Visualization with React

Peter Beshai @pbesh

ReactJS Meetup
Jan. 12, 2016

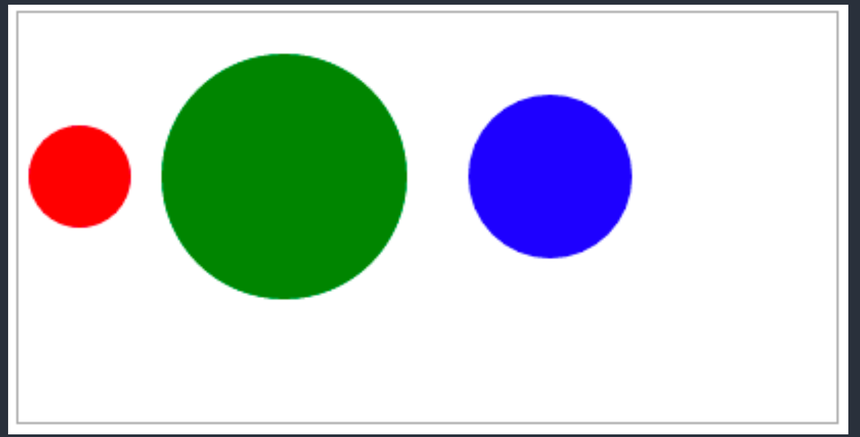
https://github.com/pbeshai/react-basic-vis-examples

Example 1 Basic SVG Drawing

```
const VisExample1 = React.createClass({
  propTypes: {
    height: React.PropTypes.number.isRequired,
    width: React.PropTypes.number.isRequired
  },

  render() {
    const { width, height } = this.props;

    return (
      <svg width={width} height={height} className='chart'>
        <circle cx={30} cy={80} r={25} fill={'red'} />
        <circle cx={130} cy={80} r={60} fill={'green'} />
        <circle cx={260} cy={80} r={40} fill={'blue'} />
      </svg>
    );
  }
});
```



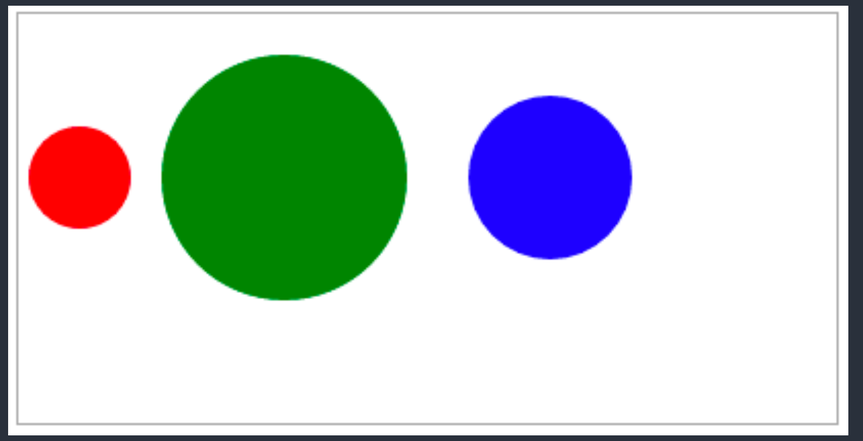
https://github.com/pbeshai/react-basic-vis-examples

Example 2 SVG Drawing from Data Array

```
render() {
  const { width, height } = this.props;

  const data = [
    { x: 30, y: 80, r: 25, color: 'red' },
    { x: 130, y: 80, r: 60, color: 'green' },
    { x: 260, y: 80, r: 40, color: 'blue' }
  ];

  return (
    <svg width={width} height={height} className='chart'>
      {data.map((d, i) => {
        return <circle key={i} cx={d.x} cy={d.y}
                       r={d.r} fill={d.color} />;
      })}
    </svg>
  );
}
```



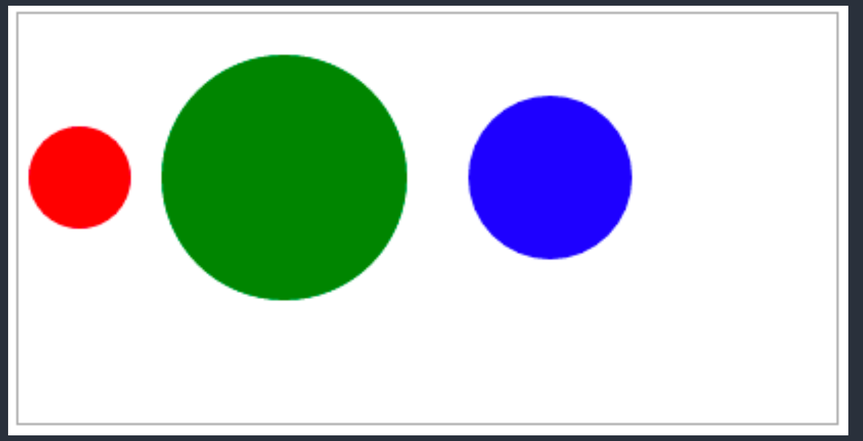
https://github.com/pbeshai/react-basic-vis-examples

Example 2 SVG Drawing from Data Array

```
render() {
  const { width, height } = this.props;

  const data = [
    { x: 30, y: 80, r: 25, color: 'red' },
    { x: 130, y: 80, r: 60, color: 'green' },
    { x: 260, y: 80, r: 40, color: 'blue' }
  ];

  return (
    <svg width={width} height={height} className='chart'>
      {data.map((d, i) => {
        return <circle key={i} cx={d.x} cy={d.y}
                       r={d.r} fill={d.color} />;
      })}
    </svg>
  );
}
```



https://github.com/pbeshai/react-basic-vis-examples

Example 3 External Data Points

```
_chartComponents() {
  /* data is of format:
     [{ locationX, locationY, frequency,
        accuracy, rank }, ...] */
  const { data } = this.props;
```

https://github.com/pbeshai/react-basic-vis-examples

Example 3 External Data Points

```
_chartComponents() {
  /* data is of format:
     [{ locationX, locationY, frequency,
        accuracy, rank }, ...] */
  const { data } = this.props;

  // convert data points into chart points
  const points = data.map(d => {
    return {
      x: d.locationX * 10,
      y: d.locationY * 10,
      r: d.frequency / 5,
      color: d.accuracy > 0.5 ? 'red' : 'blue',
      datum: d
    };
  });

  return {
    points
  };
},
```

https://github.com/pbeshai/react-basic-vis-examples

Example 3 External Data Points

```
_chartComponents() {
  /* data is of format:
     [{ locationX, locationY, frequency,
        accuracy, rank }, ...] */
  const { data } = this.props;

  // convert data points into chart points
  const points = data.map(d => {
    return {
      x: d.locationX * 10,
      y: d.locationY * 10,
      r: d.frequency / 5,
      color: d.accuracy > 0.5 ? 'red' : 'blue',
      datum: d
    };
  });

  return {
    points
  };
},
```

```
render() {
  const { width, height } = this.props;
  const { points } = this._chartComponents();

  return (
    <svg width={width} height={height}
         className='chart'>
      {points.map((d, i) => {
        return <circle key={i} cx={d.x} cy={d.y}
                       r={d.r} fill={d.color} />;
      })}
    </svg>
  );
}
```

https://github.com/pbeshai/react-basic-vis-examples

http://d3js.org

d3.**extent**()

d3.extent([1, **-20**, 5, **8**, -11, 6]) ⟶ [-20, 8]

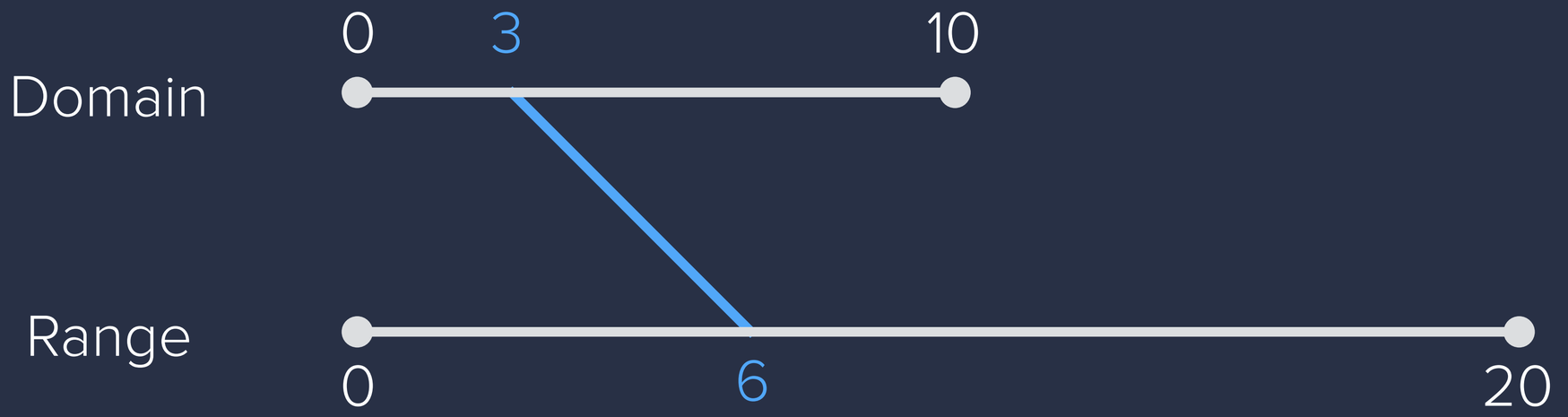
d3.**extent**()

d3.extent([1, **-20**, 5, **8**, -11, 6]) ⟶ [-20, 8]

d3.**scale.linear**()

x = d3.scale.linear().domain([0, 10]).range([0, 20])

x(3) ⟶ 6



Example 4 D3 Scales

```
_chartComponents() {
  ...
  // get the range of values from the x and y attributes in the data
  const xDomain = d3.extent(points, d => d.x);
  const yDomain = d3.extent(points, d => d.y);
```

Example 4 D3 Scales

```
_chartComponents() {
  ...
  // get the range of values from the x and y attributes in the data
  const xDomain = d3.extent(points, d => d.x);
  const yDomain = d3.extent(points, d => d.y);

  // use d3 to create scales based on the dimensions and domains
  const x = d3.scale.linear().domain(xDomain).range([0, width]);
  const y = d3.scale.linear().domain(yDomain).range([0, height]);
```

Example 4 D3 Scales

```
_chartComponents() {
  ...
  // get the range of values from the x and y attributes in the data
  const xDomain = d3.extent(points, d => d.x);
  const yDomain = d3.extent(points, d => d.y);

  // use d3 to create scales based on the dimensions and domains
  const x = d3.scale.linear().domain(xDomain).range([0, width]);
  const y = d3.scale.linear().domain(yDomain).range([0, height]);

  // create radius scale based on data
  const rDomain = d3.extent(points, d => d.r);
  const numCols = xDomain[1];
  const r = d3.scale.linear().domain(rDomain).range([0, (width / numCols) / 2]);
```

Example 4 D3 Scales

```
_chartComponents() {
  ...
  // get the range of values from the x and y attributes in the data
  const xDomain = d3.extent(points, d => d.x);
  const yDomain = d3.extent(points, d => d.y);

  // use d3 to create scales based on the dimensions and domains
  const x = d3.scale.linear().domain(xDomain).range([0, width]);
  const y = d3.scale.linear().domain(yDomain).range([0, height]);

  // create radius scale based on data
  const rDomain = d3.extent(points, d => d.r);
  const numCols = xDomain[1];
  const r = d3.scale.linear().domain(rDomain).range([0, (width / numCols) / 2]);

  // create color scale (colors from http://colorbrewer2.org)
  const color = d3.scale.linear().domain([0.3, 0.45, 0.6])
    .range(['#0571b0', '#f7f7f7', '#ca0020']).clamp(true);

  return { points, width, height, x, y, r, color };
},
```

Example 4 D3 Scales

```
render() {
  const { points, width, height, x, y, r, color } = this._chartComponents();

  return (
    <svg width={width} height={height} className='chart'>
      {points.map((d, i) => {
        return <circle key={i} cx={x(d.x)} cy={y(d.y)}
                       r={r(d.r)} fill={color(d.color)} />;
      })}
    </svg>
  );
}
```

https://github.com/pbeshai/react-basic-vis-examples

Example 5 Adding Inner Margin

```
_chartComponents() {
  ...
  // define the inner margins of the chart
  const innerMargin = { top: 10, bottom: 25, left: 10, right: 10 };
  const innerWidth = width - innerMargin.left - innerMargin.right;
  const innerHeight = height - innerMargin.top - innerMargin.bottom;

  // use d3 to create scales based on the dimensions and domains
  const x = d3.scale.linear().domain(xDomain)
    .range([innerMargin.left, innerMargin.left + innerWidth]);

  const y = d3.scale.linear().domain(yDomain)
    .range([innerMargin.top, innerMargin.top + innerHeight]);

  // create radius scale based on data
  const rDomain = d3.extent(points, d => d.r);
  const numCols = xDomain[1];
  const r = d3.scale.linear().domain(rDomain).range([0, (innerWidth / numCols) / 2]);

  ...
  return { points, width, height, innerMargin, x, y, r, color };
},
```

d3.**format**()

d3.format(‘0.1**f**’)(0.389012) → 0.4

d3.format(‘0.1**%**’)(0.389012) → 38.9%

d3.**format**()

d3.format(‘0.1**f**’)(0.389012) → 0.4

d3.format(‘0.1**%**’)(0.389012) → 38.9%

d3.**rgb**().**darker**()

d3.rgb(‘#fff’).darker() → ‘#b2b2b2’

Example 6 Highlight Behaviour

```
getInitialState() { return {}; },

_handleHighlight(d) {
  this.setState({ highlight: d });
},
```

```
render() {
  const chartComponents = this._chartComponents();
  const { points, width, height, x, y, r, color } = chartComponents;
  return (
    <svg width={width} height={height} className='chart'>
      <g className='points'>
        {points.map((d, i) => {
          return <circle key={i} cx={x(d.x)} cy={y(d.y)} r={r(d.r)} fill={color(d.color)}
                         onMouseEnter={this._handleHighlight.bind(this, d)}
                         onMouseLeave={this._handleHighlight.bind(this, null)} />;
        })}
      </g>
      {this._renderHighlight(chartComponents)}
    </svg>
  );
}
```

Example 6 Highlight Behaviour

```
getInitialState() { return {}; },

_handleHighlight(d) {
  this.setState({ highlight: d });
},

render() {
  const chartComponents = this._chartComponents();
  const { points, width, height, x, y, r, color } = chartComponents;
  return (
    <svg width={width} height={height} className='chart'>
      <g className='points'>
        {points.map((d, i) => {
          return <circle key={i} cx={x(d.x)} cy={y(d.y)} r={r(d.r)} fill={color(d.color)}
                         onMouseEnter={this._handleHighlight.bind(this, d)}
                         onMouseLeave={this._handleHighlight.bind(this, null)} />;
        })}
      </g>
      {this._renderHighlight(chartComponents)}
    </svg>
  );
}
```

Example 6 Highlight Behaviour

```
getInitialState() { return {}; },

_handleHighlight(d) {
  this.setState({ highlight: d });
},

render() {
  const chartComponents = this._chartComponents();
  const { points, width, height, x, y, r, color } = chartComponents;
  return (
    <svg width={width} height={height} className='chart'>
      <g className='points'>
        {points.map((d, i) => {
          return <circle key={i} cx={x(d.x)} cy={y(d.y)} r={r(d.r)} fill={color(d.color)}
                         onMouseEnter={this._handleHighlight.bind(this, d)}
                         onMouseLeave={this._handleHighlight.bind(this, null)} />;
        })}
      </g>
      {this._renderHighlight(chartComponents)}
    </svg>
  );
}
```

Example 6 Highlight Behaviour

```
_renderHighlight(chartComponents) {
  const { highlight } = this.state;
  const { innerMargin, height, x, y, r, color } = chartComponents;

  // no highlight, so don't show anything
  if (!highlight) return null;

  return ( // show a circle around the point and the text details of the point
    <g className='highlight'>
      <circle cx={x(highlight.x)} cy={y(highlight.y)} r={r(highlight.r) + 4}
              fill={color(highlight.color)} strokeWidth={2}
              stroke={d3.rgb(color(highlight.color)).darker()} />

      <text x={innerMargin.left} y={height - 15}>
        {`Rank #${highlight.datum.rank},
          Frequency ${d3.format('0.1f')(highlight.datum.frequency)},
          Accuracy ${d3.format('0.1%')(highlight.datum.accuracy)}`}
      </text>
    </g>
```

Example 6 Highlight Behaviour

```
_renderHighlight(chartComponents) {
  const { highlight } = this.state;
  const { innerMargin, height, x, y, r, color } = chartComponents;

  // no highlight, so don't show anything
  if (!highlight) return null;

  return ( // show a circle around the point and the text details of the point
    <g className='highlight'>
      <circle cx={x(highlight.x)} cy={y(highlight.y)} r={r(highlight.r) + 4}
              fill={color(highlight.color)} strokeWidth={2}
              stroke={d3.rgb(color(highlight.color)).darker()} />
      <text x={innerMargin.left} y={height - 15}>
        {`Rank #${highlight.datum.rank},
          Frequency ${d3.format('0.1f')(highlight.datum.frequency)},
          Accuracy ${d3.format('0.1%')(highlight.datum.accuracy)}`}
      </text>
    </g>
```

Example 6 Highlight Behaviour

```
_renderHighlight(chartComponents) {
  const { highlight } = this.state;
  const { innerMargin, height, x, y, r, color } = chartComponents;

  // no highlight, so don't show anything
  if (!highlight) return null;

  return ( // show a circle around the point and the text details of the point
    <g className='highlight'>
      <circle cx={x(highlight.x)} cy={y(highlight.y)} r={r(highlight.r) + 4}
              fill={color(highlight.color)} strokeWidth={2}
              stroke={d3.rgb(color(highlight.color)).darker()} />

      <text x={innerMargin.left} y={height - 15}>
        {`Rank #${highlight.datum.rank},
          Frequency ${d3.format('0.1f')(highlight.datum.frequency)},
          Accuracy ${d3.format('0.1%')(highlight.datum.accuracy)}`}
      </text>
    </g>
```

Example 7 Highlight Behaviour #2

```
render() {
  ...
  const maxRadius = r.range()[1];
  return (
    <svg width={width} height={height} className='chart'>
      <g className='points'>
        {points.map((d, i) => {
          return (
            <g key={i}>
              <circle cx={x(d.x)} cy={y(d.y)}  r={r(d.r)} fill={color(d.color)} />
              <rect // the invisible mouse handler
                    x={x(d.x) - maxRadius} y={y(d.y) - maxRadius} width={maxRadius * 2}
                    height={maxRadius * 2} fill={color(d.color)} style={{ opacity: 0 }}
                    onMouseEnter={this._handleHighlight.bind(this, d)}
                    onMouseLeave={this._handleHighlight.bind(this, null)} />
            </g>
          )})}
      </g>
      {this._renderHighlight(chartComponents)}
    </svg>
```

Example 8 Highlight Behaviour #3

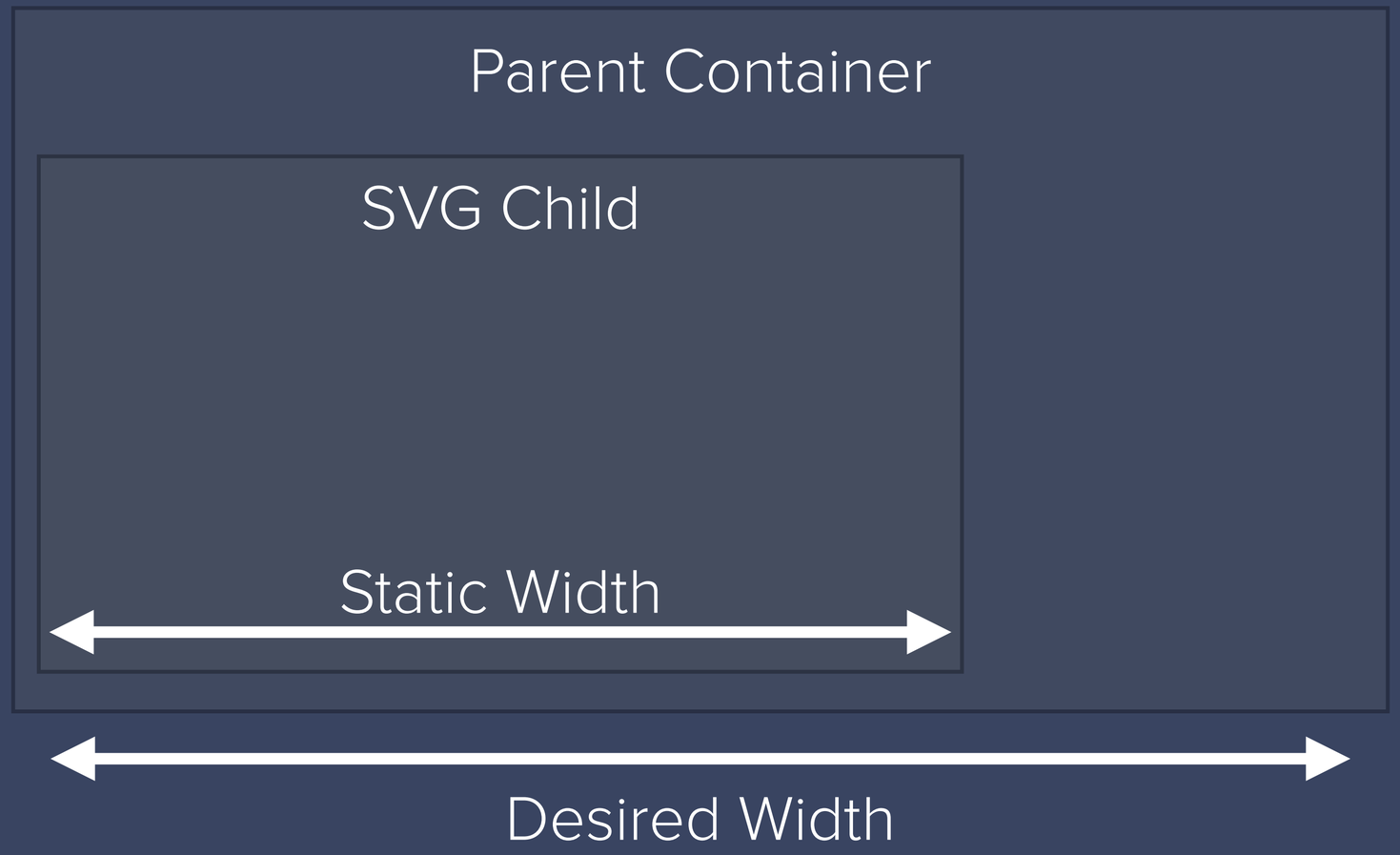
```
_renderHighlight() {
  const { highlight, chartComponents } = this.state;

  // no highlight, so don't show anything
  if (!highlight) {
    return null;
  }

  // show a circle around the point and the text details of the point
  return <VisExample8Highlight chartComponents={chartComponents} highlight={highlight} />;
},

render() {
  const { chartComponents } = this.state;
  const { width, height } = chartComponents;

  return (
    <svg width={width} height={height} className='chart'>
      <VisExample8Points chartComponents={chartComponents} onHighlight={this._handleHighlight} />
      {this._renderHighlight()}
    </svg>
  );
```


Parent Container
SVG Child
Static Width
Desired Width

```
componentDidMount() {
  this.setState({
    width: ReactDOM.findDOMNode(this).offsetWidth
  });
}
```

```
componentDidMount() {
  this.setState({
    width: ReactDOM.findDOMNode(this).offsetWidth
  });
}


// need this here to auto resize when window size changes
componentDidUpdate() {
  const domWidth = ReactDOM.findDOMNode(this).offsetWidth;

  if (this.state.width !== domWidth) {
    this.setState({
      width: domWidth
    });
  }
}
```

```
<AutoWidth>
  <VisExample8 data={data} />
</AutoWidth>
```

```
<AutoWidth>
  <VisExample8 data={data} />
</AutoWidth>
```

Window Resize?

Listen for resize events (debounced)

Window width as prop to children

```
<AutoWidth>
  <VisExample8 data={data} />
</AutoWidth>
```

Window Resize?

Listen for resize events (debounced)

Window width as prop to children

AutoWidth inside a <table>?

Use table-layout: fixed

Conclusion

Code: https://github.com/pbeshai/react-basic-vis-examples

Conclusion

Do good work and share it

Code: https://github.com/pbeshai/react-basic-vis-examples

Conclusion

Do good work and share it

You can do visualization with just React

Code: https://github.com/pbeshai/react-basic-vis-examples

Conclusion

Do good work and share it

You can do visualization with just React

D3 is still really helpful

Code: https://github.com/pbeshai/react-basic-vis-examples