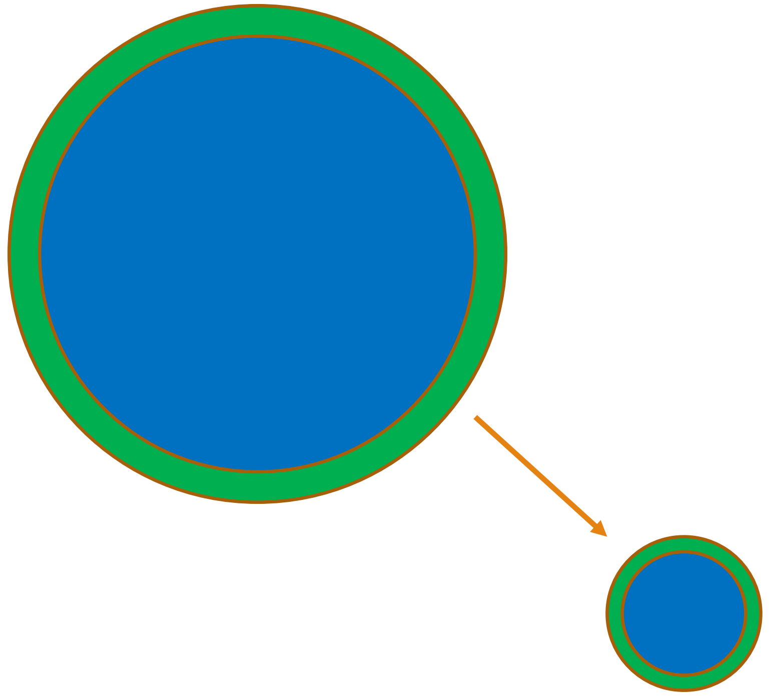
Exercise 6: Resize Circles In Real-Time When Dragging The Green Border

Similar implementation to dragging.

Think about the distance between the edge of the circle and the current mouse position.

Hint 1: Use the distance formula.

Hint 2: this.attributes.cx.value gives the position of the current selection and d3.event.x gives the x-position of the cursor (same for y).



Exercise 7: Set number of circles based on a moveable slider.

Best practice would be to write an update() function that handles exit, enter, and update cases.

Add the following HTML to the body of your page:

```
<input type="range" min="1" max="150" id="nCircles">
```