D3 Introduction

Jessica Hullman

Topics

Why Use D3
Getting started

Selections
Enter, update, exit pattern

[Some slides adapted from Mike Bostock's D3 Workshop]

Why Use D3

Visualization with Web Standards

Transformation, not representation (HTML, SVG)

Constructing a DOM from data

Benefits:
Expressivity
Debugging tools
Better documentation

Getting Started

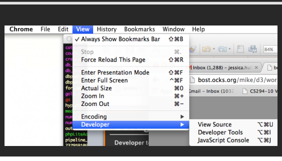
Running locally

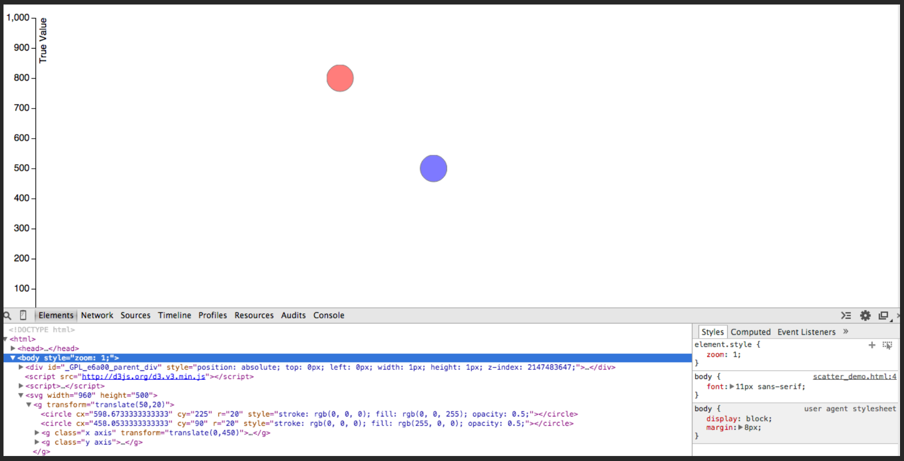
```
> python -m SimpleHTTPServer 8888 &
```

http://localhost:8888

Debugging tools

Developer tools



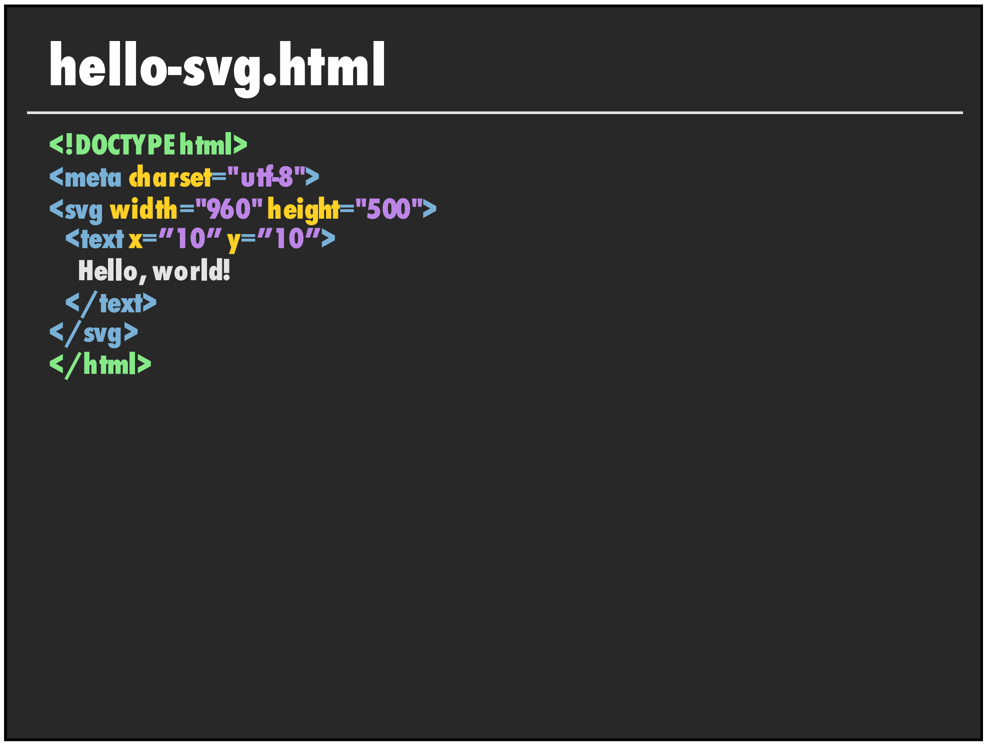

1,000
900
800
700
600
500
400
300
200
100
True Value
Elements Network Sources Timeline Profiles Resources Audits Console
Styles Computed Event Listeners »

```
<!DOCTYPE html>
▼<html>
 ▶<head>…</head>
 ▼<body style="zoom: 1;">
  ▶<div id="_GPL_e6a00_parent_div" style="position: absolute; top: 0px; left: 0px; width: 1px; height: 1px; z-index: 2147483647;">…</div>
   <script src="http://d3js.org/d3.v3.min.js"></script>
  ▶<script>…</script>
  ▼<svg width="960" height="500">
    ▼<g transform="translate(50,20)">
       <circle cx="598.6733333333333" cy="225" r="20" style="stroke: rgb(0, 0, 0); fill: rgb(0, 0, 255); opacity: 0.5;"></circle>
       <circle cx="458.0533333333333" cy="90" r="20" style="stroke: rgb(0, 0, 0); fill: rgb(255, 0, 0); opacity: 0.5;"></circle>
      ▶<g class="x axis" transform="translate(0,450)">…</g>
      ▶<g class="y axis">…</g>
     </g>
```

```
element.style {
   zoom: 1;
}
body {                    scatter_demo.html:4
   font:▶11px sans-serif;
}
body {                    user agent stylesheet
   display: block;
   margin:▶8px;
}
```

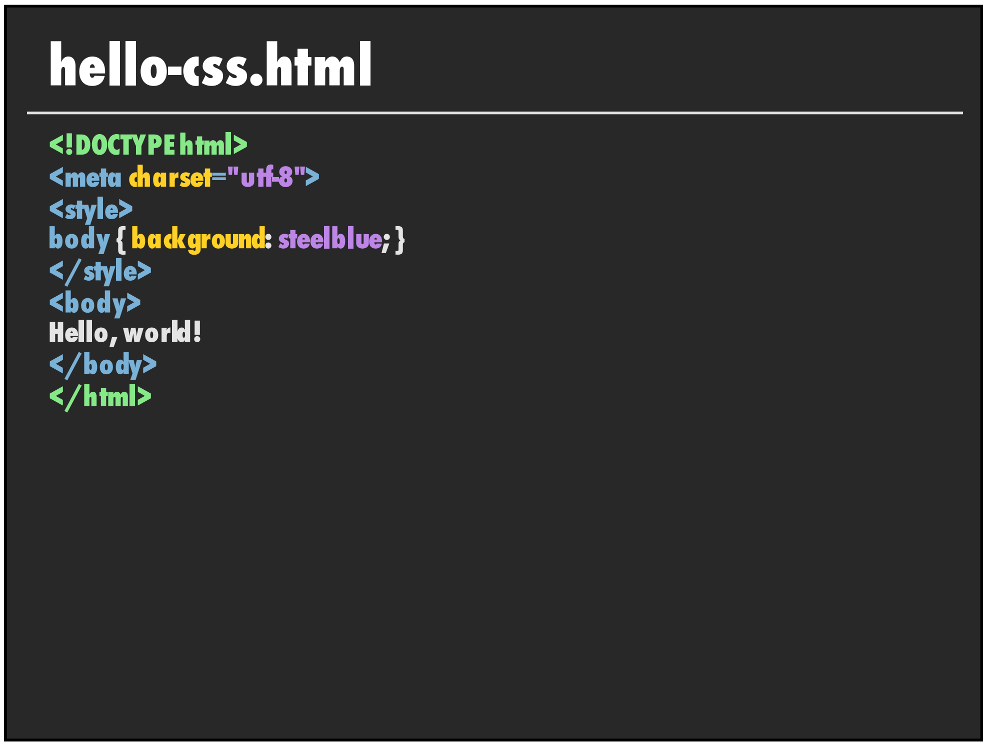
hello-world.html

```
<!DOCTYPE html>
<meta charset="utf-8">
<body>
Hello, world!
</body>
</html>
```



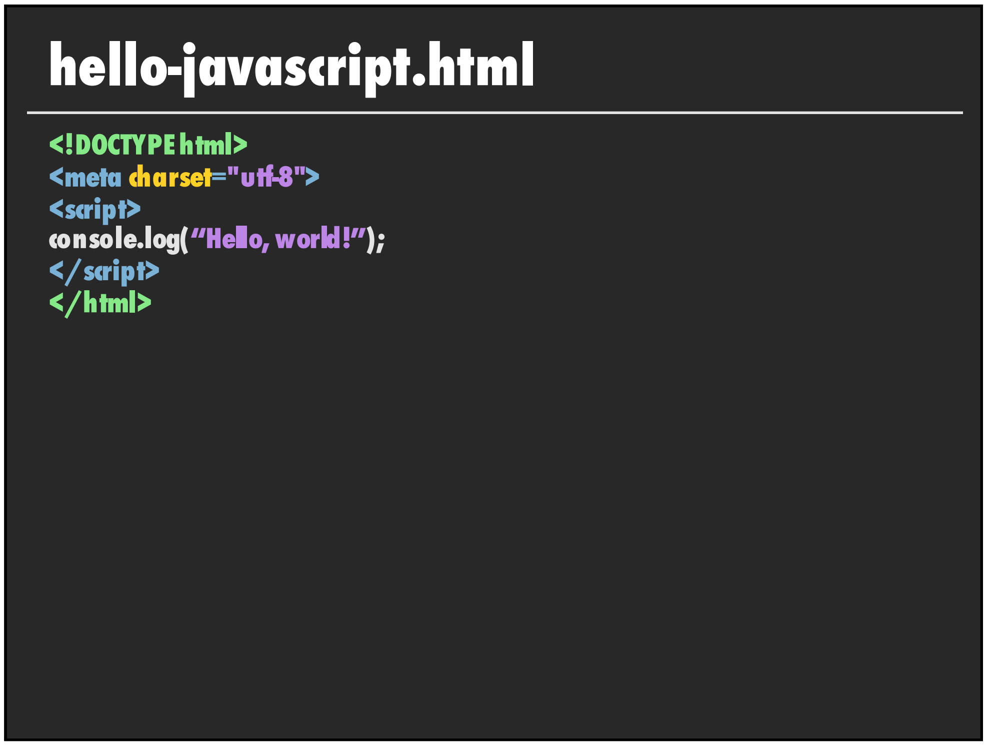
hello-svg.html

```
<!DOCTYPE html>
<meta charset="utf-8">
<svg width="960" height="500">
  <text x=”10” y=”10”>
    Hello, world!
  </text>
</svg>
</html>
```



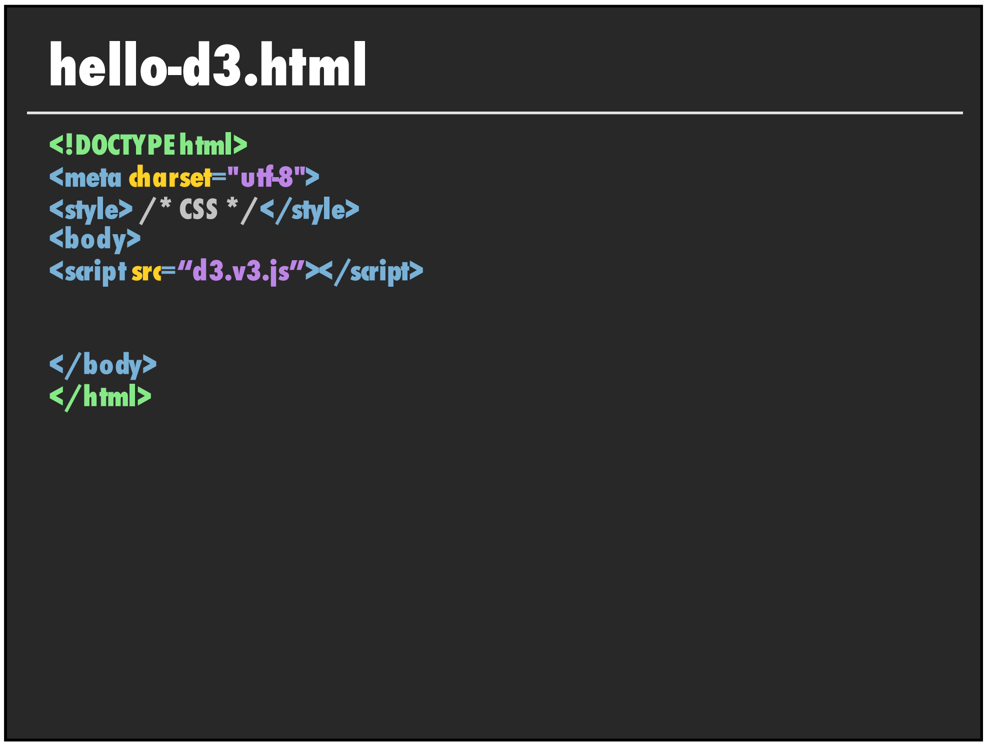
hello-css.html

```
<!DOCTYPE html>
<meta charset="utf-8">
<style>
body { background: steelblue; }
</style>
<body>
Hello, world!
</body>
</html>
```



hello-javascript.html

```
<!DOCTYPE html>
<meta charset="utf-8">
<script>
console.log(“Hello, world!”);
</script>
</html>
```

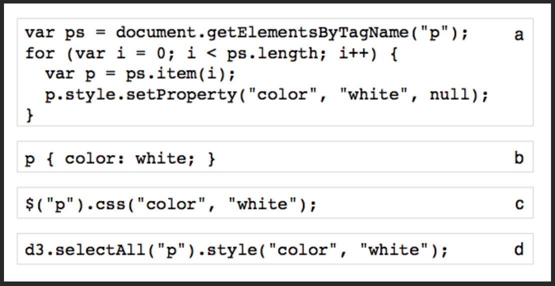


hello-d3.html

```
<!DOCTYPE html>
<meta charset="utf-8">
<style> /* CSS */</style>
<body>
<script src=“d3.v3.js”></script>


</body>
</html>
```

Selections

Operating on a selection



```
var ps = document.getElementsByTagName("p");
for (var i = 0; i < ps.length; i++) {
  var p = ps.item(i);
  p.style.setProperty("color", "white", null);
}
```
a

```
p { color: white; }
```
b

```
$("p").css("color", "white");
```
c

```
d3.selectAll("p").style("color", "white");
```
d

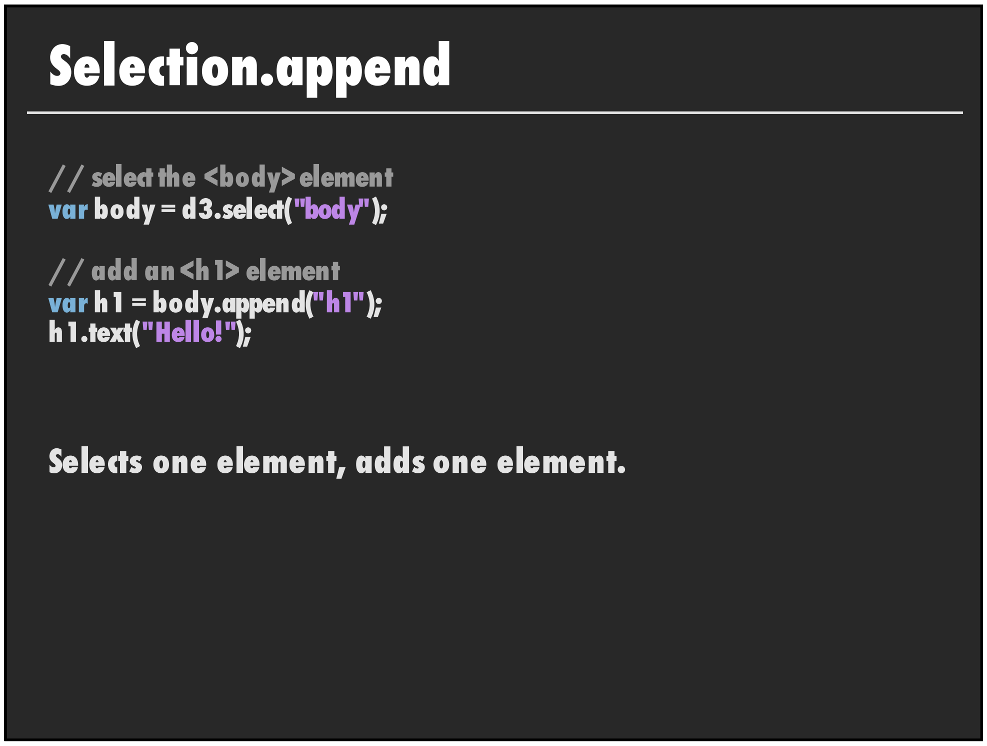
Selections in d3 are associated with operators to set properties.

Select SVG circles

```
//select all SVG circle elements
var circle=d3.selectAll("circle")


//set attributes and styles
circle.attr("cx", 20);
circle.attr("cy", 12);
circle.attr("r", 24);
circle.style("fill", "red");


//method chaining
d3.selectAll("circle")
    .attr("cx", 20)
    .attr("cy", 12)
    .attr("r", 24)
    .style("fill", "red");
```

Other basic shapes

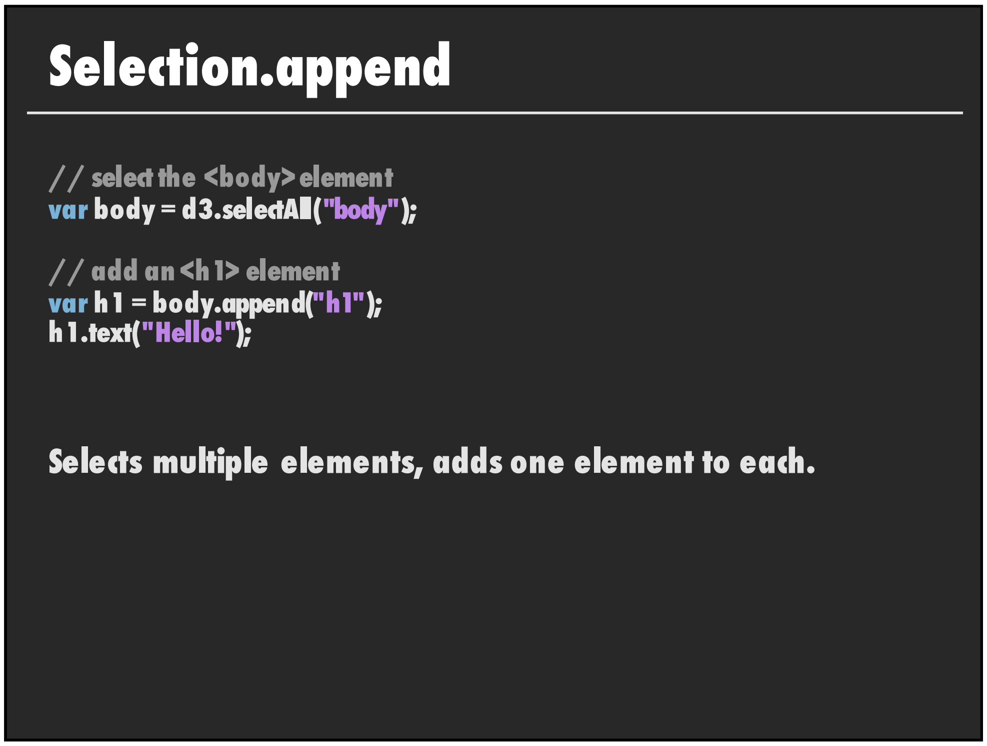
```
var rect = d3.selectAll("rect")
    .attr("x", 20)
    .attr("y", 12)
    .attr("width", 24)
    .attr("height", 24);

var line = d3.selectAll("line")
    .attr("x1", 20)
    .attr("y1", 12)
    .attr("x2", 40)
    .attr("y2", 24);

var text = d3.selectAll("text")
    .attr("x", 20)
    .attr("y", 12);
```



Selection.append

```
// select the <body> element
var body = d3.select("body");

// add an <h1> element
var h1 = body.append("h1");
h1.text("Hello!");
```

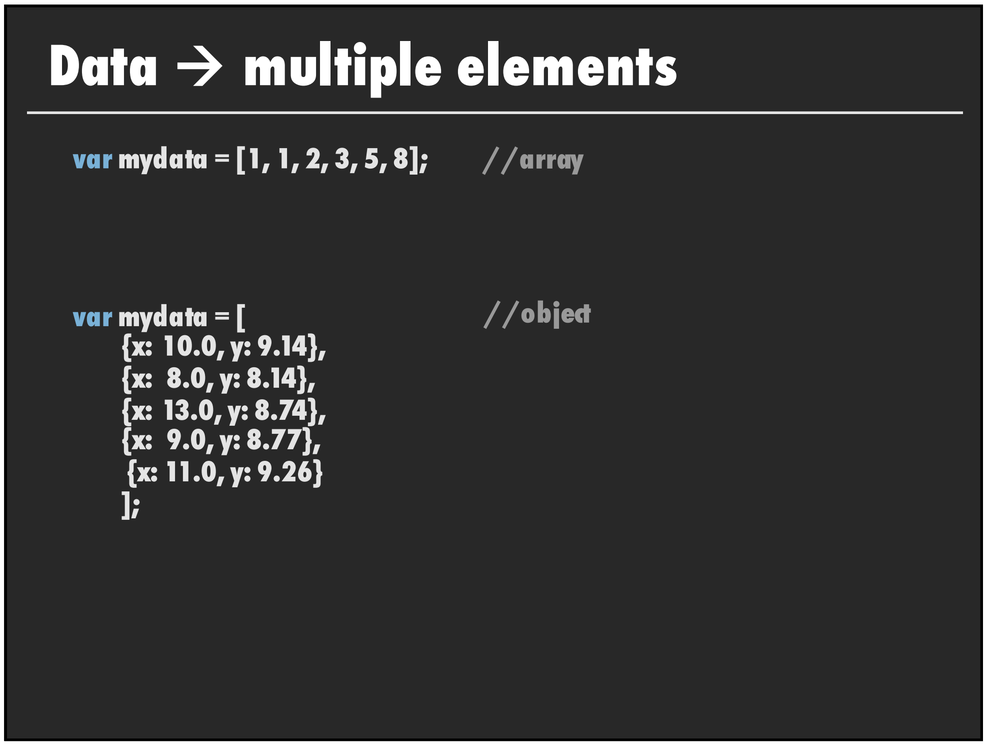
Selects one element, adds one element.



Selection.append

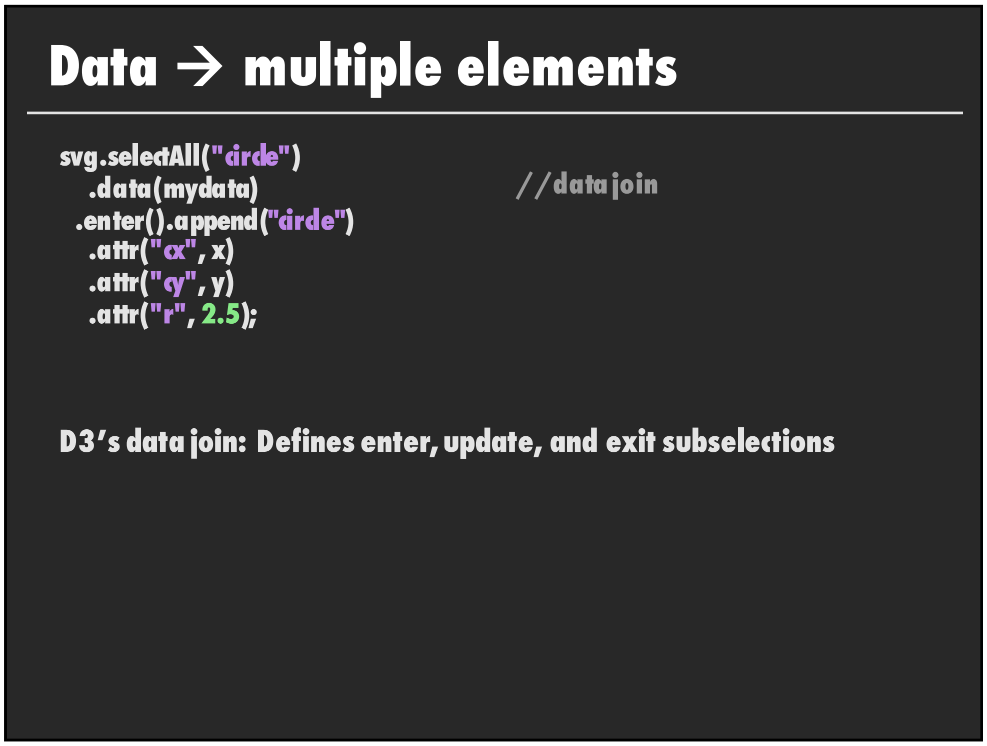
```
// select the <body> element
var body = d3.selectAll("body");

// add an <h1> element
var h1 = body.append("h1");
h1.text("Hello!");
```

Selects multiple elements, adds one element to each.



Data → multiple elements

```
var mydata = [1, 1, 2, 3, 5, 8];        //array



var mydata = [                          //object
      {x: 10.0, y: 9.14},
      {x:  8.0, y: 8.14},
      {x: 13.0, y: 8.74},
      {x:  9.0, y: 8.77},
       {x: 11.0, y: 9.26}
      ];
```

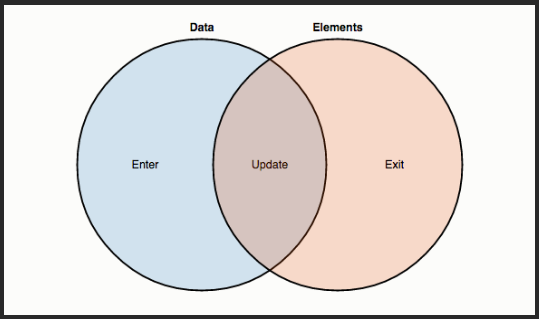


Data → multiple elements

```
svg.selectAll("circle")
   .data(mydata)                        //data join
  .enter().append("circle")
   .attr("cx", x)
   .attr("cy", y)
   .attr("r", 2.5);
```

D3's data join: Defines enter, update, and exit subselections

Data → multiple elements



3 selections:

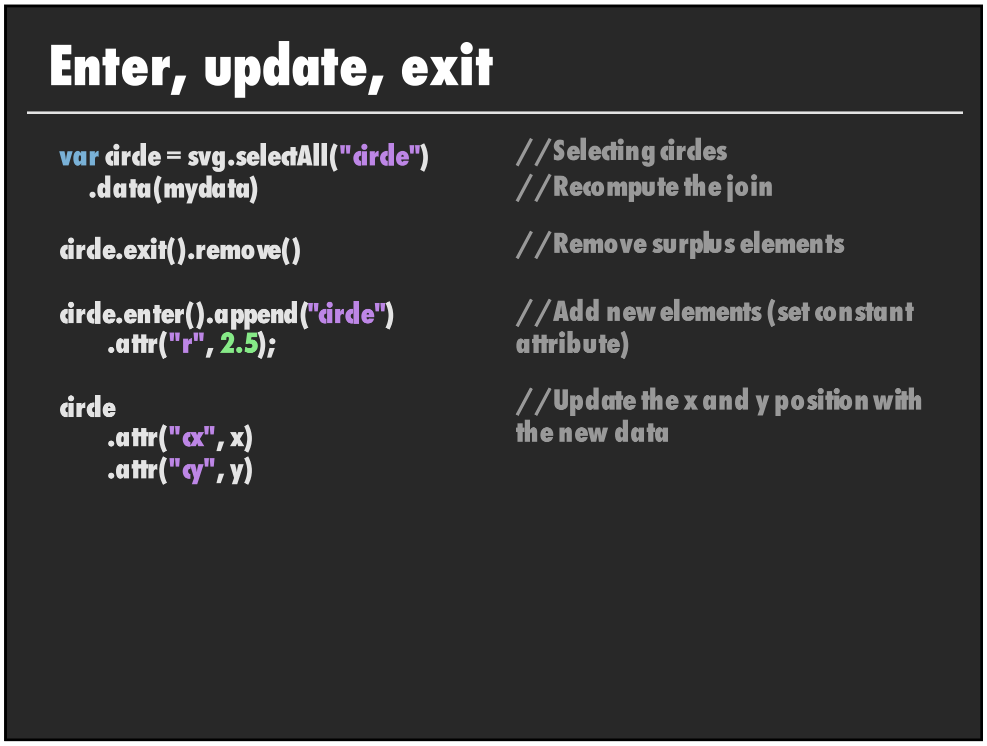
- *Enter*: Missing elements
- *Update*: Data points joined to existing elements
- *Exit*: Leftover unbound elements

Data → multiple elements

```
var circle = svg.selectAll("circle")      //Returns a new empty selection
   .data(mydata)                          //Join the selection to data: 3 new
                                          selections (enter, update, exit)

circle.enter().append("circle")           //Appending to the enter selection
     .attr("cx", x)                       adds the missing elements to the
     .attr("cy", y)                       SVG container
     .attr("r", 2.5);
```

Accessor functions: `function x(d) { return d.x; }`

Why joins?



Enter, update, exit

```
var circle = svg.selectAll("circle")       //Selecting circles
   .data(mydata)                           //Recompute the join

circle.exit().remove()                     //Remove surplus elements

circle.enter().append("circle")            //Add new elements (set constant
      .attr("r", 2.5);                     attribute)

circle                                     //Update the x and y position with
      .attr("cx", x)                       the new data
      .attr("cy", y)
```