EmberJS & D3.js

Data visualization

Summary

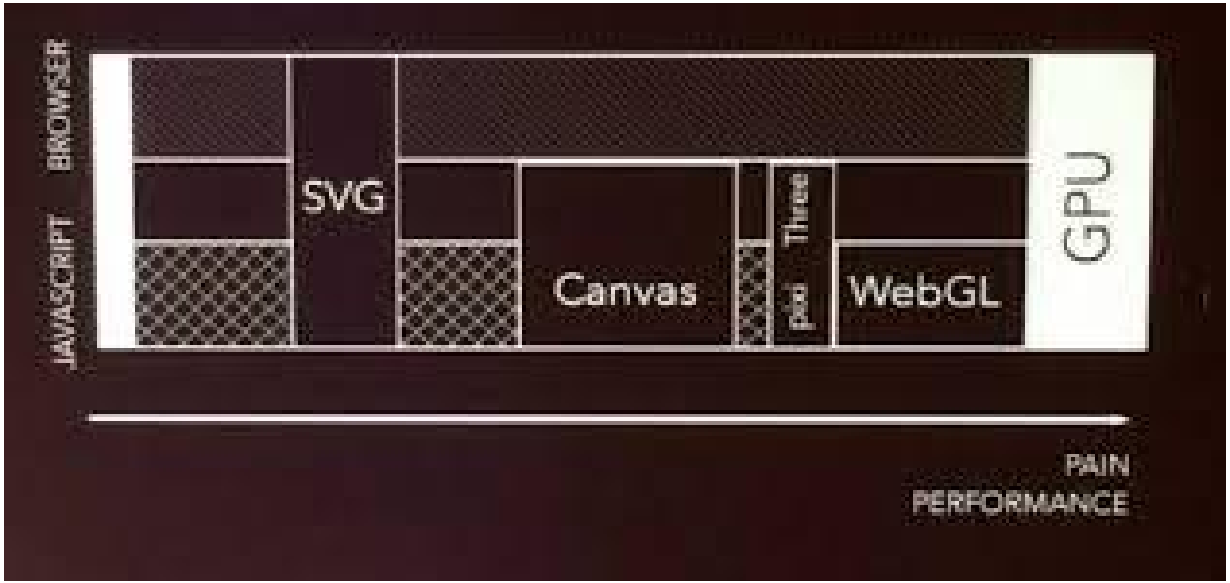
- Web graphics technologies
- SVG element in HTML
- D3.js
- EmberJS with D3

Graphics technologies

There are 3 primary HTML5 graphics APIs:

- SVG
- Canvas
- WebGL

Graphics technologies



Basic SVG elements

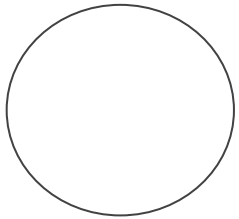
<rect/>

x: x-coordinate of top-left

y: y-coordinate of top-left

width

height



<circle/>

cx: x-coordinate of center

cy: y-coordinate of center

r: radius

HI!

<text></text>

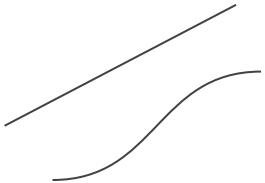
x: x-coordinate

y: y-coordinate

dx: x-coordinate offset

dy: y-coordinate offset

text-anchor: horizontal text alignment

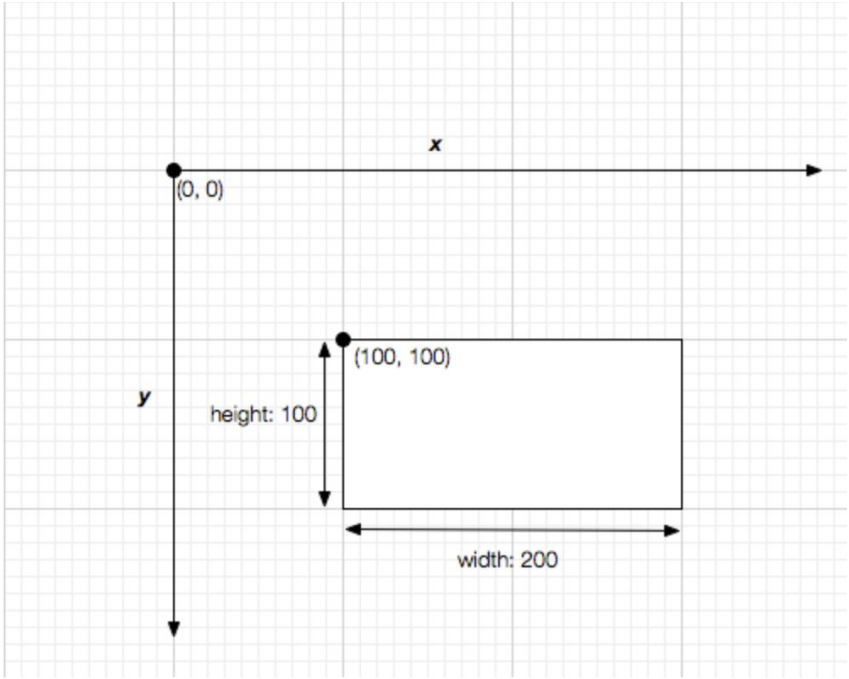


<path/>

d: path to follow

Moveto, Lineto, Curveto, Arcto

Basic SVG elements - Example

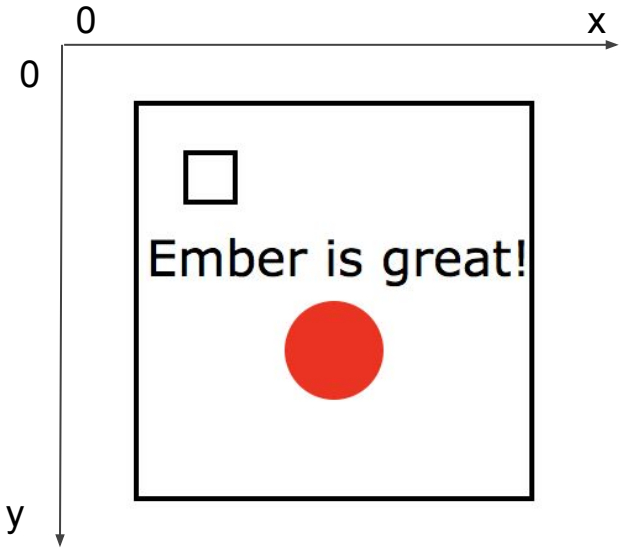



```
<svg>

 <rect x="100" y="100" width="200"
       height="100"/>

</svg>
```

Basic SVG elements - Example




```svg
<svg width="200" height="200" xmlns="http://www.w3.org/2000/svg">
  <!-- path -->
  <path d="M10 10 H 90 V 90 H 10 Z" fill="transparent" stroke="black"/>
  <!-- Circle -->
  <circle cx="50" cy="60" r="10" fill="red"/>
  <!-- Text -->
  <text x="12" y="45" font-family="Verdana" font-size="10">
    Ember is great!
  </text>
  <!-- rectangle -->
  <rect x=20 y=20 width="10" height="10" fill="transparent" stroke="black"/>
</svg>
```

D3 Ecosystem

- API Documentation
- Bl.ocks
- Blockbuilder
- search

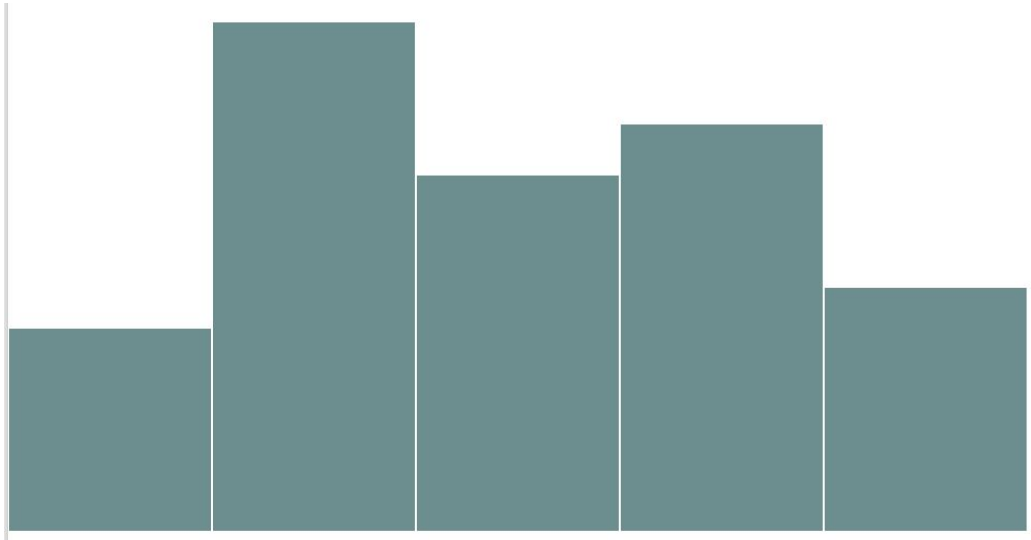
Main modules

- Selections
- Scales
- Axes
- Shapes

Selection & Data

```
<body>
  <svg>
    <rect />
    <rect />
    <rect />
    <rect />
    <rect />
  </svg>
  <script>
    var rectWidth = 100;
    var height = 300;
    var data = [100, 250, 175, 200, 120];

    var rect = d3.selectAll('rect')
      .data(data)
      .attr('x', function(d, i) {return i * rectWidth})
      .attr('y', function(d) {return height - d})
      .attr('width', rectWidth)
      .attr('height', function(d) {return d})
      .attr('fill', "#658f90")
      .attr('stroke', '#fff')
  </script>
```



Selection & Data

```
<body>
  <svg>
    <rect />
    <rect />
    <rect />
    <rect />
    <rect />
  </svg>
  <script>
    var rectWidth = 100;
    var height = 300;
    var data = [100, 250, 175, 200, 120];

    var rect = d3.selectAll('rect')
      .data(data)
      .attr('x', function(d, i) {return i * rectWidth})
      .attr('y', function(d) {return height - d})
      .attr('width', rectWidth)
      .attr('height', function(d) {return d})
      .attr('fill', "#658f90")
      .attr('stroke', '#fff')
  </script>
```

5 rectangle elements

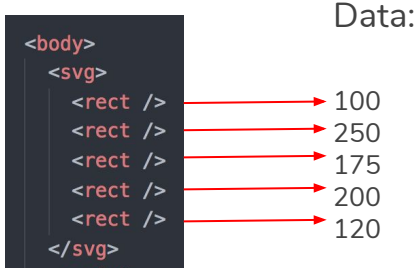
Select all rectangle elements

```
▼ut {_groups: Array(1), _parents: Array(1)}
  ▼ _groups: Array(1)
    ▼ 0: NodeList(5)
      ▶ 0: rect
      ▶ 1: rect
      ▶ 2: rect
      ▶ 3: rect
      ▶ 4: rect
        length: 5
      ▶ __proto__: NodeList
        length: 1
    ▶ __proto__: Array(0)
  ▶ _parents: [html]
  ▶ __proto__: Object
```

Selection & Data

```
var rect = d3.selectAll('rect')
  .data(data)
  .attr('x', function(d, i) {return i * rectWidth})
  .attr('y', function(d) {return height - d})
  .attr('width', rectWidth)
  .attr('height', function(d) {return d})
  .attr('fill', "#658f90")
  .attr('stroke', '#fff')
</script>
```

“Binding” data to the selections



Selection & Data

```
var rect = d3.selectAll('rect')
  .data(data)
  .attr('x', function(d, i) {return i * rectWidth})
  .attr('y', function(d) {return height - d})
  .attr('width', rectWidth)
  .attr('height', function(d) {return d})
  .attr('fill', "#658f90")
  .attr('stroke', '#fff')
</script>
```

Loop through each rectangle selection

Value get set by passed in (data, index)

Output

```
<rect x="0" y="200" width="100" height="100" fill="#658f90" stroke="#fff"></rect>
<rect x="100" y="50" width="100" height="250" fill="#658f90" stroke="#fff"></rect>
<rect x="200" y="125" width="100" height="175" fill="#658f90" stroke="#fff"></rect>
<rect x="300" y="100" width="100" height="200" fill="#658f90" stroke="#fff"></rect>
<rect x="400" y="180" width="100" height="120" fill="#658f90" stroke="#fff"></rect>
```

Scales

Scales maps from data attributes (domain) to display (range)

For example:

```
Var yScale = d3.scaleLinear()
    .domain([min, max]) // input
    .range([min, max]); // output
```

yScale(20) => 0.2

yScale(20) => 400

Value to opacity

Domain	Range
[0, 100]	[0, 1]
0	0
20	0.2
50	0.5
75	0.75
100	1

Y-value get set correctly

Domain	Range
[0, 100]	[500, 0]
0	500
20	400
50	250
75	125
100	0

Scales

```
data = [{ id:1, value:100 }, { id:2, value:200 } ,...]

// get min/max
var min = d3.min(data, d => d.value);
var max = d3.max(data, d => d.value);

// or use extent, which gives back [min, max]
var extent = d3.extent(data, d => d.value);

var yScale = d3.scaleLinear()
  .domain(extent)
  .range([height, 0]);
```

Scales

Most common scales:

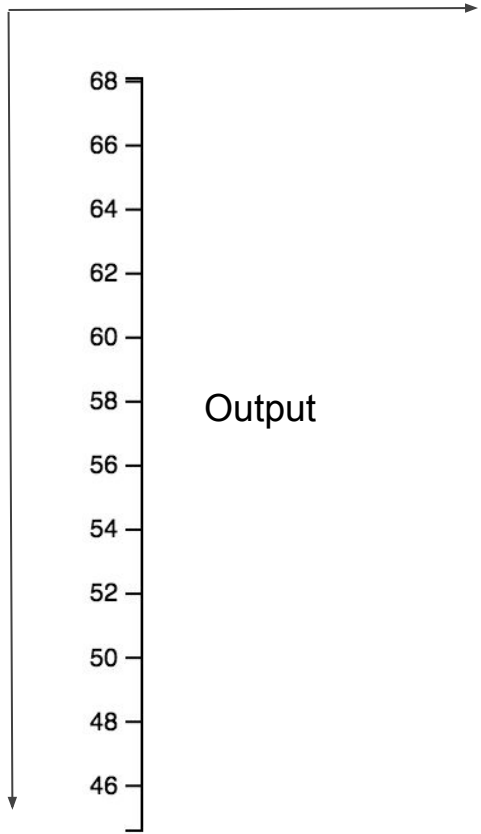
- // continuous:
  d3.scaleLinear()
  d3.scaleLog()
  d3.scaleTime()

- // ordinal:
  d3.scaleBand() => Map categories (ex: animals) in your x or y value

Axis

```
var yAxis = d3.axisLeft()
  .scale(yScale); // pass in a scale

d3.select('svg')
  // create a group element we can translate
  // so that the axis will be visible in SVG
  .append('g')
  .attr('transform', 'translate(40, 20)')
  // selection.call(yAxis) is the same as yAxis(selection)
  // and an axis will be created within the selection
  .call(yAxis);
```

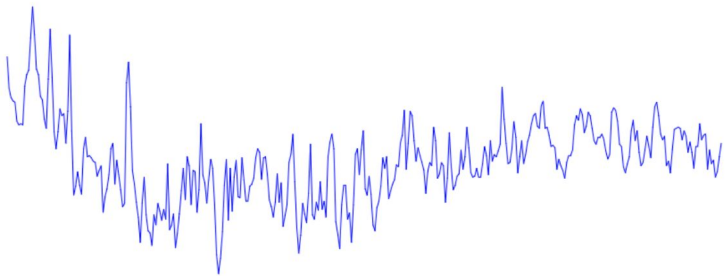


Shapes

The d3-shape module take care of computing the d attribute of your path especially for :

- Line-chat
- Pie-chart
- Arcs
- Areas

Shapes



```
<path d="M0,56.17021276595736L2.191780821917808,91.91489361702125
L4.383561643835616,102.12765957446803L6.575342465753424,105.95744680851061
L8.767123287671232,107.23404255319141L10.95890410958904,128.93617021276592
L13.150684931506849,132.76595744680844L15.342465753424658,131.48936170212764
L17.534246575342465,132.76595744680844L19.726027397260275,89.3617021276595
L21.91780821917808,76.595744680851L24.10958904109589,71.48936170212761
L26.301369863013697,35.74468085106383L28.493150684931507,0L30.684931506849317,31
L32.87671232876712,70.21276595744675L35.06849315068493,76.595744680851
L37.26027397260274,100.85106382978714L39.45205479452055,104.68085106382978
L41.64383561643836,126.38297872340416L43.83561643835616,136.59574468085106
L46.02739726027397,81.70212765957439L48.21917808219178,25.531914893617
L50.41095890410959,79.14893617021275L52.602739726027394,140.42553191489355
L54.794520547945204,159.57446808510636L56.986301369863014,140.42553191489355
L59.178082191780824,114.89361702127653L61.369863013698634,122.55319148936167
L63.561643835616444,119.99999999999994L65.75342465753424,153.19148936170208
```

Shapes - d3.line()

```
var data = [
  {date: new Date(2007, 3, 24), value: 93.24},
  {date: new Date(2007, 3, 25), value: 95.35},
  {date: new Date(2007, 3, 26), value: 98.84},
  {date: new Date(2007, 3, 27), value: 99.92},
  {date: new Date(2007, 3, 30), value: 99.80},
  {date: new Date(2007, 4,  1), value: 99.47},
  …
];

var line = d3.line()
  .x((d) => { return xScale(d.date); })
  .y((d) => { return yScale(d.value); });

d3.select('svg')
  .append('path')
  .attr('d', line(data));
```

d3.ScaleTime()
d3.ScaleLinear()

Output: path that connects each point (object) with lines or curves

Charts with EmberJS

- Ember has a lot of good addons that you can use for rendering simple charts
- Pros:
  - Fast to implement
  - Easy to maintain
- Cons:
  - Limitations on configuration options
  - Feature doesn’t exist => hack a solution => loose time digging into the addon code

D3 within EmberJS

- It has its own addon which is ember-d3. A wrapper of the d3 library, making it possible to import any individual D3 package
- Pros:
  - Gives you a lot of flexibility. You can build whatever you want
- Cons:
  - D3 has its own concept of data binding
  - Learning curve can be though

D3 within EmberJS

- It has its own addon which is ember-d3. A wrapper of the d3 library, making it possible to import any individual D3 package
- Pros:
  - Gives you a lot of flexibility. You can build whatever you want
- Cons:
  - D3 has its own concept of data binding
  - Learning curve can be though

References

SVG:

- http://slides.com/sdrasner/frontendmasters1#/

D3:

- http://www.d3noob.org/2014/02/d3js-elements.html
- http://www.recursion.org/d3-for-mere-mortals/

Ember with d3:

- https://github.com/benlesh/ember-d3-example/tree/master/app

Thanks!

https://emberjsparis.herokuapp.com/