Using D3.js with React

A Software Presentation From ZERRTECH

Jeremy Zerr

Site: https://www.zerrtech.com

LinkedIn: http://www.linkedin.com/in/jrzerr

What I Do

- ☐ I build custom software
- ☐ We create web applications, mobile apps, front to back
- ☐ We have used D3.js on a handful of projects over the years

ZERRTECH

Why do I want to use D3.js with React?

- ☐ Build reusable components
- ☐ Embed charts within an existing React application
- ☐ React + Redux is great at managing data changes within a web app
- ☐ Potential better performance with your charts

Why is it hard to use D3.js with React?

- Both libraries do data-driven DOM manipulation. They don't like to share.
- Where do you draw the line?

Where do they cross over?

- D3 likes to bind directly to data, so does React
- React likes to handle DOM updates in an efficient way, D3 wants to do all the DOM updates too
- Transitions are critical to most data visualizations, D3 handles transitions, React can do this but D3 is specialized to handle

Goals

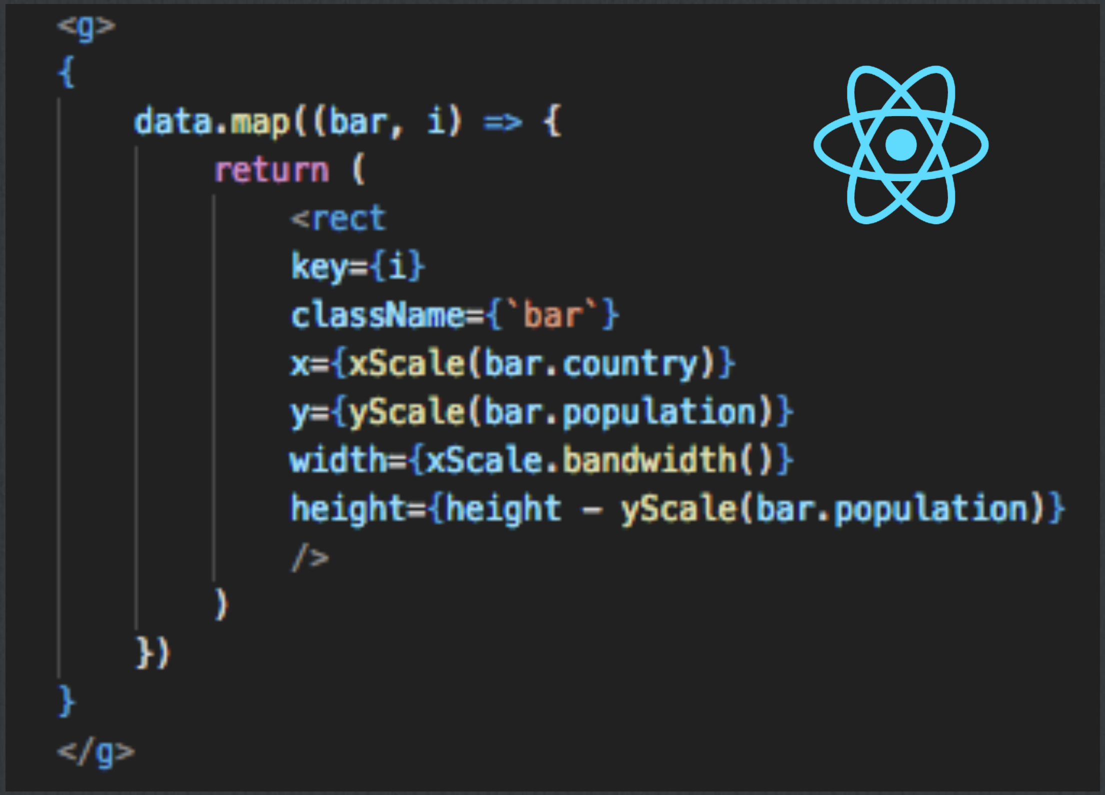
- ☐ Keep the central data storage principles of Redux
- ☐ Keep functionality of React lifecycle events and change detection
- ☐ Should be able to implement new charts by looking at D3 code (learning curve for existing D3 devs)
- ☐ Change monolithic D3 code into component-based charts

Options

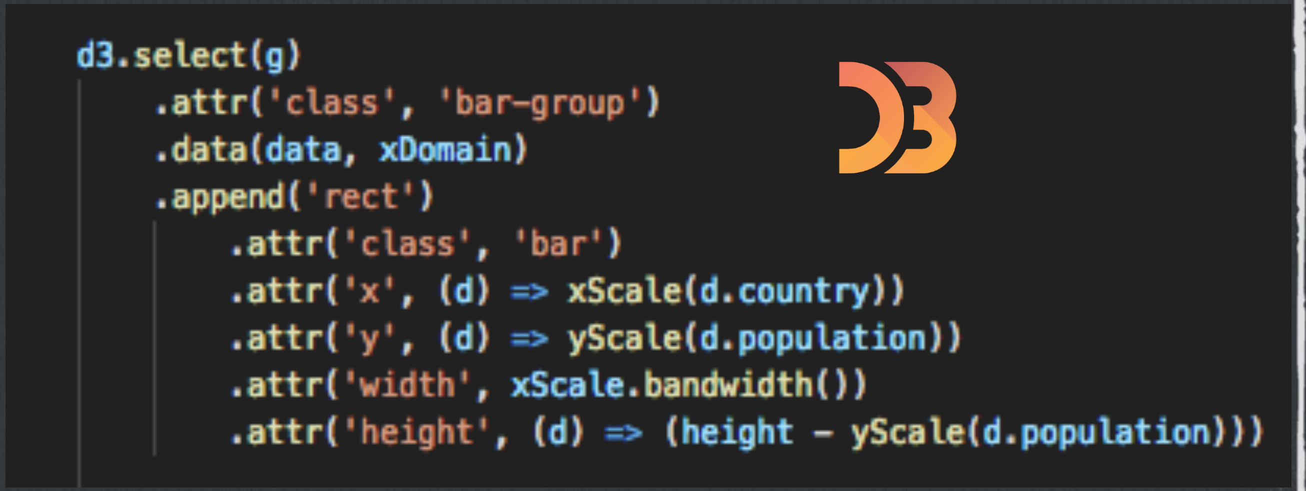
- Disable React when doing D3, like always shouldComponentUpdate = false
- Drawbacks
  - losing a lot of functionality React gives you
  - Giving up React’s Virtual DOM performance boost

Options

- Use React to draw the SVG elements instead of D3



```
<g>
{
    data.map((bar, i) => {
        return (
            <rect
            key={i}
            className={`bar`}
            x={xScale(bar.country)}
            y={yScale(bar.population)}
            width={xScale.bandwidth()}
            height={height - yScale(bar.population)}
            />
        )
    })
}
</g>
```



```
d3.select(g)
    .attr('class', 'bar-group')
    .data(data, xDomain)
    .append('rect')
        .attr('class', 'bar')
        .attr('x', (d) => xScale(d.country))
        .attr('y', (d) => yScale(d.population))
        .attr('width', xScale.bandwidth())
        .attr('height', (d) => (height - yScale(d.population)))
```

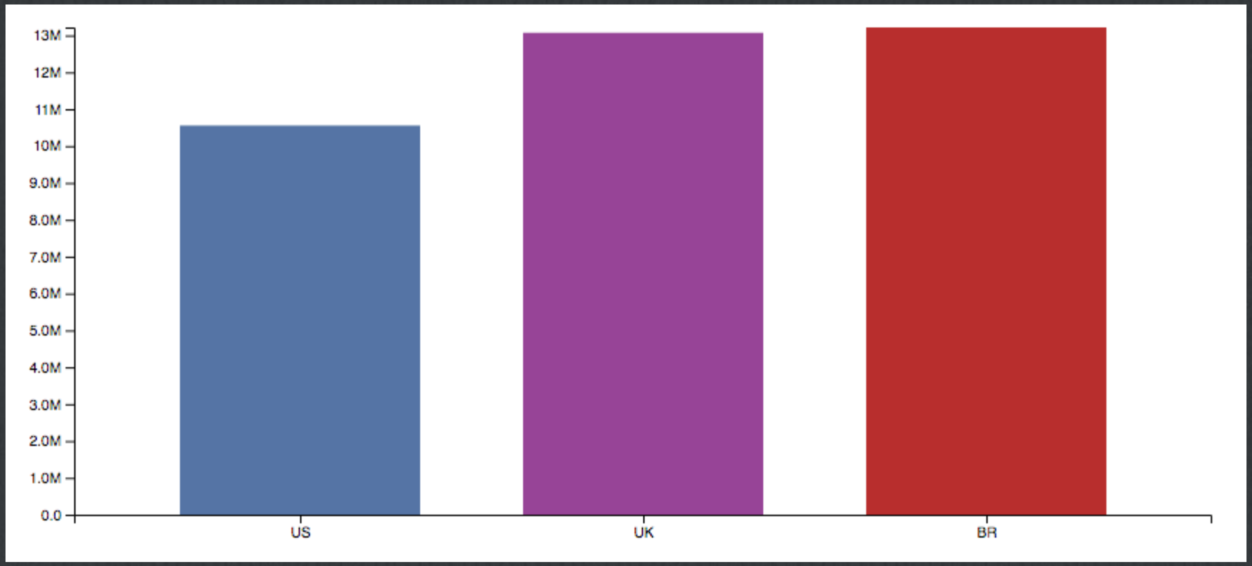
- Drawbacks:
  - not familiar to D3 devs
  - D3 transition capability lost

Options

- Use a fake DOM with D3, render it as a React element
- Still uses React lifecycle and trigger changes based on data/filters stored in Redux
- Can still use normal looking D3 code (good for D3 devs)
- Can use all D3 coolness, transitions, events, colors, etc

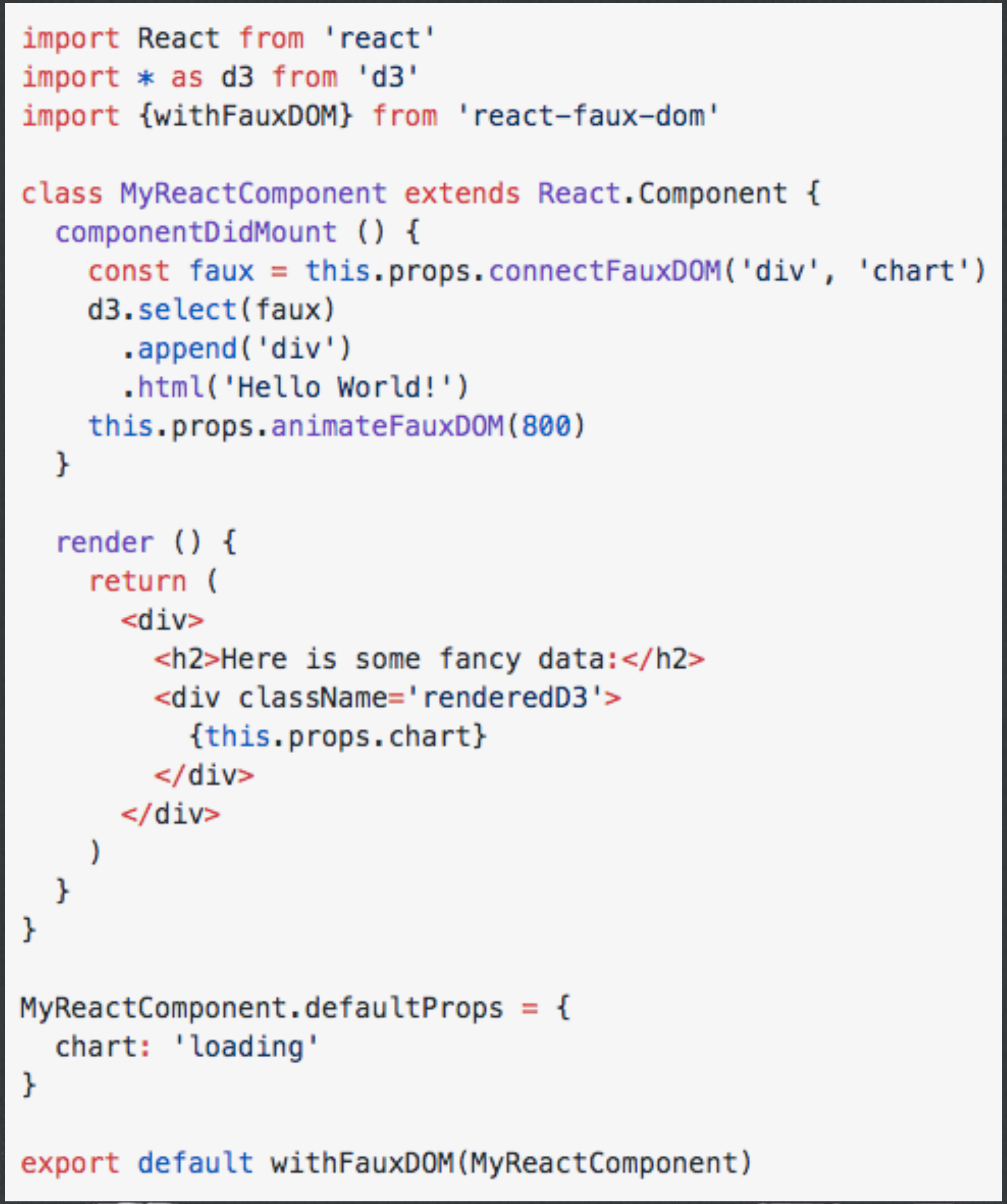
Zerrtech/react-d3-zerrtech

- Live on Github Pages at: https://zerrtech.github.io/react-d3-zerrtech/
- On Github at Zerrtech/react-d3-zerrtech
- Started an app using create-react-app
    - Added minimal packages:
        - redux
        - d3
        - react-faux-dom



Implementation

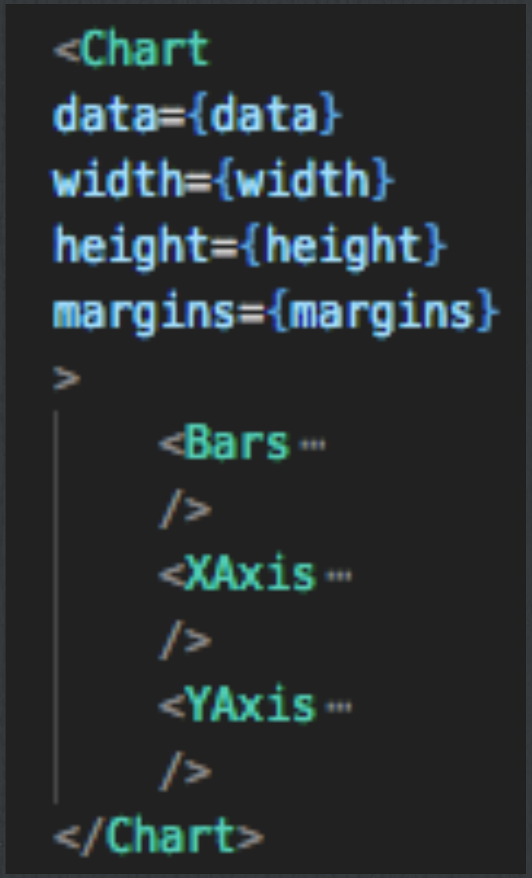
- Use a fake DOM
- Olical/react-faux-dom



```
import React from 'react'
import * as d3 from 'd3'
import {withFauxDOM} from 'react-faux-dom'

class MyReactComponent extends React.Component {
  componentDidMount () {
    const faux = this.props.connectFauxDOM('div', 'chart')
    d3.select(faux)
      .append('div')
      .html('Hello World!')
    this.props.animateFauxDOM(800)
  }

  render () {
    return (
      <div>
        <h2>Here is some fancy data:</h2>
        <div className='renderedD3'>
          {this.props.chart}
        </div>
      </div>
    )
  }
}

MyReactComponent.defaultProps = {
  chart: 'loading'
}

export default withFauxDOM(MyReactComponent)
```

Implementation

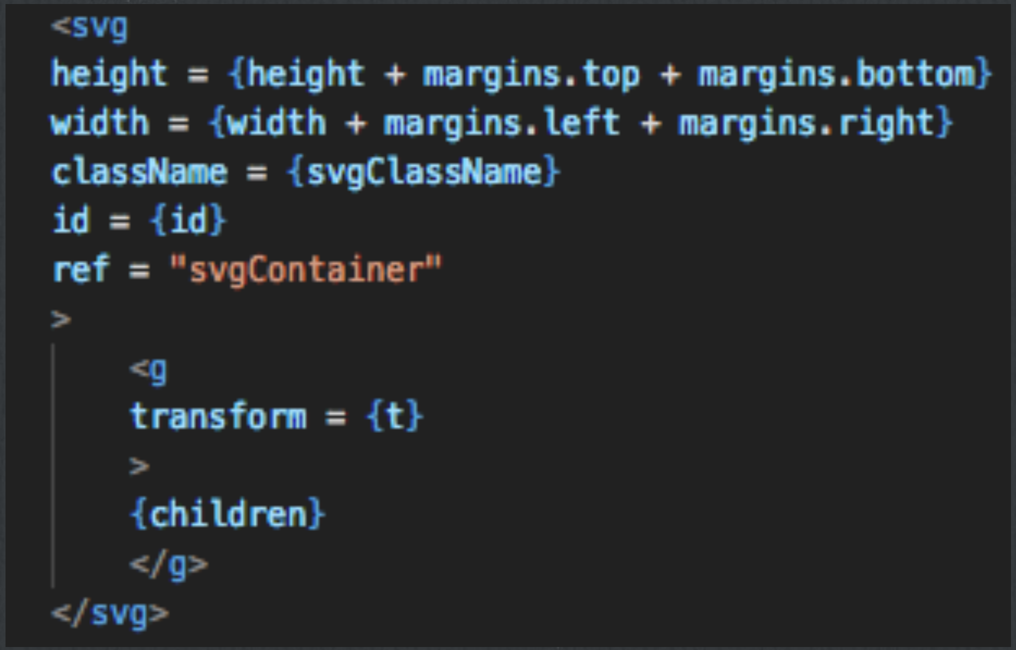
- Build a set if chart components from the vanilla D3.js here:
  - https://github.com/andrewchumich/d3-zerrtech/blob/master/index.js
- Components:
  - Chart
  - Bars
  - XAxis
  - YAxis



```
<Chart
data={data}
width={width}
height={height}
margins={margins}
>
    <Bars ⋯
    />
    <XAxis ⋯
    />
    <YAxis ⋯
    />
</Chart>
```

Chart

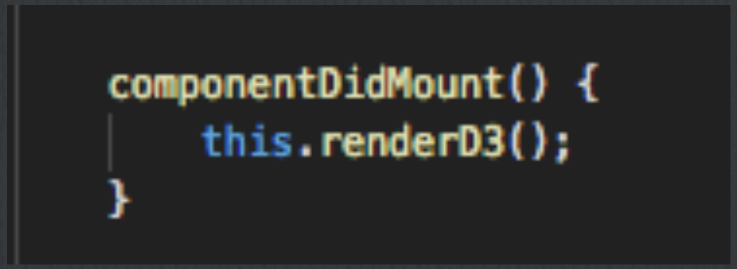
- Puts an SVG out in the DOM



```
<svg
height = {height + margins.top + margins.bottom}
width = {width + margins.left + margins.right}
className = {svgClassName}
id = {id}
ref = "svgContainer"
>
    <g
    transform = {t}
    >
    {children}
    </g>
</svg>
```

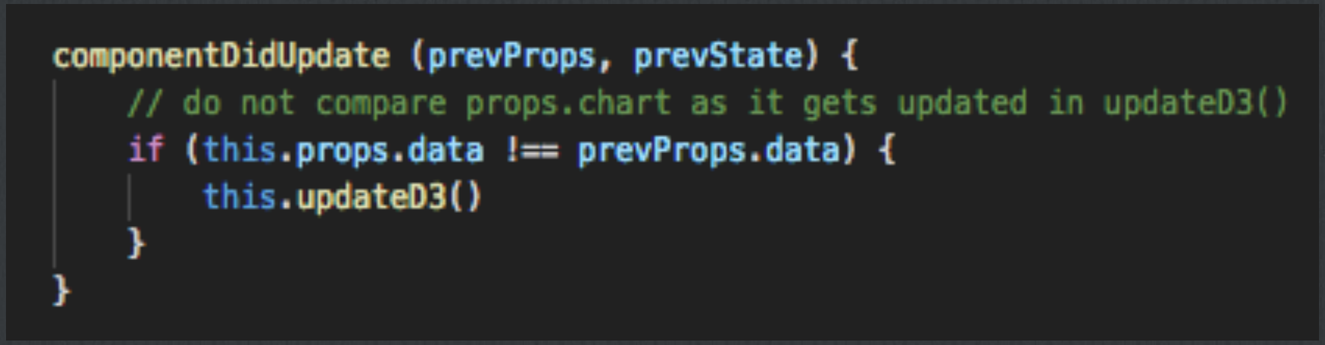
D3 first render then updates

- We add the initial render on componentDidMount



```
componentDidMount() {
    this.renderD3();
}
```

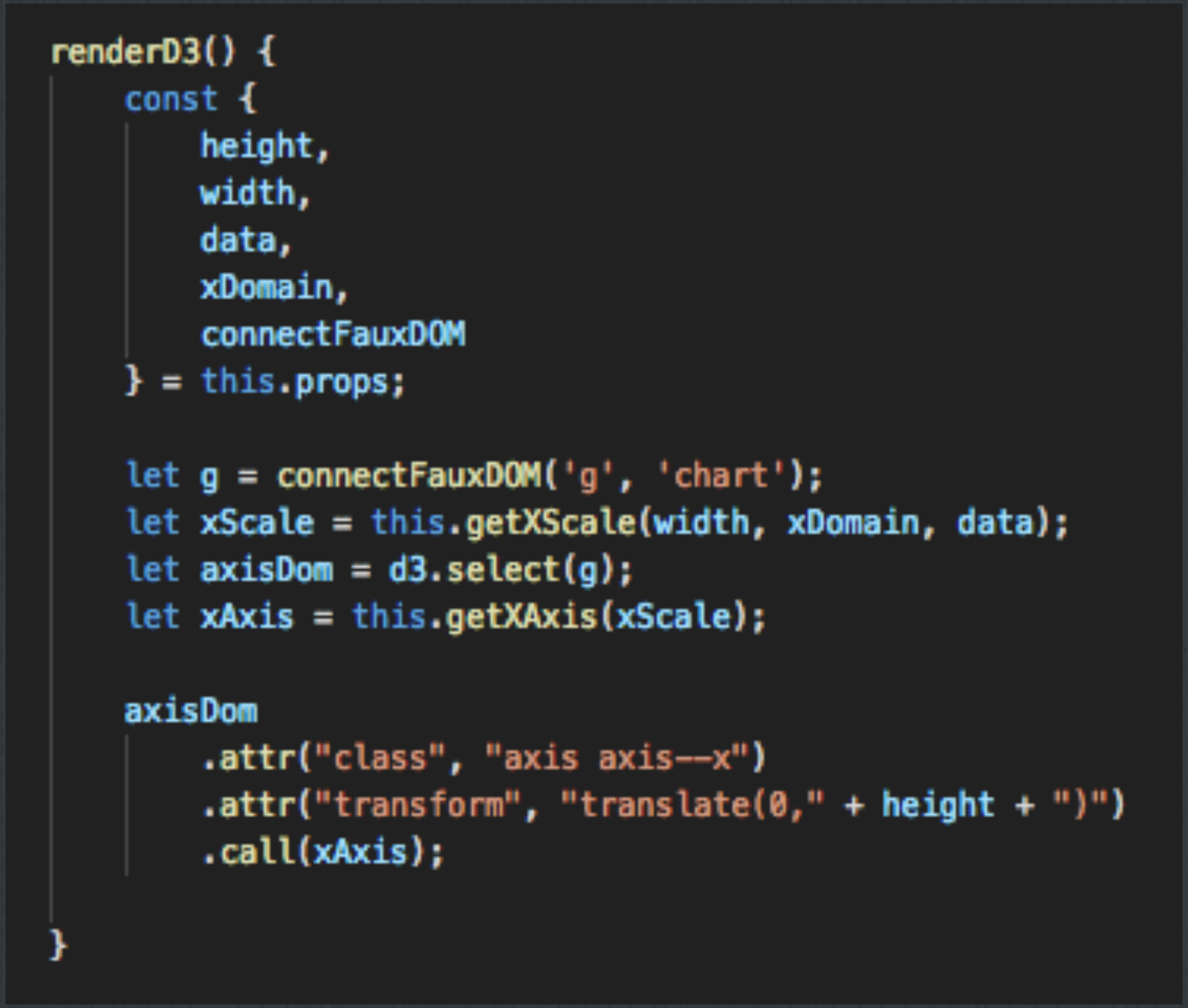
- We do updates in componentDidUpdate



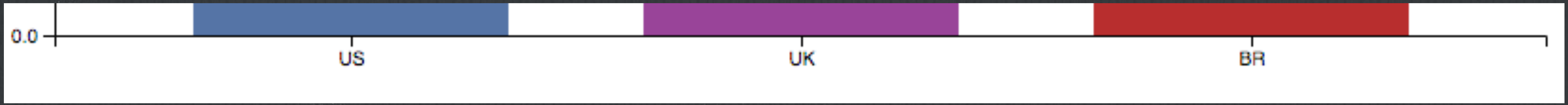
```
componentDidUpdate (prevProps, prevState) {
    // do not compare props.chart as it gets updated in updateD3()
    if (this.props.data !== prevProps.data) {
        this.updateD3()
    }
}
```

XAxis initial render

- Adds the D3 X Axis

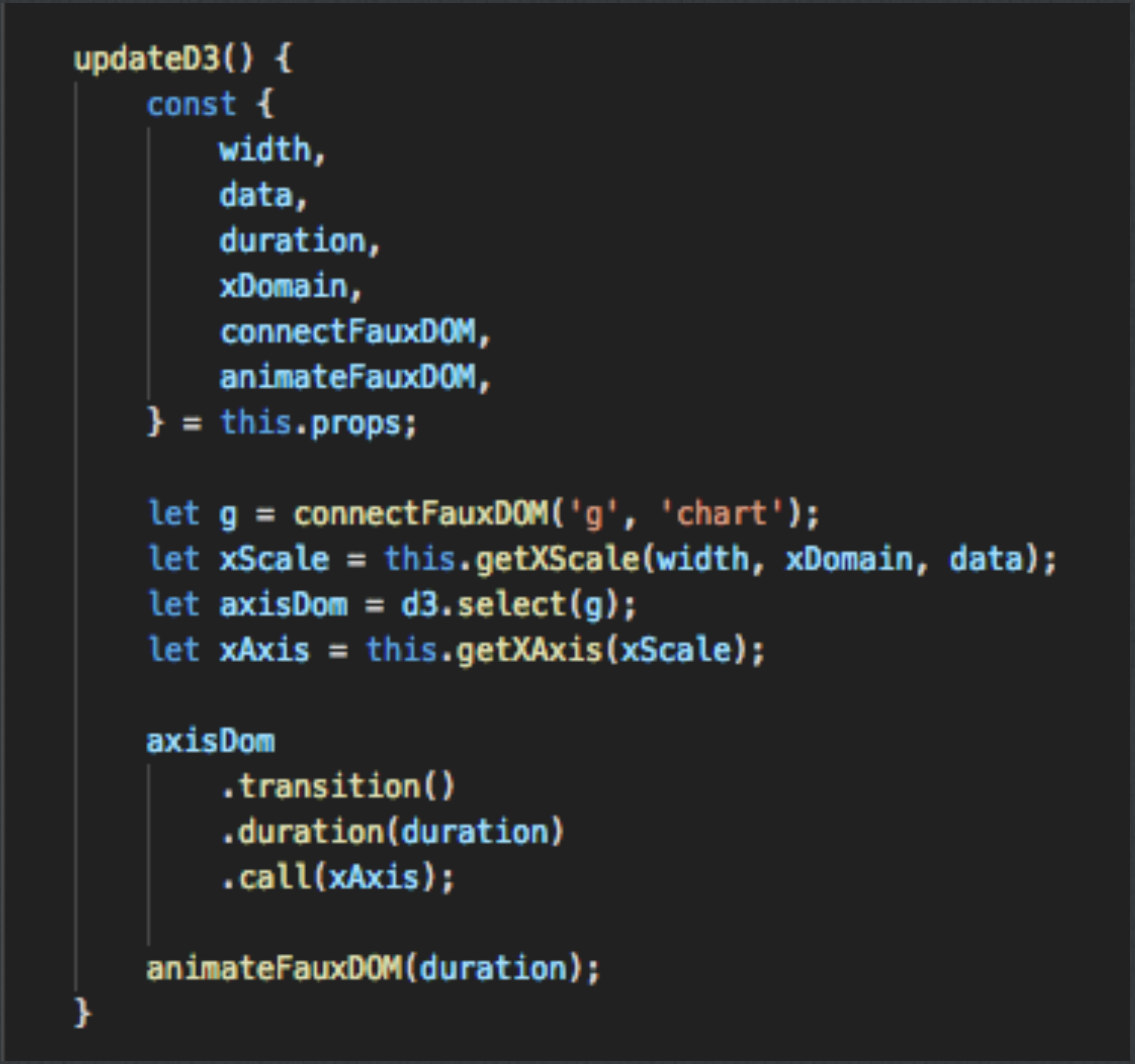


```
renderD3() {
    const {
        height,
        width,
        data,
        xDomain,
        connectFauxDOM
    } = this.props;

    let g = connectFauxDOM('g', 'chart');
    let xScale = this.getXScale(width, xDomain, data);
    let axisDom = d3.select(g);
    let xAxis = this.getXAxis(xScale);

    axisDom
        .attr("class", "axis axis--x")
        .attr("transform", "translate(0," + height + ")")
        .call(xAxis);

}
```



XAxis updates

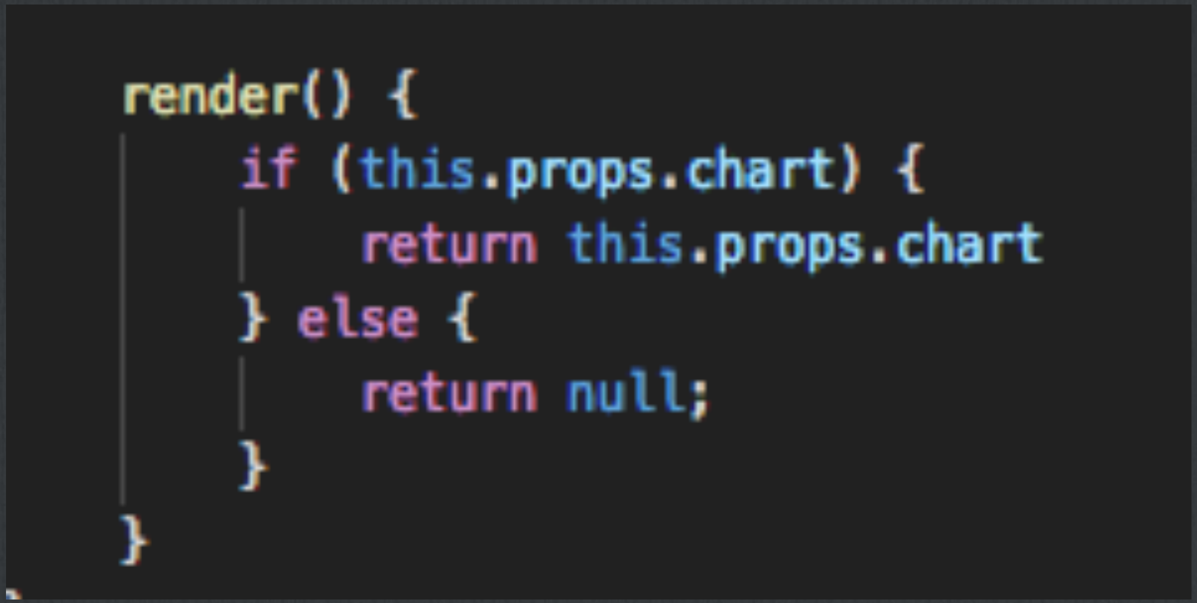
- Updates and transitions the X Axis
- animateFauxDOM updates this.props.chart every 16ms (60fps)
- Normal D3 transitions still work!



```
updateD3() {
    const {
        width,
        data,
        duration,
        xDomain,
        connectFauxDOM,
        animateFauxDOM,
    } = this.props;

    let g = connectFauxDOM('g', 'chart');
    let xScale = this.getXScale(width, xDomain, data);
    let axisDom = d3.select(g);
    let xAxis = this.getXAxis(xScale);

    axisDom
        .transition()
        .duration(duration)
        .call(xAxis);

    animateFauxDOM(duration);
}
```

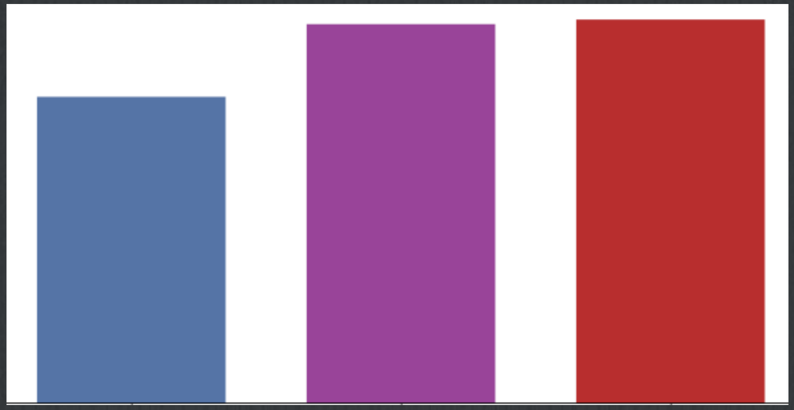
XAxis render()

- The connectFauxDOM puts the DOM on this.props.chart
- Render function becomes simple

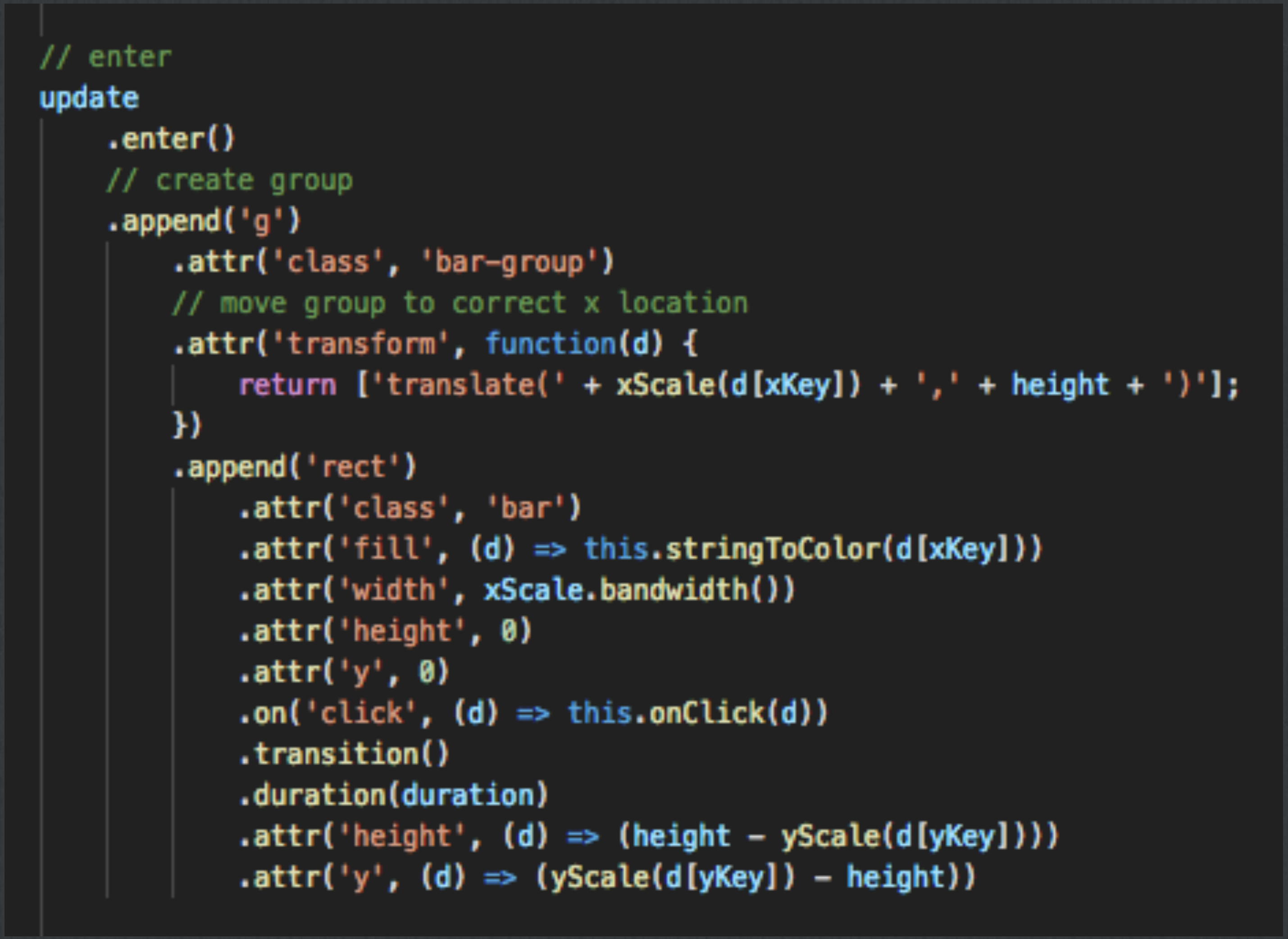


```
render() {
    if (this.props.chart) {
        return this.props.chart
    } else {
        return null;
    }
}
```

Bars - updateD3



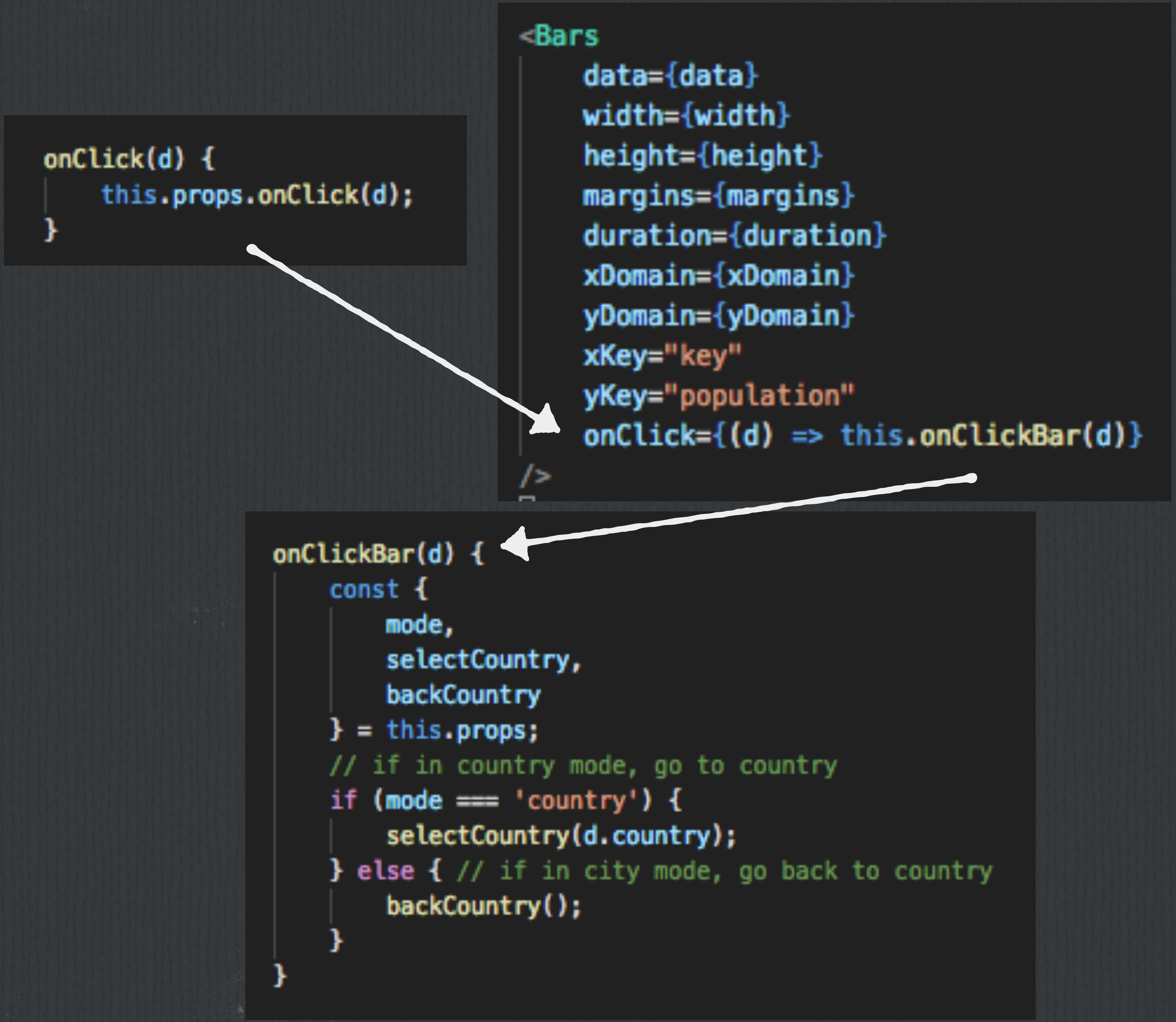
- Looks like normal D3 code



```
// enter
update
    .enter()
    // create group
    .append('g')
        .attr('class', 'bar-group')
        // move group to correct x location
        .attr('transform', function(d) {
            return ['translate(' + xScale(d[xKey]) + ',' + height + ')'];
        })
        .append('rect')
            .attr('class', 'bar')
            .attr('fill', (d) => this.stringToColor(d[xKey]))
            .attr('width', xScale.bandwidth())
            .attr('height', 0)
            .attr('y', 0)
            .on('click', (d) => this.onClick(d))
            .transition()
            .duration(duration)
            .attr('height', (d) => (height - yScale(d[yKey])))
            .attr('y', (d) => (yScale(d[yKey]) - height))
```

Bars - onClick

- Our Bars component is a dumb component
- When a bar is clicked, we just call an onClick handler passed in from the outside
- Then from outside (Population component) calls a Redux action
- We are still just using React and Redux!!!



```
onClick(d) {
    this.props.onClick(d);
}
```

```
<Bars
    data={data}
    width={width}
    height={height}
    margins={margins}
    duration={duration}
    xDomain={xDomain}
    yDomain={yDomain}
    xKey="key"
    yKey="population"
    onClick={(d) => this.onClickBar(d)}
/>
```

```
onClickBar(d) {
    const {
        mode,
        selectCountry,
        backCountry
    } = this.props;
    // if in country mode, go to country
    if (mode === 'country') {
        selectCountry(d.country);
    } else { // if in city mode, go back to country
        backCountry();
    }
}
```

Resources

- react-d3 library
- Oliver Caldwell, the guy who made react-faux-dom, blog post "D3 within React the right way"
- Thibaut Tiberghien, goes through lots of the options about integrating D3 with React, blog post "React + D3.js: Balancing Performance & Developer Experience"

Thanks! Connect with Me! We would love to build your next web app

A Software Presentation From ZERRTECH

Jeremy Zerr

Site: https://www.zerrtech.com

LinkedIn: https://www.linkedin.com/in/jrzerr