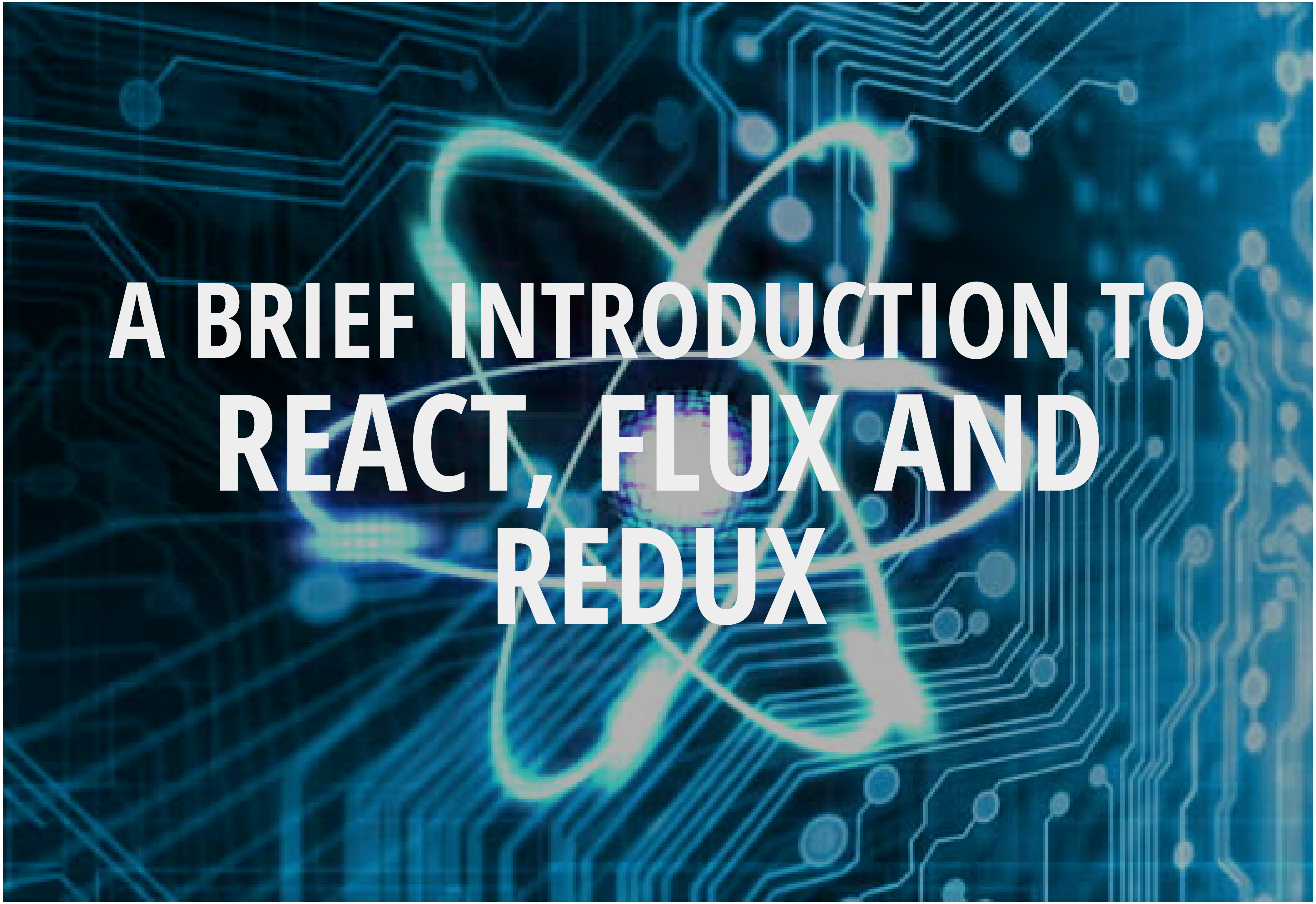


A BRIEF INTRODUCTION TO REACT, FLUX AND REDUX

React

WHAT IS IT?

Library, not framework

Built to solve one problem: building large applications with data that changes over time.

"Simply express how your app should look at any given point in time, and React will automatically manage all UI updates when your underlying data changes."

- Why React? (React documentation)

CHARACTERISTICS

- Component based
- Declarative
- *Learn once, write everywhere*

CHARACTERISTICS (II)

- JSX (e.g. `<Lol>` could render)
- CSS in JS (a bit controversial)
- Virtual DOM

STRUCTURE OF A COMPONENT

- Render
- Props
- State
- Lifecycle hooks
  - componentWillMount
  - componentDidMount
  - componentWillUnmount
  - shouldComponentUpdate

EXAMPLE - SIMPLE COMPONENT

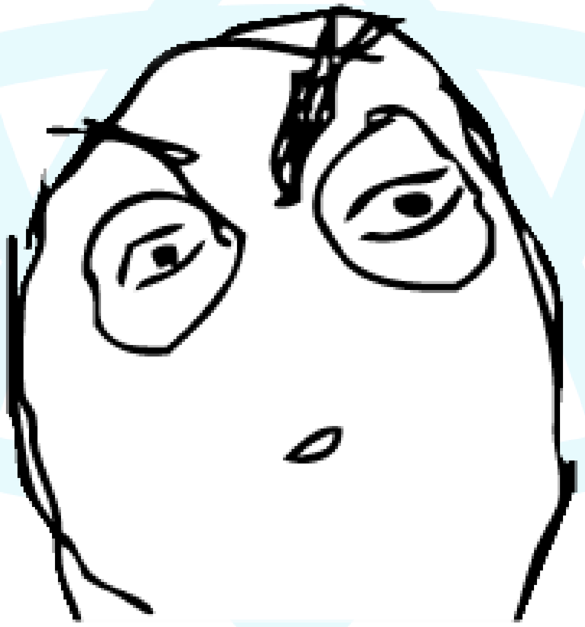
```
var HelloMessage = React.createClass({
  render: function() {
    return <div>Hello {this.props.name}</div>;
  }
});

ReactDOM.render(<HelloMessage name='John' />, mountNode);
```

EXAMPLE 2 - STATEFUL COMPONENT

```
var Timer = React.createClass({
  getInitialState: function() {
    return {secondsElapsed: 0};
  },
  tick: function() {
    this.setState({secondsElapsed: this.state.secondsElapsed + 1});
  },
  componentDidMount: function() {
    this.interval = setInterval(this.tick, 1000);
  },
  componentWillUnmount: function() {
    clearInterval(this.interval);
  },
  render: function() {
    return (
      <div>Seconds Elapsed: {this.state.secondsElapsed}</div>
    );
  }
});
```

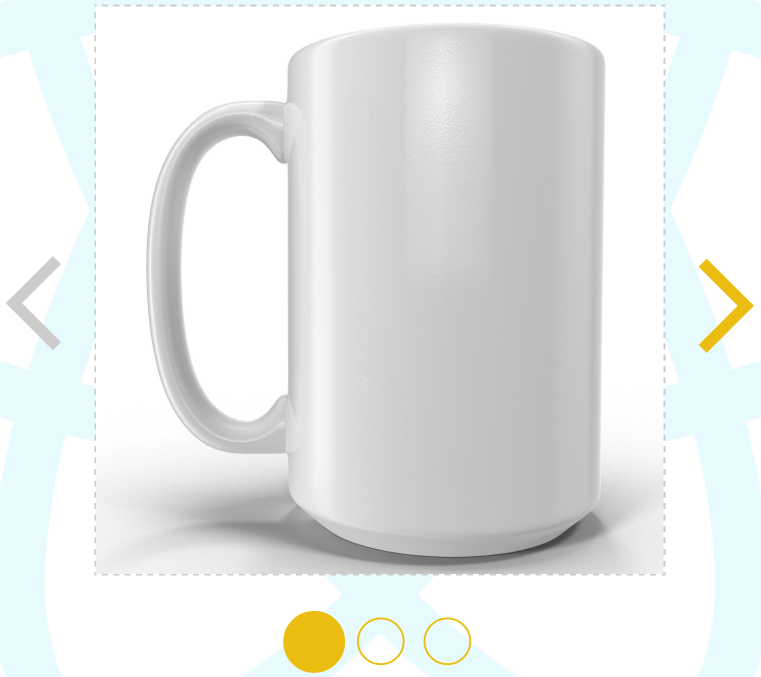
Yeah, really nice, but... What can all this be REALLY used for?



EXAMPLE 3 - ?

```
<div className={'preview'}>
    <Chevron direction='left' disabled={!this.previousExists()} onClick={() => this.previous()} />
    <img src={displayedImage} className={'preview-image'} alt={displayedImage} />
    <Chevron direction='right' disabled={!this.nextExists()} onClick={() => this.next()} />
    <NavigationDots numberOfDots={this.props.images.length} selectedDot={this.state.activeIndex} onClick={index => thi
</div>
```

EXAMPLE 3 - PRODUCT PREVIEW



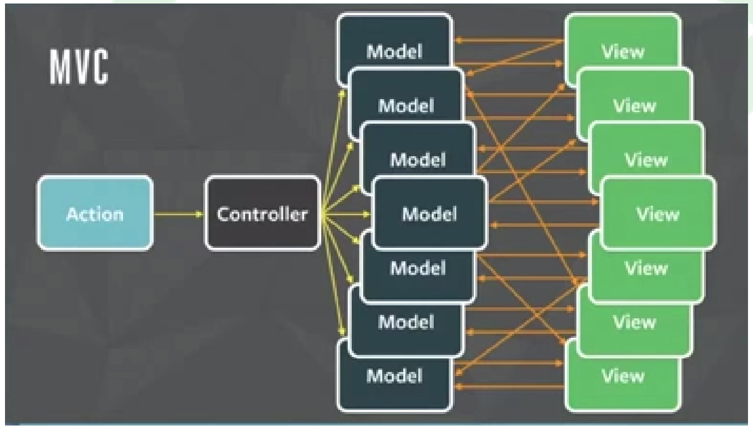
* This was intended to be an example of a possible usage of React in the company

Now go check the code. I will wait here.

Flux

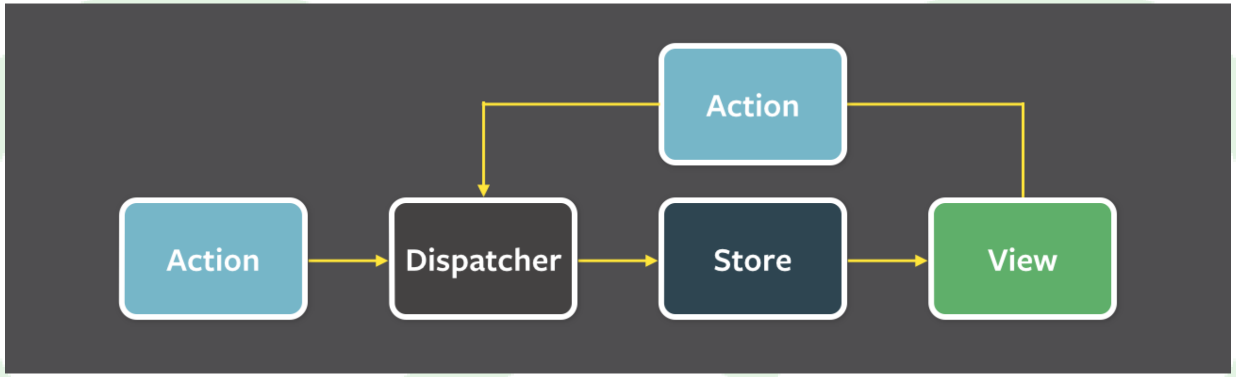
MOTIVATION

MVC not scaling well



Annoying bug

FLUX DATA FLOW



Store - manages the application state for a particular domain
View - typically a React component
Action - any user interaction with UI, change in the server, etc.
Dispatcher - notifies stores about actions triggered

CHARACTERISTICS

- Explicit data, instead of derived data
  - Client has more control and ability to stay consistent
- Wants to separate data from view state
  - Not a surprise, just like MVC
- Avoids cascading effects by preventing nested updates
  - Single direction of data flow
  - Finish processing before triggering new actions

BENEFITS

- *"If you understand where your action starts, and what are the changes inside the data layer, you know all the downstream effects"*- Jing Chen (Facebook)
- Improves data consistency
- Makes easier to pinpoint root of a bug
- Meaningful UT: *state A + input* *state B*

Time to check some code again.

Redux

MOTIVATION: FRUSTRATION

(BECAUSE OF THE WORKFLOW)

WHY "REDUX"?

Reducers + Flux

REDUCERS?

```
(previousState, action) => newState
```

This signature is the same as *Array.reduce:*

```
(accumulator, value) => accumulator
```

THE THREE PRINCIPLES OF REDUX

- Single source of truth
- State is read-only
- Reducers must be pure functions

DIFFERENCES WITH FLUX

- Only one store - state tree
- There is no Dispatcher
- Redux, unlike Flux, assumes data is not mutated - otherwise the cool stuff does not work

BENEFITS (AKA *THE COOL STUFF*)

- Hot reloading
- Time travel
- It makes easy to implement:
    - A logger
    - The undo functionality
    - Etc.

Code time

Oh, and do not forget the awesome devTools

react-redux

- Used to abstract the Redux logic in container components
- `<Provider>` provides (   ) the store to its child components
- Container components created via `connect`
  - `mapStateToProps`
  - `mapDispatchToProps`

We're almost finished. Go check the last piece of code.
It's not much, I promise.

THERE'S MUCH MORE COOL STUFF OUT THERE

- Immutable
- GraphQL + Relay
- React Native
- Isomorphism
- Flow

THAT'S ALL FOLKS

ANY QUESTIONS?

OK, it wasn't so brief