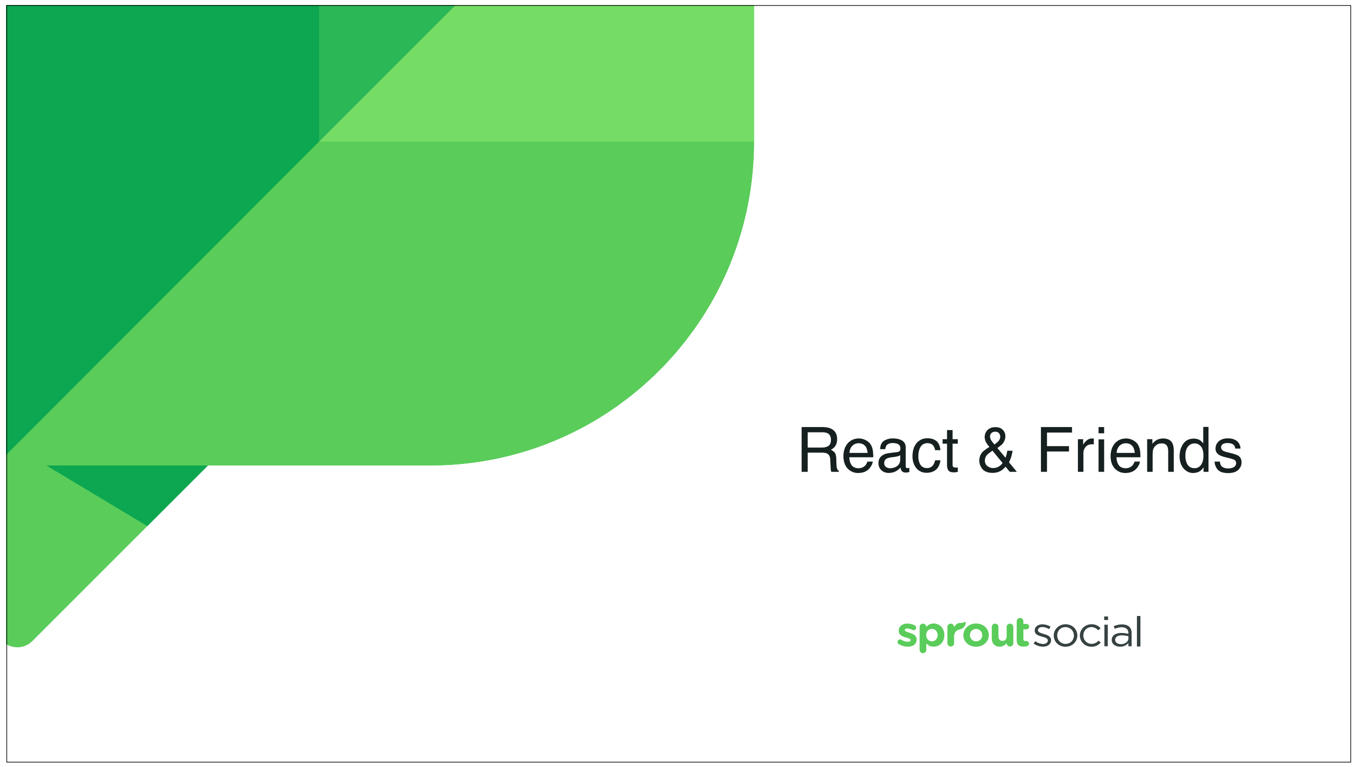


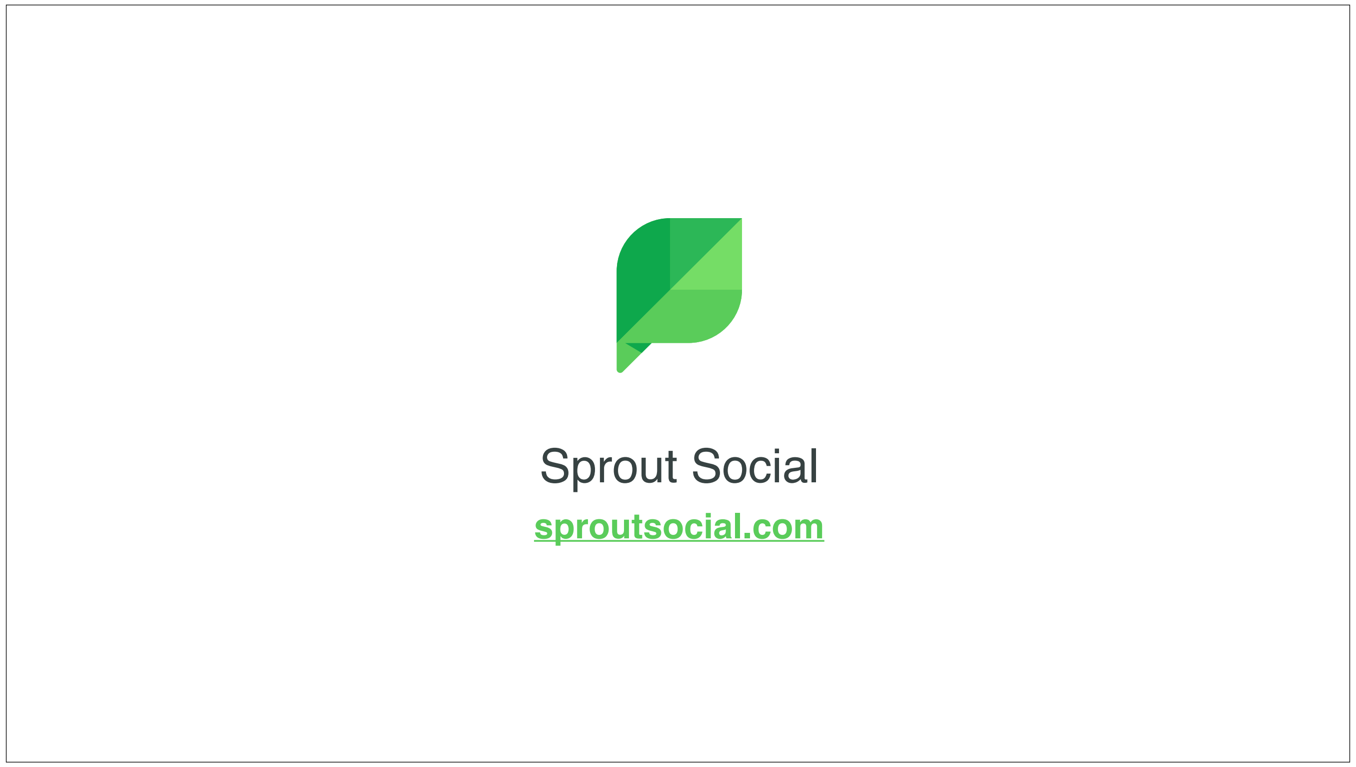
React & Friends

sproutsocial

- Welcome
- Thanks to Bill for organizing
- Thanks to Sprout for hosting
- We’re glad to have the ReactJS meetup back

Gordon Kristan

Software Engineer (~5 years)
Using React for ~1 year

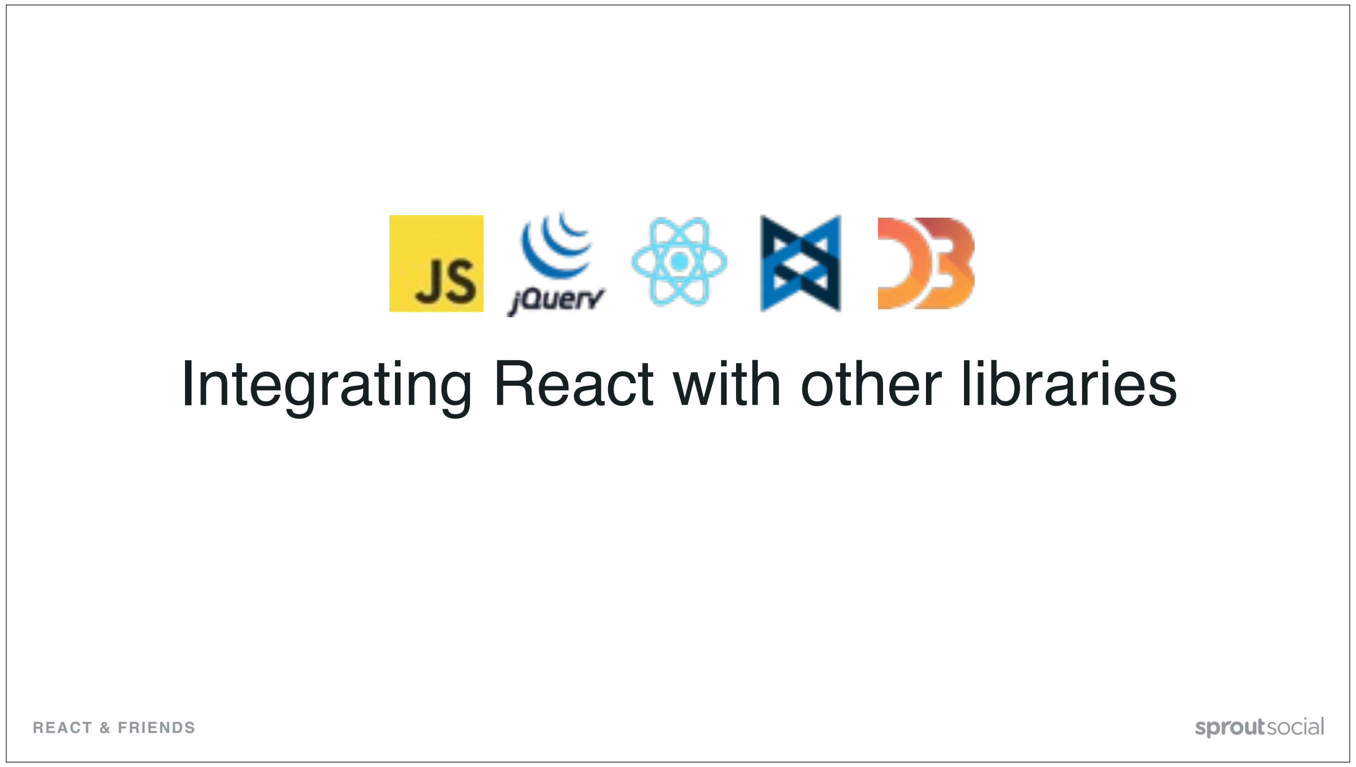


Sprout Social

sproutsocial.com

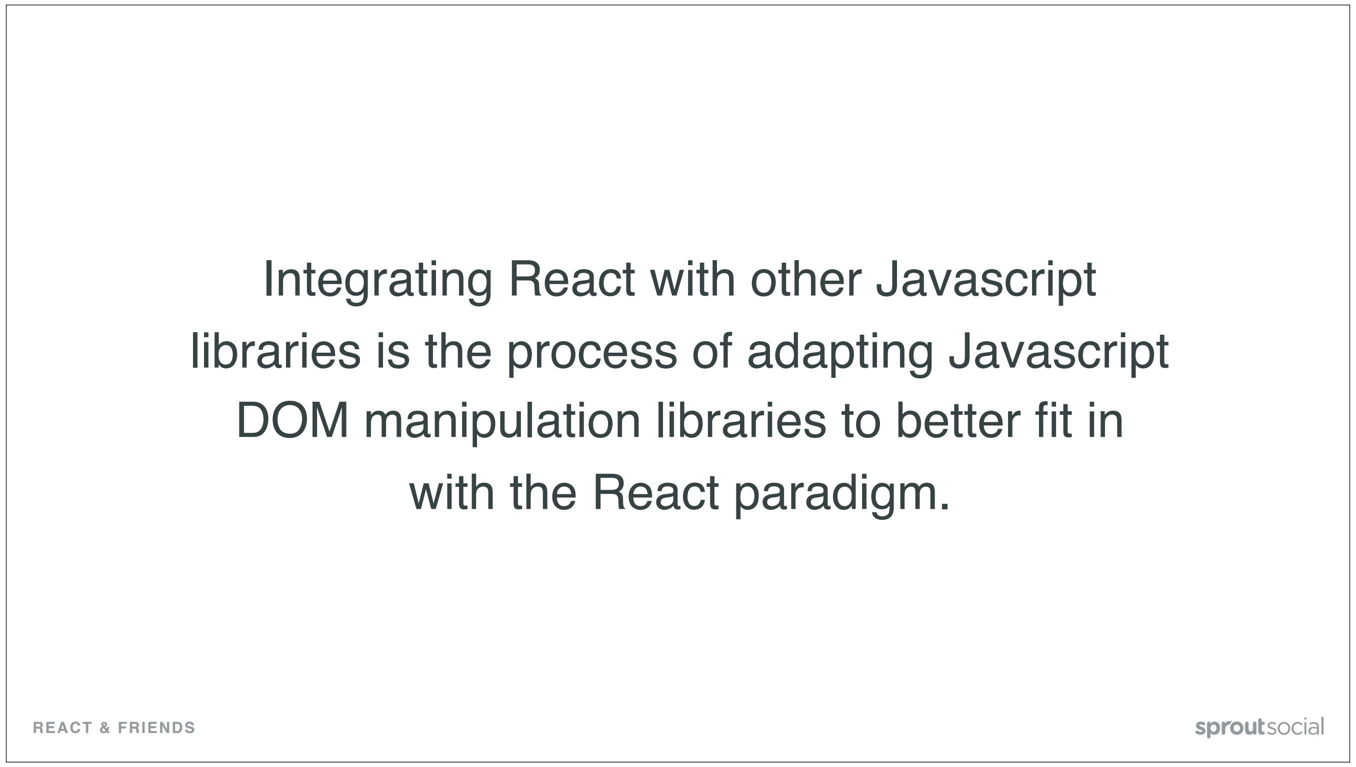
A little bit about Sprout:
- who we are
- what we do
- who are customers are

Let’s dive into it:



Integrating React with other libraries

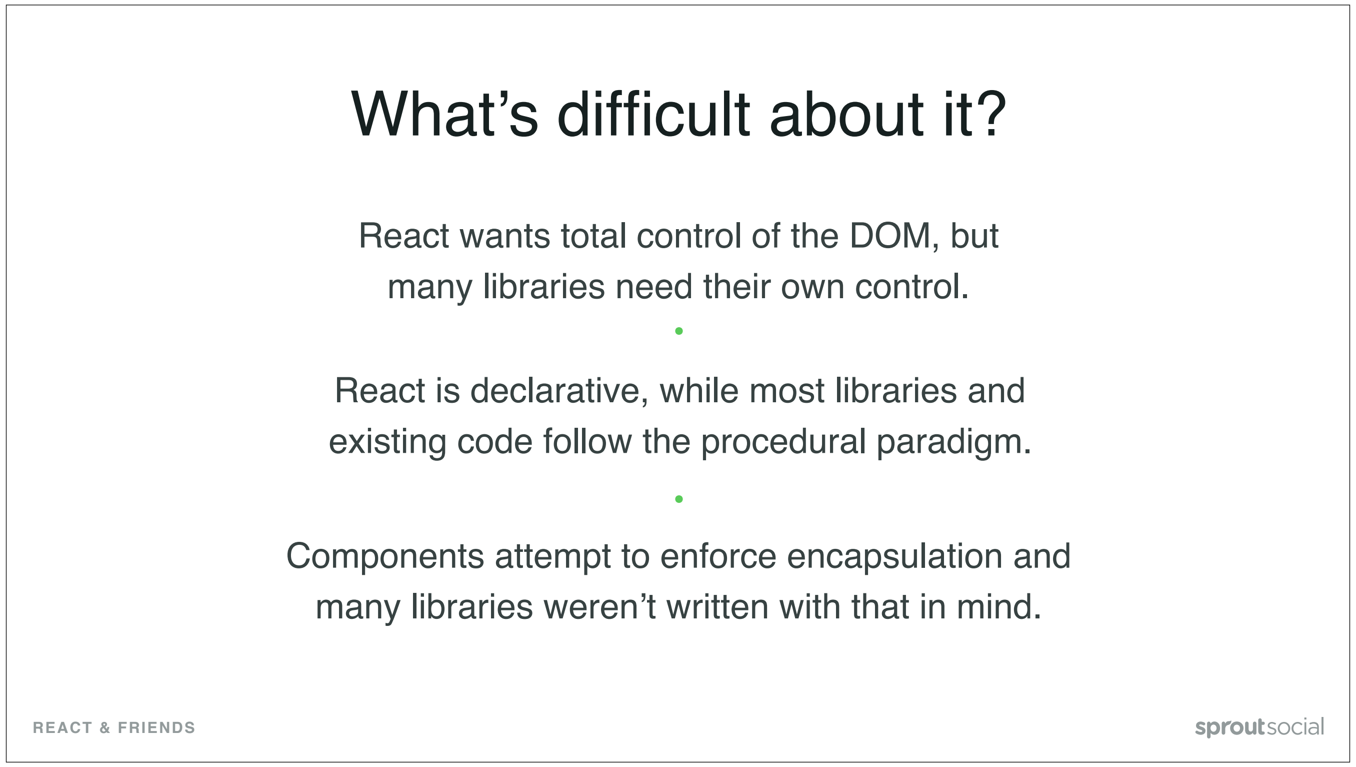
What do I mean when I say that? —>


Integrating React with other Javascript libraries is the process of adapting Javascript DOM manipulation libraries to better fit in with the React paradigm.

-Most libraries don’t follow React paradigms.
- They very procedural and not many are broken into views or components.
- Why is this important? —>

Code Re-use

Code re-use is everything. Not everybody starts with a clean slate. We have existing code that we want to use, either internal or external.



What’s difficult about it?

React wants total control of the DOM, but many libraries need their own control.

•

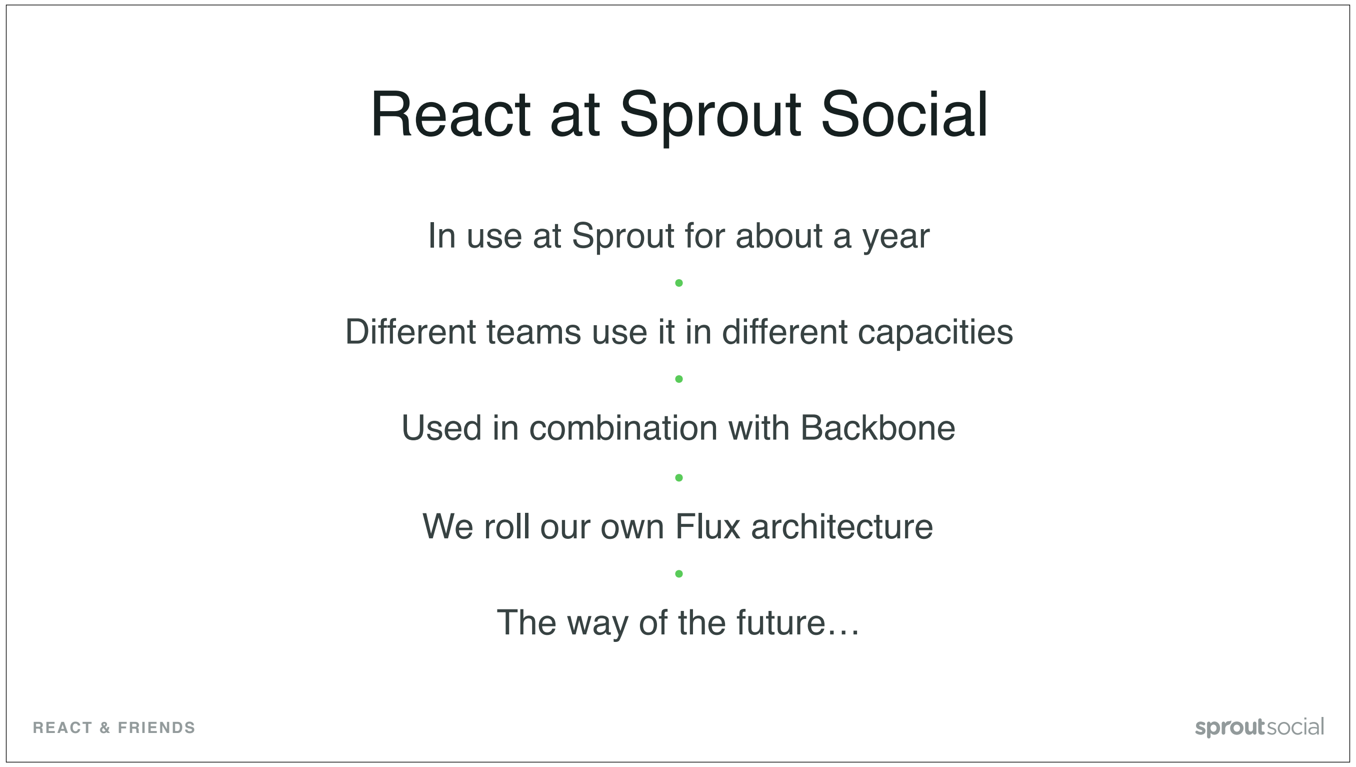
React is declarative, while most libraries and existing code follow the procedural paradigm.

•

Components attempt to enforce encapsulation and many libraries weren’t written with that in mind.

sproutsocial

Before I begin, I’d like to give you a bit of context by talking about where we are at Sprout Social.



React at Sprout Social

In use at Sprout for about a year

•

Different teams use it in different capacities

•

Used in combination with Backbone

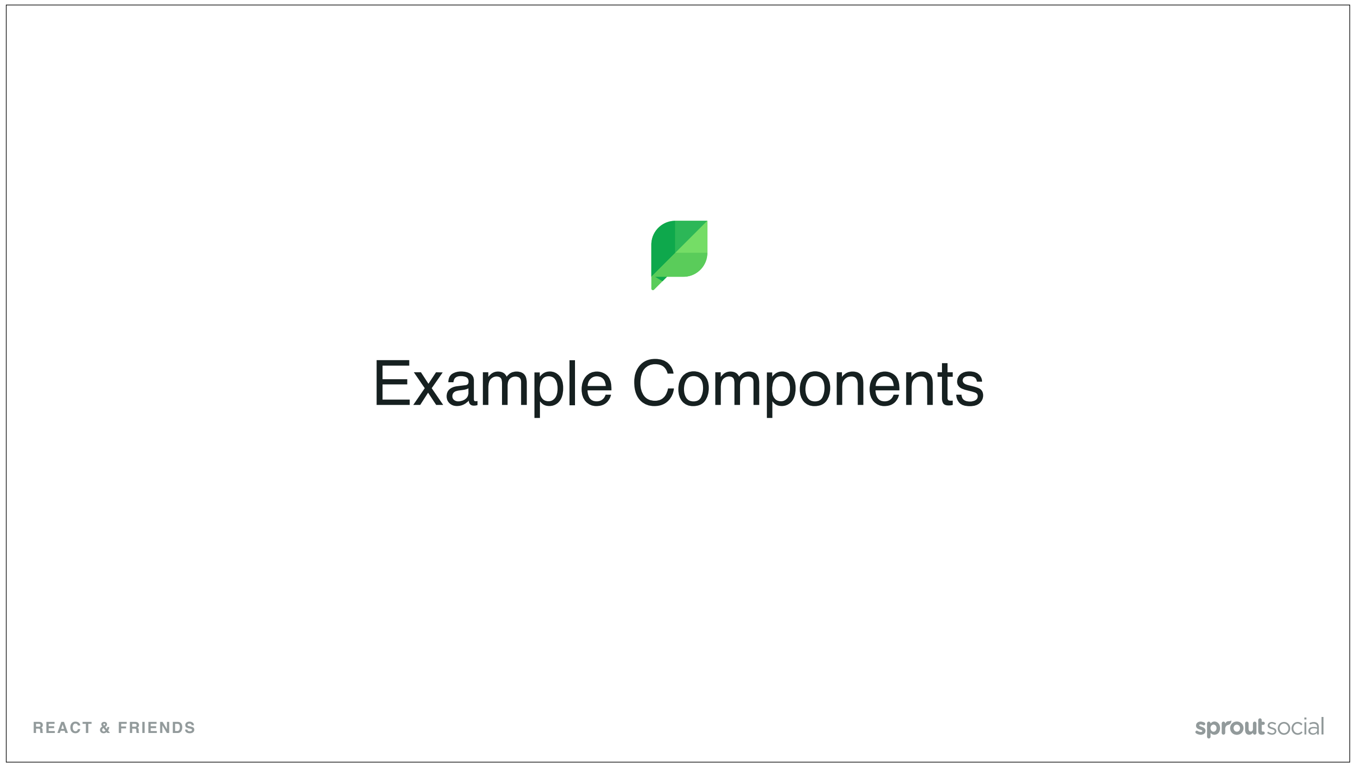
•

We roll our own Flux architecture

•

The way of the future…

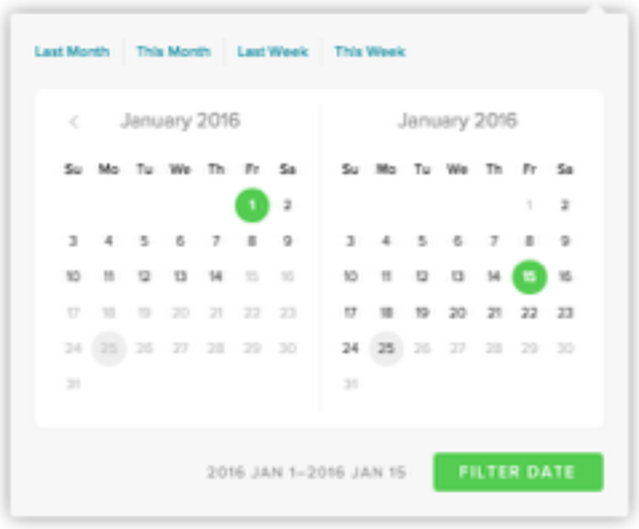
We've placed big bets on React and we've seen every one pay off. We've seen huge increases in productivity, and we've even started a pattern library. We haven't been using it long, but we're already seeing benefits.



Example Components

To start out, a few examples. Things we’ve done, problems we’ve run into, and how we solved those problems.

The Date Range Picker

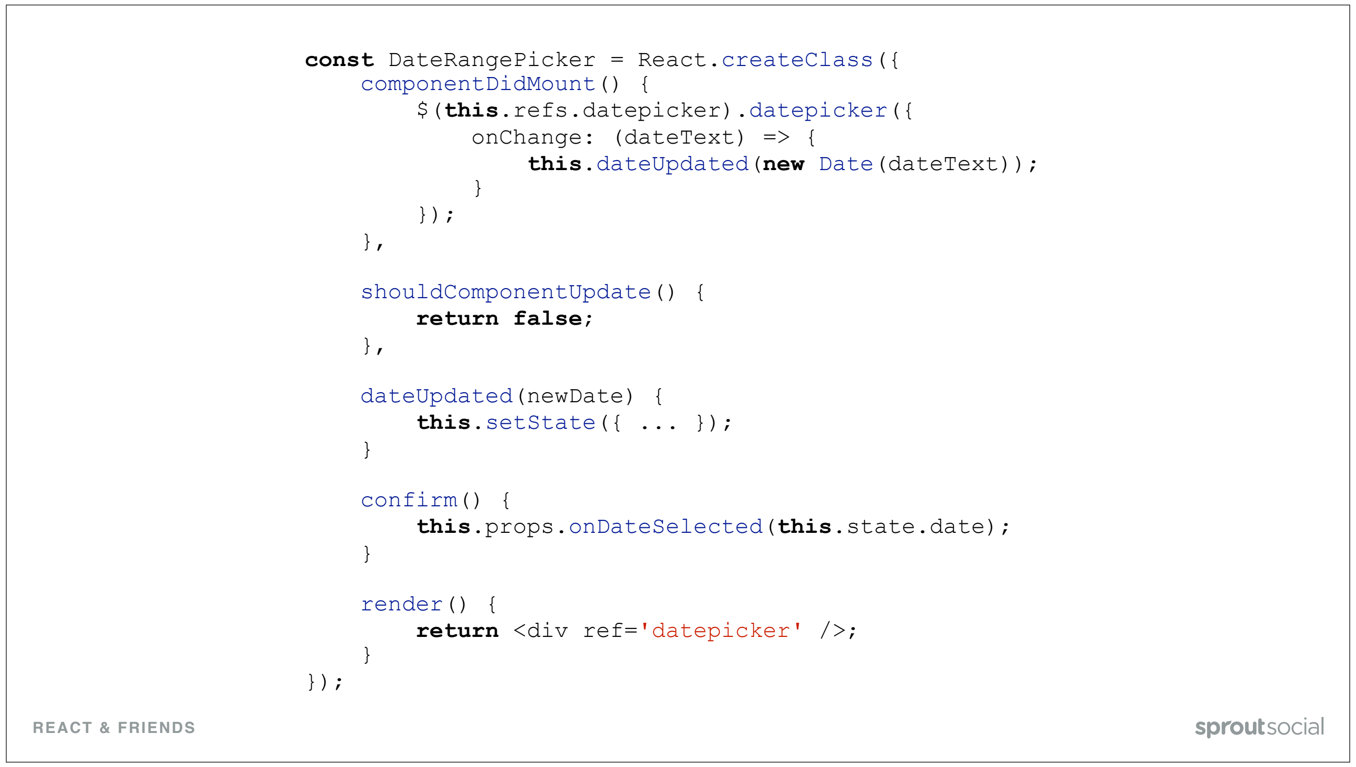


- The jQuery UI date range picker
- Our first integration with a 3rd party library
- Important because of the widespread use (a dozen reports)
- A problem that everybody runs into, but every body struggles with
- We looked at the React documentation and came up with this —>

```
const DateRangePicker = React.createClass({
    componentDidMount() {
        $(this.refs.datepicker).datepicker({ ... });
    },

    shouldComponentUpdate() {
        return false;
    },

    render() {
        return <div ref='datepicker' />;
    }
});
```

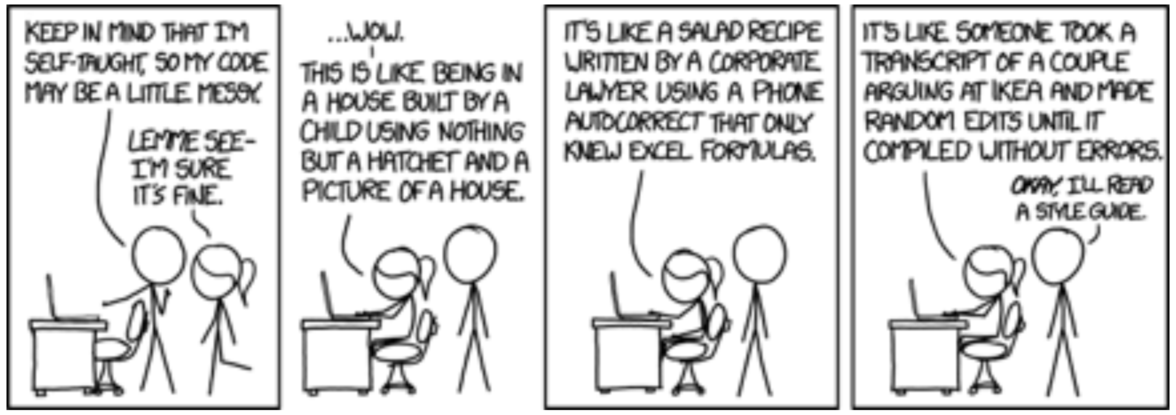
sproutsocial

- It makes sense once you see it
- shouldComponentUpdate is a bit weird, but we went with it
- From there, we set up some callbacks for change events —>

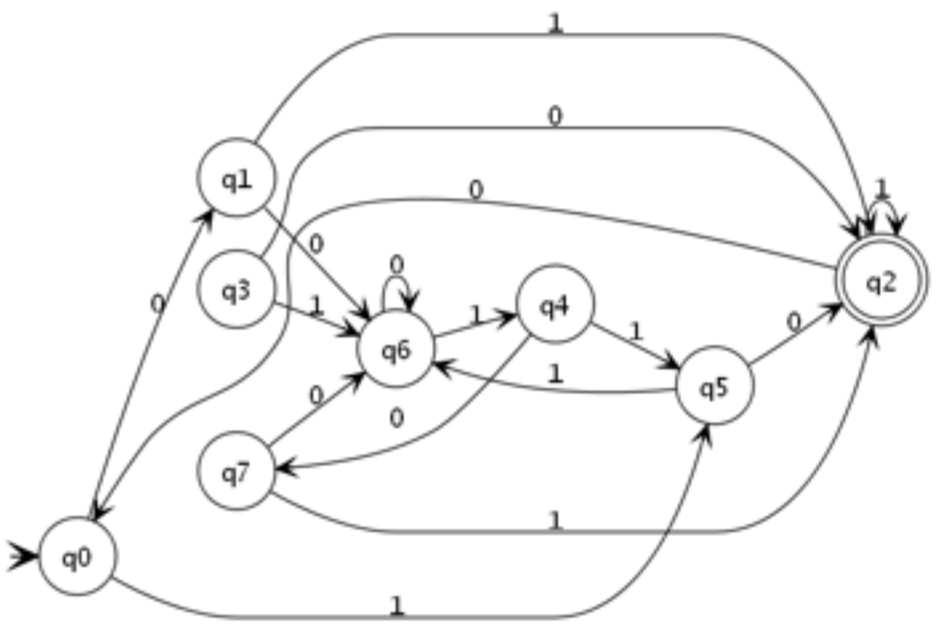


```
const DateRangePicker = React.createClass({
    componentDidMount() {
        $(this.refs.datepicker).datepicker({
            onChange: (dateText) => {
                this.dateUpdated(new Date(dateText));
            }
        });
    },

    shouldComponentUpdate() {
        return false;
    },

    dateUpdated(newDate) {
        this.setState({ ... });
    }

    confirm() {
        this.props.onDateSelected(this.state.date);
    }

    render() {
        return <div ref='datepicker' />;
    }
});
```

- A little more complicated, but still easy to grasp
- We manage state inside of the component
- We call a passed in callback when the user confirms their choice
- It worked well for a while, but we soon ran into issues —>


KEEP IN MIND THAT I'M SELF-TAUGHT, SO MY CODE MAY BE A LITTLE MESSY.
LEMME SEE- I'M SURE IT'S FINE.
...WOW.
THIS IS LIKE BEING IN A HOUSE BUILT BY A CHILD USING NOTHING BUT A HATCHET AND A PICTURE OF A HOUSE.
IT'S LIKE A SALAD RECIPE WRITTEN BY A CORPORATE LAWYER USING A PHONE AUTOCORRECT THAT ONLY KNEW EXCEL FORMULAS.
IT'S LIKE SOMEONE TOOK A TRANSCRIPT OF A COUPLE ARGUING AT IKEA AND MADE RANDOM EDITS UNTIL IT COMPILED WITHOUT ERRORS.
OKAY, I'LL READ A STYLE GUIDE.

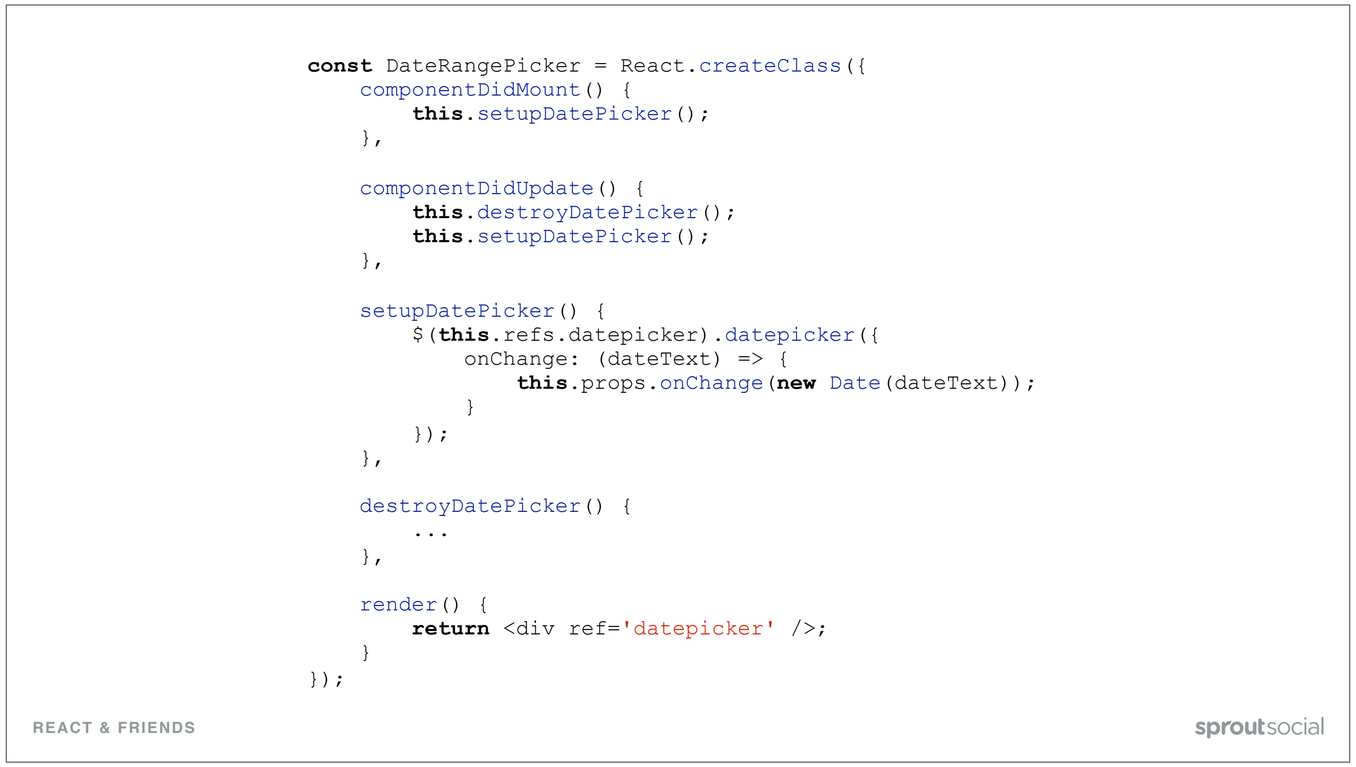
- The state was a tangled mess
- Our business requirements grew: preset ranges, upper and lower bounds, same day restrictions, time zones, etc
- It was becoming more and more complicated —>



- Before you knew it, we had 10 states and 100 ways to transition between them
- The issues weren't that bad, we were managing, but we couldn't help but feel as if there was a better way
- We were doing the exact opposite of what React wanted – what we loved so much about React
- So we came up with a crazy idea —>

- What if, instead of managing state, we made the component more declarative?
- Instead of transitioning between states, we would simply rebuild the component every time.
- Just like React, when something changes we tear down everything and re-render it all again
- And I know what you're thinking —> there's no way that's going to work

- But we tried it anyway. What’s the worst that could happen?
- After a few hours of playing around with it, we had something that looked like this —>



```
const DateRangePicker = React.createClass({
    componentDidMount() {
        this.setupDatePicker();
    },

    componentDidUpdate() {
        this.destroyDatePicker();
        this.setupDatePicker();
    },

    setupDatePicker() {
        $(this.refs.datepicker).datepicker({
            onChange: (dateText) => {
                this.props.onChange(new Date(dateText));
            }
        });
    },

    destroyDatePicker() {
        ...
    },

    render() {
        return <div ref='datepicker' />;
    }
});
```

- We eliminated the internal state entirely
- We pushed the state logic into the consumer, it’s a controlled component now
- We destroyed and rebuilt the jQuery picker every time
- To our surprise, it worked beautifully, even in IE9!
- We committed our changes and lived happily ever after in the land of React, and learned a few lessons along the way

Lessons Learned

Try something crazy, that’s what led to the invention of React

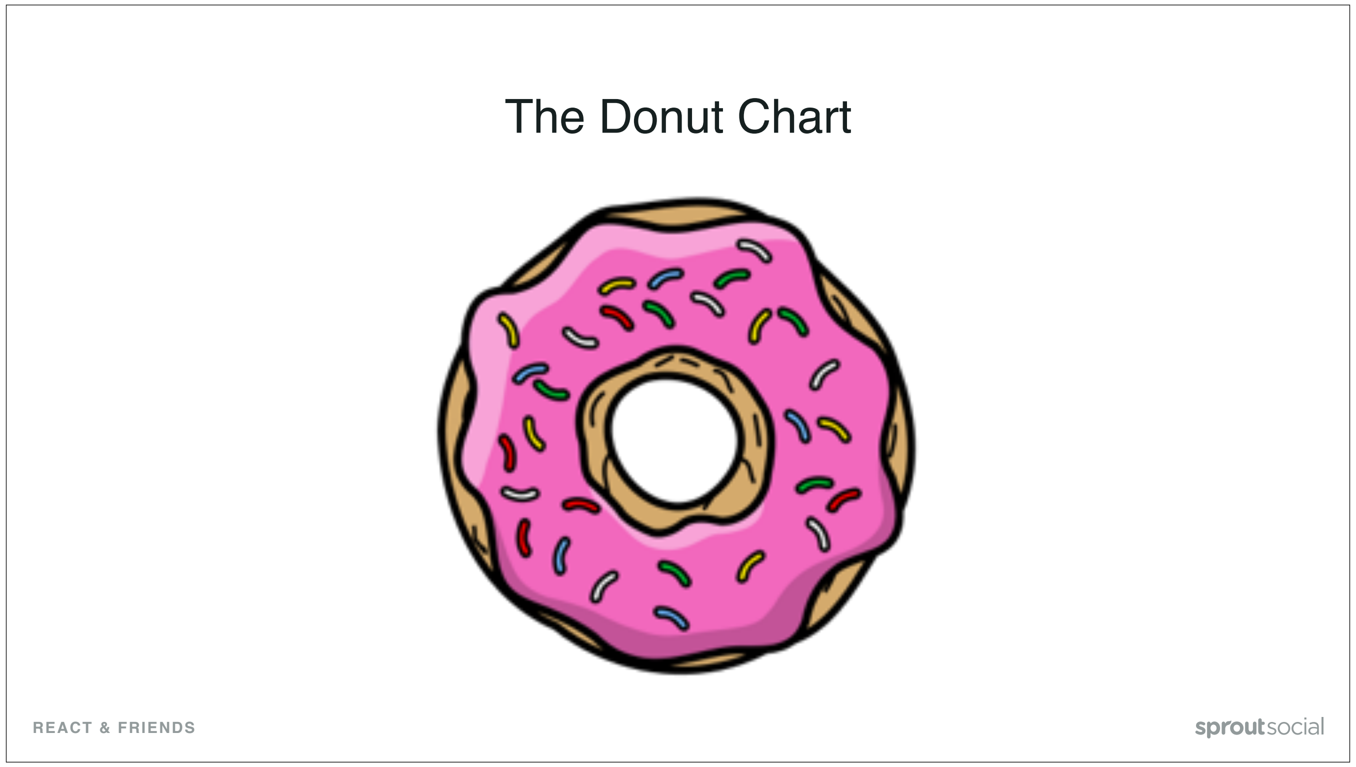
•

Twist the external library to work with React, not the other way around

•

“…premature optimization is the root of all evil…”

- Donald Knuth

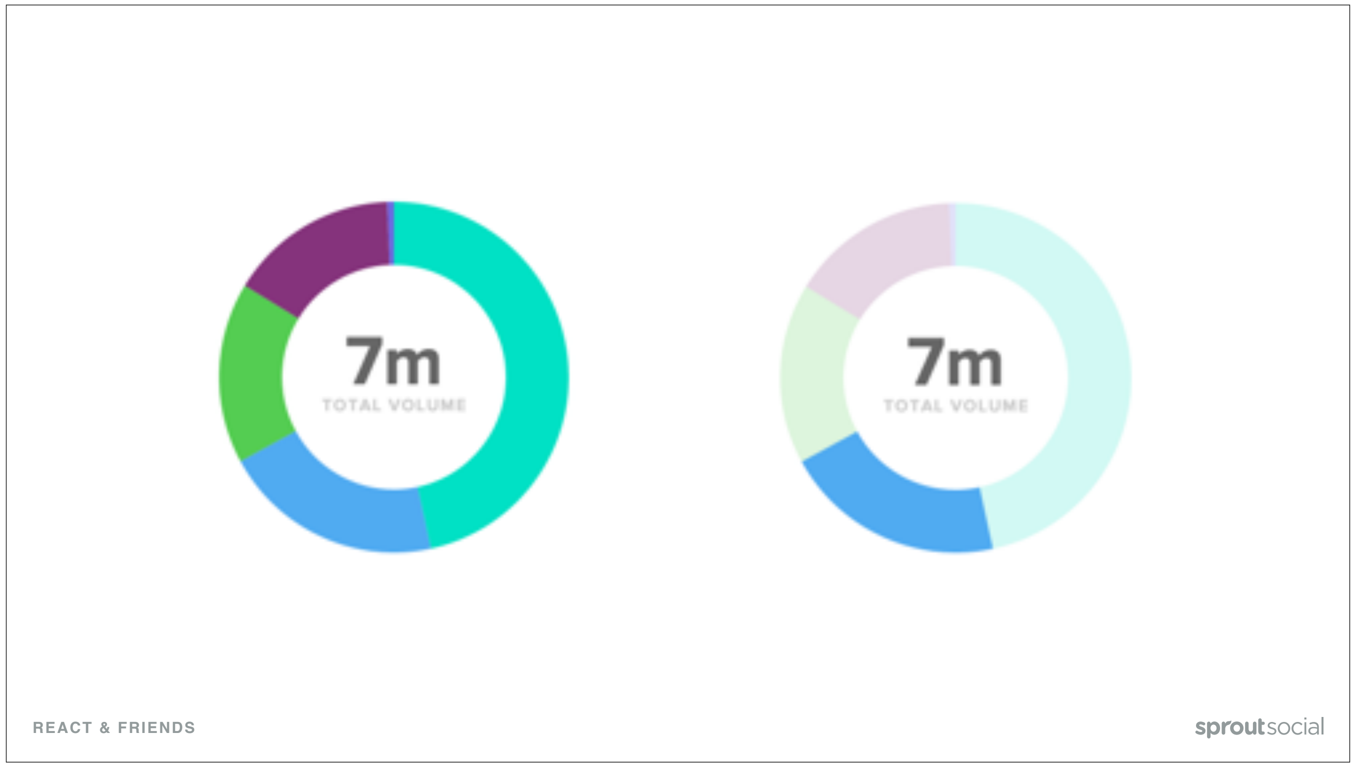


The Donut Chart

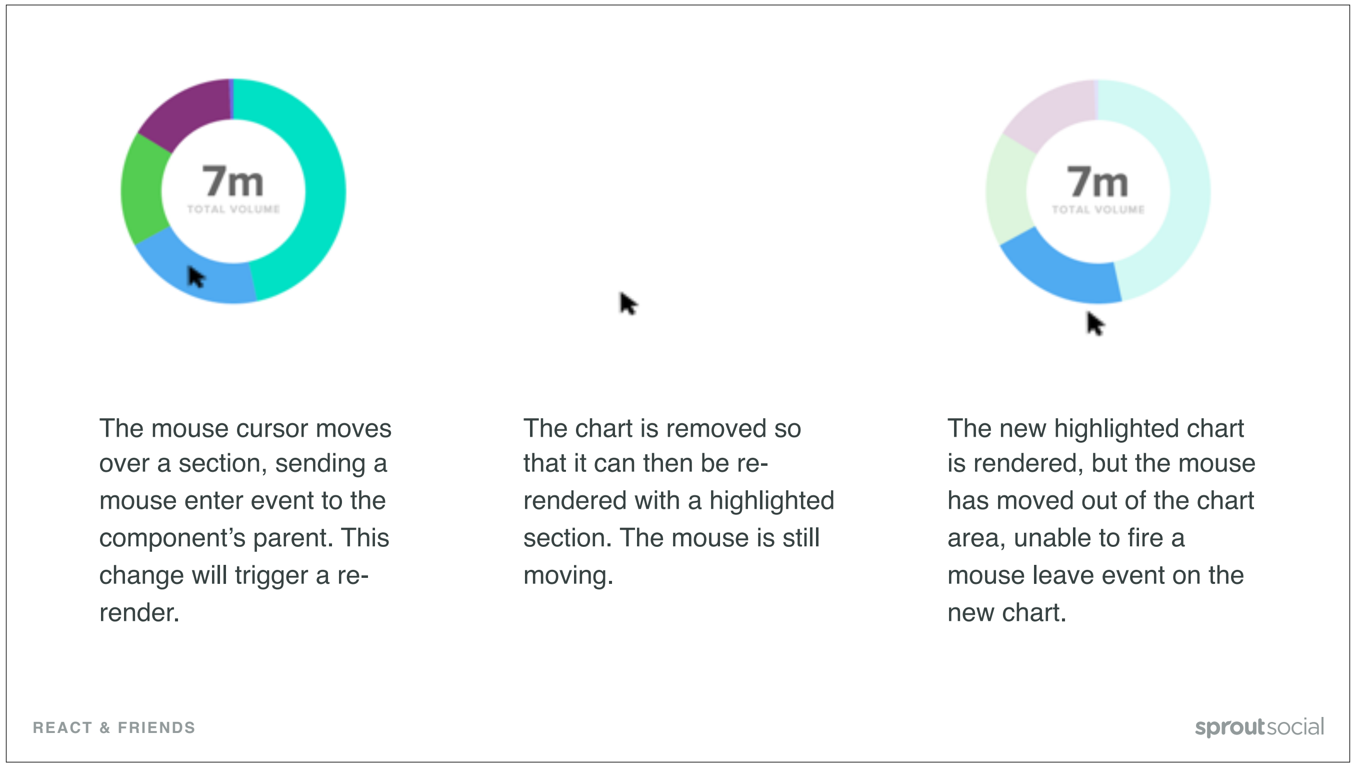
REACT & FRIENDS

sproutsocial

- After the date picker we were tasked with creating the donut chart
- We had done other components, but most of them followed the same pattern as the date picker
- This one required animation, a well-known problem in the React community
- To give you a bit of context, here’s our donut charts —>



- They're built with D3, a data visualization library
- We feed in numbers, and D3 builds the chart according to the ratios
- In addition, we have a hover state (light vs dim)
- We had two problems: one with the rough transition
- One that was a little more complicated

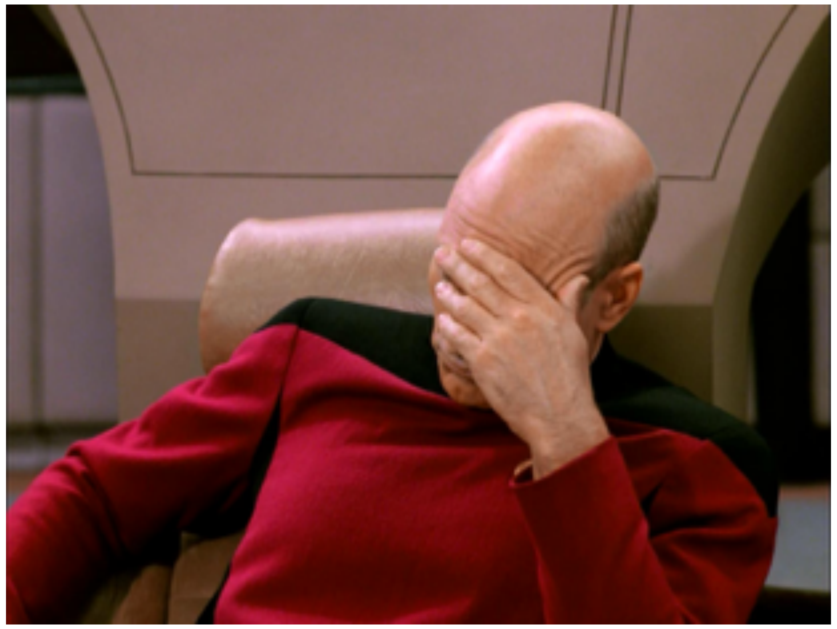


The mouse cursor moves over a section, sending a mouse enter event to the component's parent. This change will trigger a re-render.

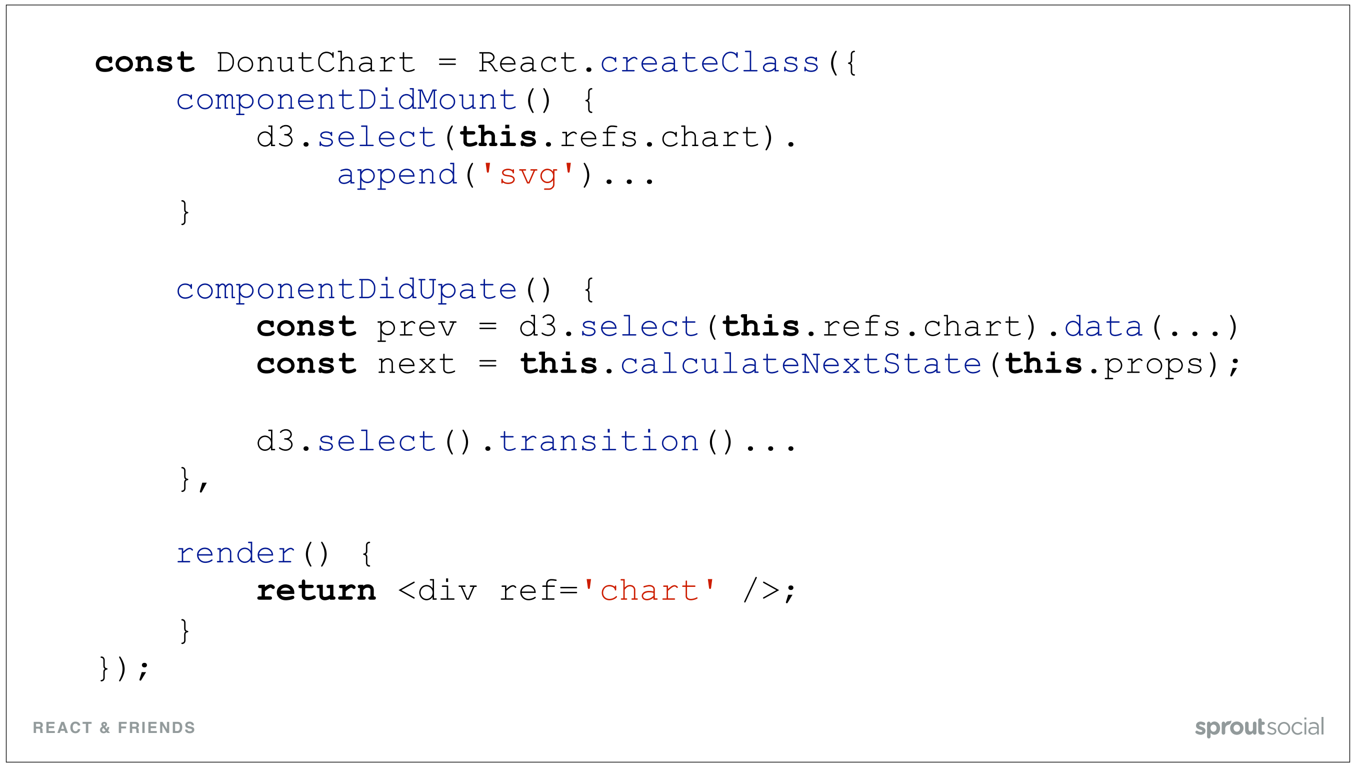
The chart is removed so that it can then be re-rendered with a highlighted section. The mouse is still moving.

The new highlighted chart is rendered, but the mouse has moved out of the chart area, unable to fire a mouse leave event on the new chart.

- We were running into issues with the hover state
- If you quickly moved your mouse, the chart would say in the hover state
- (Explanation)
- It turned out to be caused by us removing the elements from the DOM
- We knew what we had to do: start managing DOM state again

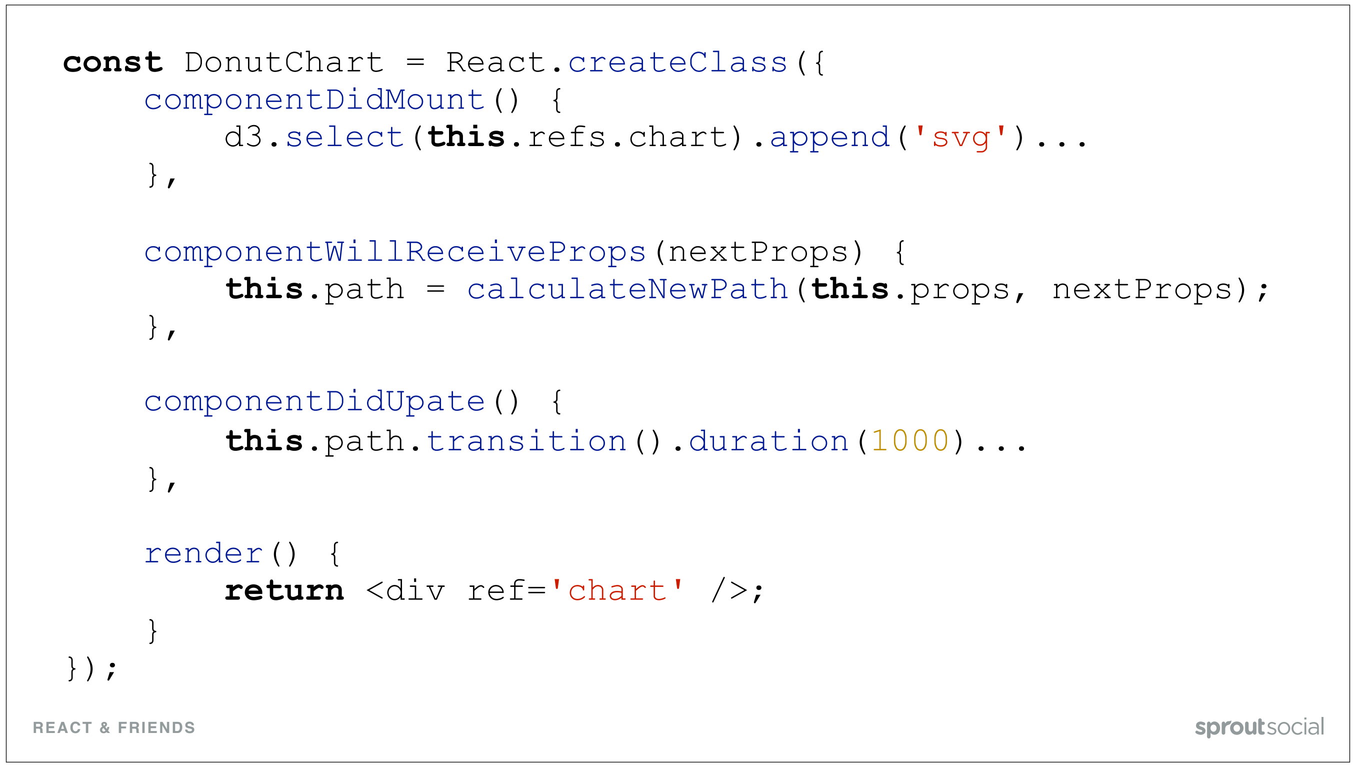


- After we had learned our lesson about keeping DOM state, we were going to try it again
- That meant state diffing and transitions
- We determined the only way to do what we wanted was to gradually change from one state to another
- Our code would look something like this —>



```
const DonutChart = React.createClass({
    componentDidMount() {
        d3.select(this.refs.chart).
            append('svg')...
    }

    componentDidUpate() {
        const prev = d3.select(this.refs.chart).data(...)
        const next = this.calculateNextState(this.props);

        d3.select().transition()...
    },

    render() {
        return <div ref='chart' />;
    }
});
```

- We would perform different logic in initial render vs updates
- But we found a few things to help along the way
- React helped us with the lifecycle hooks (that’s what they’re for!)
- D3 helped us with its transition support
- Our new component looked something like this —>

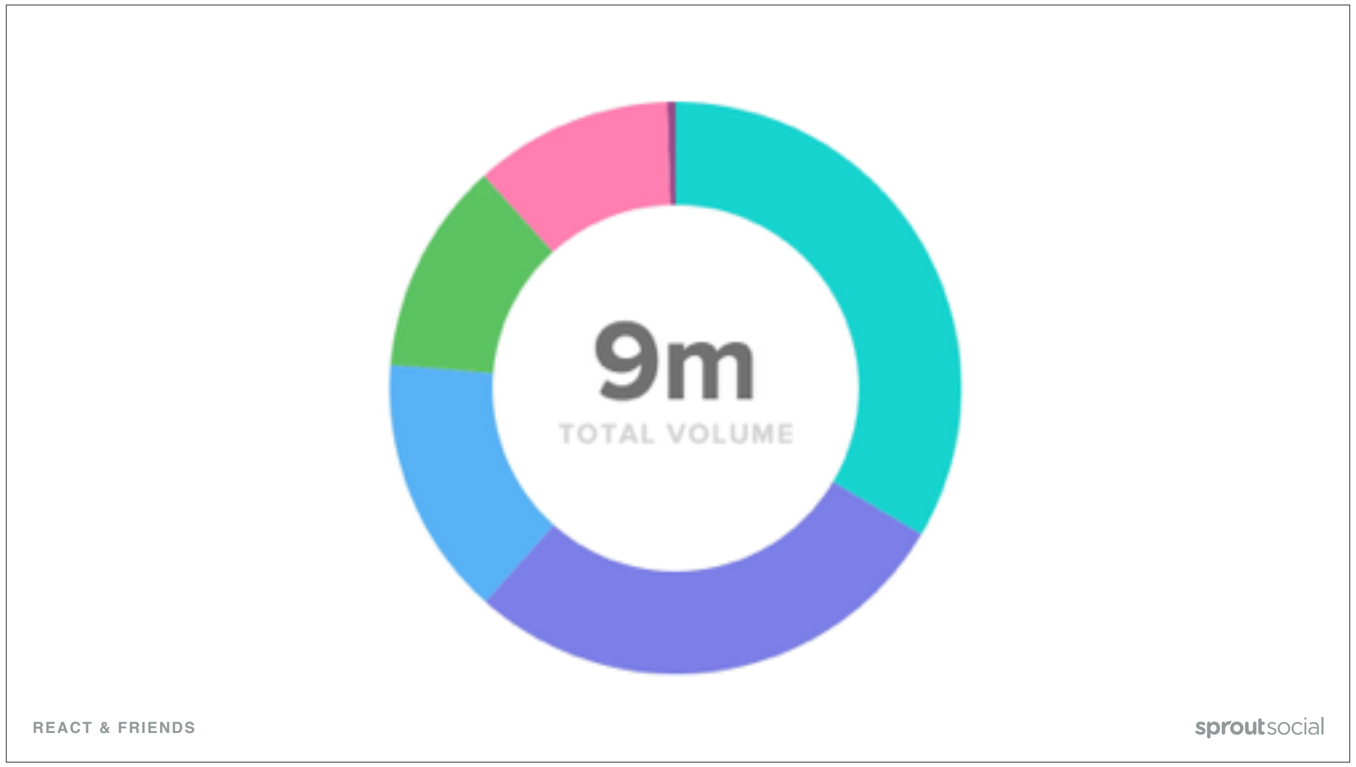


```
const DonutChart = React.createClass({
    componentDidMount() {
        d3.select(this.refs.chart).append('svg')...
    },

    componentWillReceiveProps(nextProps) {
        this.path = calculateNewPath(this.props, nextProps);
    },

    componentDidUpate() {
        this.path.transition().duration(1000)...
    },

    render() {
        return <div ref='chart' />;
    }
});
```

REACT & FRIENDS

sproutsocial

- React helped us with the componentWillRecieveProps hook (told us the previous and next states)
- D3 helped us by having built-in transition support
- We still had separate render/update logic, but it was clean and understandable
- What was the final result? —>



sproutsocial

- We not only fixed our hover state bug, but we had some fancy animations to go along with it
- We also learned a few lessons

Lessons Learned

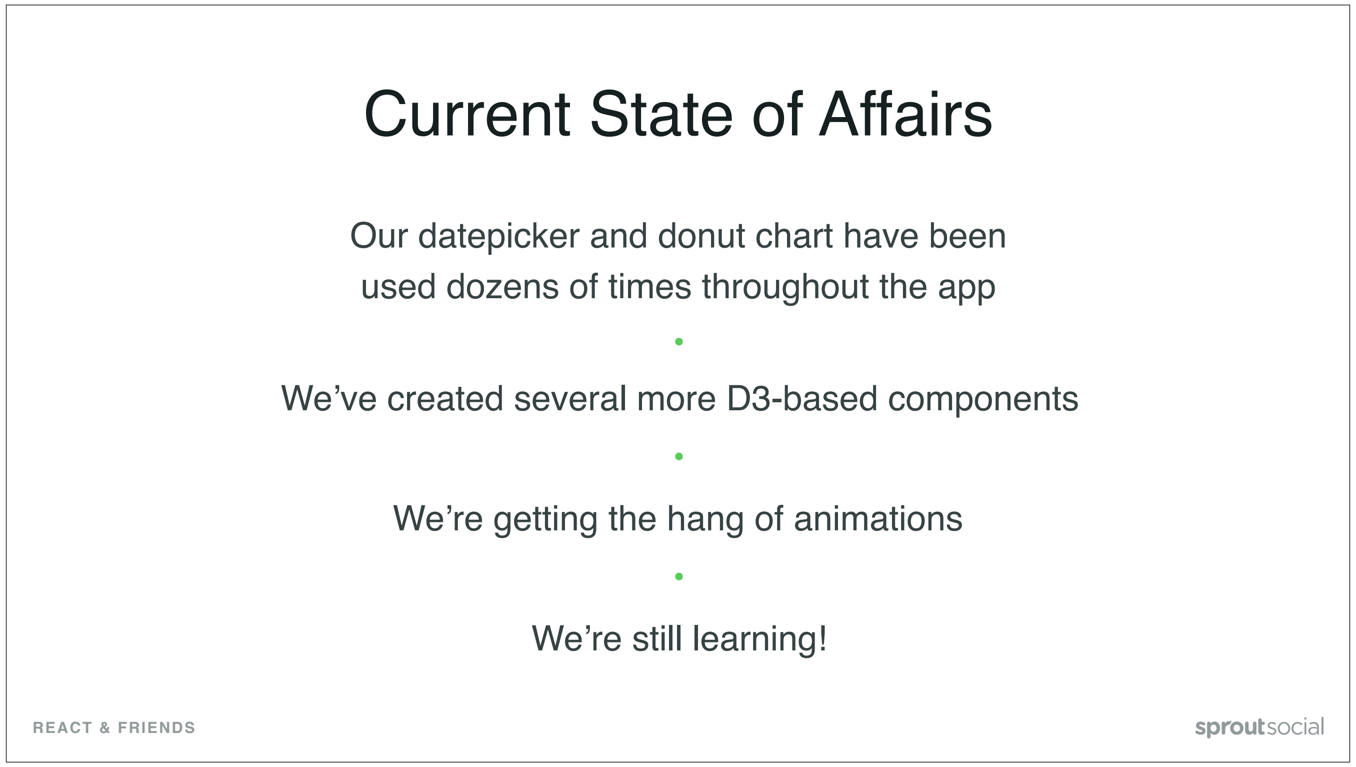
There are no one-size-fits-all solutions, what works for one problem might not work for another

•

There are no bad ideas in programming, just ideas that have specific use case

•

Sometimes the obvious solutions are the best



Current State of Affairs

Our datepicker and donut chart have been
used dozens of times throughout the app

•

We've created several more D3-based components

•

We're getting the hang of animations

•

We're still learning!

The best advice I can give —>

Takeaways

Make your React APIs the best they can be,
even if the internal implementation is ugly.

•

Build only what you need, then iterate. You'll
discover new and better solutions along the way.

•

There are no silver bullets, so don't look for one.

•

Don't go it alone, take your team with!

Thank You!

Gordon Kristan

gordon@sproutsocial.com

@gordonkristan