DATA DRIVEN DOCUMENTS

It's just a toolkit…

a really powerful toolkit, but you still have
to do all of the programming and design.

LEARNING OBJECTIVES

- Gain familiarity with d3.js web programming operations & toolkit
- Enter, Update, Exit

HW 3

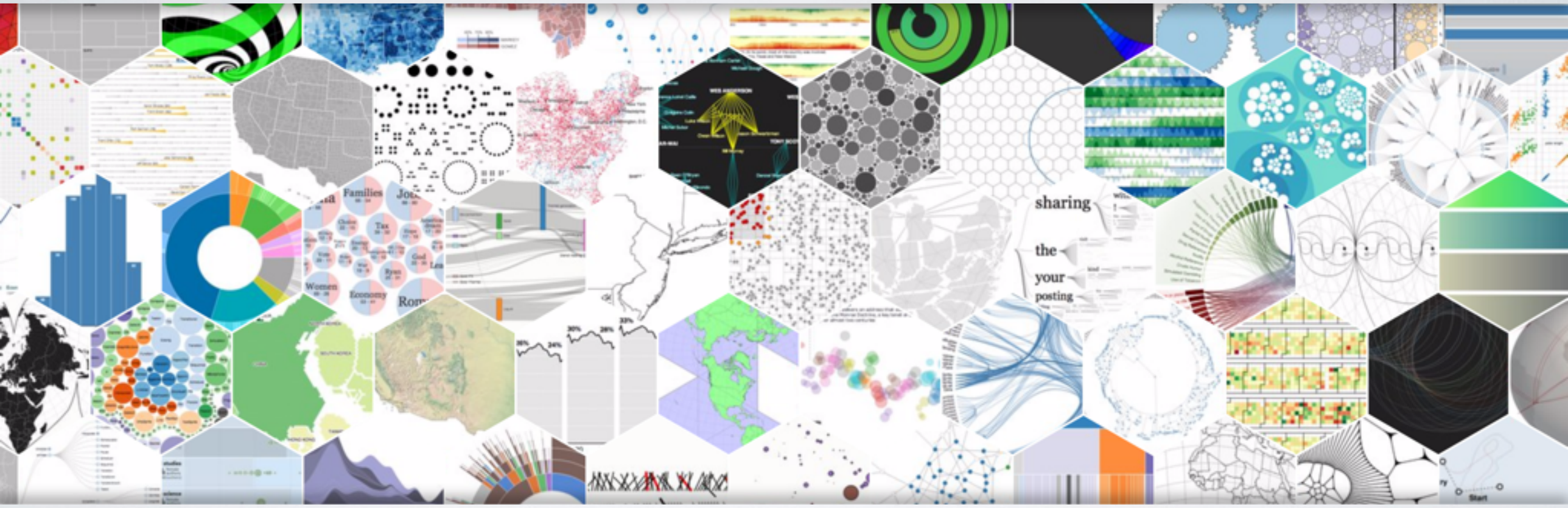
- Three Options
  - d3.js version
  - processing version
  - hand-drawn

DATA DRIVEN DOCUMENTS

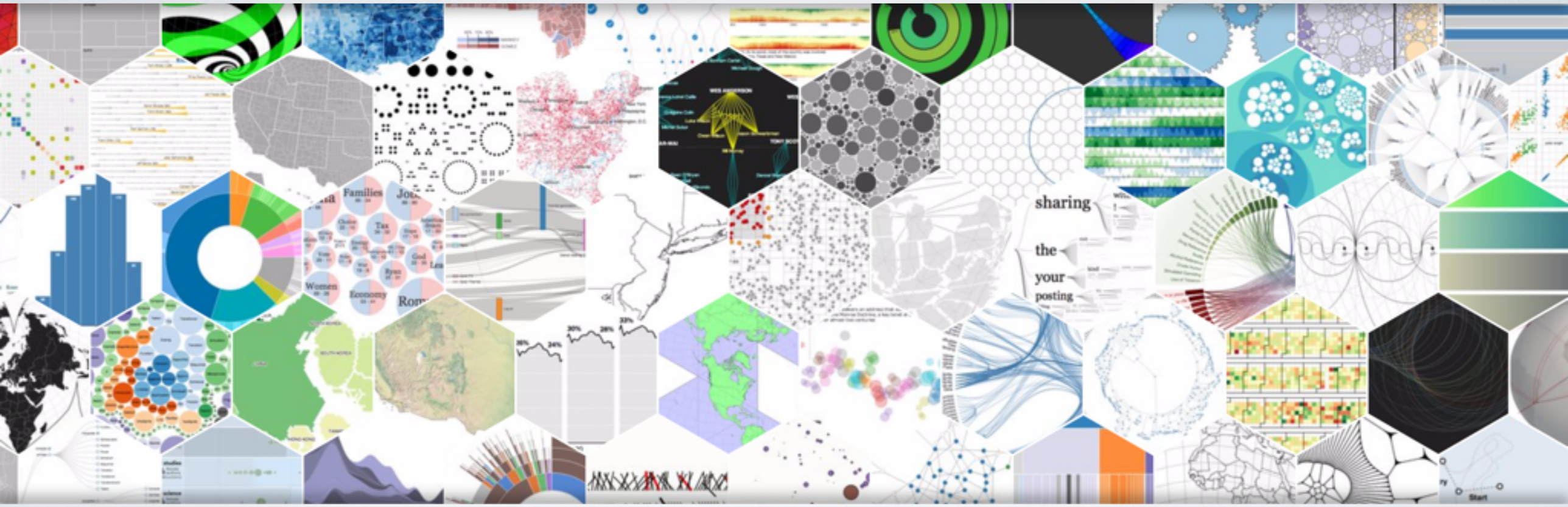
It's just a toolkit…

a really powerful toolkit, but you still have
to do all of the programming and design.

It’s just a toolkit…



It's just a toolkit…



a really powerful toolkit, but you still have to do all of the programming and design.

WHAT IS D3?

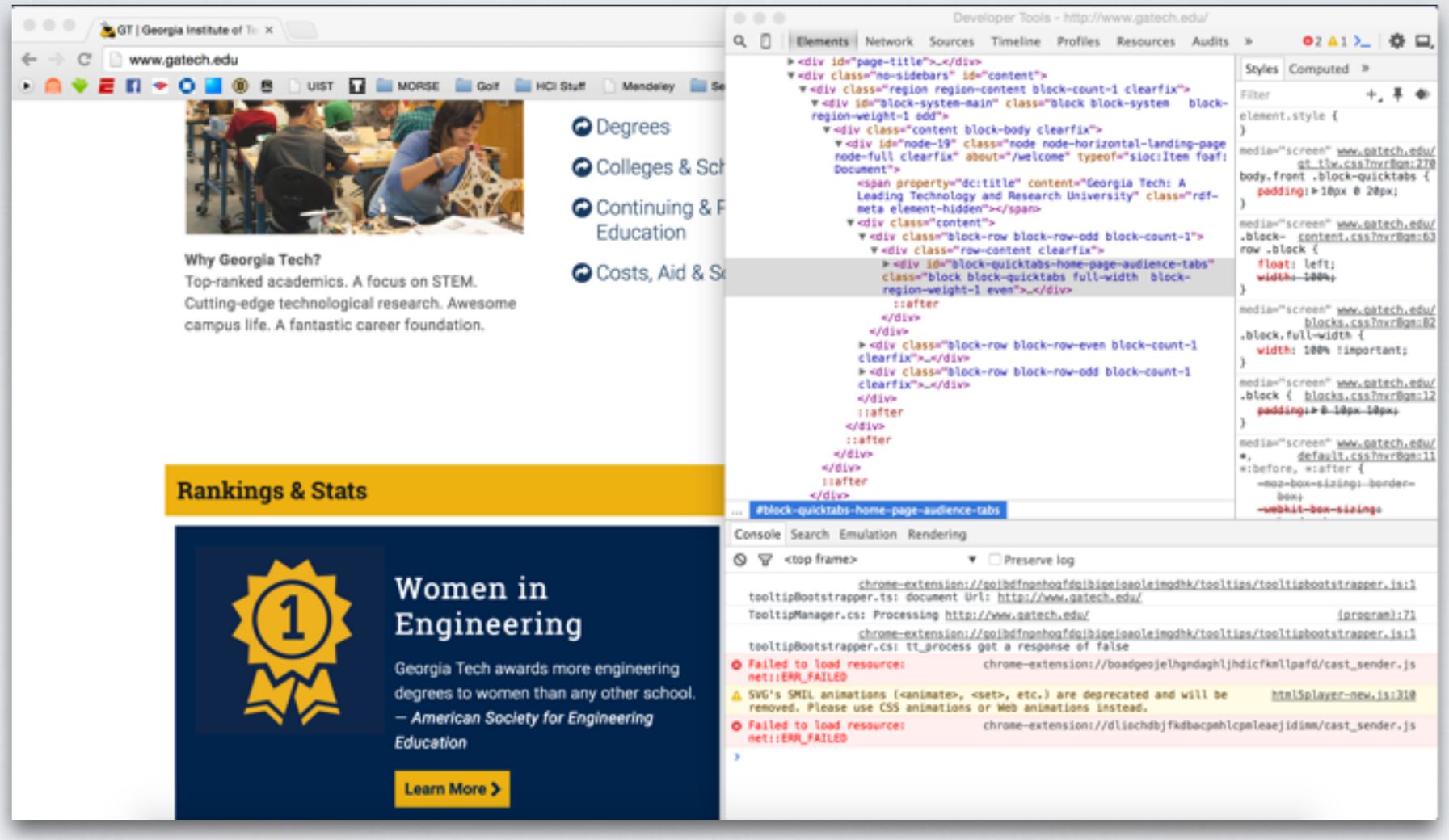
- A javascript visualization toolkit
- Binds HTML elements to input data
- Great helper functions for:
  - layout (force-directed, treemap, bundle, radial, geographic, … and many more)
  - scales & axes
  - interaction (drag, zoom & pan)
  - data loading (csv, json, tsv)
  - sort and filter data
  - …and many more functions [check out the API]

WHAT IT ISN'T

- D3 does not create visualizations for you
- Use Tableau or Spotfire if you want automatically generated charts
- D3 assists you in creating visualizations but does not specify the visual encodings for data

HTML PRIMER

- Open up web browser - go to gatech.edu
- Inspect element by Right-clicking anywhere & select inspect element



Chrome

On Mac **⌘ + Shift + C.**

On Windows **Ctrl + Shift + C.**

Firefox

On Mac **⌘ + Opt + C.**

On Windows **Ctrl + Shift + C.**

CSS PRIMER

- classes specify styling attributes for HTML elements (dot indicates class)

```
.bar {
   fill: steelblue;
}
```

CSS PRIMER

- classes specify styling attributes for HTML elements (dot indicates class)

```
.bar {
   fill: steelblue;
}
```

- specify styling for html element types (no prefix indicates html type)

```
circle {
   stroke: black;
}
```

CSS PRIMER

- classes specify styling attributes for HTML elements (dot indicates class)

```
.bar {
   fill: steelblue;
}
```

- specify styling for html element types (no prefix indicates html type)

```
circle {
   stroke: black;
}
```

- combine class and html types in css as well (space means a child of that element)

```
.axis text {
   font-size: 9px;
}
```

JAVASCRIPT PRIMER

- variables point to objects, arrays or primitives (number, boolean, string)

```
var imageHeight = 70;
var arrayExample = [0, 1 , 2 , 4];
var objectExample = {name: 'John', year: '2nd'};
```

JAVASCRIPT PRIMER

- variables point to objects, arrays or primitives (number, boolean, string)

```
var imageHeight = 70;
var arrayExample = [0, 1 , 2 , 4];
var objectExample = {name: 'John', year: '2nd'};
```

- objects have methods, they can be chained together for d3

```
objectExample.exampleMethod(inputValue).appendMethod()
```

JAVASCRIPT PRIMER

- variables point to objects, arrays or primitives (number, boolean, string)

```
var imageHeight = 70;
var arrayExample = [0, 1 , 2 , 4];
var objectExample = {name: 'John', year: '2nd'};
```

- objects have methods, they can be chained together for d3

```
objectExample.exampleMethod(inputValue).appendMethod()
```

- call backs a declared and called at a later point in time

```
objectExample.onEvent('mouseclick',
   function(event){
      event.source.highlight(true);
});
```

CSS & JAVASCRIPT

- Class = style classification for html elements
  - specify class in d3 using: `.attr('class', 'bar')`
  - select all bar classified objects with: `d3.selectAll('.bar')`

CSS & JAVASCRIPT

- Class = style classification for html elements
  - specify class in d3 using: `.attr('class', 'bar')`
  - select all bar classified objects with: `d3.selectAll('.bar')`

- ID = unique identifier for *a single* html element in page
  - specify the id using d3 using: `.attr('id', 'bar-0')`
  - select all bar classified objects with: `d3.select('#bar-0')`

CSS & JAVASCRIPT

- Class = style classification for html elements
  - specify class in d3 using: `.attr('class', 'bar')`
  - select all bar classified objects with: `d3.selectAll('.bar')`

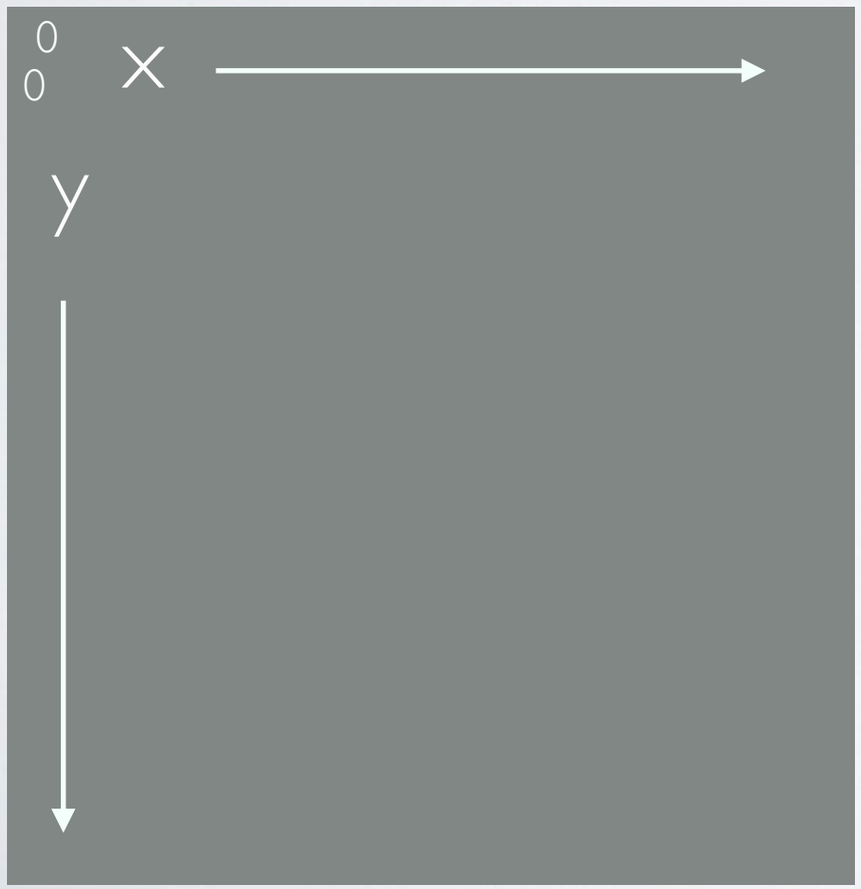
- ID = unique identifier for *a single* html element in page
  - specify the id using d3 using: `.attr('id', 'bar-0')`
  - select all bar classified objects with: `d3.select('#bar-0')`

- Change style or attributes directly in javascript once selection is made

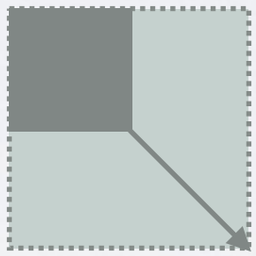
```
d3.select('#bar-0').attr('fill', 'red');
```

SVG PRIMER

- SVG = Scalable Vector Graphics
  - create primitive shapes [rect, circle], path, text, or image
  - look at cheatsheet or on W3C for specifications
- Coordinate System



- Scaling

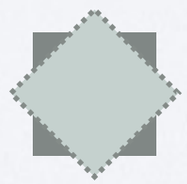


```
scale(2,2)
```

- Translate

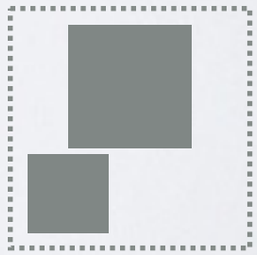
```
translate(2,0)
```

- Rotate



```
rotate(45)
```

- Groups



```
<g></g>
```

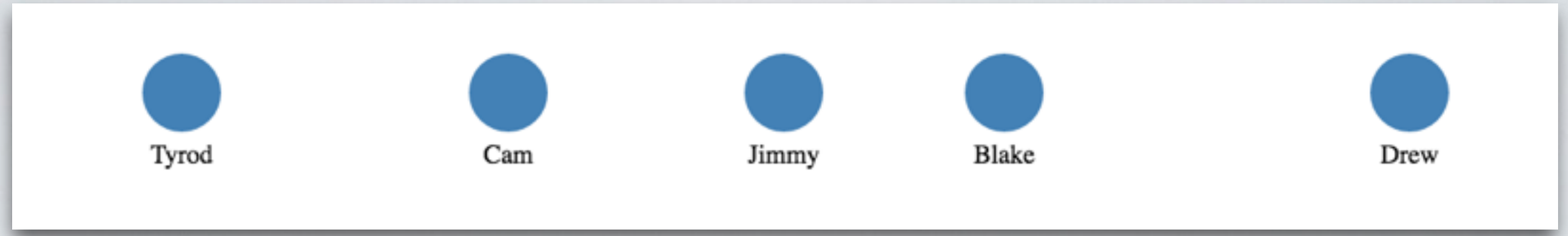
HOW DOES D3 TIE IN?

- D3 binds data to HTML elements
- D3 is a declarative language
  - allows you to declare how the data will be visually encoded
  - how interactions will be handled
- Binding allows you to make selections to add new data, remove old data or update data representations

HOW DOES D3 TIE IN?

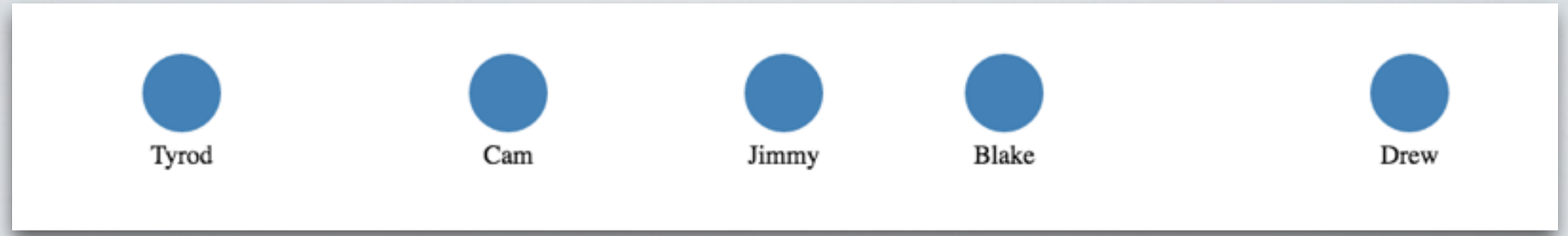
- D3 binds data to HTML elements
- D3 is a declarative language
  - allows you to declare how the data will be visually encoded
  - how interactions will be handled
- Binding allows you to make selections to add new data, remove old data or update data representations
- d3.data() is where the magic happens

WITHOUT D3




```
<svg id="vis" width="720" height="500">
  <g transform="translate(634.5,100)" class="node">
      <circle r="15"></circle>
      <text dy="2.7em">Drew</text>
      </g>
  <g transform="translate(396,100)" class="node">
      <circle r="15"></circle>
      <text dy="2.7em">Jimmy</text>
  </g>
  <g transform="translate(166.5,100)" class="node">
      <circle r="15"></circle>
      <text dy="2.7em">Tyrod</text>
  </g>
  <g transform="translate(480,100)" class="node">
      <circle r="15"></circle>
      <text dy="2.7em">Blake</text>
  </g>
  <g transform="translate(291,100)" class="node">
      <circle r="15"></circle>
      <text dy="2.7em">Cam</text>
  </g>
</svg>
```

WITH D3




```
var svg = d3.select(“#vis”);

var circle = svg.selectAll(“circle")
  .data(datum.concat(add),function(d){return d.name;});

var circleG = circle.enter().append('g')
  .attr('transform',function(d){return ‘translate('+
  (d.val*1.5)+',100)';}).attr('class','node');

circleG.append(‘circle').attr('r',15);

circleG.append(‘text').attr('dy','2.7em')
  .text(function(d){return d.name;});
```

STEP-BY-STEP

<g> <g> <g> <g> <g>

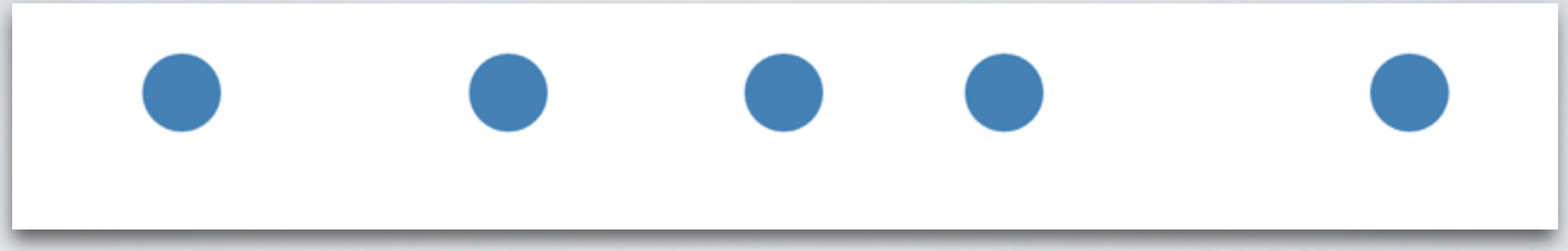
```
var svg = d3.select(“#vis”);

var circle = svg.selectAll(“circle")
  .data(datum.concat(add),function(d){return d.name;});

var circleG = circle.enter().append('g')
  .attr('transform',function(d){return ‘translate('+
  (d.val*1.5)+',100)';}).attr('class','node');

circleG.append(‘circle').attr('r',15);

circleG.append(‘text’).attr('dy','2.7em')
  .text(function(d){return d.name;});
```

STEP-BY-STEP

<g> <g> <g> <g> <g>

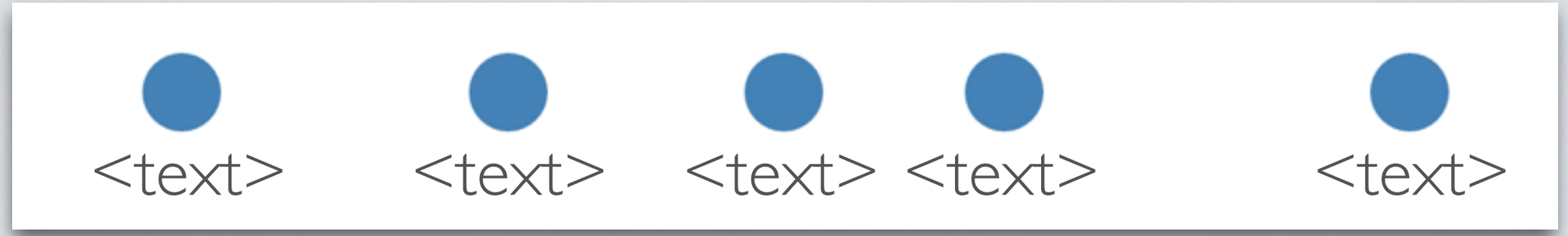
```
var svg = d3.select(“#vis”);

var circle = svg.selectAll(“circle")
  .data(datum.concat(add),function(d){return d.name;});

var circleG = circle.enter().append('g')
  .attr('transform',function(d){return ‘translate('+
  (d.val*1.5)+',100)';}).attr('class','node');

circleG.append(‘circle').attr('r',15);

circleG.append(‘text').attr('dy','2.7em')
  .text(function(d){return d.name;});
```

STEP-BY-STEP



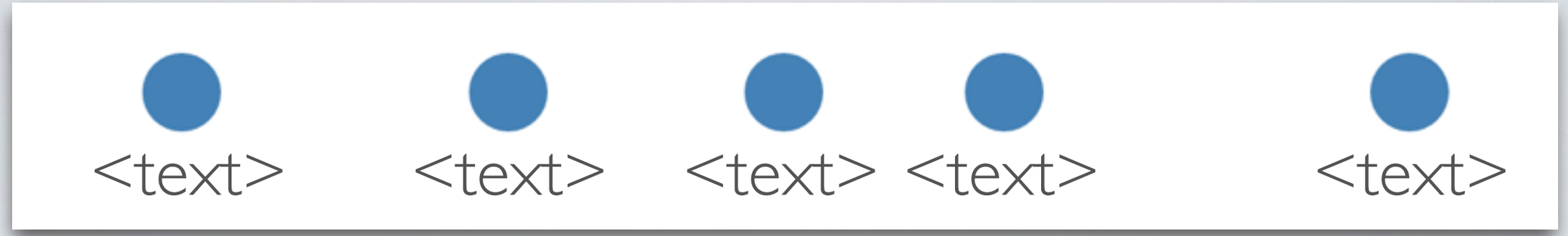
```
var svg = d3.select(“#vis”);

var circle = svg.selectAll(“circle")
  .data(datum.concat(add),function(d){return d.name;});

var circleG = circle.enter().append('g')
  .attr('transform',function(d){return ‘translate('+
  (d.val*1.5)+',100)';}).attr('class','node');

circleG.append(‘circle').attr('r',15);

circleG.append(‘text').attr('dy','2.7em')
  .text(function(d){return d.name;});
```

STEP-BY-STEP



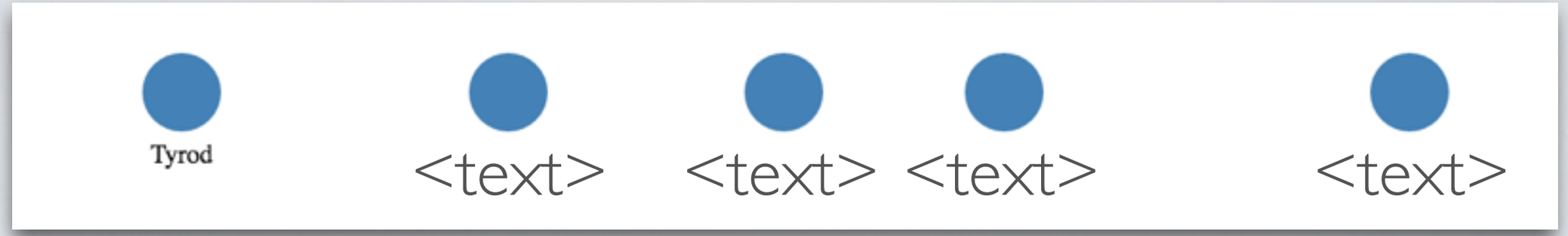

```
var svg = d3.select(“#vis”);

var circle = svg.selectAll(“circle")
  .data(datum.concat(add),function(d){return d.name;});

var circleG = circle.enter().append('g')
  .attr('transform',function(d){return ‘translate('+
  (d.val*1.5)+',100)';}).attr('class','node');

circleG.append(‘circle').attr('r',15);

circleG.append(‘text').attr('dy','2.7em')
  .text(function(d){return d.name;});
```

STEP-BY-STEP



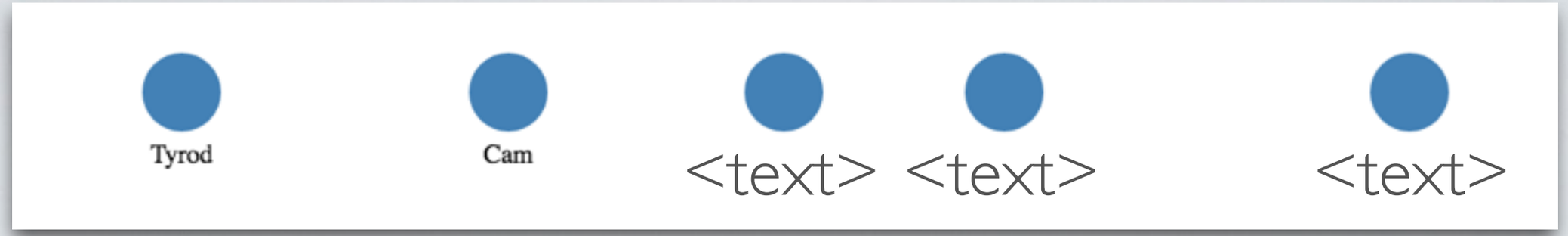

```
var svg = d3.select(“#vis”);

var circle = svg.selectAll(“circle")
  .data(datum.concat(add),function(d){return d.name;});

var circleG = circle.enter().append('g')
  .attr('transform',function(d){return ‘translate('+
  (d.val*1.5)+',100)';}).attr('class','node');

circleG.append(‘circle').attr('r',15);

circleG.append(‘text’).attr('dy','2.7em')
  .text(function(d){return d.name;});
```

STEP-BY-STEP



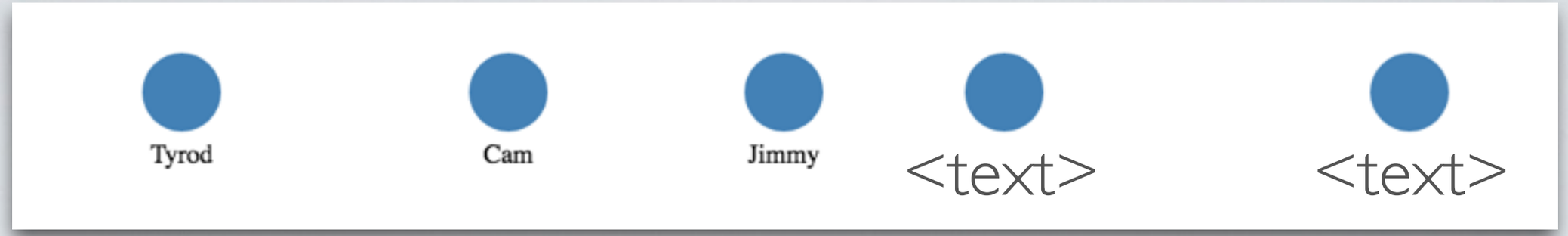

```
var svg = d3.select(“#vis”);

var circle = svg.selectAll(“circle")
  .data(datum.concat(add),function(d){return d.name;});

var circleG = circle.enter().append('g')
  .attr('transform',function(d){return ‘translate('+
  (d.val*1.5)+',100)';}).attr('class','node');

circleG.append(‘circle').attr('r',15);

circleG.append(‘text’).attr('dy','2.7em')
  .text(function(d){return d.name;});
```

STEP-BY-STEP



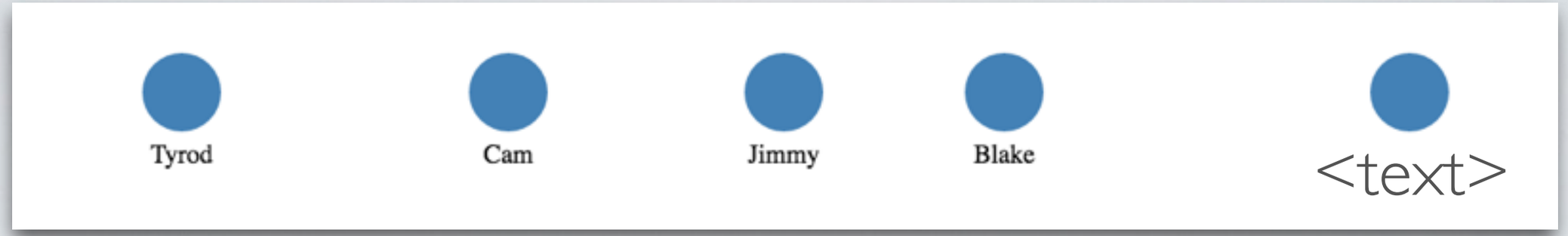

```
var svg = d3.select(“#vis”);

var circle = svg.selectAll(“circle")
  .data(datum.concat(add),function(d){return d.name;});

var circleG = circle.enter().append('g')
  .attr('transform',function(d){return ‘translate('+
  (d.val*1.5)+',100)';}).attr('class','node');

circleG.append(‘circle').attr('r',15);

circleG.append(‘text’).attr('dy','2.7em')
  .text(function(d){return d.name;});
```

STEP-BY-STEP



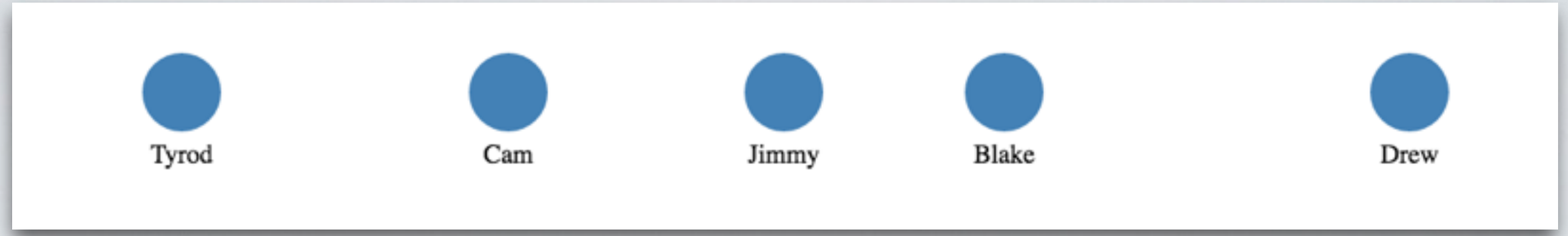

```
var svg = d3.select(“#vis”);

var circle = svg.selectAll(“circle")
  .data(datum.concat(add),function(d){return d.name;});

var circleG = circle.enter().append('g')
  .attr('transform',function(d){return ‘translate('+
  (d.val*1.5)+',100)';}).attr('class','node');

circleG.append(‘circle').attr('r',15);

circleG.append(‘text’).attr('dy','2.7em')
  .text(function(d){return d.name;});
```

STEP-BY-STEP




```
var svg = d3.select(“#vis”);

var circle = svg.selectAll(“circle")
  .data(datum.concat(add),function(d){return d.name;});

var circleG = circle.enter().append('g')
  .attr('transform',function(d){return ‘translate('+
  (d.val*1.5)+',100)';}).attr('class','node');

circleG.append(‘circle').attr('r',15);

circleG.append(‘text’).attr('dy','2.7em')
  .text(function(d){return d.name;});
```

STEP-BY-STEP




```
var svg = d3.select(“#vis”);

var circle = svg.selectAll(“circle")
  .data(datum.concat(add),function(d){return d.name;});

var circleG = circle.enter().append('g')
  .attr('transform',function(d){return ‘translate('+
  (d.val*1.5)+',100)';}).attr('class','node');

circleG.append(‘circle').attr('r',15);

circleG.append(‘text’).attr('dy','2.7em')
  .text(function(d){return d.name;});
```

DATA JOIN

- Given array of names
- Select the SVG with id ‘vis’
- Select all old elements classed html ‘name’ (currently none)
- Join new data to create a new selection

```
var names = [‘John’, ‘Arjun’,
‘Chad’, ‘Alex’, ‘Bahador’];

var updateN = d3.select(‘#vis’)
  .selectAll(‘.name’).data(names);
```

- UPDATE

DATA FUNCTION

- UPDATE
- update old elements as needed

```
var names = [‘John’, ‘Arjun’,
‘Chad’, ‘Alex’, ‘Bahador’];

var updateN = d3.select(‘#vis’)
  .selectAll(‘.name’).data(names);
```

ENTER & APPEND

- Now we add new html elements
- ENTER
- all elements without a bound html element (all in this case)
- APPEND
- add element with tag type, in this case ‘text’
- .text() - specify the content
- function(d){} is a callback that passes each data case

```
var names = [‘John’, ‘Arjun’, ‘Chad’, ‘Alex’, ‘Bahador’];

var updateN = d3.select(‘#vis’)
   .selectAll(‘.name’).data(names);

var enterN = updateN.enter()
   .append(‘text’)
   .text(function(d){return d;});
```

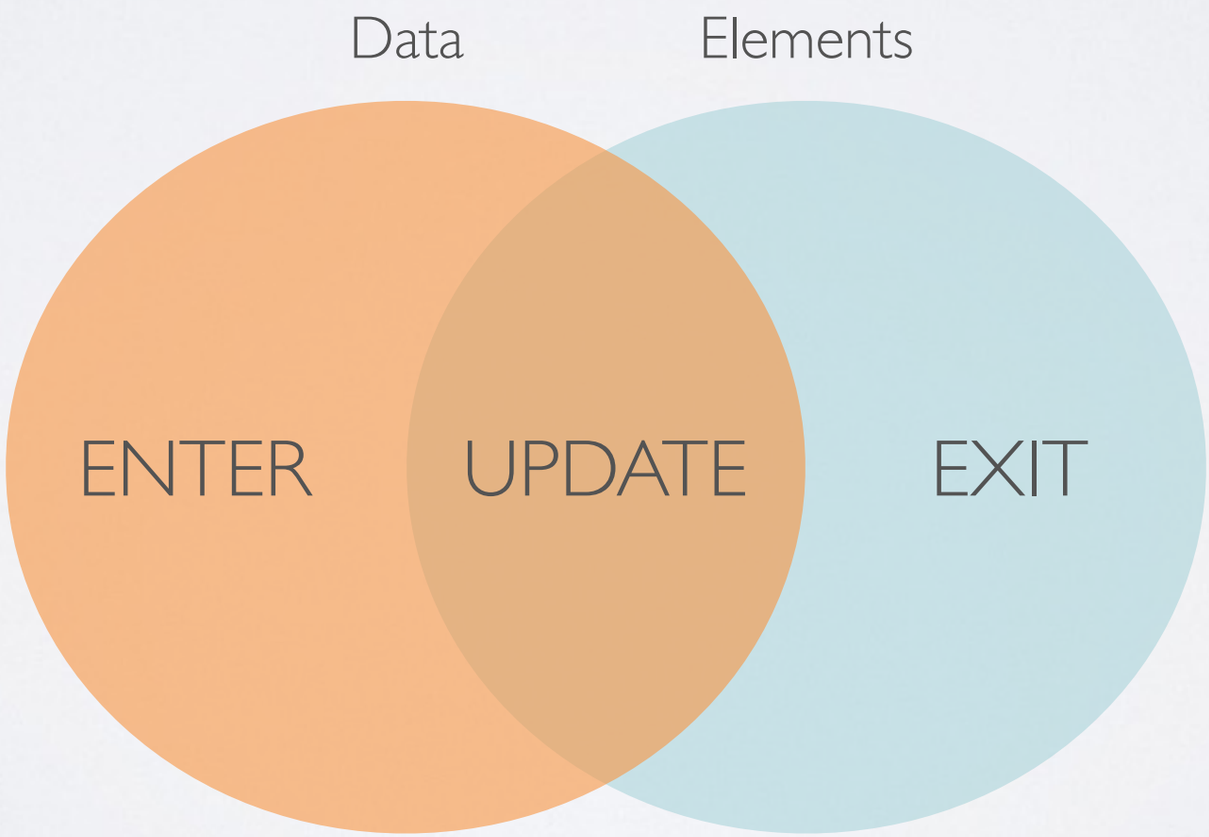
UPDATE?

- I thought update selection had all of the elements
- It does now!
- ENTER + UPDATE
- use merge() to combine selections
- Any changes we want to consistently make to new and already existing elements happens here
- Move the text element to the right position
- i = index of data element

```
var names = [‘John’, ‘Arjun’,
‘Chad’, ‘Alex’, ‘Bahador’];

var updateN = d3.select(‘#vis’)
   .selectAll(‘.name’).data(names);

var enterN = updateN.enter()
   .append(‘text’)
   .text(function(d){return d;})
   merge(updateN)
   .attr(‘x’,function(d,i){return
   i*100;});
```

THERE'S ALSO EXIT

- Exit selects all elements that are no longer bound to data in the array
- ENTER - UPDATE - EXIT
  - allows us to add new data, update representation, remove old data



D3 HELPER FUNCTIONS

- Load Data

```
d3.csv(“.data.csv”,
function(error,datum){
 if(error) return;
 else //use data here
})
```

Loads data asynchronously

IMPORTANT:
Use callback scope!

- Linear Scale

```
d3.scaleLinear()
 .domain([0,max])
 .range([height,0]);
```

Converts [0,…,max] to [height,…,0]

- Ordinal Scale

```
d3.scaleOrdinal()
 .domain(bookNames)
 .rangeRoundBands(
 [0,width],1);
```

Converts book name to [20,…,width-20]

- Axis

```
d3.axisLeft(yAxis);
```

Creates y axis from linear scale

- Force-Directed Layout

```
d3.forceSimulation()
 .nodes(chars)
 .links(book.links)
 .size([w,h])
 .start();
```

Creates a force layout based on nodes and links.

Basically an animation.

QUICK NOTE

- d3.js version 4 released this summer
- main changes from v3:

QUICK NOTE

- d3.js version 4 released this summer
- main changes from v3:
  - Namespaces:

	v3	v4
Linear Scale	d3.linear.scale()	d3.linearScale()
Axis	d3.svg.axis().orient('left) .scale(yScale)	d3.axisLeft(yScale)
Force Layout	d3.layout.force()	d3.forceSimulation()

QUICK NOTE

- d3.js version 4 released this summer
- main changes from v3:
  - Namespaces:

	v3	v4
Linear Scale	d3.linear.scale()	d3.linearScale()
Axis	d3.svg.axis().orient('left) .scale(yScale)	d3.axisLeft(yScale)
Force Layout	d3.layout.force()	d3.forceSimulation()

- selection-merge - not present in v3

THREE LITTLE CIRCLES

LEARNING OBJECTIVES

- Gain familiarity with d3.js web programming operations & toolkit
- Enter, Update, Exit

VISUALIZATION OF THE DAY

- First person id up today
- Instructions on website, details on t-square
- Find which day you are assigned to

PROJECT

- Teams are set?
- Hopefully have good topics

- Proposal due Wednesday
- Follow the directions
- **Bring 3 copies**

READING

- If you're doing d3.js, look over the websites linked on our class pages for today

UPCOMING

- Multivariate Visual Representations 1 & 2