levi nine
Technology Services

React

Introduction to React framework for SPA

Miljan Veljović
Miloš Cvetković

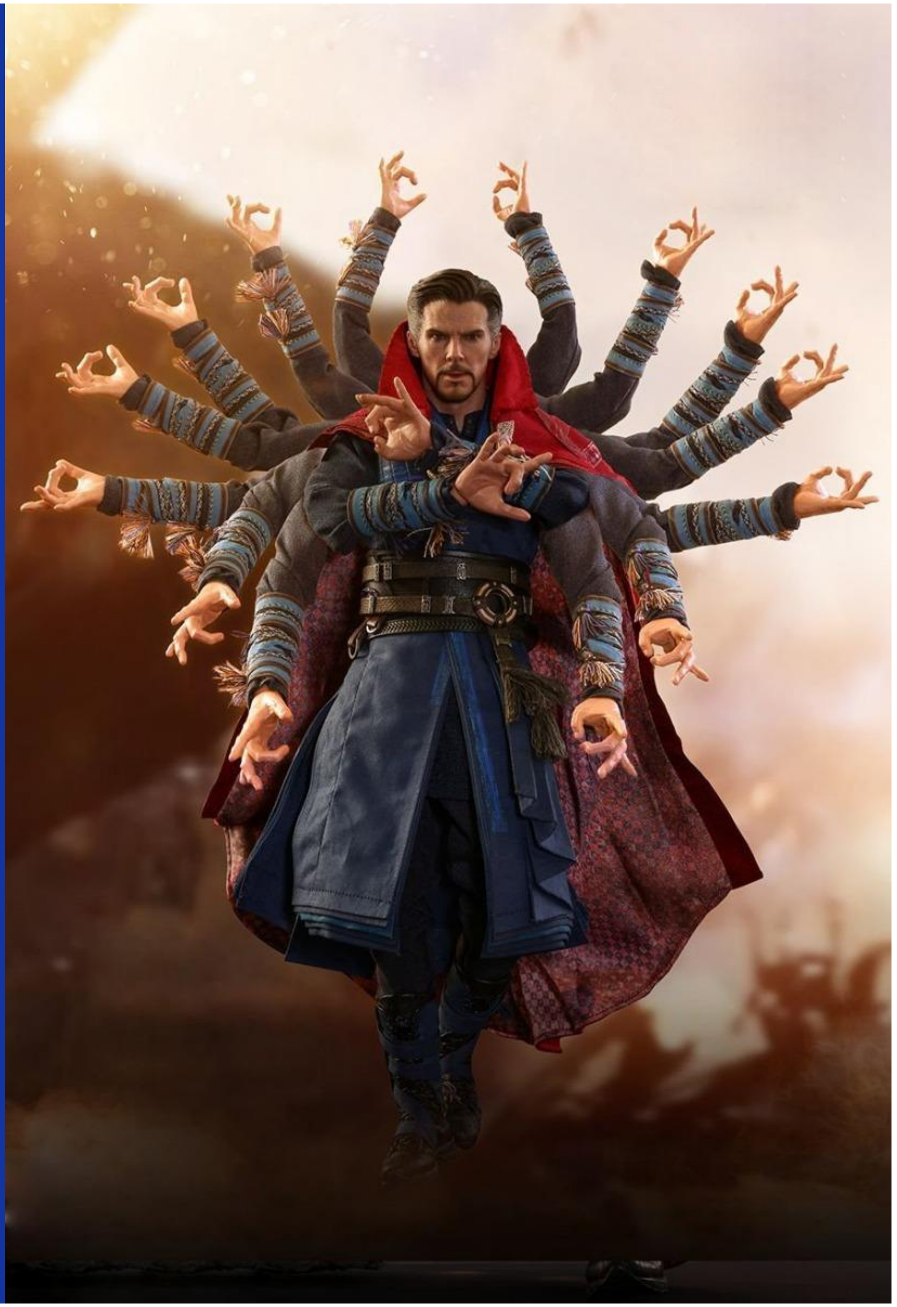
MATF, Beograd, 12.12.2019.

AGENDA SLIDE

01

WHAT IS REACT?

React in a nutshell.



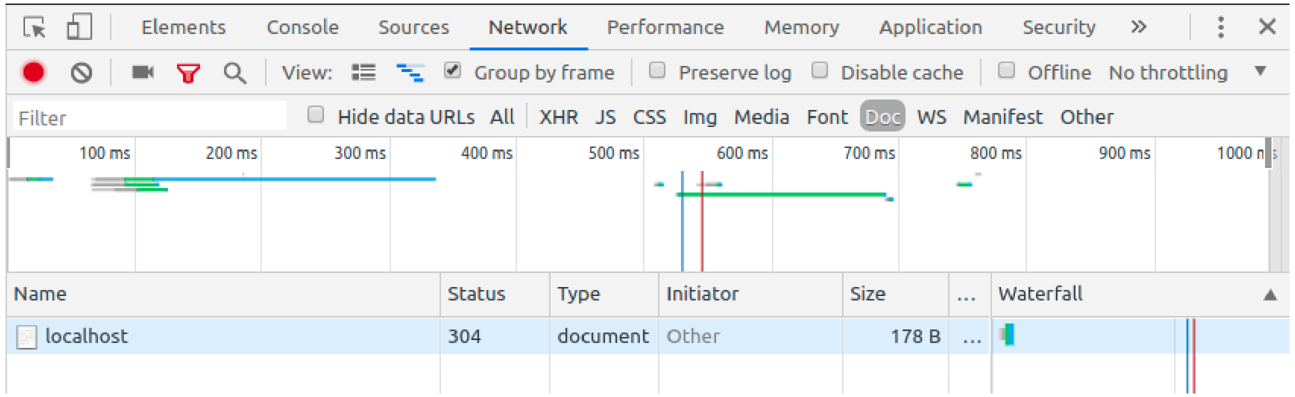
What is React?

React is JavaScript framework for building user interfaces as a SPA (Single Page Application).

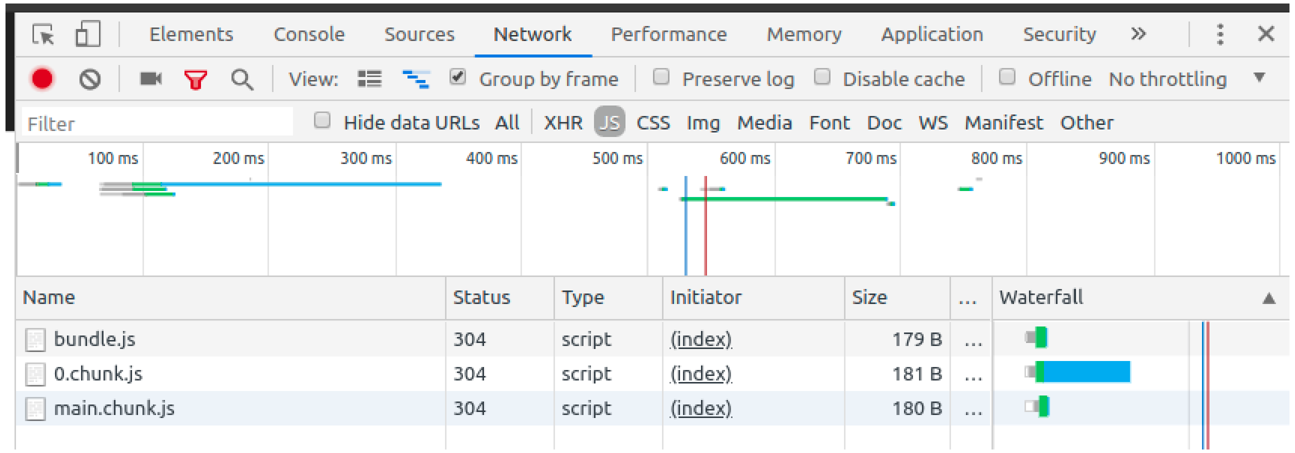
Static sites vs. SPAs?

- Traditional static sites have one HTML file per page and navigation on the site triggers HTTP request to fetch the new HTML page
- SPAs have **one single** HTML page for the whole application, and JavaScript file associated to that HTML which controls the execution of application and its state (including navigation and DOM manipulation).

Bundle files



Name	Status	Type	Initiator	Size	...	Waterfall
localhost	304	document	Other	178 B	...	

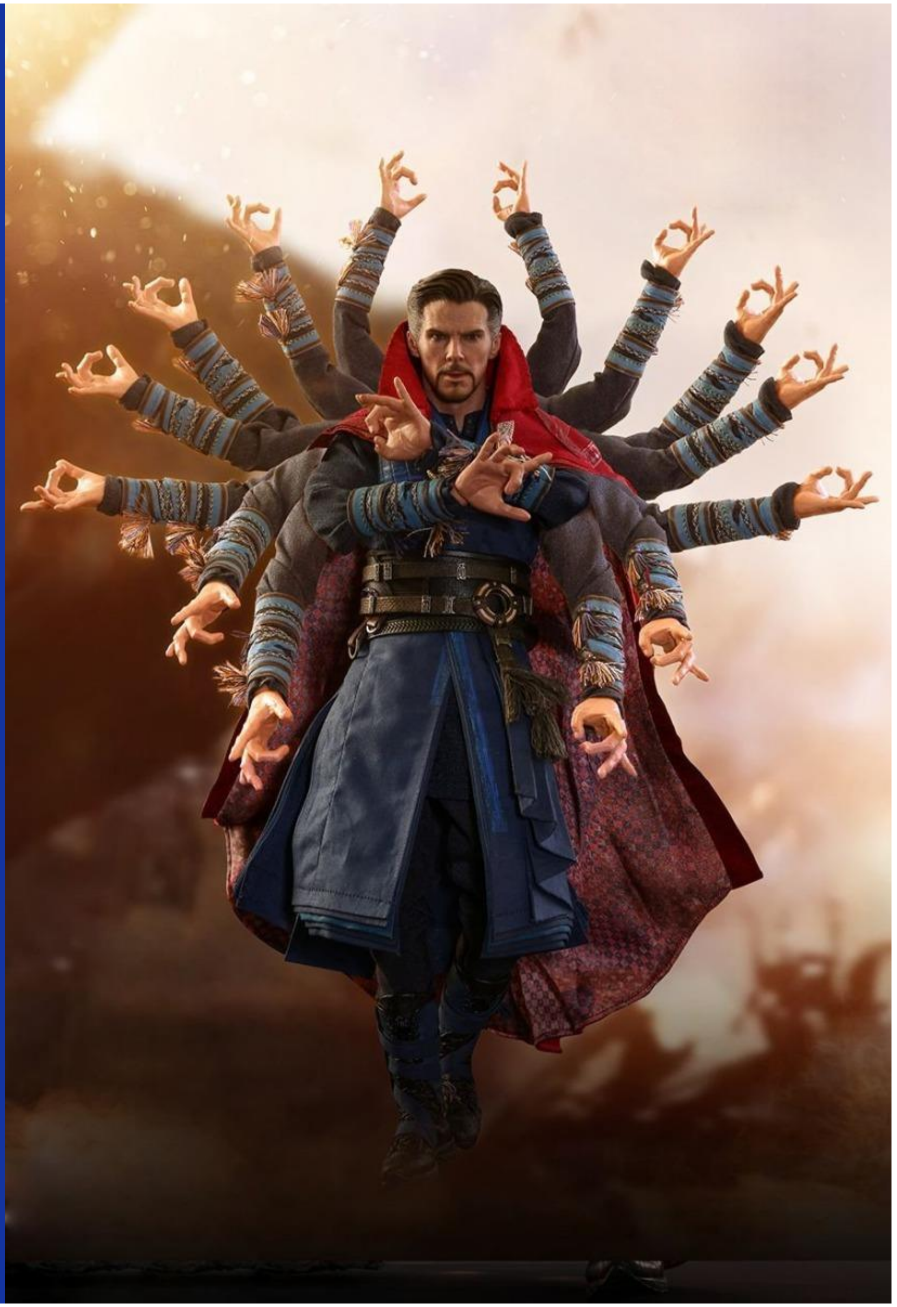


Name	Status	Type	Initiator	Size	...	Waterfall
bundle.js	304	script	(index)	179 B	...	
0.chunk.js	304	script	(index)	181 B	...	
main.chunk.js	304	script	(index)	180 B	...	

02

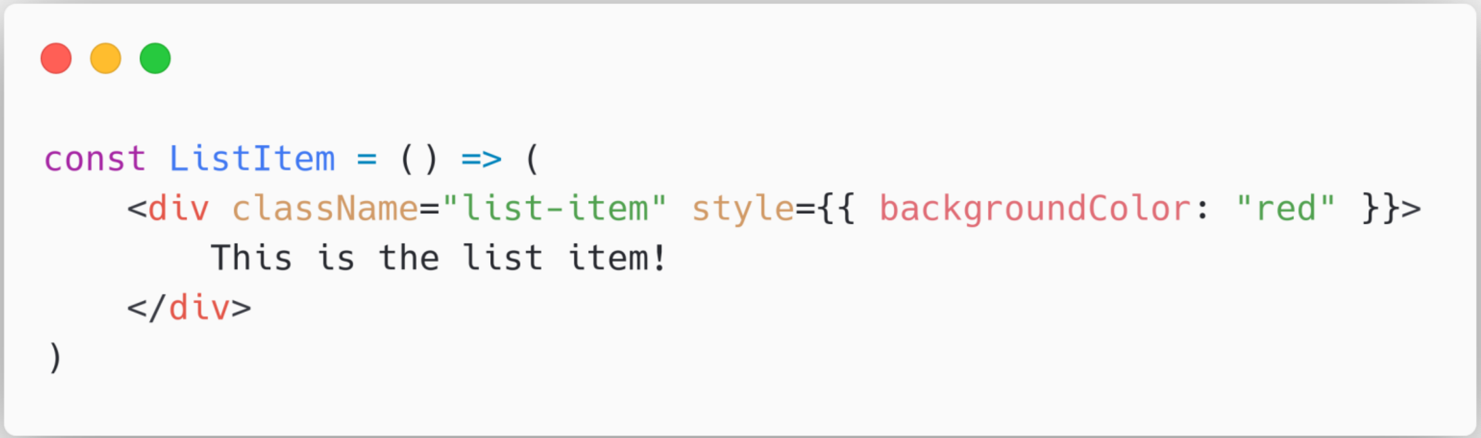
JSX

It's not HTML.



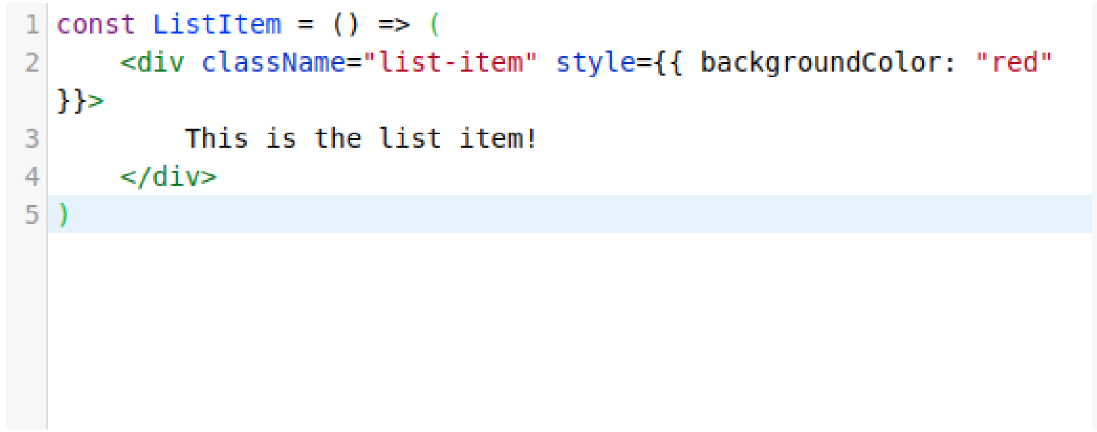
What is JSX?

- It's extension to JavaScript, **not** HTML, and it's compiled into **pure** JavaScript.

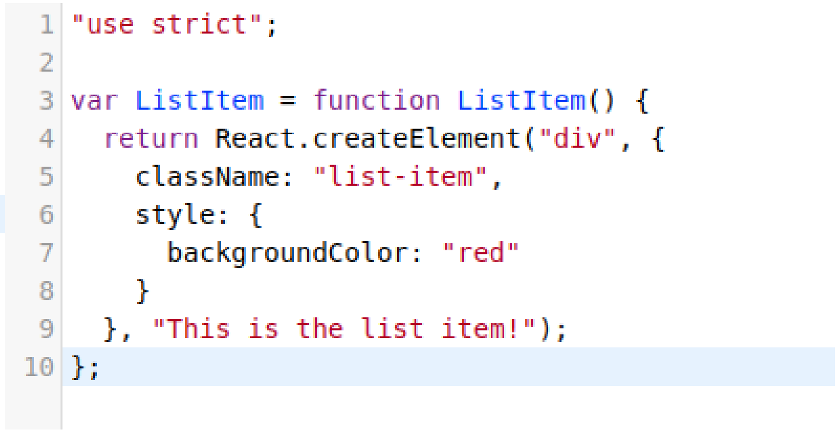


```
const ListItem = () => (
    <div className="list-item" style={{ backgroundColor: "red" }}>
        This is the list item!
    </div>
)
```

Babel



```
const ListItem = () => (
    <div className="list-item" style={{ backgroundColor: "red" }}>
        This is the list item!
    </div>
)
```

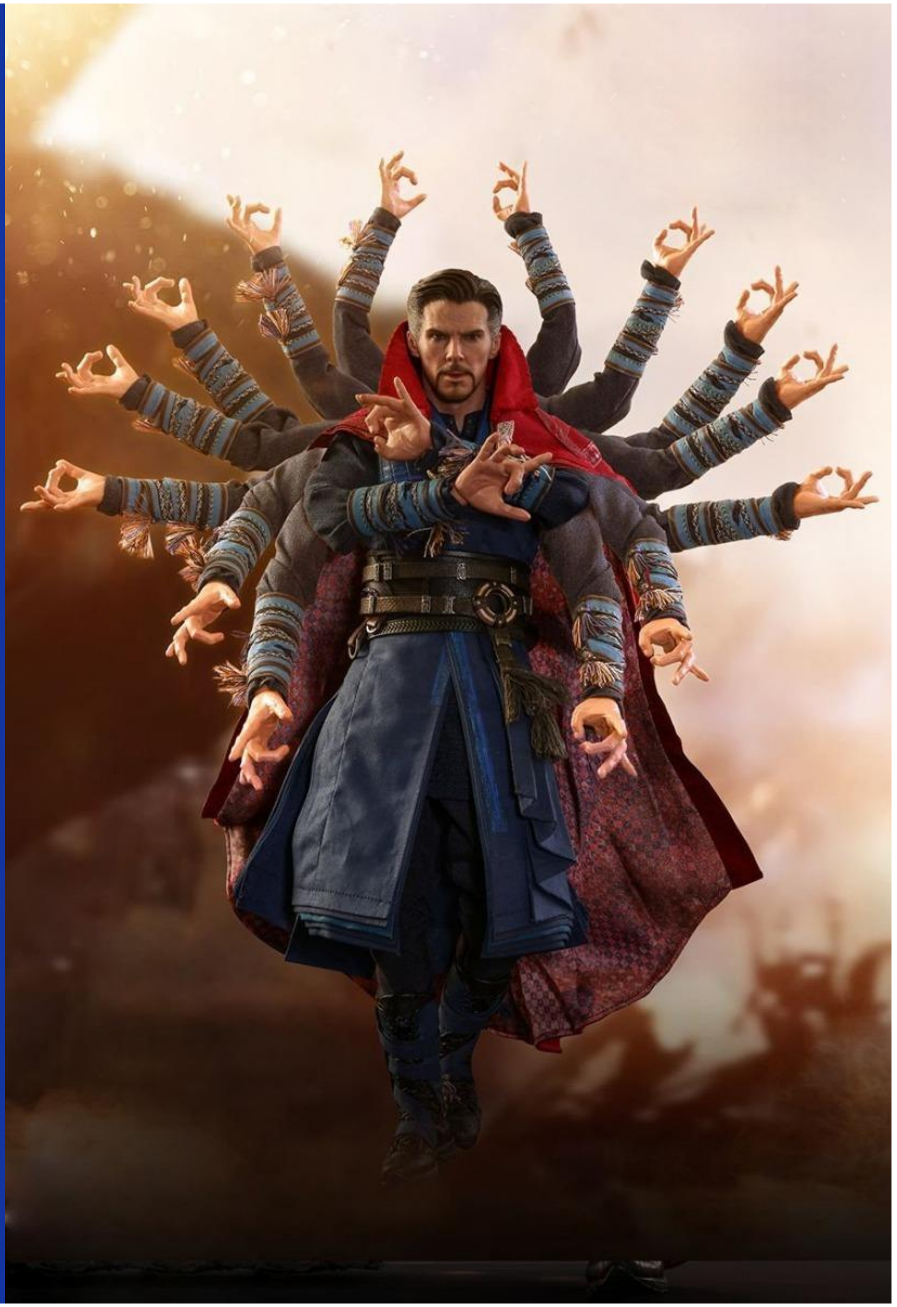


```
"use strict";

var ListItem = function ListItem() {
  return React.createElement("div", {
    className: "list-item",
    style: {
      backgroundColor: "red"
    }
  }, "This is the list item!");
};
```

04

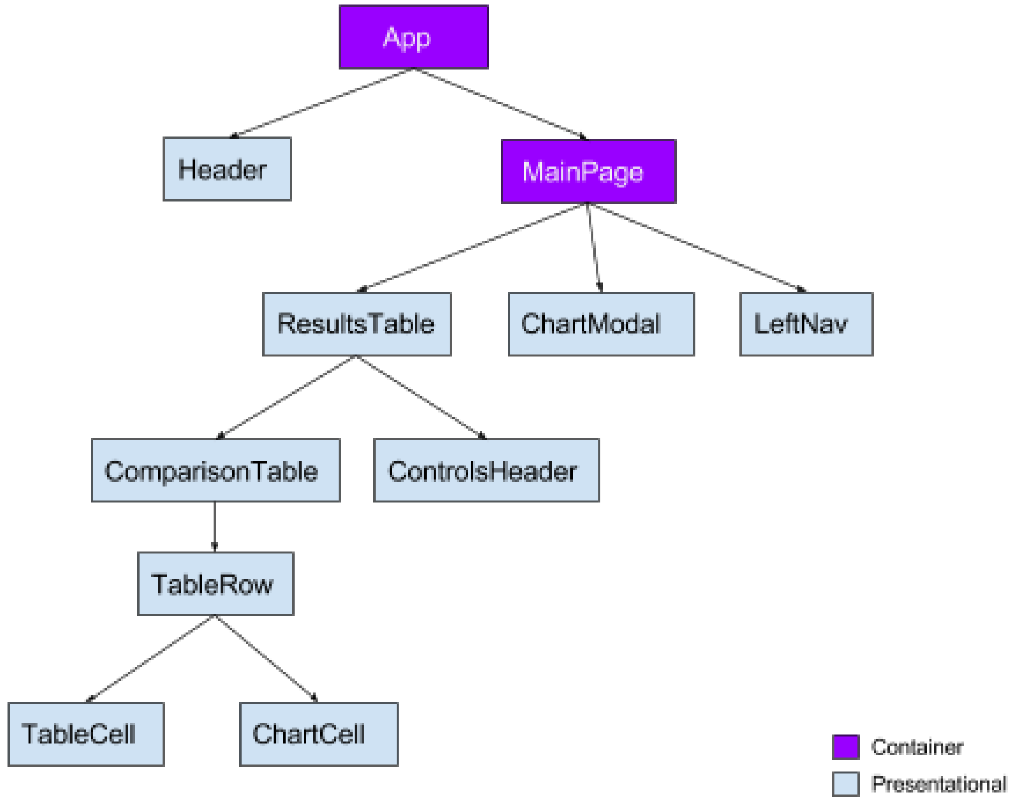
COMPONENTS

The building blocks of React applications.



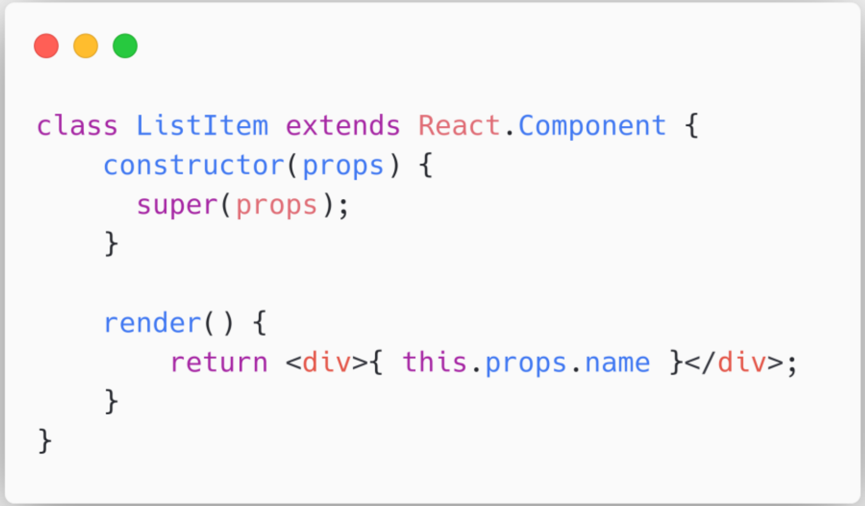
React components overview

- Basic building blocks in React
- Class-based components **must have render method**, and optionally constructor and other lifecycle methods, and **must extends React.Component.**
- Function-based components must return valid JSX.
- The goal: components as simple as possible.
- Guide: simple, presentational components should be function-based, and more complex components should be class-based

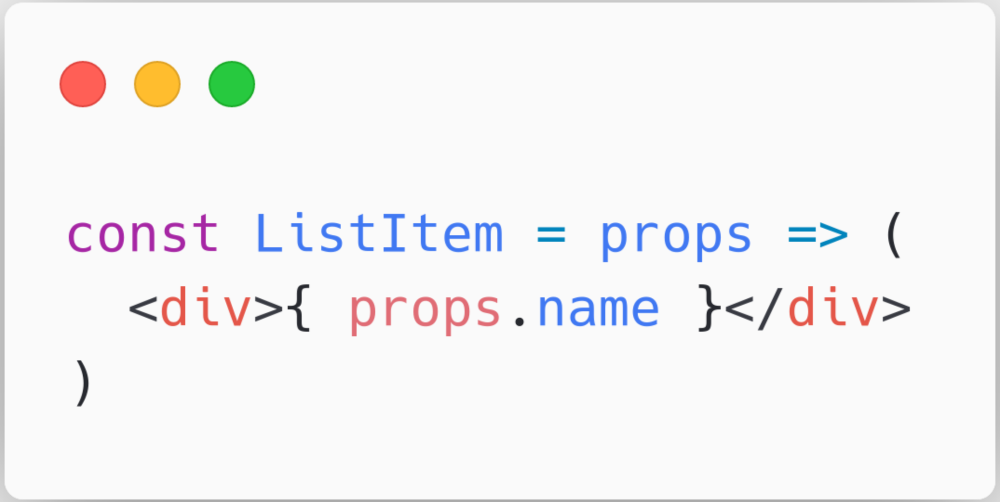


Props

- **Props** (properties) are component's inputs that are passed from parent component to child component.
- It's the core of the component composition idea that React is built on.



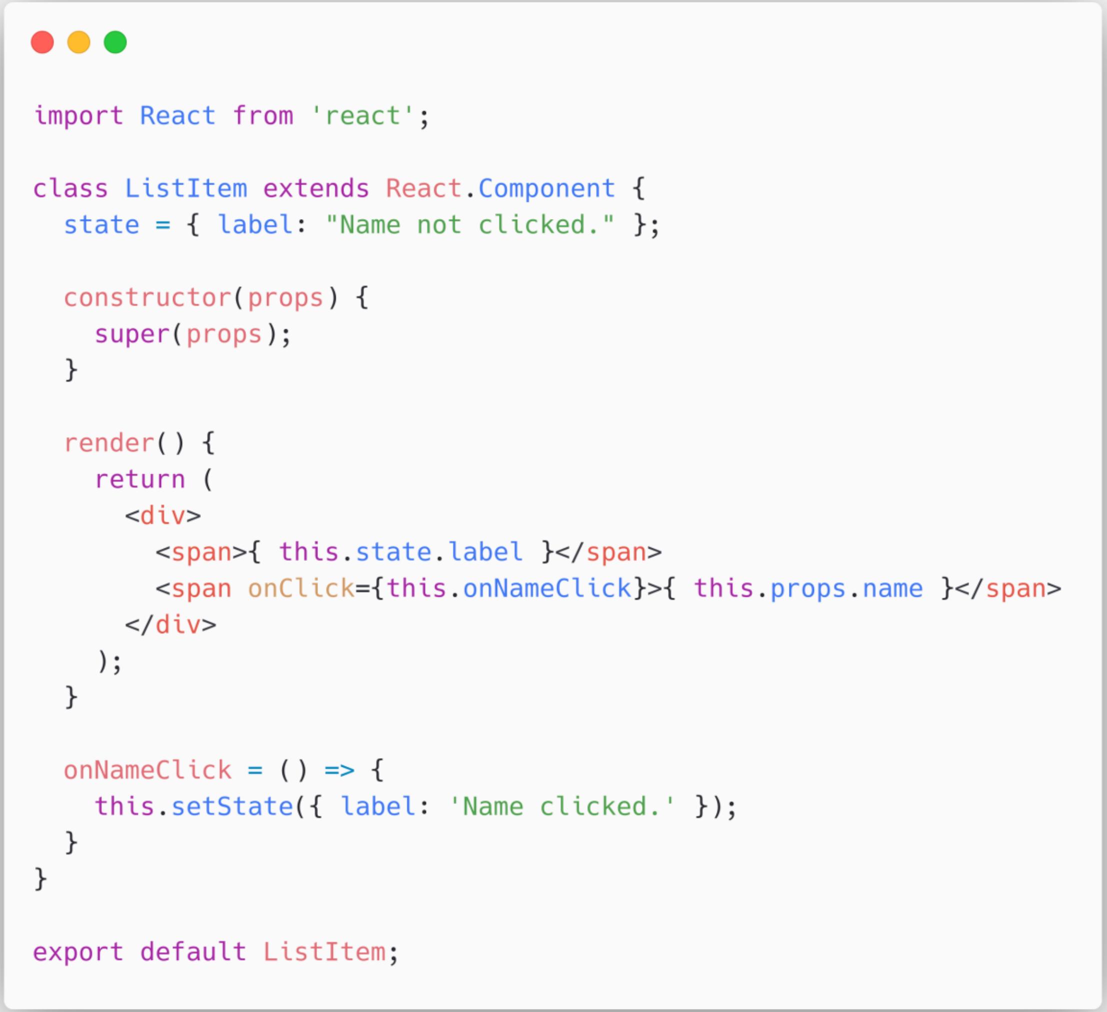
```
class ListItem extends React.Component {
    constructor(props) {
      super(props);
    }

    render() {
        return <div>{ this.props.name }</div>;
    }
}
```



```
const ListItem = props => (
  <div>{ props.name }</div>
)
```

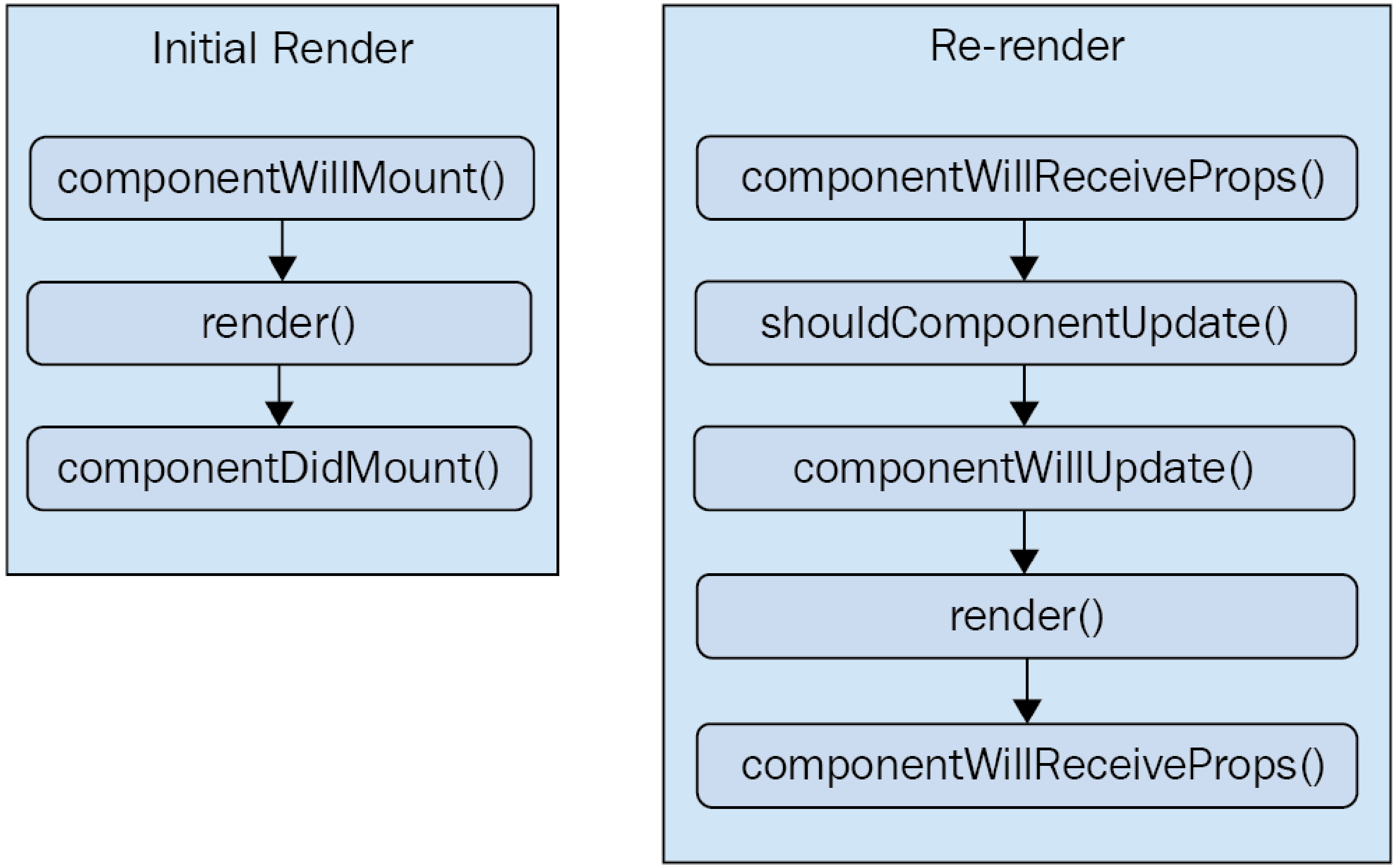
State

- State is object that represents the current state of the component, and whose changes triggers re-rendering of the component.
- State **must** be initialized and can be change **only** with **setState**.
- Function-based components does not have state.



```
import React from 'react';

class ListItem extends React.Component {
  state = { label: "Name not clicked." };

  constructor(props) {
    super(props);
  }

  render() {
    return (
      <div>
        <span>{ this.state.label }</span>
        <span onClick={this.onNameClick}>{ this.props.name }</span>
      </div>
    );
  }

  onNameClick = () => {
    this.setState({ label: 'Name clicked.' });
  }
}

export default ListItem;
```

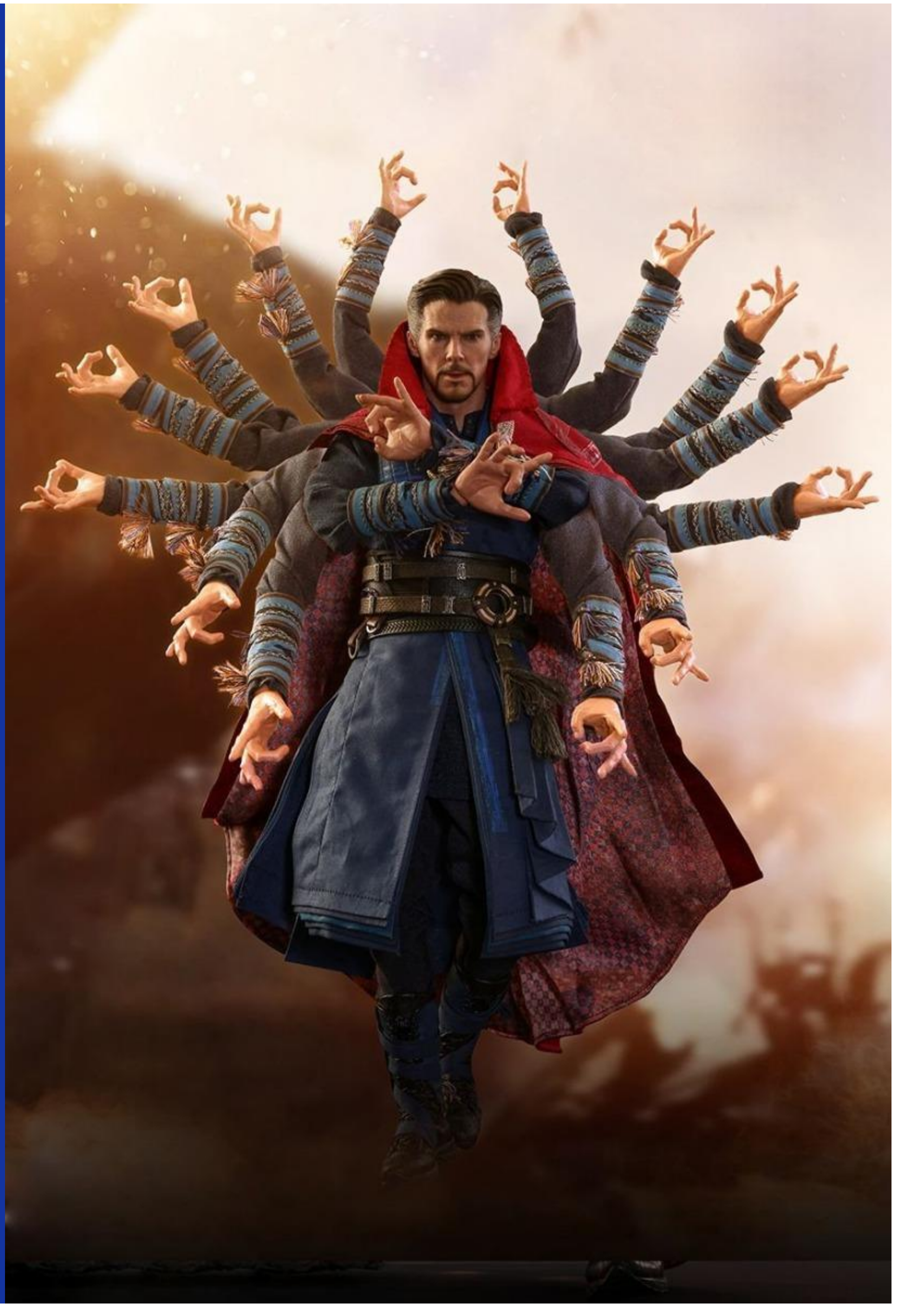
React component lifecycle methods



04

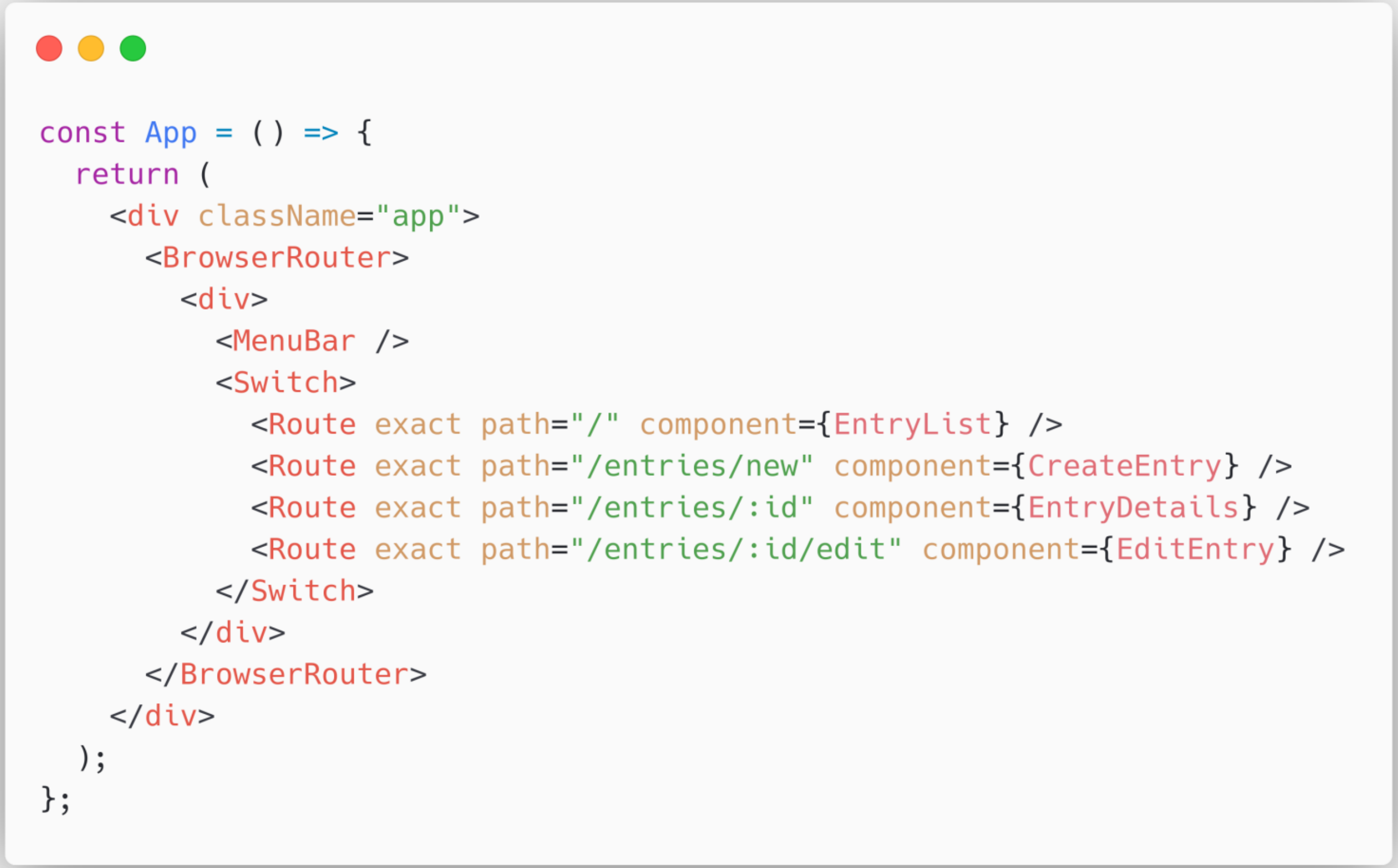
REACT ROUTER

SPA does not mean only one URL.



React Router

- Provides a way to conditionally render components depending on the current URL in the browser.
- It's not really navigation, it's just removing and appending components as described in the code.

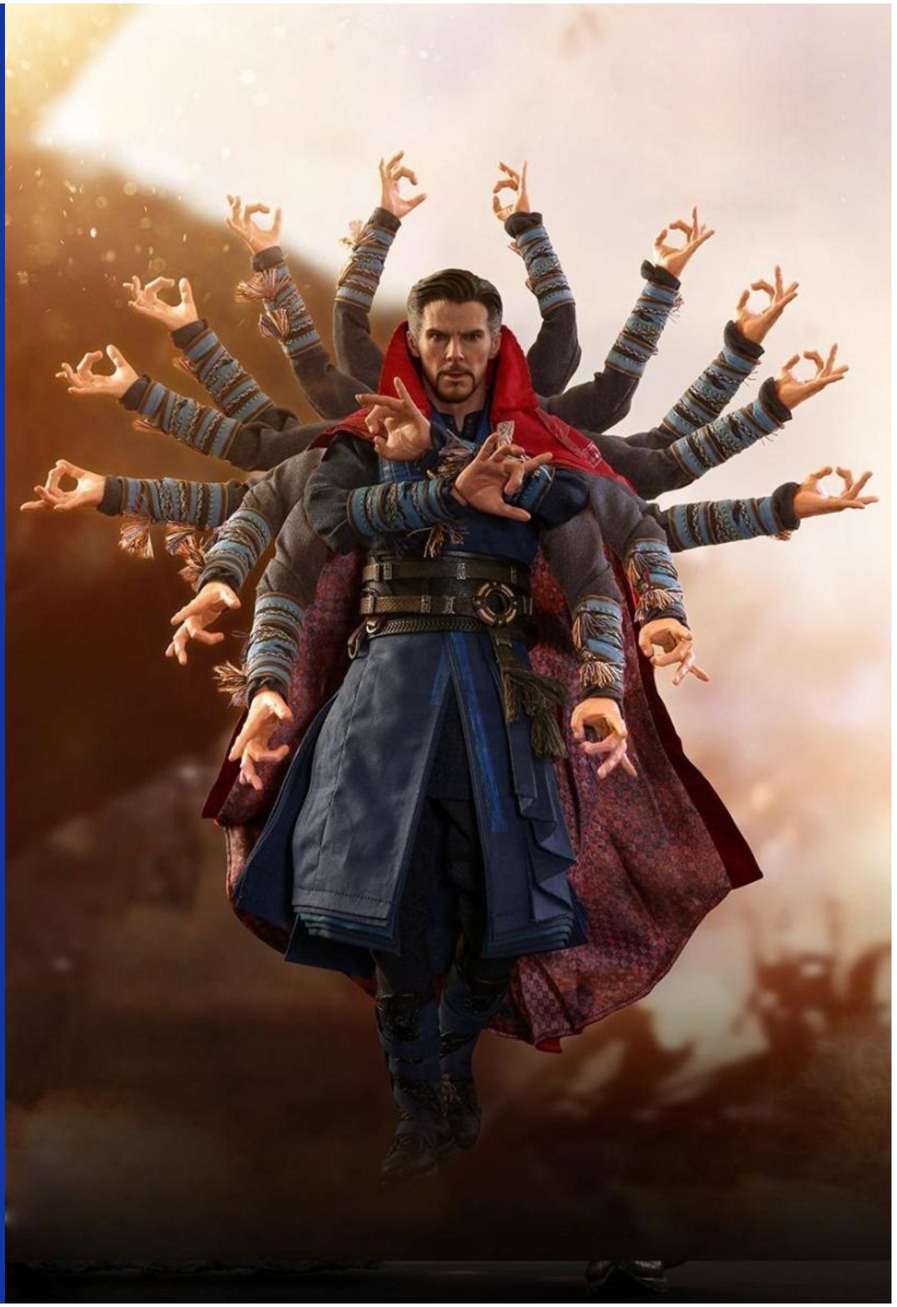


```
const App = () => {
  return (
    <div className="app">
      <BrowserRouter>
        <div>
          <MenuBar />
          <Switch>
            <Route exact path="/" component={EntryList} />
            <Route exact path="/entries/new" component={CreateEntry} />
            <Route exact path="/entries/:id" component={EntryDetails} />
            <Route exact path="/entries/:id/edit" component={EditEntry} />
          </Switch>
        </div>
      </BrowserRouter>
    </div>
  );
};
```

05

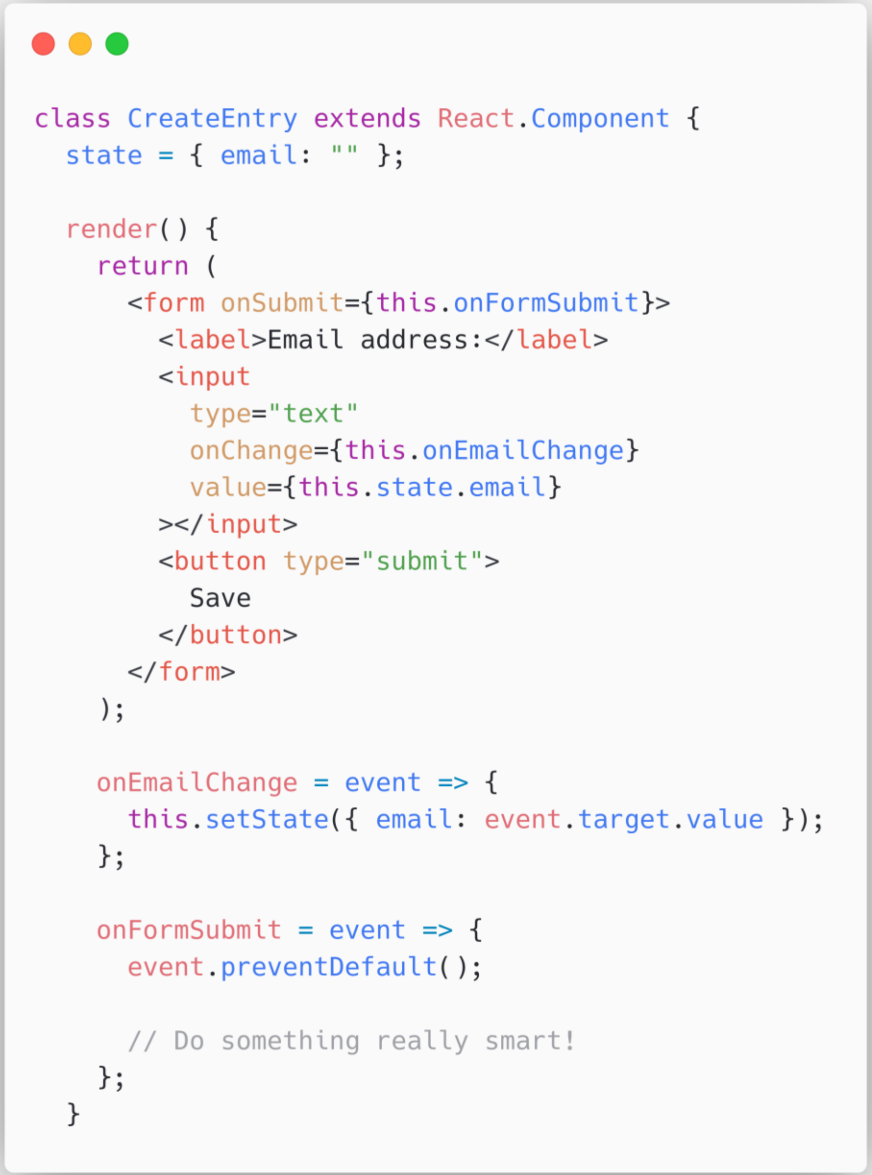
FORMS

If you'll ever need user interaction.



Handing user inputs

- Instead of traditional HTML forms, submit function is implemented separately.
- **Controlled** inputs are completelly controlled within JavaScript code.
- **Uncontrolled** inputs are not fully controlled by JavaScript code.
- Goal: all inputs should be **controlled** and single source of truth should be component's code.

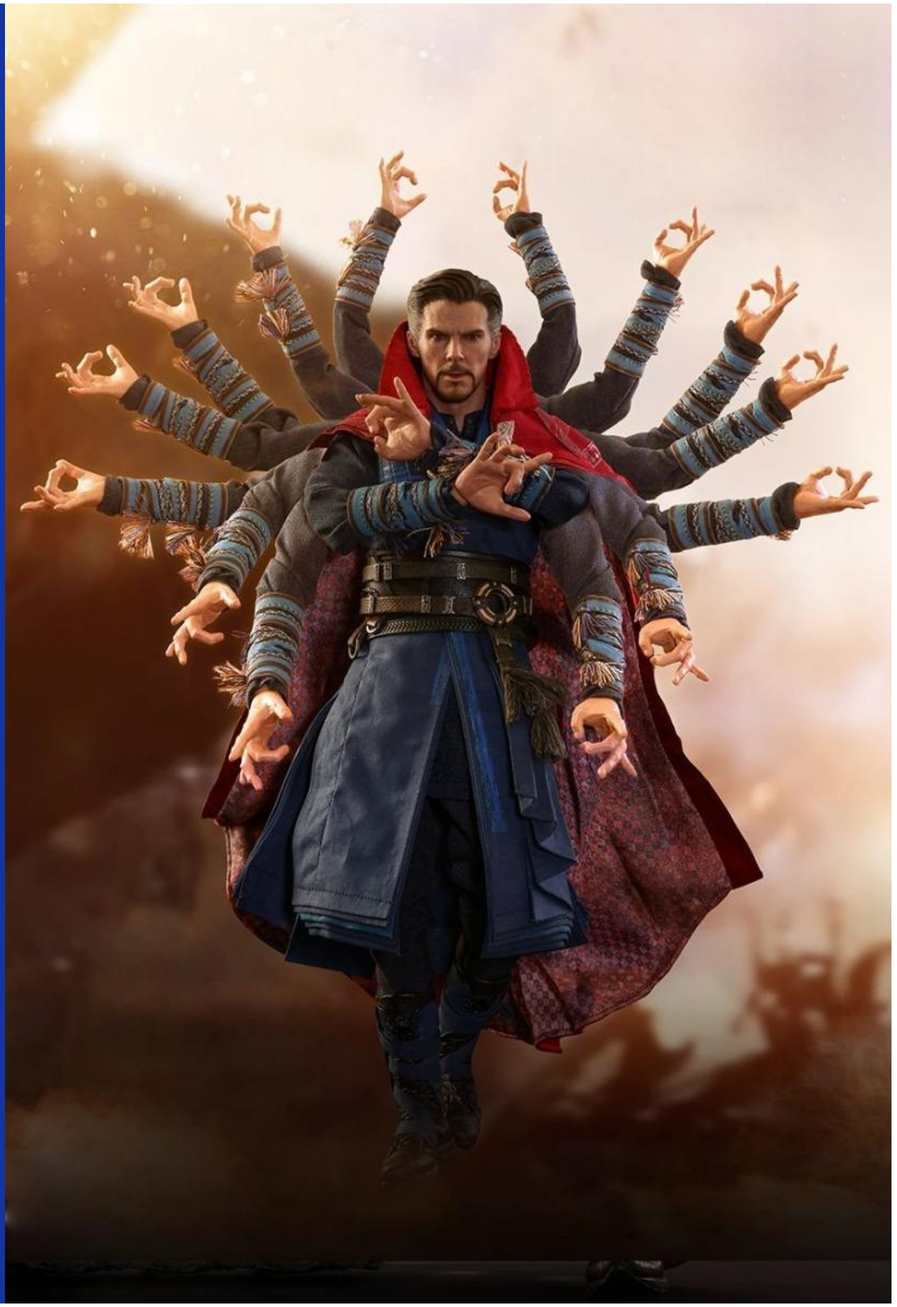


```
class CreateEntry extends React.Component {
  state = { email: "" };

  render() {
    return (
      <form onSubmit={this.onFormSubmit}>
        <label>Email address:</label>
        <input
          type="text"
          onChange={this.onEmailChange}
          value={this.state.email}
        ></input>
        <button type="submit">
          Save
        </button>
      </form>
    );

    onEmailChange = event => {
      this.setState({ email: event.target.value });
    };

    onFormSubmit = event => {
      event.preventDefault();

      // Do something really smart!
    };
  }
```

06

REDUX

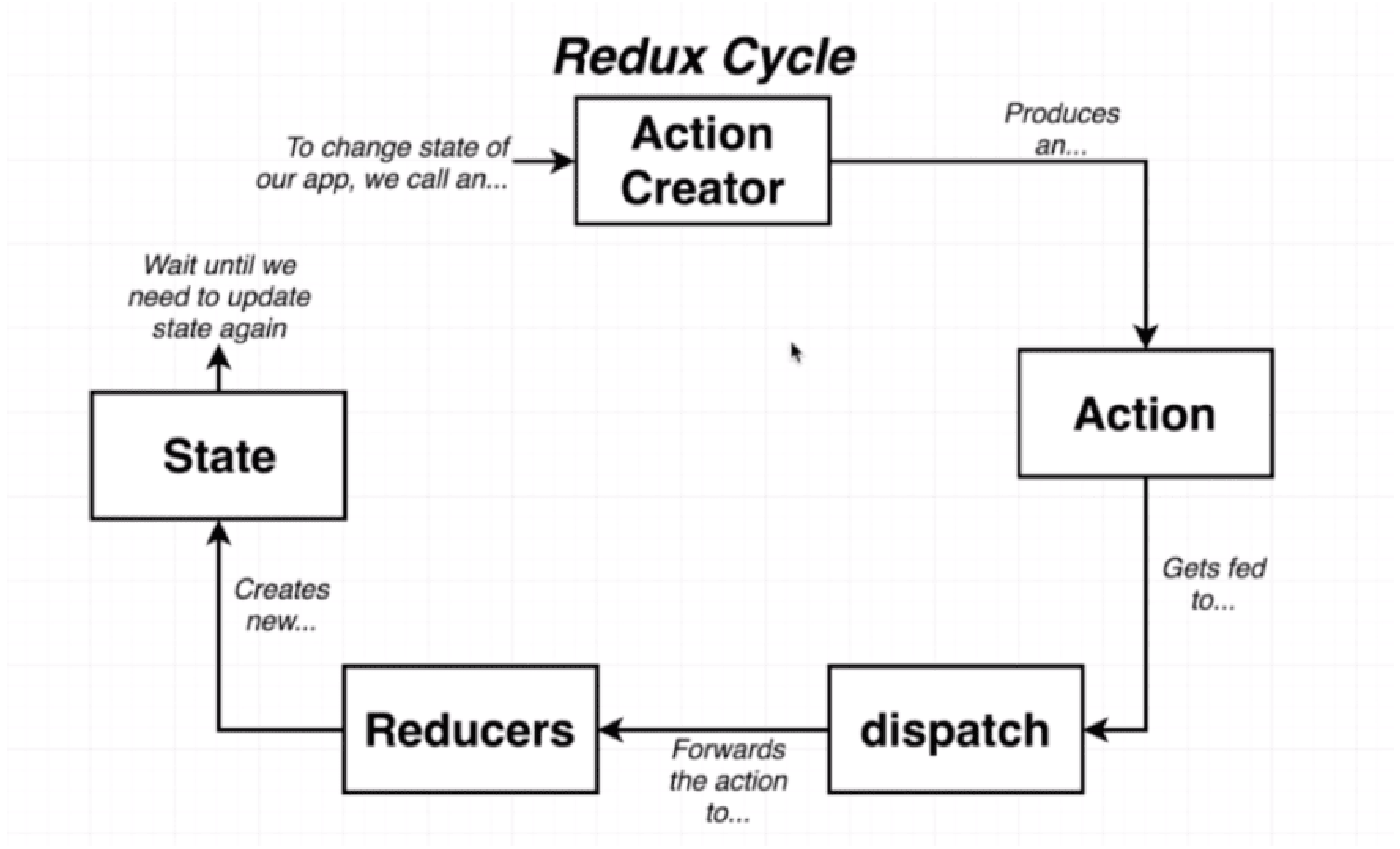
Manage state like a pro.



Redux in React

- Redux is state management library, independent from React, but it's a great addition to it.
- It provides **Store**, which is a set of **Action Creators**, **Actions** and **Reducers**.
- Root component does not have to pass props to leaf component through every component in between.
- Root (or any other) component dispatches an action, and leaf (or any other) component receives the new State from the Store.

Redux flow


Redux Cycle
To change state of our app, we call an...
Action Creator
Produces an...
Action
Gets fed to...
dispatch
Forwards the action to...
Reducers
Creates new...
State
Wait until we need to update state again

QUESTIONS ?

levi nine
Technology Services