REACT + D3 = <3

Jonathan Kaufman - @kauffecup - 12.01.15

JONATHAN KAUFMAN

DEVELOPER ADVOCATE

IBM

@kauffecup

jdkaufma@us.ibm.com

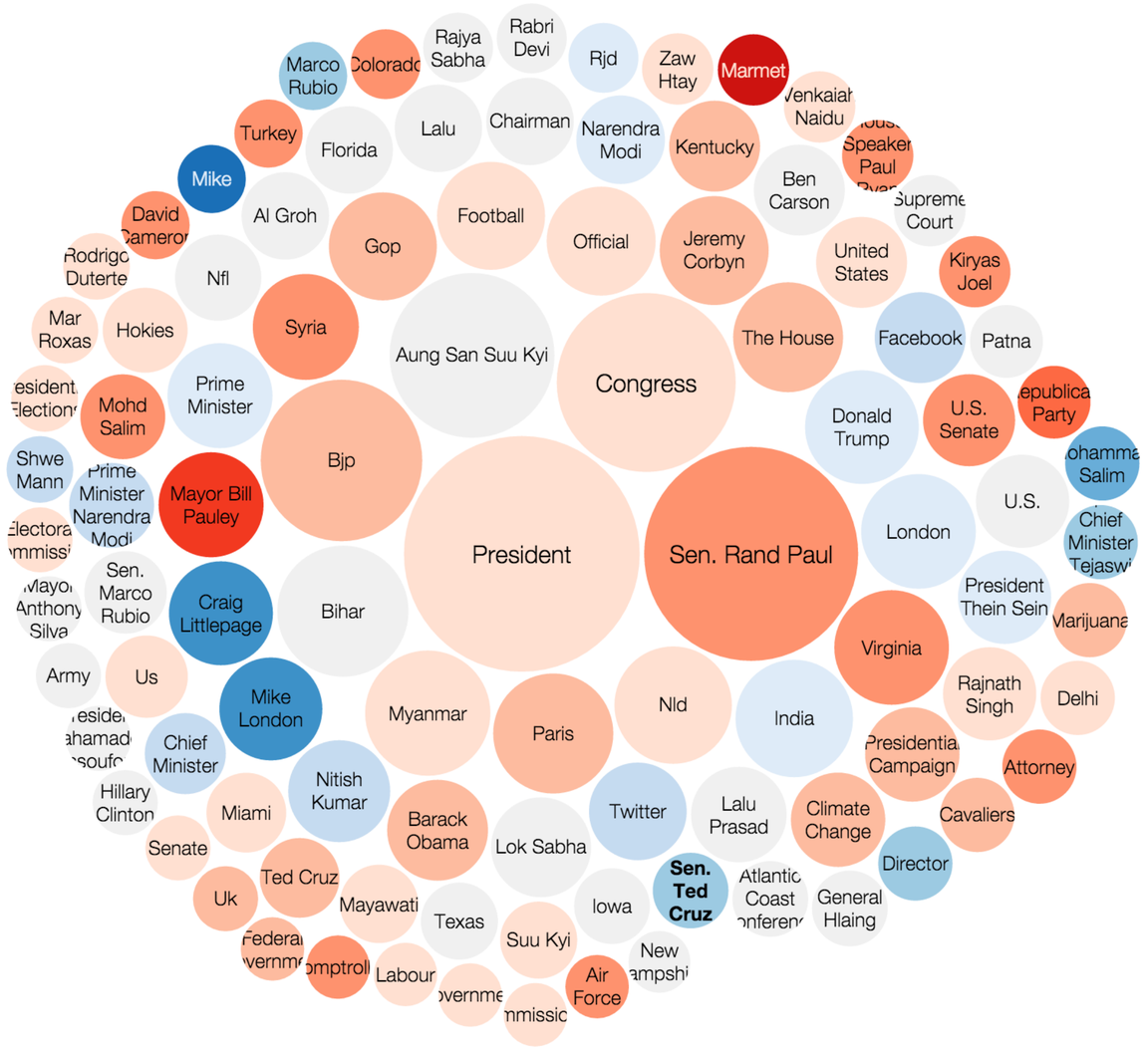
jkaufman.io

THE APP

Election Insights

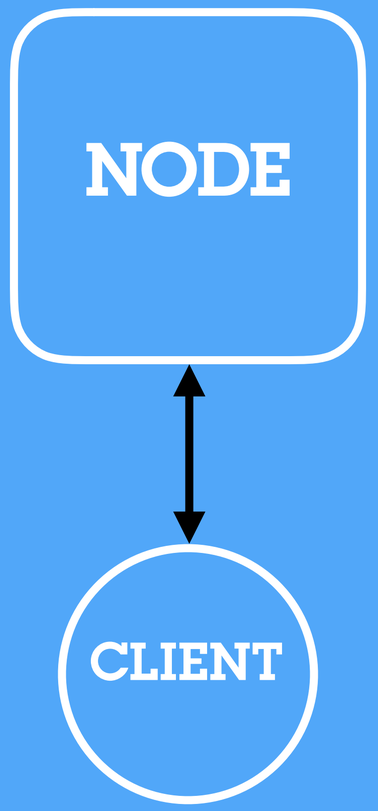
Nov 23, 1:06pm — Nov 29, 1:06pm — a day — Nov 30, 1:06pm

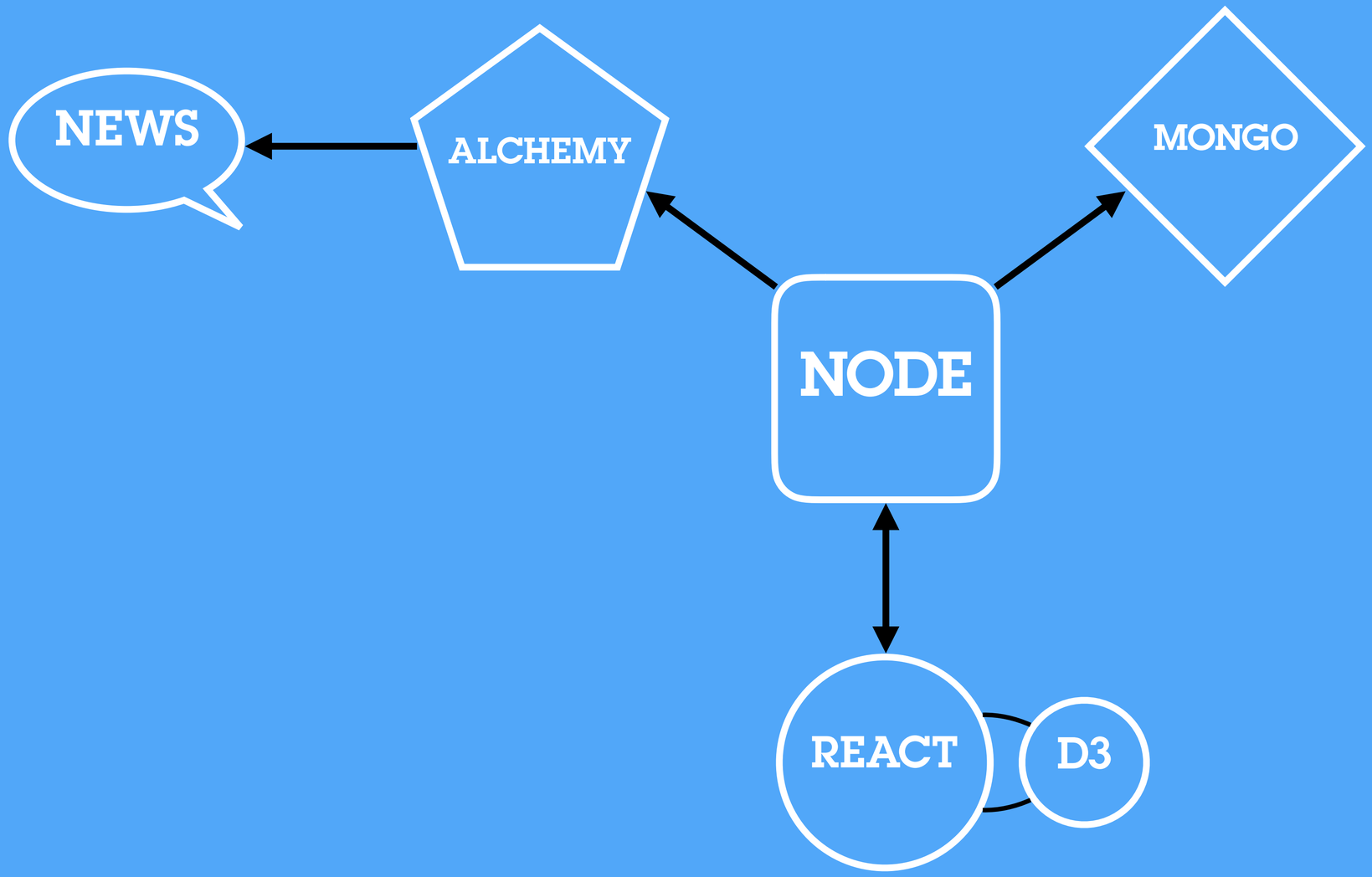
Nov 30, 1:06pm

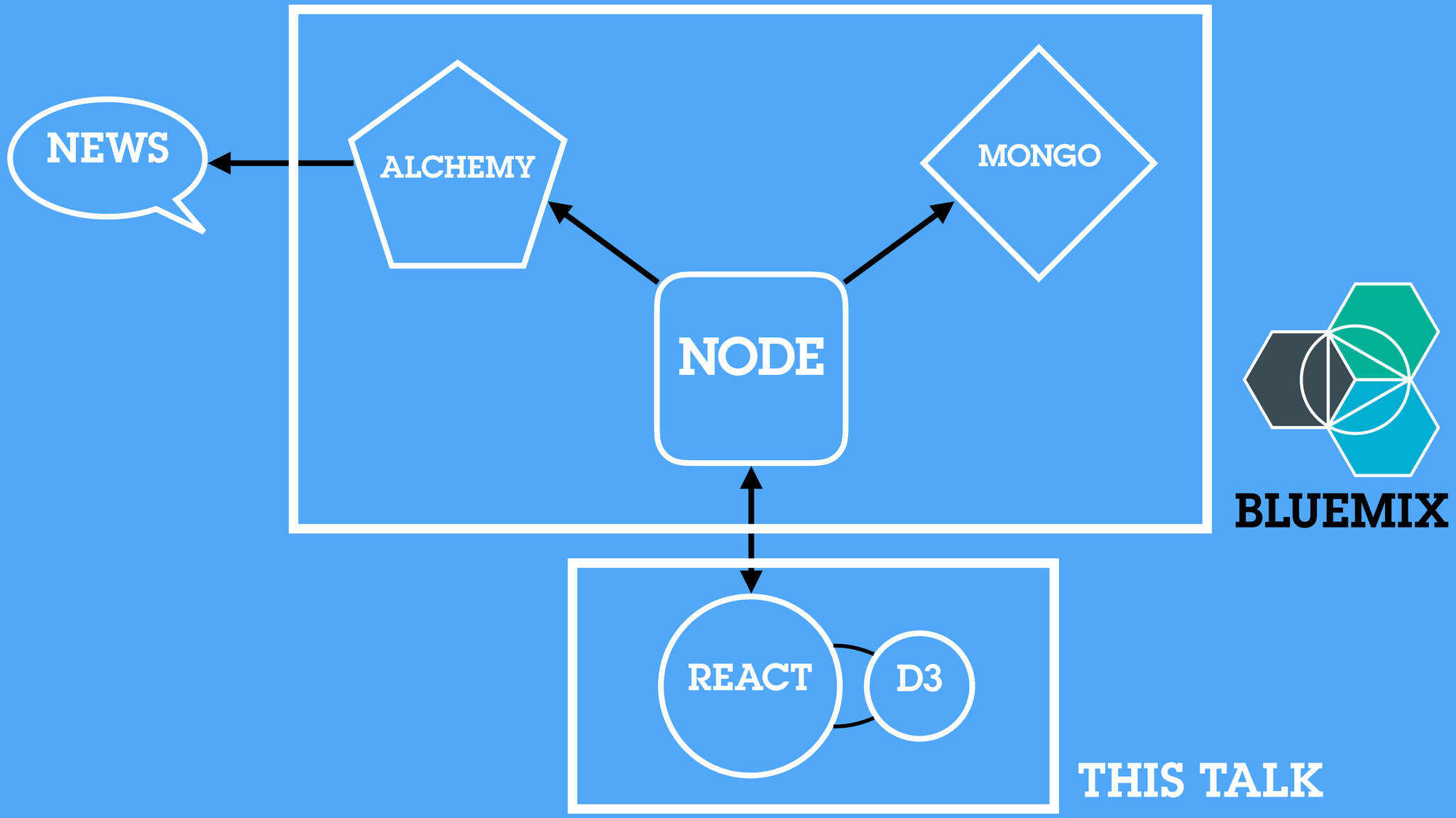


100 Circles

(might need to adjust for screen size or if animations are laggy)


NODE
CLIENT


NEWS
ALCHEMY
MONGO
NODE
REACT
D3


NEWS
ALCHEMY
MONGO
NODE
REACT
D3
BLUEMIX
THIS TALK

WHY D3?

DATA DRIVEN DOCUMENTS

Election Insights

Nov 22, 7:00pm — a day — Nov 29, 7:00pm

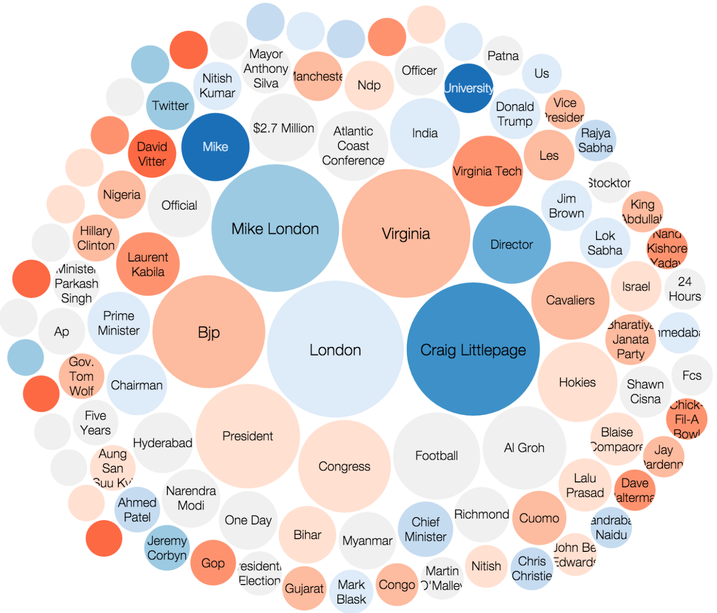
Nov 28, 7:00pm

Nov 29, 7:00pm

Positive

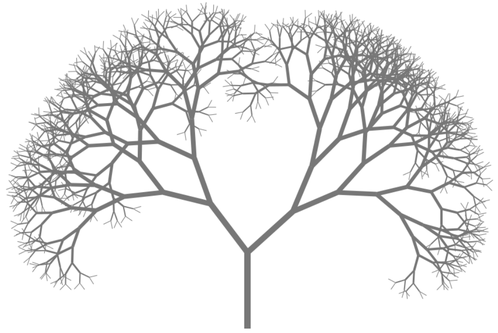
Neutral

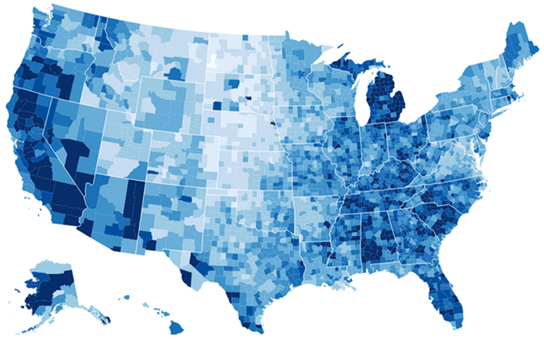
Negative

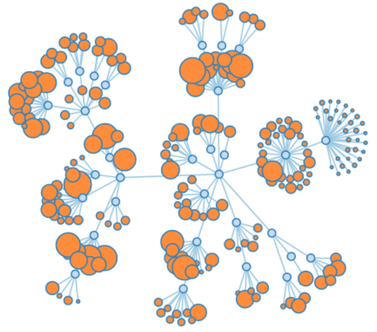


100 Circles

(might need to adjust for screen size or if animations are laggy)







THE DATA

+

IBM Bluemix

```
q.enriched.url.enrichedTitle.taxonomy.taxonomy_=
|label=elections,score=>0.75|
```

```
{
  "status": "OK",
  "totalTransactions": 13,
  "result": {
    "docs": [
      {
        "id": "NjA4MDQ2NzM4M3wxNDQ4ODk0MDAy",
        "timestamp": 1448894002,
        "source": {
          "enriched": {
            "url": {
              "title": "Tech Active Runners in IBM, Frontier Communications Corporation (NASDAQ:FTR), NXP Semiconduc
              "url": "http://streetwisereport.com/tech-active-runners-in-thrust-international-business-machines-corp
              "enrichedTitle": {
                "entities": [
                  {
                    "count": 7,
                    "sentiment": -0.333,
                    "text": "International Business Machines Corporation",
                  },
                  ...
                ]
              }
            }
          }
        }
      }
    ]
  }
}
```

```
{
  "count": 7,
  "sentiment": -0.333,
  "text": "International Business Machines Corporation",
  "date": "2015-11-30T09:09:33:22.0000Z"
}

{
  "count": 3,
  "sentiment": 0,
  "text": "President",
  "date": "2015-11-30T09:09:33:22.0000Z"
}

{
  "count": 2,
  "sentiment": 0.469637007,
  "text": "School District",
  "date": "2015-11-30T09:09:33:22.0000Z"
}
```

```
{
  "count": 7,
  "sentiment": -0.333,
  "text": "International Business Machines Corporation",
  "date": "2015-11-30T09:09:33:22.0000Z"
}

{
  "count": 3,
  "sentiment": 0,
  "text": "President",
  "date": "2015-11-30T09:09:33:22.0000Z"
}

{
  "count": 2,
  "sentiment": 0.469637007,
  "text": "School District",
  "date": "2015-11-30T09:09:33:22.0000Z"
}
```


MONGO DB

1. QUERY ENTITIES BY DATE
2. GROUP THEM BY TEXT
3. AVERAGE THE SENTIMENT
4. SUM UP THE COUNT
5. SORT BY THE SUMMED COUNT
6. LIMIT TO AN ARBITRARY NUMBER

MONGO AGGREGATION PIPELINE

```
aggregateEntities: function (start = 0, end = 9999999999999, limit = 100) {
  return new Promise((resolve, reject) => {
    Entity.aggregate(
      { $match: { date: { $gte: new Date(start) , $lt: new Date(end) } } },
      { $group: {
        _id: { $toLower: '$text'},
        value: { $sum: '$count'},
        sentiment: { $avg: '$sentiment'}
      } },
      { $sort: { value: -1} },
      { $limit: limit },
      (err, res) => {
        if (err) { reject(err); }
        else { resolve(res); }
      }
    );
  });
}
```

```
[{
  "_id":"President",
  "sentiment":-0.09158288680865387,
  "value":270
},{
  "_id":"United States",
  "sentiment":-0.03118193632258064,
  "value":125
},{
  "_id":"Mauricio Macri",
  "sentiment":-0.09327133250000001,
  "value":112
}]
```

CONNECTING D3 AND REACT

BOTH ARE
FUNCTIONAL

ALIGN THE LIFECYCLES

```
componentWillMount()

componentDidMount()

componentWillReceiveProps()

componentShouldUpdate()

componentWillUpdate()

componentDidUpdate()

componentWillUnmount()
```

ponentWillMount()

ponentDidMount() → new D3Thing(domNode, props)

ponentWillReceiveProps()

ponentShouldUpdate()

ponentWillUpdate()

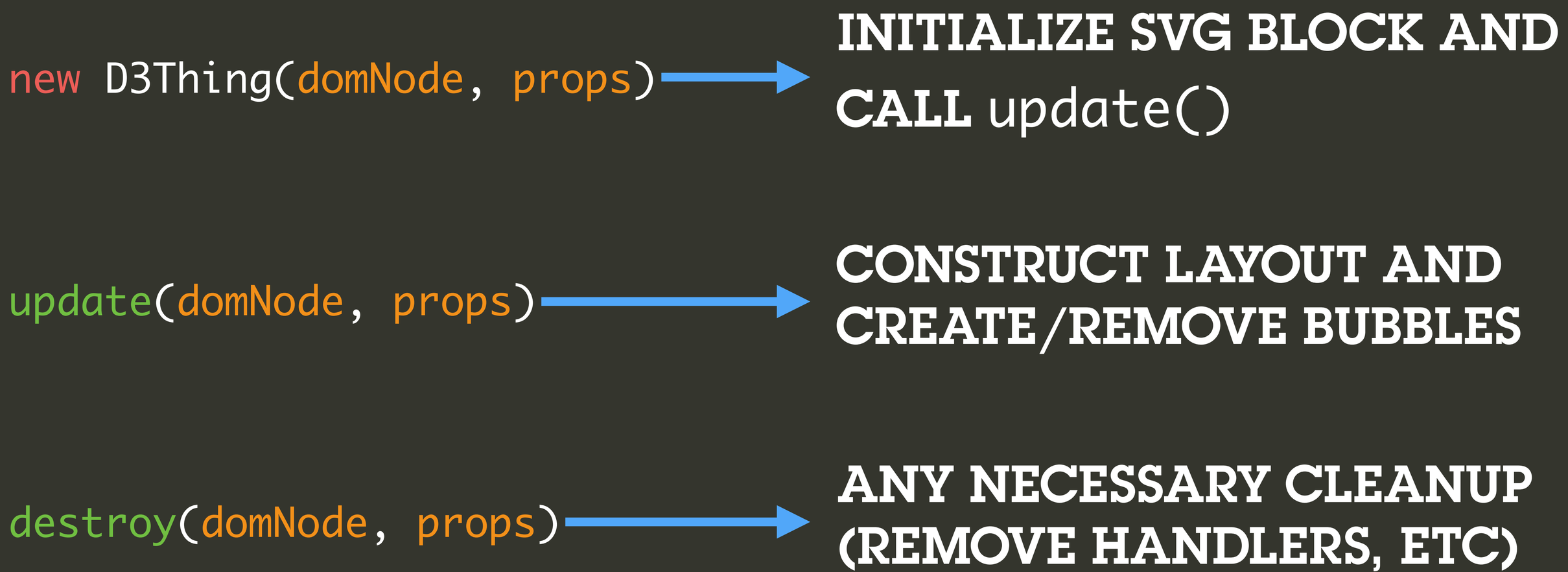
ponentDidUpdate() → update(domNode, props)

ponentWillUnmount() → destroy(domNode, props)

```
import React, { Component, PropTypes } from 'react';
import ReactDOM from 'react-dom';
import SimpleComponentD3 from './SimpleComponentD3';

export default class SimpleComponent extends Component {
  render() {
    return <div className="simple-component"></div>;
  }
  componentDidMount() {
    this.simpleComponentD3 = new SimpleComponentD3(ReactDOM.findDOMNode(this), this.getProps());
  }
  componentDidUpdate() {
    this.simpleComponentD3.update(ReactDOM.findDOMNode(this), this.getProps());
  }
  componentWillUnmount() {
    this.simpleComponentD3.destroy(ReactDOM.findDOMNode(this));
  }
  getProps() {
    return this.props;
  }
}
```

KEEP D3 STATELESS


new D3Thing(domNode, props)
INITIALIZE SVG BLOCK AND
CALL update()
update(domNode, props)
CONSTRUCT LAYOUT AND
CREATE/REMOVE BUBBLES
destroy(domNode, props)
ANY NECESSARY CLEANUP
(REMOVE HANDLERS, ETC)

ONE SOURCE OF TRUTH

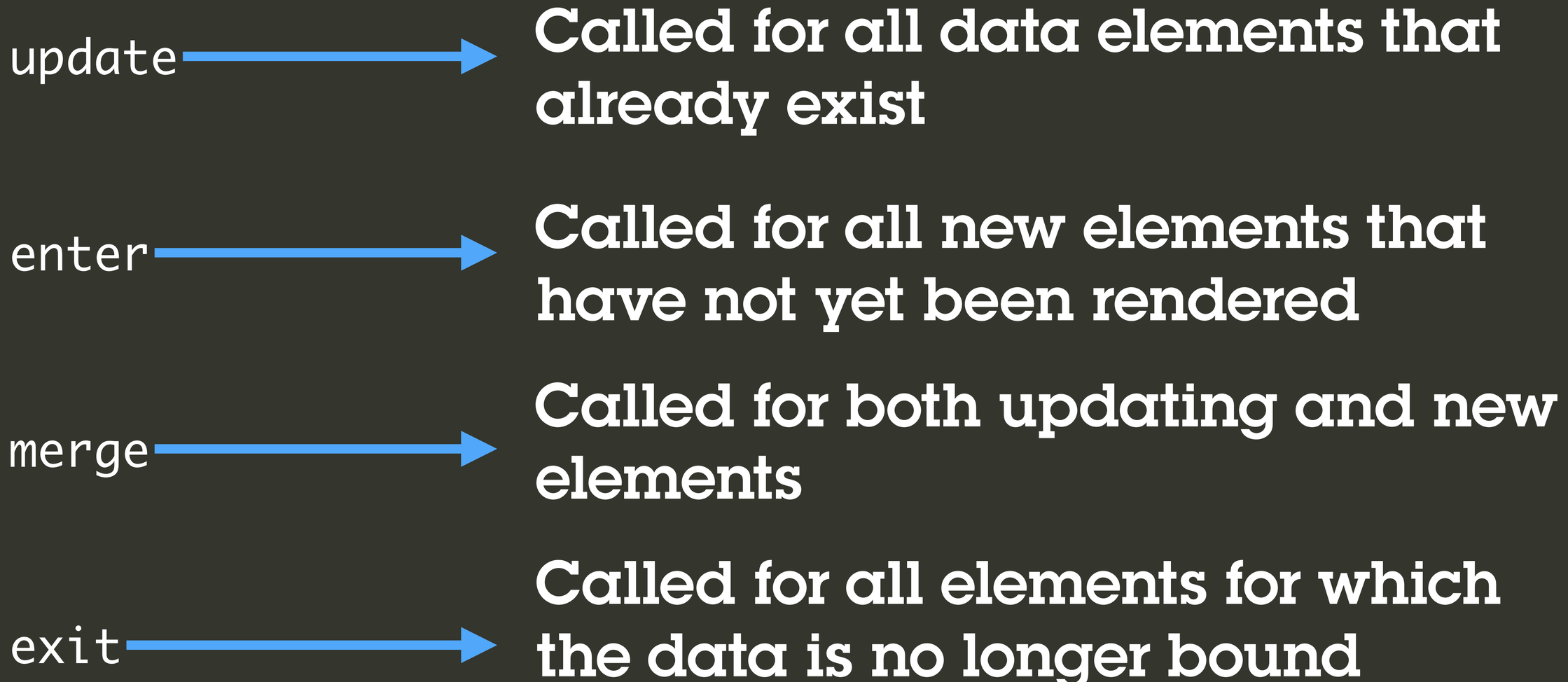
D3 LIFE-CYCLE EVENTS

BIND DATA TO SVG/HTML

```
this.svg = d3.select(el).append('svg')

...

const circles = this.svg.selectAll('circle')
  .data(data, d => d._id);
```


update
Called for all data elements that already exist
enter
Called for all new elements that have not yet been rendered
merge
Called for both updating and new elements
exit
Called for all elements for which the data is no longer bound

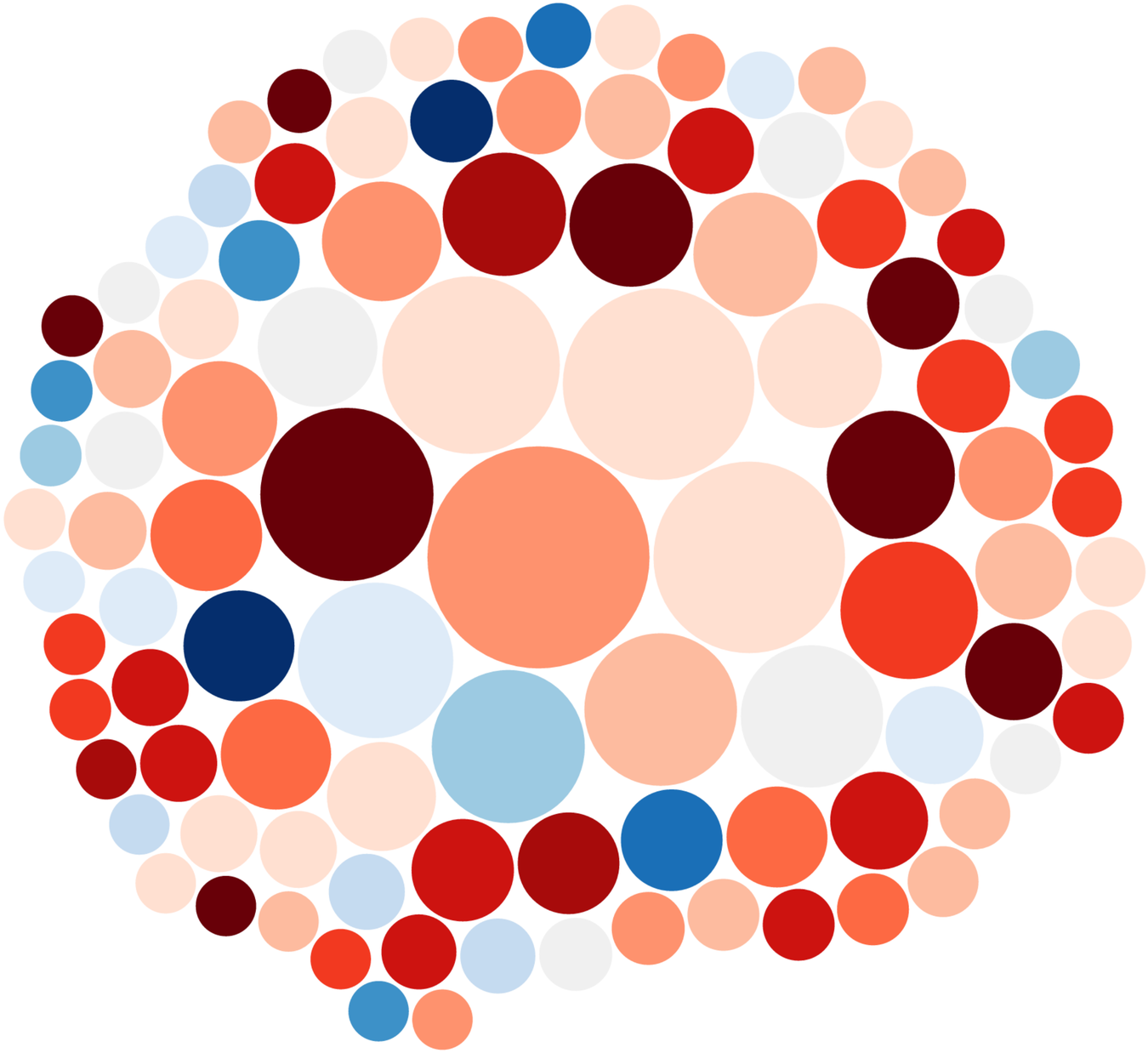
```
const circles = this.svg.selectAll('circle')
  .data(data, d => d._id);

circles.enter()
  .append('circle')
  .attr('transform', d => 'translate(' + d.x + ',' + d.y + ')')
  .attr('r', d => d.r)
```

```
const circles = this.svg.selectAll('circle')
  .data(data, d => d._id);

circles.exit().remove()
```

EXAMPLE APPLICATION



BASIC D3 COMPONENT

```
constructor(el, props = {}) {
  this.svg = d3.select(el).append('svg')
    .attr('class', 'bubble-chart-d3')
    .style('overflow', 'visible');
  this.update(el, props);
}
```

```
update(el, props) {
  this.adjustSize(el);
  const { data } = props;
  // get our layout data
  const nodes = this.bubble.nodes(data.length ? {children: data} : data)
    .filter(d => d.depth); // filter out the outer bubble

  // link our nodes to d3
  const circles = this.svg.selectAll('circle')
    .data(nodes, d => 'g' + d._id);

  // move any existing nodes to their new location
  circles.attr('transform', d => 'translate(' + d.x + ',' + d.y + ')')
    .attr('r', d => d.r);
  // create any new nodes and postion them
  circles.enter().append('circle')
    .attr('transform', d => 'translate(' + d.x + ',' + d.y + ')')
    .attr('r', d => d.r);
  // remove any nodes that ain't there
  circles.exit()
    .remove();
}
```

```
adjustSize(el) {
  this.diameter = Math.min(el.offsetWidth, el.offsetHeight);
  const top = Math.max((el.offsetHeight - this.diameter)/2, 0);
  // center some stuff vertically
  this.svg.attr('width', this.diameter)
    .attr('height', this.diameter)
    .style('position', 'relative')
    .style('top', top + 'px');    // center vertically

  // create the bubble layout that we will use to position our bubbles
  this.bubble = d3.layout.pack()
    .sort(null)
    .size([this.diameter, this.diameter])
    .padding(3);
}
```

ZERO SIDE EFFECTS*

*EXCEPT FOR REFERENCE TO SVG BLOCK

ADDING COLOR

```
update(el, props) {
  this.adjustSize(el);
  const { data } = props;
  // get our layout data
  const nodes = this.bubble.nodes(data.length ? {children: data} : data)
    .filter(d => d.depth); // filter out the outer bubble

  // link our nodes to d3
  const circles = this.svg.selectAll('circle')
    .data(nodes, d => 'g' + d._id);

  // move any existing nodes to their new location
  circles.attr('transform', d => 'translate(' + d.x + ',' + d.y + ')')
    .attr('r', d => d.r);
  // create any new nodes and postion them
  circles.enter().append('circle')
    .attr('transform', d => 'translate(' + d.x + ',' + d.y + ')')
    .attr('r', d => d.r);
  // remove any nodes that ain't there
  circles.exit()
    .remove();
}
```

```
this.adjustSize(el);
const { data, colorLegend } = props;

// define a color scale for our colorValues
const color = d3.scale.quantize()
  .domain([
    d3.min(data, d => d.colorValue),
    d3.max(data, d => d.colorValue)
  ])
  .range(colorLegend);

// get our layout data
const nodes = this.bubble.nodes(data.length ? {children: data} : data)
  .filter(d => d.depth); // filter out the outer bubble

// link our nodes to d3
const circles = this.svg.selectAll('circle')
  .data(nodes, d => 'g' + d._id);

// move any existing nodes to their new location
circles.attr('transform', d => 'translate(' + d.x + ',' + d.y + ')')
  .attr('r', d => d.r)
  .style('fill', d => color(d.colorValue));
// create any new nodes and position them
circles.enter().append('circle')
  .attr('transform', d => 'translate(' + d.x + ',' + d.y + ')')
  .attr('r', d => d.r)
  .style('fill', d => color(d.colorValue));
// remove any nodes that ain't there
circles.exit()
```

```
const color = d3.scale.quantize()
  .domain([
    d3.min(data, d => d.colorValue),
    d3.max(data, d => d.colorValue)
  ])
  .range(colorLegend);
```

```
const data = [{ colorValue:-0.9 }, { colorValue:0.75 }, { colorValue:0.1 }];
```

```
const colorLegend = [
  // reds from dark to light
  "#67000d", "#a50f15", "#cb181d", "#ef3b2c", "#fb6a4a", "#fc9272", "#fcbba1", "#fee0d2",
  //neutral grey
  "#f0f0f0",
  // blues from light to dark
  "#deebf7", "#c6dbef", "#9ecae1", "#6baed6", "#4292c6", "#2171b5", "#08519c", "#08306b"
];
```

```
const color = d3.scale.quantize()
  .domain([
    d3.min(data, d => d.colorValue),
    d3.max(data, d => d.colorValue)
  ])
  .range(colorLegend);
```

-0.9 0.75

"#67000d" "#08306b"

color(0.1) === "#deebf7"

ADDING
ANIMATIONS

```
    // define a color scale for our colorValues
    const color = d3.scale.quantize()
      .domain([
        d3.min(data, d => d.colorValue),
        d3.max(data, d => d.colorValue)
      ])
      .range(colorLegend);

    // get our layout data
    const nodes = this.bubble.nodes(data.length ? {children: data} : data)
      .filter(d => d.depth); // filter out the outer bubble

    // link our nodes to d3
    const circles = this.svg.selectAll('circle')
      .data(nodes, d => 'g' + d._id);

    // move any existing nodes to their new location
    circles.attr('transform', d => 'translate(' + d.x + ',' + d.y + ')')
      .attr('r', d => d.r)
      .style('fill', d => color(d.colorValue));
    // create any new nodes and postion them
    circles.enter().append('circle')
      .attr('transform', d => 'translate(' + d.x + ',' + d.y + ')')
      .attr('r', d => d.r)
      .style('fill', d => color(d.colorValue));
    // remove any nodes that ain't there
    circles.exit()
      .remove();
  }
```

```
    // move any existing nodes to their new location
    circles.transition()
      .duration(duration)
      .delay((d, i) => i * 7)
      .attr('transform', d => 'translate(' + d.x + ',' + d.y + ')')
      .attr('r', d => d.r)
      .style('fill', d => color(d.colorValue));
    // create any new nodes and postion them
    circles.enter().append('circle')
      .attr('transform', d => 'translate(' + d.x + ',' + d.y + ‘)')
      .attr('r', d => 0)
      .style('fill', d => color(d.colorValue))
      .transition()
      .duration(duration * 1.2)
      .attr('r', d => d.r);
    // remove any nodes that ain't there
    circles.exit()
      .transition()
      .duration(duration)
      .attr('transform', d => {
        const dy = d.y - this.diameter/2;
        const dx = d.x - this.diameter/2;
        const theta = Math.atan2(dy,dx);
        const destX = this.diameter * (1 + Math.cos(theta) )/2;
        const destY = this.diameter * (1 + Math.sin(theta) )/2;
        return 'translate(' + destX + ',' + destY + ')'; })
      .attr('r', 0)
      .remove();
  }
```

ADDING TEXT

WRAPPING TEXT IN SVGS IS HARD

1. CREATE HTML BLOCK
2. CREATE s
3. ANIMATE s

```
constructor(el, props = {}) {
  this.svg = d3.select(el).append('svg')
    .attr('class', 'bubble-chart-d3')
    .style('overflow', ‘visible');
  this.update(el, props);
}
```

```
constructor(el, props = {}) {
  this.svg = d3.select(el).append('svg')
    .attr('class', 'bubble-chart-d3')
    .style('overflow', ‘visible');
  this.html = d3.select(el).append('div')
    .attr('class', 'bubble-chart-text')
    .style('position', 'absolute')
    .style('left', 0)              // center horizontally
    .style('right', 0)
    .style('margin-left', 'auto')
    .style('margin-right', 'auto');
  this.update(el, props);
}
```

```
adjustSize(el) {
  this.diameter = Math.min(el.offsetWidth, el.offsetHeight);
  const top = Math.max((el.offsetHeight - this.diameter)/2, 0);
  // center some stuff vertically
  this.svg.attr('width', this.diameter)
    .attr('height', this.diameter)
    .style('position', 'relative')
    .style('top', top + 'px');    // center vertically

  // create the bubble layout that we will use to position our bubbles
  this.bubble = d3.layout.pack()
    .sort(null)
    .size([this.diameter, this.diameter])
    .padding(3);
}
```

```
adjustSize(el) {
  this.diameter = Math.min(el.offsetWidth, el.offsetHeight);
  const top = Math.max((el.offsetHeight - this.diameter)/2, 0);
  // center some stuff vertically
  this.svg.attr('width', this.diameter)
    .attr('height', this.diameter)
    .style('position', 'relative')
    .style('top', top + 'px');    // center vertically
  this.html.style('width', this.diameter + 'px')
    .style('height', this.diameter + 'px')
    .style('top', top + 'px');   // center vertically;
  // create the bubble layout that we will use to position our bubbles
  this.bubble = d3.layout.pack()
    .sort(null)
    .size([this.diameter, this.diameter])
    .padding(3);
}
```

```
update(el, props) {
  this.adjustSize(el);
  const duration = 500;
  const delay = 0;
  const { data, colorLegend } = props;

  // define a color scale for our colorValues
  const color = d3.scale.quantize()
    .domain([
      d3.min(data, d => d.colorValue),
      d3.max(data, d => d.colorValue)
    ])
    .range(colorLegend);


  // get our layout data
  const nodes = this.bubble.nodes(data.length ? {children: data} : data)
    .filter(d => d.depth); // filter out the outer bubble

  // link our nodes to d3
  const circles = this.svg.selectAll('circle')
    .data(nodes, d => 'g' + d._id);

  // move any existing nodes to their new location
  circles.transition()
    .duration(duration)
    .delay((d, i) => i * 7)
    .attr('transform', d => 'translate(' + d.x + ',' + d.y + ')')
    .attr('r', d => d.r)
```

```
const labels = this.html.selectAll('.bubble-label')
  .data(nodes, d => 'g' + d._id);

// move any existing nodes to their new location
circles.transition()
  .duration(duration)
  .delay((d, i) => i * 7)
  .attr('transform', d => 'translate(' + d.x + ',' + d.y + ')')
  .attr('r', d => d.r)
  .style('fill', d => color(d.colorValue));
labels.transition()
  .duration(duration)
  .delay((d, i) => i * 7)
  .style('height', d => 2 * d.r + 'px')
  .style('width', d => 2 * d.r + 'px')
  .style('left', d =>  d.x - d.r + 'px')
  .style('top', d =>  d.y - d.r + 'px')
  .style('opacity', 1);
// create any new nodes and position them
circles.enter().append('circle')
  .attr('transform', d => 'translate(' + d.x + ',' + d.y + ')')
  .attr('r', d => 0)
  .style('fill', d => color(d.colorValue))
  .transition()
  .duration(duration * 1.2)
  .attr('r', d => d.r);
labels.enter().append('div')
    .attr('class', 'bubble-label')
    .text(d => d.displayText || d._id)
```

REFERENCES

GITHUB	https://github.com/kauffecup/react-d3-examples/
MY POST	http://www.jkaufman.io/react-d3-love/
REACT BUBBLE CHART	https://github.com/kauffecup/react-bubble-chart
ELECTION INSIGHTS	http://electioninsights.mybluemix.net/
D3	http://d3js.org/
ALCHEMY	http://www.alchemyapi.com/
REDUX	http://redux.js.org/
NICOLAS HERY BLOG	http://nicolashery.com/integrating-d3js-visualizations-in-a-react-app/