THINKING IN REACT

EVERYTHING IS A (PURE) FUNCTION

```
const Namebox = (props) => {
    return (
      <Label fontWeight={props.fontWeight}>
        ${props.labelContent}
      </Label>
    );
}

function NameBox(name) {
  return { fontWeight: 'bold', labelContent: name };
}
```

- Think of a UI as a simple function that projects data to a different form of data
- Same inputs give the same output => Pure Function

BUILDING VIEWS BY COMPOSITION

```
class UserBox implements React.Component {
  render() {
    return (
      <FancyBox label={this.props.labelContent}>
        <NameBox firstName={this.props.firstname}
                 surname={this.props.surname} />
      </FancyBox>
    );
  }
}
```

- Abstract components in reusable pieces
- Compose complex components from reusable pieces

STATE TRAVELS TOP DOWN

```
class App implements React.Component {
  updateUserBox() {
    this.setState({
      user: {firstname: 'John', surname: 'doe'}
    })
  }
  render() {
    const {firstname, surname} = this.state.user;

    return (<App>
      <UserBox firstname={firstname} surname={surname} />
    </App>);
  }
}
```

- Identify the common parent where your state lives
- Identify state to trickle down the component tree

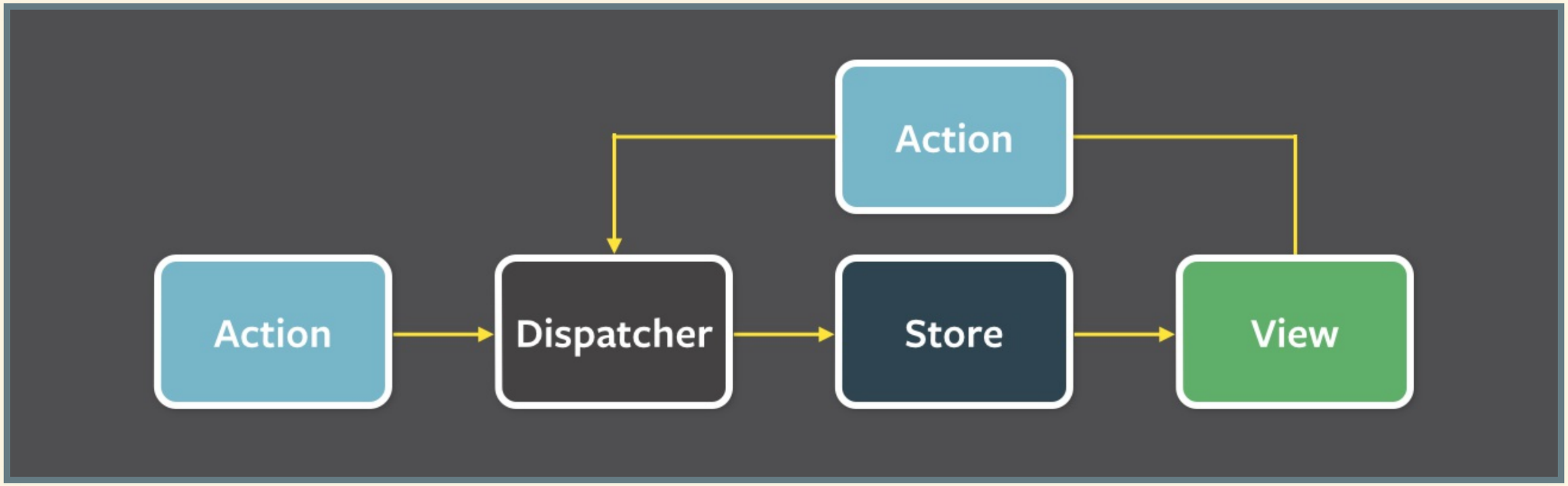
STATE EVENTS TRAVEL BOTTOM UP

```
class UserBox implements React.Component {
  handleInputChange(e) {
    this.props.onInputChange(e.target.value);
  }

  render() {
    return (<FancyBox name={this.props.name}>
      <NameBox firstName={this.props.firstName}
               surname={this.props.surname} />
    </FancyBox>);
  }
}
```

- Identify parent component state handlers
- Push state handlers down the tree through props
- Be aware of callback binding issues

THINK DATA FLOW AS FLUX



Source: Facebook - Flux

- Write sane, testable apps with an one-way data flow
- Use one of the common flux approaches e.g. Redux, Mobx

THINK OF SITE EFFECTS AS A MIDDLEWARE NOT AS FLUX

- Ajax-Calls, (Service) Workers, Local Storage, etc.
- Evaluate approaches properly for best fit, e.g.:
  - redux-promise, redux-thunk, redux-saga, etc.
  - mobx-rest, etc.
  - ...

THINK ABOUT TEST APPROACHES

Test type	Approach
Testing structure	`TestUtils.renderIntoDocument()`
Shallow rendering	`TestUtils.createRenderer().rend`
Simulating events	`TestUtils.Simulate`

THINK ABOUT TEST APPROACHES

Test type	Approach
Simulating events	Directly `setState`
Testing behaviour	Calling properties and methods

TYPICAL REACT STARTER DECISIONS

Decision	Directions
Universal App?	Isomorphic render? ReactNative?
Static typing?	Flow, TypeScript
Styling?	CSS Modules? PostCSS? Preprocessor?

TYPICAL REACT STARTER DECISIONS

Decision	Directions
State handling?	setState? Redux? MobX?
Bundling?	Webpack? Bower?
Test strategy?	Structure? Behaviour? Shallow render?

SO LONG AND THANKS FOR THE FISH!

Periklis Tsirakidis

Github: github.com/Pericles

FURTHER READING

- React - Basic Theoretical Concepts
- Approaches to testing React Components
- The Hitchhiker's Guide to Modern ES Tooling
- Flux - In depth overview
- Redux - Core Concepts
- MobX - Concepts & Principles
- Book: SurviveJS - Become a React Master
- Book: SurviveJS - Become a Webpack Master
- Book: Fullstack React