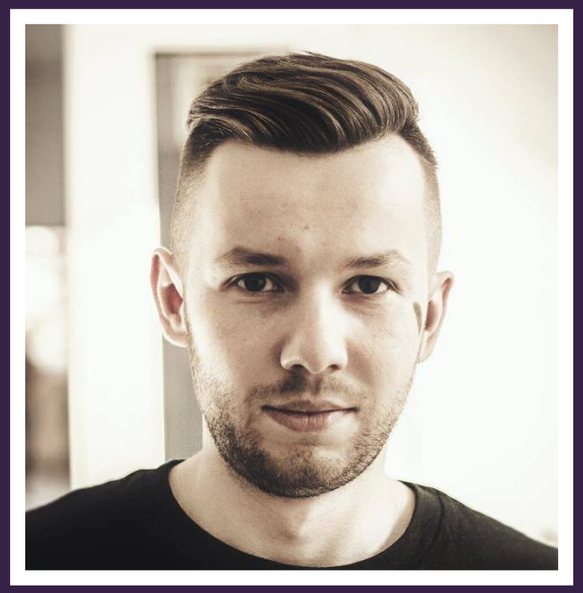
REACT JS



PRZEMYSŁAW WISZOWATY

HELLO!

SOFTware HUT

TENDERHUT GROUP

meet.js
białystok

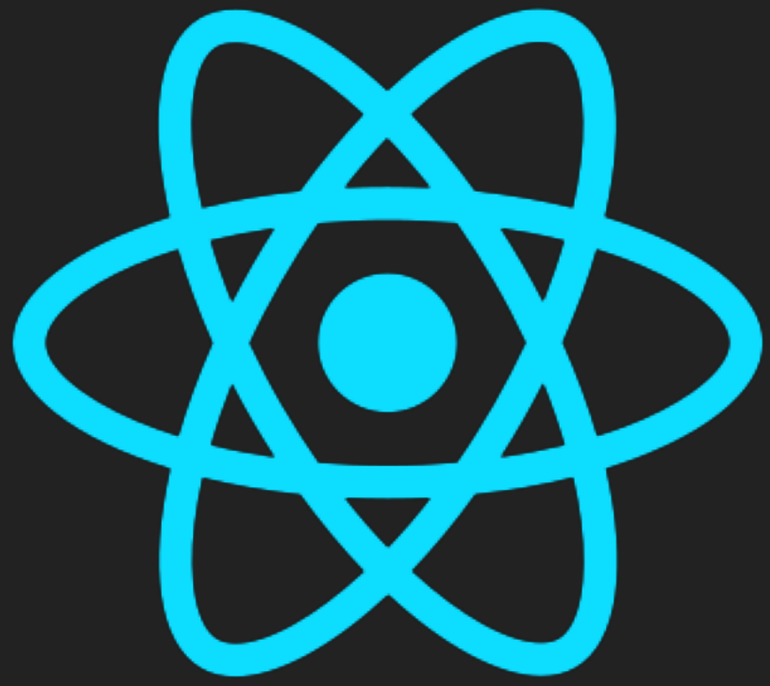
REACT IS EVERYWHERE

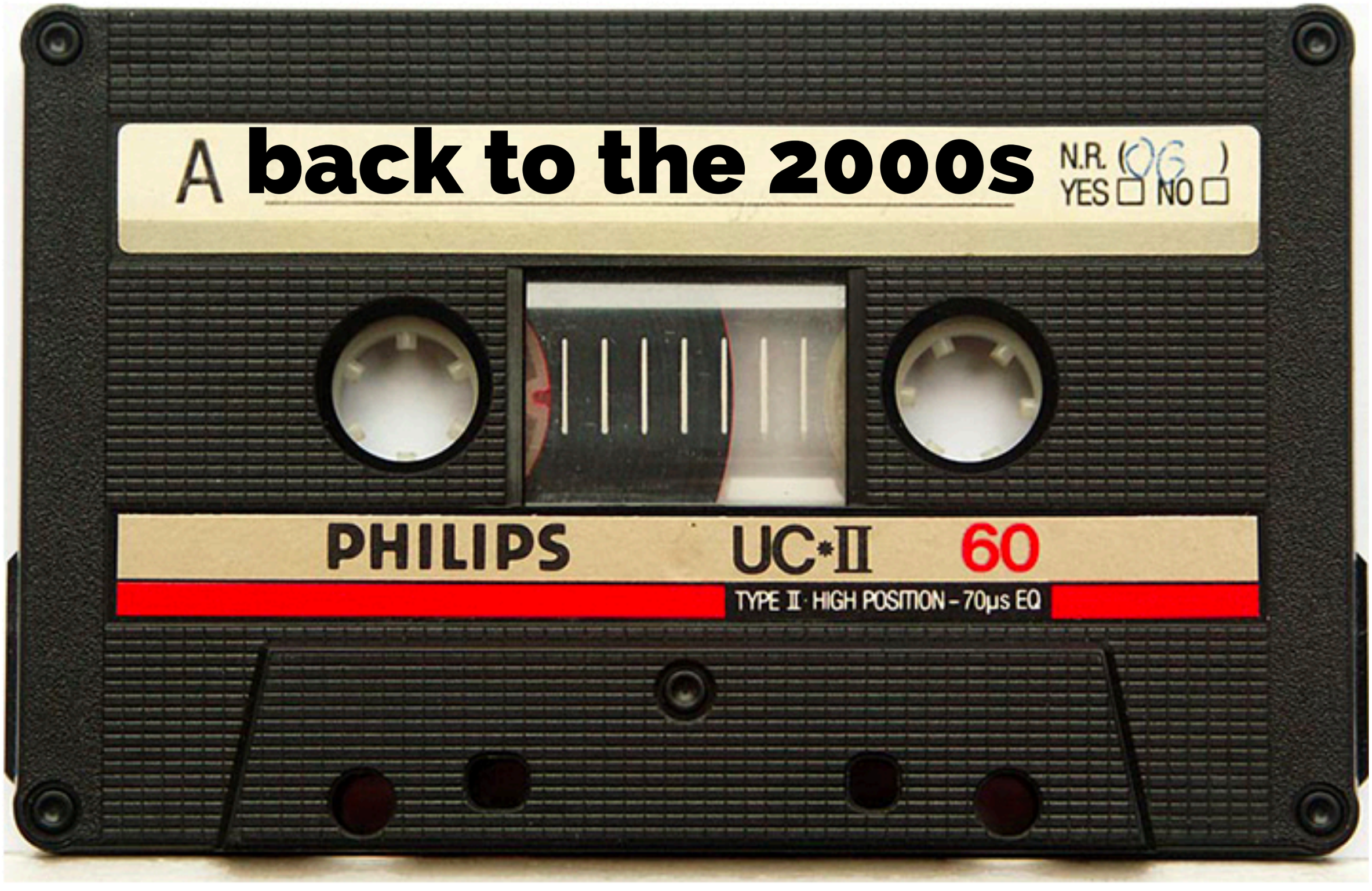
„Learn Once, Write Anywhere“

FRONTEND

BACKEND

MOBILE




A back to the 2000s
N.R. (06)
YES ☐ NO ☐
PHILIPS
UC*II
60
TYPE II · HIGH POSITION – 70µs EQ

The Geocities-izer

ge Look Like It Was Made By A 13 Year-Old In 1996

Type any URL in the box below and click Submit to see how it would look as a Geocities page.

Or Try one of these: The New York Times YouTube BoingBoing

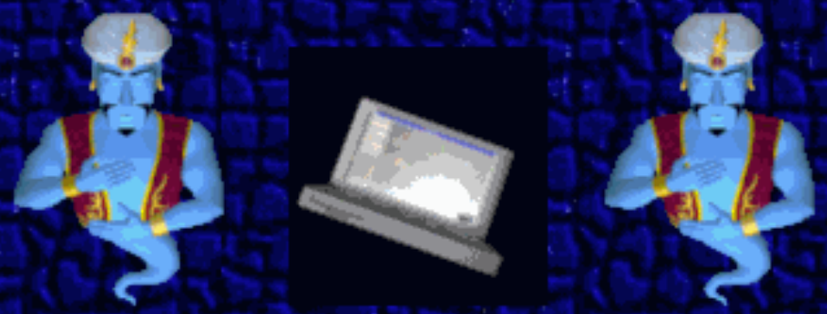
http://

Submit

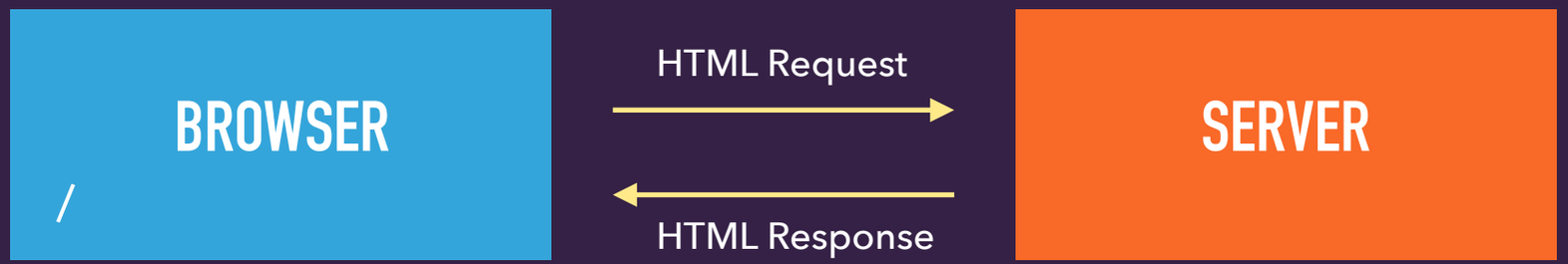
Some pages may work very slowly or not all. Many webapps are just too advanced for Geocities.

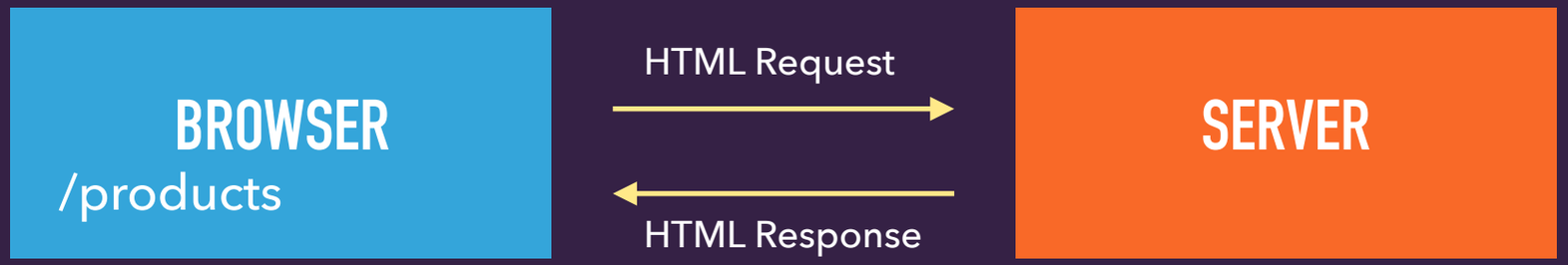
Turn your sound up for the full effect.

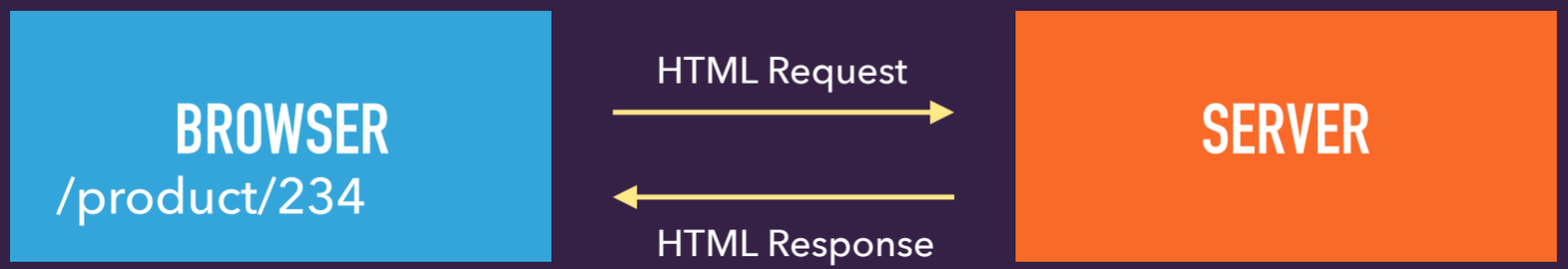
Created by Mike Lacher



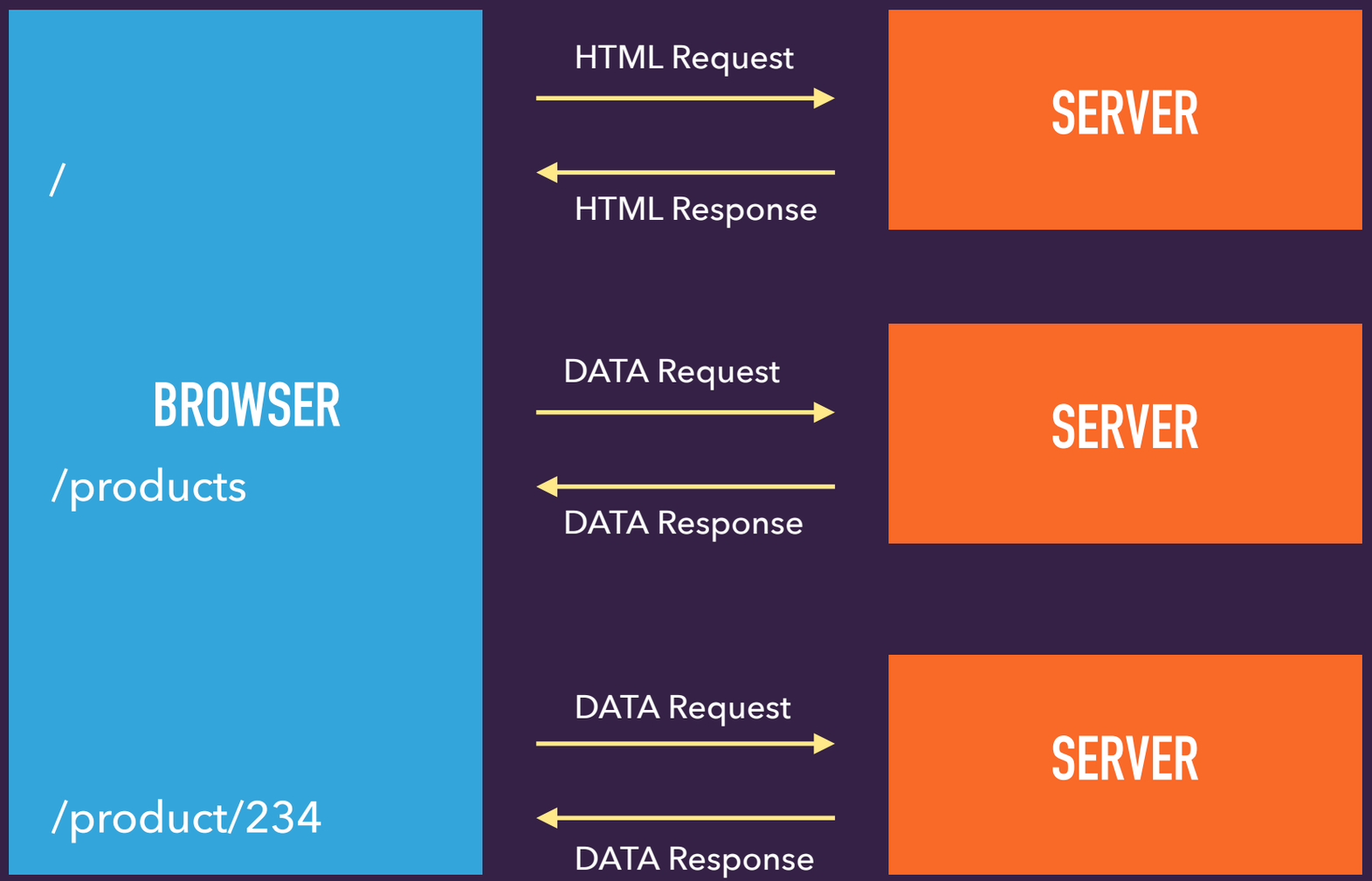
TRADITIONAL WEBSITE







SINGLE PAGE APPLICATION



Chrome Plik Edytuj Widok Historia Zakładki Osoby Okno Pomoc

TenderHut S.A. - Technologicz

https://tenderhut.com

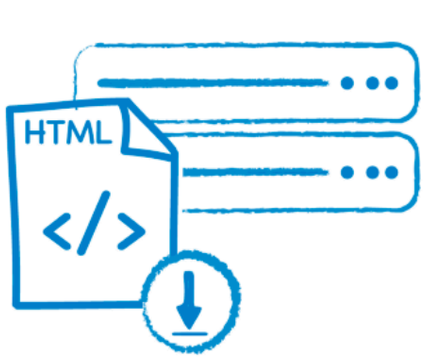
414 × 736

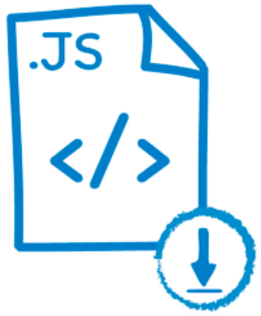
Elements Console Sources Network Performance Memory Application Security Audits

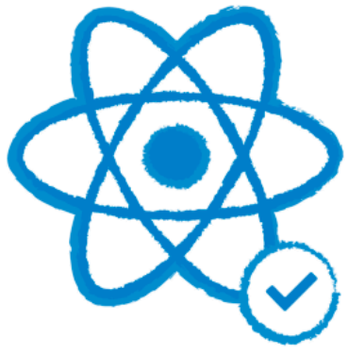
(index) :formatted

```
<!DOCTYPE html>
<html lang="en" prefix="og: http://ogp.me/ns#">
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width,initial-scale=1,shrink-to-fit=no">
        <link rel="apple-touch-icon" sizes="180x180" href="/icons/apple-touch-icon.png?v=694AzX47gQ">
        <link rel="icon" type="image/png" sizes="32x32" href="/icons/favicon-32x32.png?v=694AzX47gQ">
        <link rel="icon" type="image/png" sizes="16x16" href="/icons/favicon-16x16.png?v=694AzX47gQ">
        <link rel="mask-icon" href="/icons/safari-pinned-tab.svg?v=694AzX47gQ" color="#5bbad5">
        <link rel="shortcut icon" href="/icons/favicon.ico?v=694AzX47gQ">
        <meta name="msapplication-TileColor" content="#2d89ef">
        <meta name="msapplication-config" content="/icons/browserconfig.xml?v=694AzX47gQ">
        <meta name="theme-color" content="#ffffff">
        <meta name="google-site-verification" content="NCJvpBpTbp9pUJdDk7t6dhIisOj4kZJEphgntRU-TbM"/>
        <link rel="manifest" href="/manifest.json">
        <title>TenderHut</title>
        <link href="/static/css/main.5ca042e5.css" rel="stylesheet">
    </head>
    <body>
        <noscript>You need to enable JavaScript to run this app.</noscript>
        <div id="app-root"></div>
        <div id="modal-root"></div>
        <script type="text/javascript" src="/static/js/main.dc65024f.js"></script>
    </body>
</html>
```

CSR





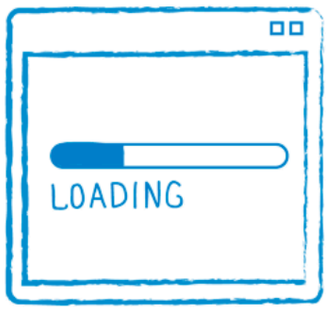


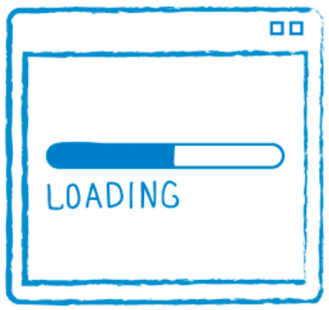
Server Sending Response to Browser

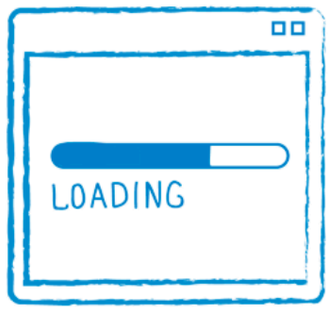
Browser Downloads JS

Browser executes React

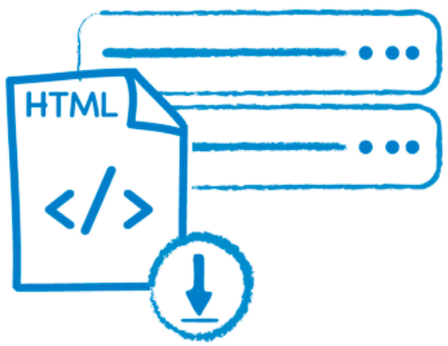
Page Now **Viewable** and **Interactable**



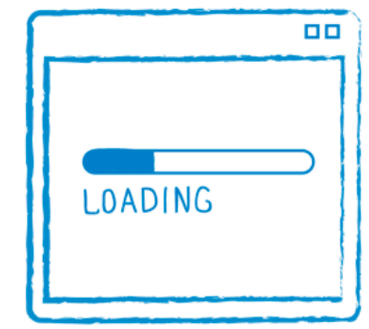


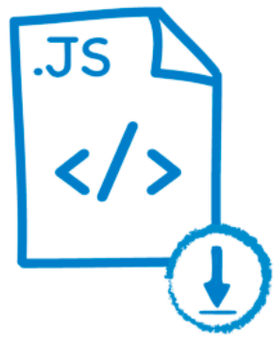


SSR

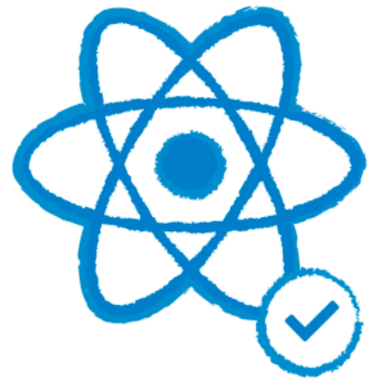


Server Sending Ready to be rendered HTML Response to Browser





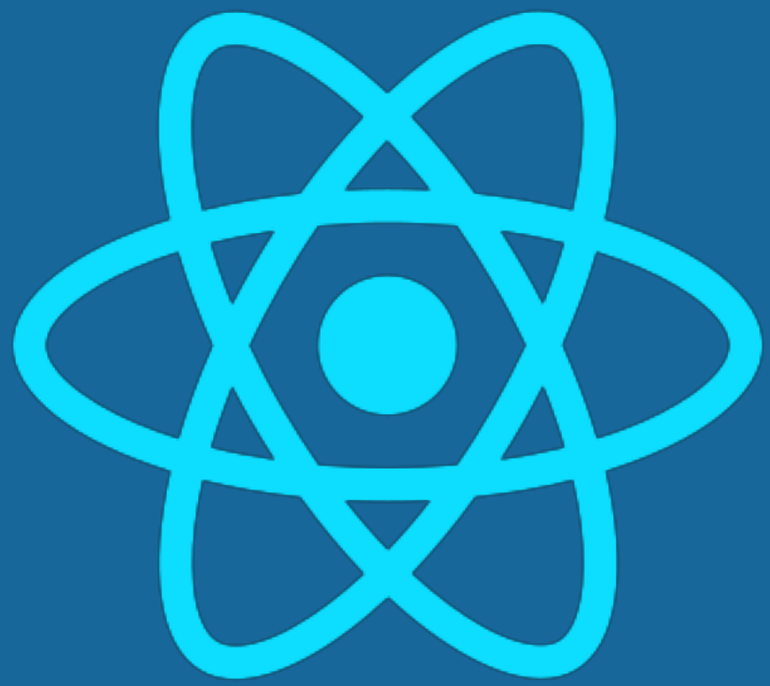
Browser Renders the page, **Now Viewable**, and Browser Downloads JS



Browser executes React

Page Now **Interactable**

WHAT IS REACT?

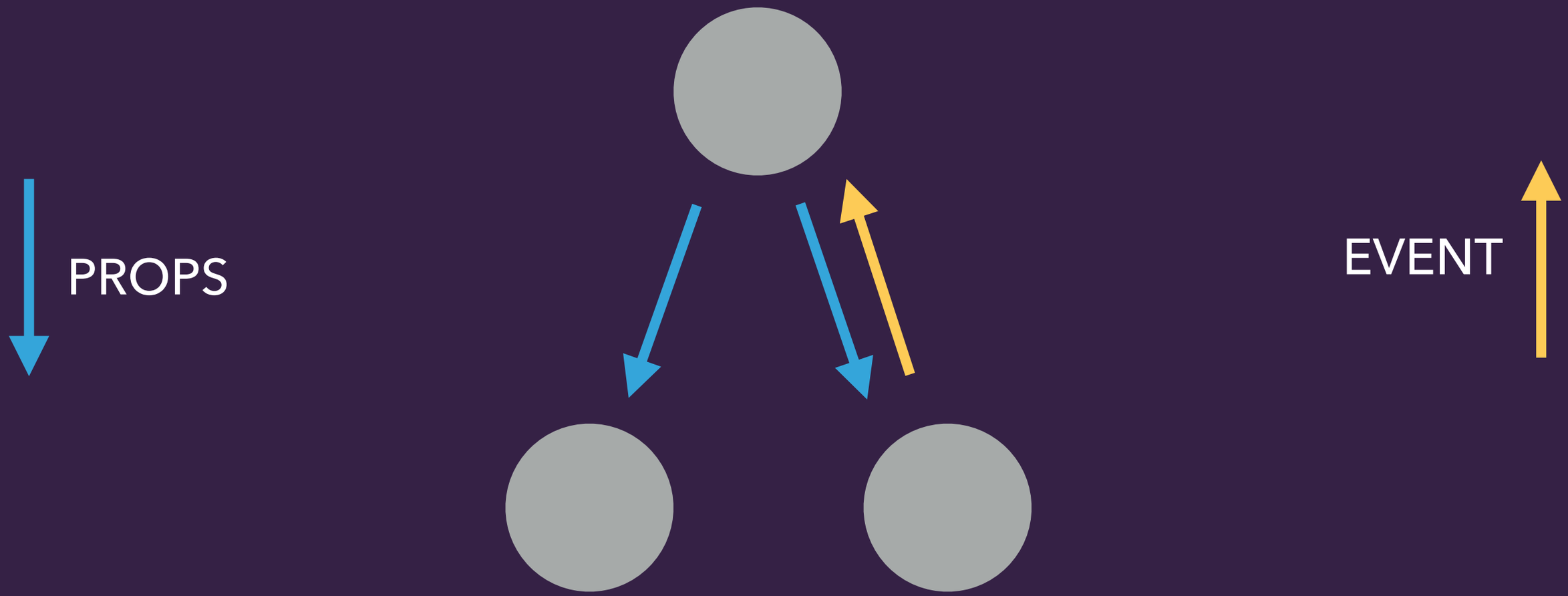


LIBRARY NOT ~~FRAMEWORK~~

LOW LEARNING CURVE

ONE WAY DATA FLOW

ONE WAY DATA FLOW



NO CONTROLLERS

NO MODELS

NO DIRECTIVES

NO GLOBAL EVENT LISTENER

JUST COMPONENT

COMPONENT

ISOLATED

REUSABLE

TESTABLE

COMPONENT TYPES

FUNCTIONAL

```
function Welcome(props) {
    return <h1>Hello, {props.name}</h1>;
}
```

FUNCTIONAL

```
const Welcome = props => {
    return <h1>Hello, {props.name}</h1>;
};
```

FUNCTIONAL

```
const Welcome = ({name}) => {
    return <h1>Hello, {name}</h1>;
};
```

CLASS

```
class Welcome extends React.Component {
    render() {
        const {name} = this.props;
        return (
            <h1>Hello, {name}</h1>
        )
    }
}
```

EVERYTHING IS A COMPONENT

CardComponent

CardHeaderComponent

CardBodyComponent

UserPhotoComponent

```jsx
import React from 'react';

const FacebookComponent = () => (
    <CardComponent>
        <CardHeaderComponent>
            <SelectorComponent />
            <SelectorComponent />
            <SelectorComponent />
        </CardHeaderComponent>

        <CardBodyComponent>
            <UserPhotoComponent />
        </CardBodyComponent>

        <CardFooterComponent />
    </CardComponent>
);

export default FacebookComponent;

```

JSX

```jsx
const Welcome = () => {
  return <h1>Hello, World</h1>;
};
```

JSX

```js
const Welcome = function Welcome() {
    return React.createElement(
        type: "h1",
        props: null,
        children: "Hello, World"
    );
};
```

JS

REACT DOESN'T REQUIRE USING JSX

VIRTUAL DOM

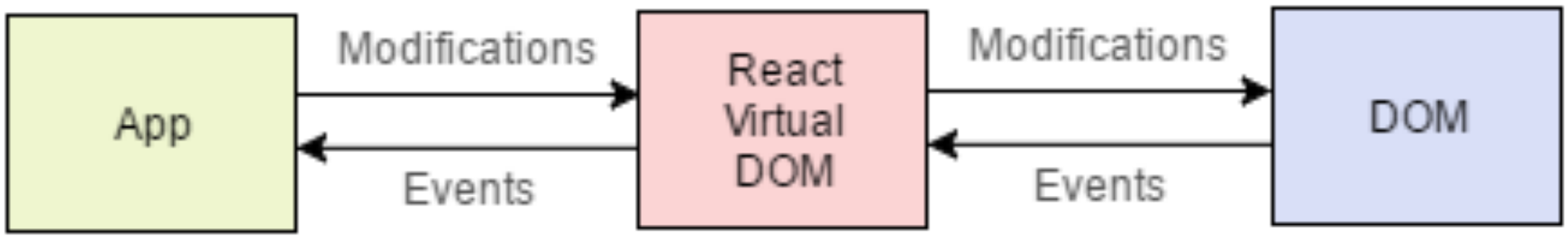
IT'S FAST

IT'S PURE

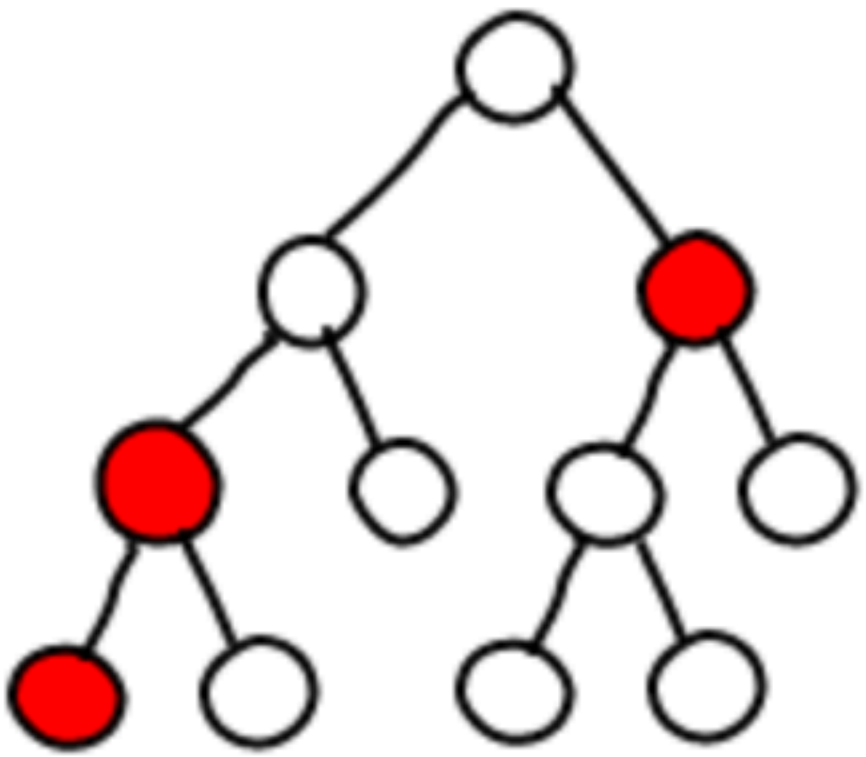
IT WORKS

Traditional Web Application

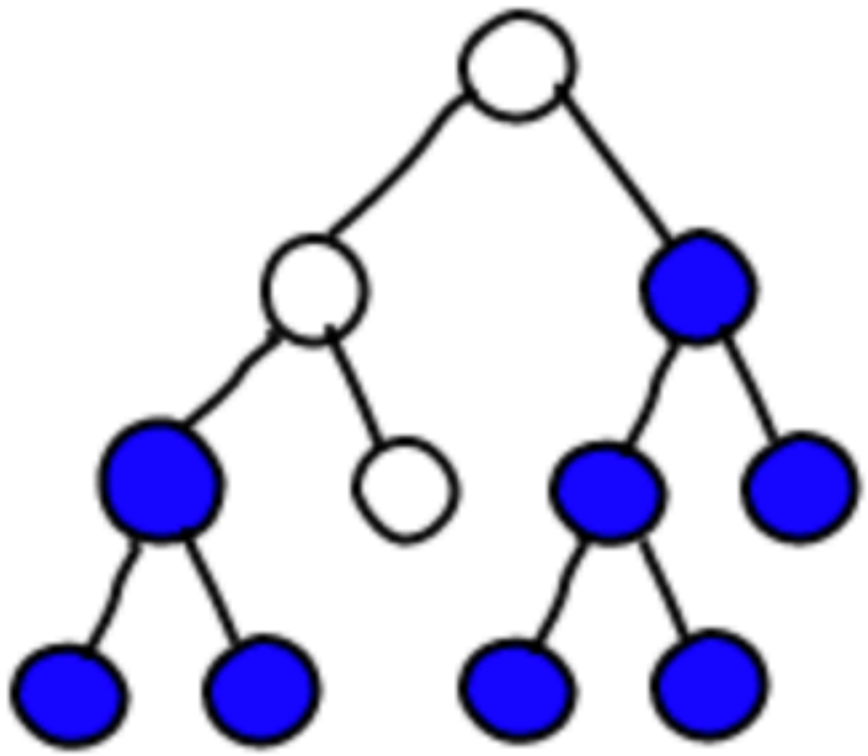
React.js



Dirty

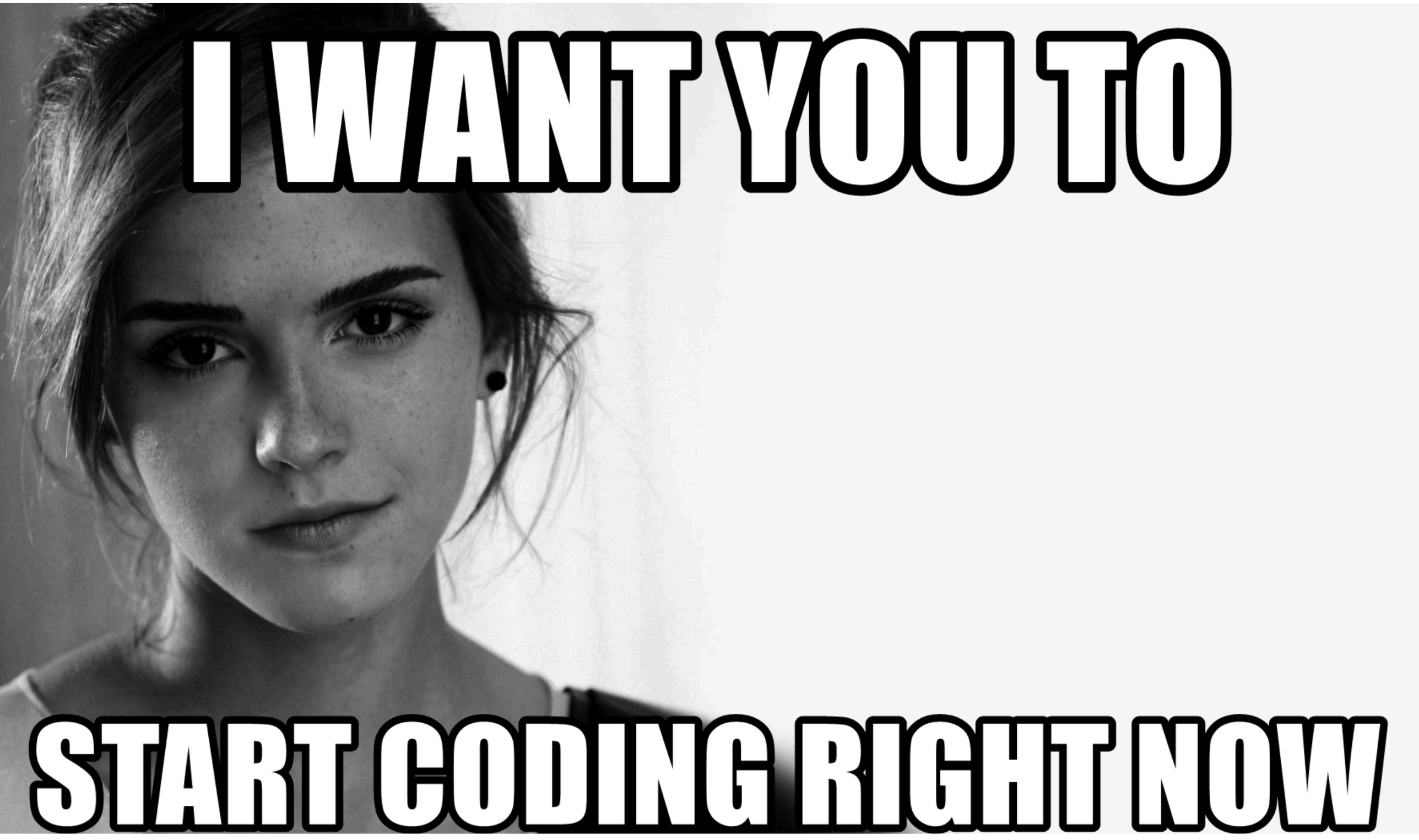


Re-rendered



Hello, world!

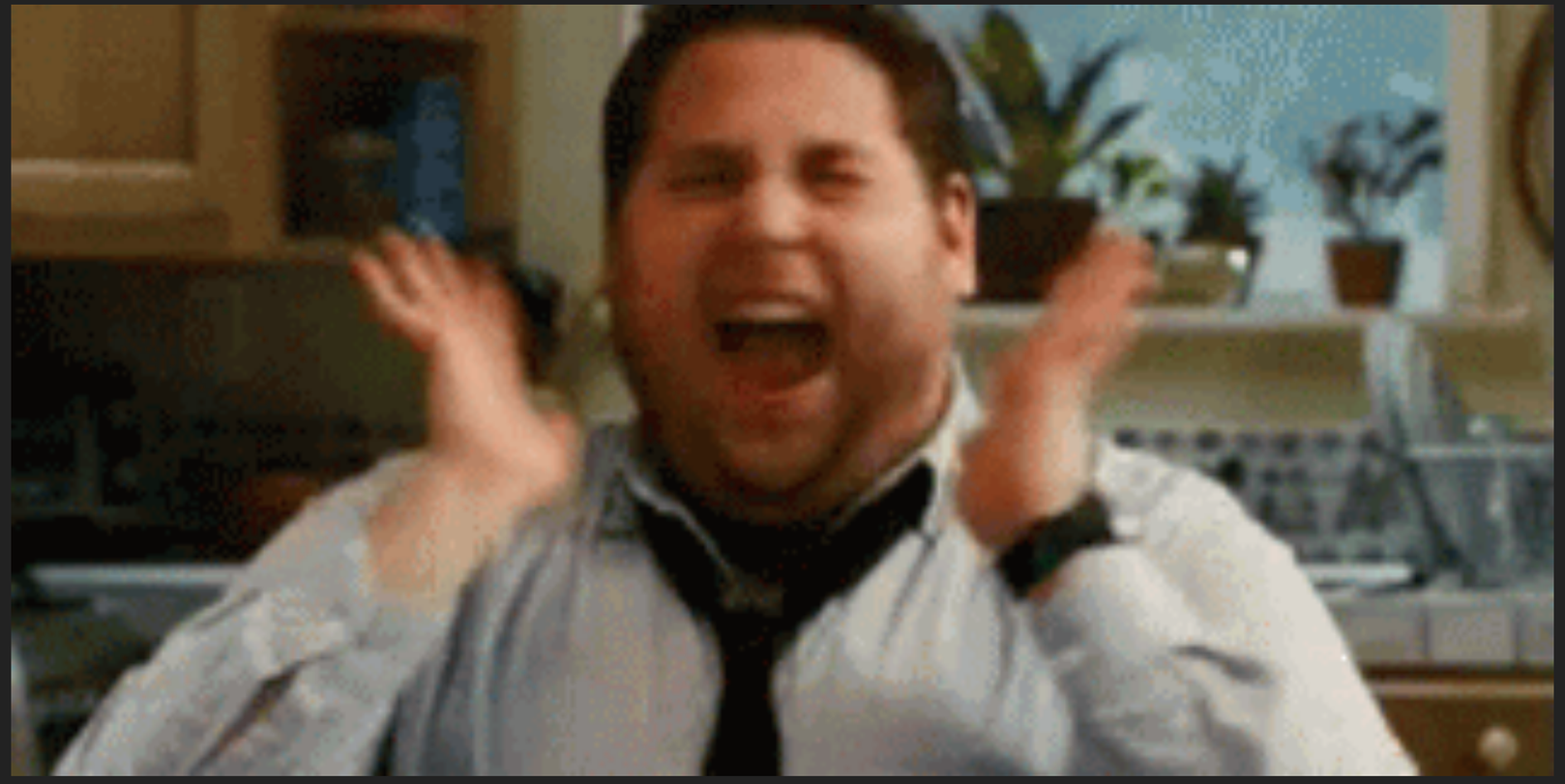
It is 12:26:46 PM.


I WANT YOU TO
START CODING RIGHT NOW

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>React App</title>
    <script src="https://unpkg.com/react@16/umd/react.production.min.js" crossorigin></script>
    <script src="https://unpkg.com/react-dom@16/umd/react-dom.production.min.js" crossorigin></script>
  </head>
  <body>
    <div id="root"></div>

    <script type="text/javascript">
        var Welcome = function Welcome() {
            return React.createElement("h1", null, "Hello, World");
        };

        ReactDOM.render(Welcome, document.getElementById("root"));
    </script>
  </body>
</html>
```

CREATE REACT APP

npx create-react-app myapp



DEMO

REACT DEV TOOLS

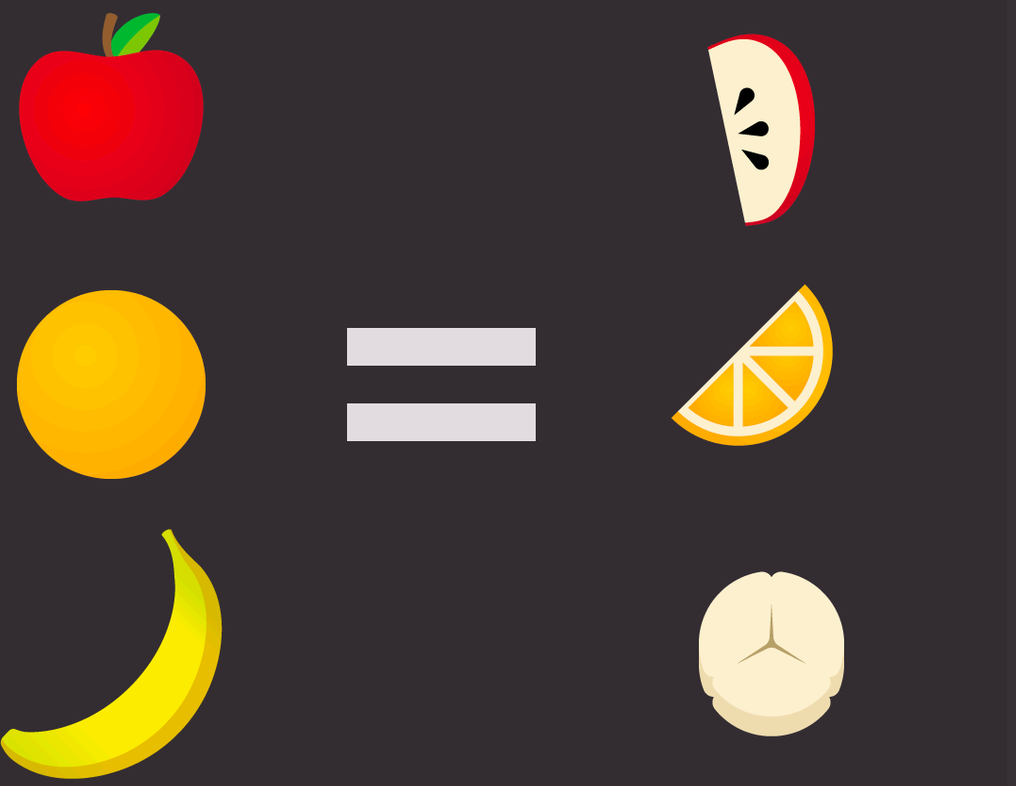
DEMO

THANKS

REACT JS

FUNCTIONAL PROGRAMMING

MAP



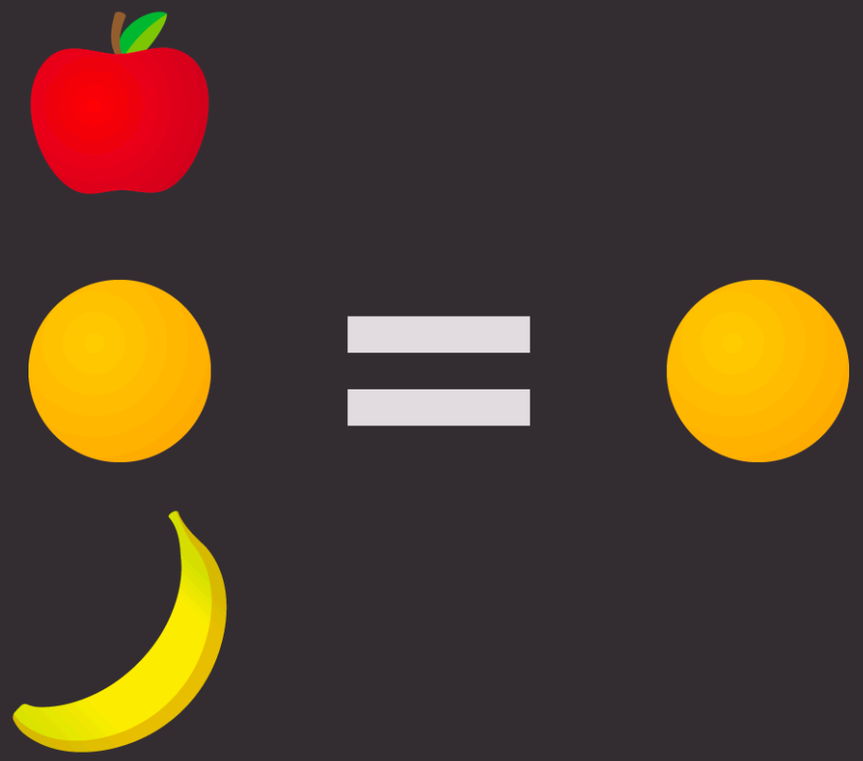
MAP

```
const array = [1,2,3,4,5];
```

```
const newArray = array.map(element => element + 1);
```

```
> newArray[2,,,,]
> newArray[2,3,,,]
> newArray[2,3,4,,]
> newArray[2,3,4,5,]
> newArray[2,3,4,5,6]
```

FILTER



FILTER

```
const array = [1,2,3,4,5];
const newArray = array.map(element => element > 3);
```

```
> newArray[]
> newArray[]
> newArray[]
> newArray[4]
> newArray[4,5]
```

RENDER MULTIPLE COMPONENT

RENDER MULTIPLE COMPONENT

```jsx
const App = () => (
  <div>
    <h4>Asia</h4>
    <h4>Paweł</h4>
    <h4>Gosia</h4>
    <h4>Darek</h4>
  </div>
);
```

```jsx
const App = () => (
  <div>
  {[
    <h4>Asia</h4>,
    <h4>Paweł</h4>,
    <h4>Gosia</h4>,
    <h4>Darek</h4>
  ]}
  </div>
);
```

RENDER MULTIPLE COMPONENT

```
const names = [
    "Asia",
    "Paweł",
    "Gosia",
    "Darek"
];
```

map()

```
{[
    <h4>Asia</h4>,
    <h4>Paweł</h4>,
    <h4>Gosia</h4>,
    <h4>Darek</h4>
]}
```

RENDER MULTIPLE COMPONENT

```
const names = ["Asia", "Paweł", "Gosia", "Darek"];
```

```
names.map(name => <h4>{name}</h4>);
```

```
{[
  <h4>Asia</h4>,
  <h4>Paweł</h4>,
  <h4>Gosia</h4>,
  <h4>Darek</h4>
]}
```

RENDER MULTIPLE COMPONENT

```
const names = ["Asia", "Paweł", "Gosia", "Darek"];

const App = () => (
  <div>
    {names.map(name => (
      <h4>{name}</h4>
    ))}
  </div>
);
```

```
00:35:03.447 ▶Warning: Each child in an array or iterator should have  index.js:1452
a unique "key" prop.

Check the render method of `App`. See https://fb.me/react-warning-keys
for more information.
    in h4 (at App.js:10)
    in App (at src/index.js:7)
```

KEY SHOULD BE UNIQUE

NOT

~~TIMESTAMP~~

~~RANDOM~~

GOOD

```
▼ <App> == $r
  ▼ <div>
    ▼ <div className="content">
        <h4 key="0">Asia</h4>
        <h4 key="1">Paweł</h4>
        <h4 key="2">Gosia</h4>
        <h4 key="3">Darek</h4>
      </div>
    ▼ <div className="sidebar">
        <h4 key="0">Skoda</h4>
        <h4 key="1">VW</h4>
        <h4 key="2">BMW</h4>
        <h4 key="3">Porsche</h4>
      </div>
    </div>
</App>
```

ANTI PATTERN

KEY={INDEX}

BUT.....

DEMO

KEY ISN'T PASSED TO COMPONENT

PROPS

PROPS = PRIMITIVE VALUES, REACT ELEMENTS FUNCTIONS

PROPS
DEFAULT IS TRUE

```jsx
<MyComponent dark />
```

```jsx
<MyComponent dark={true} />
```

PROPS
SPREAD ATTRIBUTES

```
const postData = {
  title: "Post Title",
  image: "https://pic.com/1.jpg",
  text: "Lorem ipsum dolor sit."
};

<Post {...postData} />
```

=

```
<Post
    title="Post Title"
    image="https://pic.com/1.jpg"
    text="Lorem ipsum dolor sit."
/>
```

NOTHING RENDER

```
<div />
```

```
<div></div>
```

```
<div>{false}</div>
```

```
<div>{null}</div>
```

```
<div>{undefined}</div>
```

```
<div>{true}</div>
```

CONDITIONAL RENDERING

CONDITIONAL RENDERING

```
const App = () => {
  if (post) {
    return <Post {...post} />;
  } else {
    return null;
  }
};
```

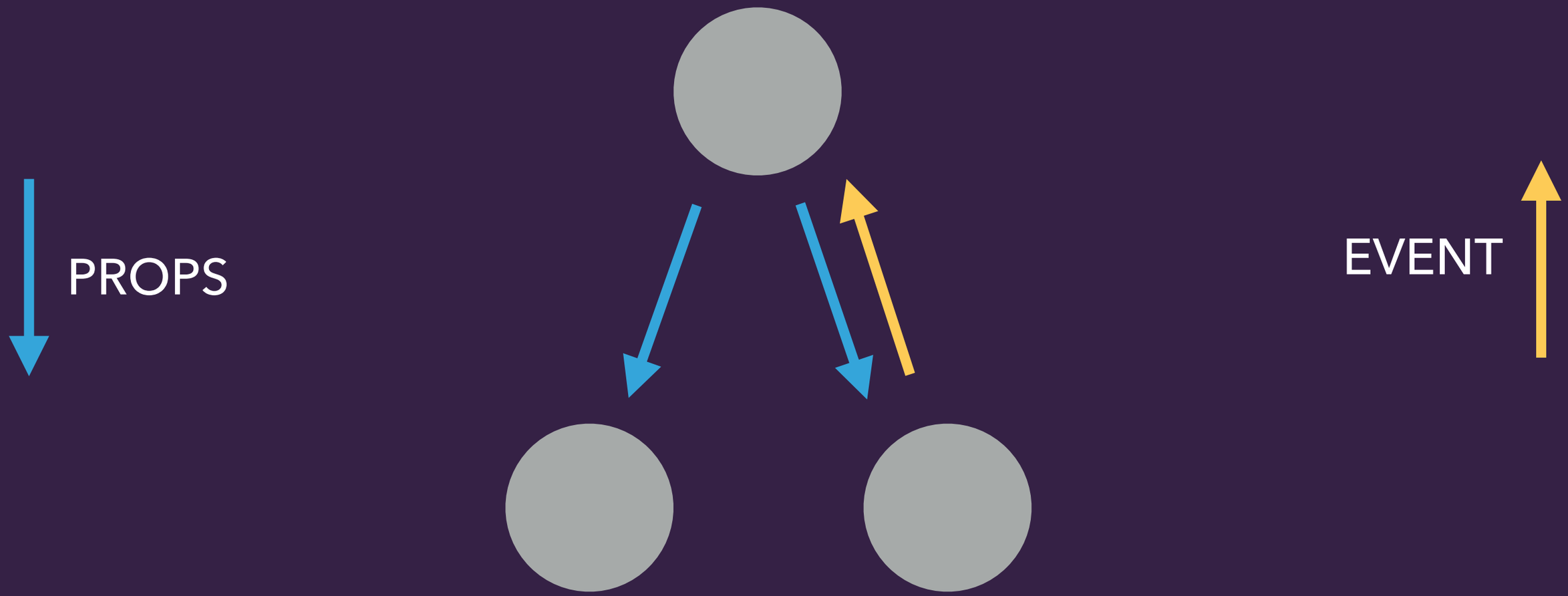
```
const App = () => {
  return post ? <Post {...post} /> : null;
};
```

CONDITIONAL RENDERING

```
const App = () => {
  return post && <Post {...post} />;
};
```

```
true && expression = expression
false && expression = false
```

ONE WAY DATA FLOW



REACT

EVENTS

event props of React element

```
// ReactElement
<button onClick={ this.handleClick }>Click!</button>
<input type="text" defaultValue="" onBlur={ this.handleBlur } />

// HTML DOM element
<button onclick="handle_click();">Click!</button>
<input type="text" value="" onblur="handle_blur();" />
```

event attributes of HTML DOM element

REACT
EVENTS

```
const App = () => {
  return <Button name="Save" />;
};


const Button = ({ name }) => (
  <button onClick={() => alert("Click Button”)}>
    {name}
  </button>
);
```

REACT
EVENTS

```
const onButtonClick = () => alert("Click Button");

const App = () => {
  return <Button name="Save" onClick={onButtonClick}/>;
};

const Button = ({ name, onClick }) => (
  <button onClick={onClick}>{name}</button>
);
```

PROPTYPES

PROPTYPES

```
import PropTypes from "prop-types";

const Hello = ({ name }) => <h1>Hello, {name}</h1>;

Hello.propTypes = {
  name: PropTypes.string
};

Hello.defaultProps = {
  name: 'unknown'
};
```

PROPTYPES

```
22:52:11.651 ▶Warning: Failed prop type: Invalid prop `name` of type `array`
              supplied to `Hello`, expected `string`.
                  in Hello (at App.js:6)
                  in App (at src/index.js:7)
```

PROPTYPES

https://github.com/facebook/prop-types

CONTEXT API

CONTEXT

```
const App = () => (
  <Wrapper lang="pl">
    <Content lang={props.lang}>
      <BlogPosts lang={props.lang}>
        <Post lang={props.lang} />
      </BlogPosts>
    </Content>
    <Sidebar lang={props.lang}>
      <SomeComponent lang={props.lang}>
        <NewsletterForm lang={props.lang} />
      </SomeComponent>
    </Sidebar>
  </Wrapper>
);
```

CONTEXT

```
const Context = React.createContext();
```

```
<Context.Provider />
```

```
<Context.Consumer />
```

<PROVIDER>

<CONSUMER>

```jsx
const LanguageContext = React.createContext();

const App = () => (
  <LanguageContext.Provider value="pl">
    <Wrapper>
      <Content>
        <BlogPosts>
          <LanguageContext.Consumer>
            {lang => <Post lang={lang} />}
          </LanguageContext.Consumer>
        </BlogPosts>
      </Content>
      <Sidebar>
        <SomeComponent>
          <LanguageContext.Consumer>
            {lang => <NewsletterForm lang={lang} />}
          </LanguageContext.Consumer>
        </SomeComponent>
      </Sidebar>
    </Wrapper>
  </LanguageContext.Provider>
);
```

THANKS

REACT JS

HOC

HIGH ORDER COMPONENT

HOC

```
component => componentOnSteroids;
```

HOC

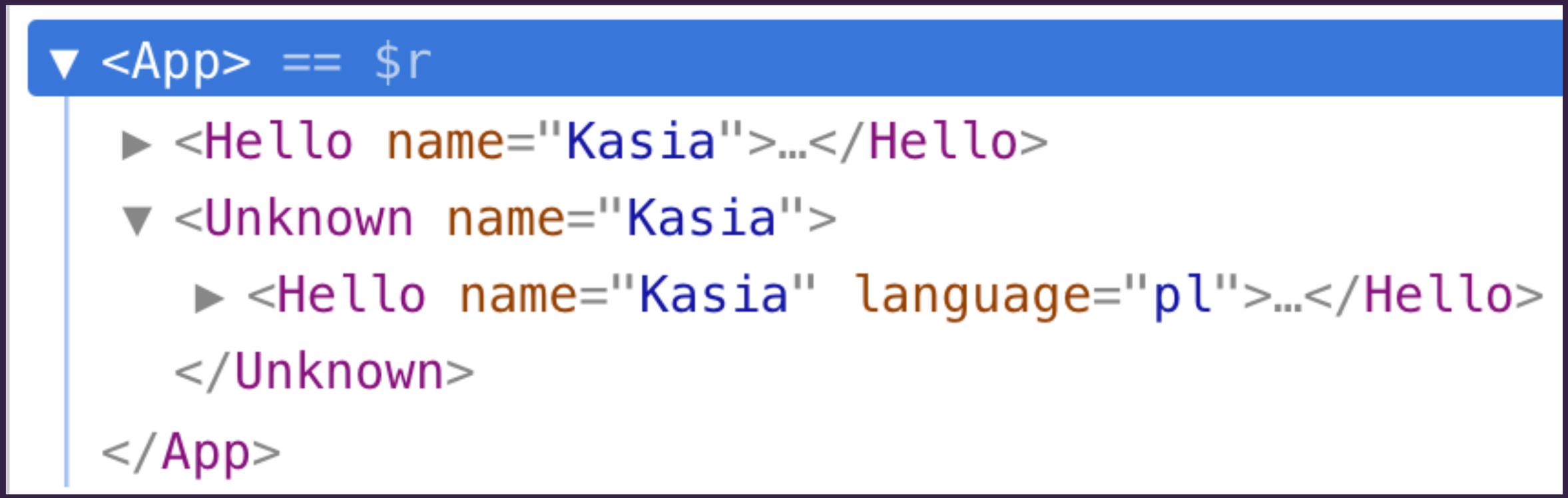
```
const EnhancedComponent = higherOrderComponent(WrappedComponent);
```

HOC

```
const withLanguage = WrappedComponent => {
  return props => (
    <WrappedComponent {...props} language="pl" />
  );
};
```

```
const HelloWithLang = withLanguage(Hello);

const App = () => (
  <>
    <Hello name="Kasia" />
    <HelloWithLang name="Kasia" />
  </>
);
```

HOC


▼ <App> == $r
▶ <Hello name="Kasia">…</Hello>
▼ <Unknown name="Kasia">
▶ <Hello name="Kasia" language="pl">…</Hello>
</Unknown>
</App>

HOC

HOC

```
function logProps(InputComponent) {
    InputComponent.prototype.componentWillReceiveProps = function(nextProps) {
        console.log('Current props: ', this.props);
        console.log('Next props: ', nextProps);
    };
    return InputComponent;
}

const EnhancedComponent = logProps(InputComponent);
```

HOC

```
function logProps(WrappedComponent) {
    return class extends React.Component {
        componentWillReceiveProps(nextProps) {
            console.log('Current props: ', this.props);
            console.log('Next props: ', nextProps);
        }
        render() {
            return <WrappedComponent {...this.props} />;
        }
    }
}
```

HOC

```
const EnhancedComponent = hoc(Component);

const EnhancedComponent = anotherHoc(Component, config);
```

```
const ConnectedComponent = connect(param1, param2)(Component);
const enhance = connect(param1, param2);
const ConnectedComponent = enhance(Component);
```

HOC

```
const EnhancedComponent = withRouter(connect(param)(WrappedComponent))
```

```
const enhance = compose(
    withRouter,
    connect(commentSelector)
)
const EnhancedComponent = enhance(WrappedComponent)
```

HOC

```
@withRouter
@connect(param)
export default class MyFancyComponent extends React.Component {

}
```

HOC

react-fns

https://github.com/jaredpalmer/react-fns

CLASS COMPONENT

CLASS COMPONENT

```
const Welcome = props => {
    return <h1>Hello, {props.name}</h1>;
}
```

```
class Welcome extends React.Component {
    render() {
        return <h1>Hello, {this.props.name}</h1>;
    }
}
```

CLASS COMPONENT

„When should I use a function and when a class?"

~undefined

CLASS COMPONENT DIFFERENCES

CLASS COMPONENT SYNTAX

CLASS COMPONENT DIFFERENCES

```
var Welcome = function Welcome(props) {
  return React.createElement(
        "h1",
        null,
        "Hello, ",
        props.name);
};
```

CLASS COMPONENT DIFFERENCES

```
var Welcome = (function(_React$Component) {
    _inherits(Welcome, _React$Component);

    function Welcome() {
        _classCallCheck(this, Welcome);

        return _possibleConstructorReturn(
            this,
            (Welcome.__proto__ || Object.getPrototypeOf(Welcome)).apply(
            this,
            arguments
            )
        );
    }

    _createClass(Welcome, [
        {
            key: "render",
            value: function render() {
                return React.createElement("h1", null, "Hello, ", this.props.name);
            }
        }
    ]);

    return Welcome;
```

CLASS COMPONENT STATE

CLASS COMPONENT STATE

STATE VS PROPS

CLASS
COMPONENT
STATE

COMMON

plain JS objects

render update

deterministic

CLASS
COMPONENT
STATE

PROPS

Component's configuration

Received from above

Immutable

CLASS
COMPONENT
STATE

STATE

Component's information

Created in component

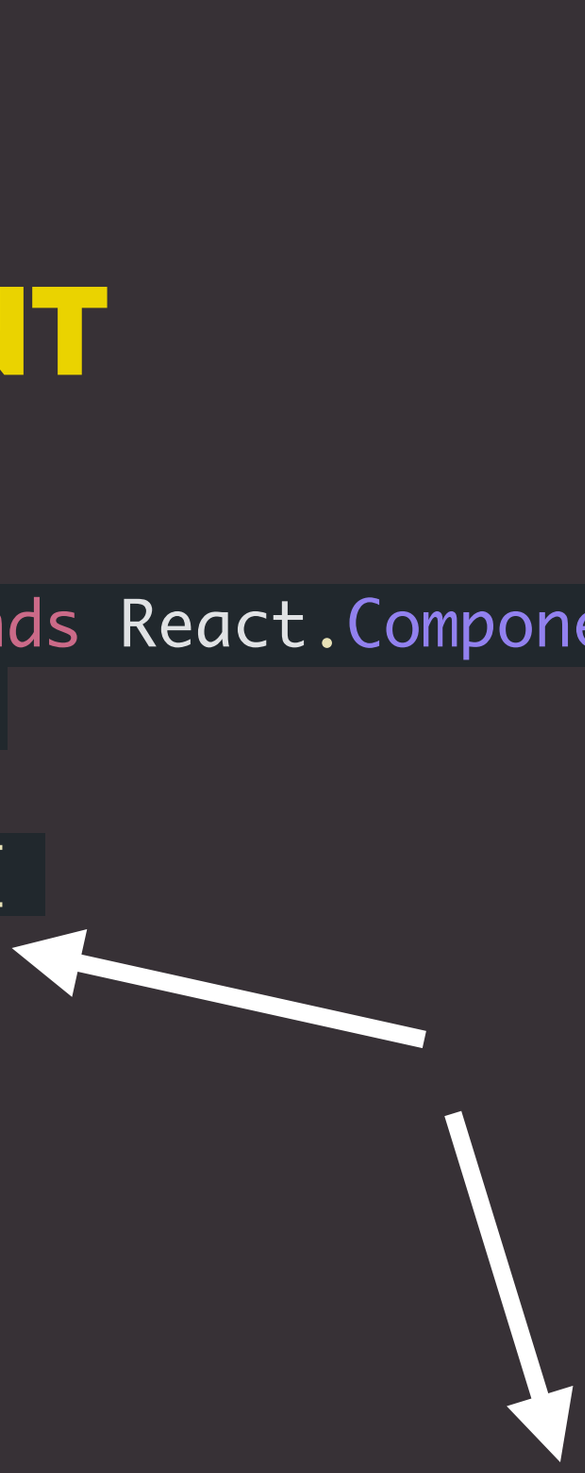
Changeable

CLASS COMPONENT STATE

```
class Button extends React.Component {
    constructor() {
        super();
        this.state = {
            count: 0,
        };
    }

    render() {
      ……
    }
}
```

CLASS COMPONENT STATE

```
class Button extends React.Component {
  constructor() {
    super();
    this.state = {
      count: 0
    };
  }

  render() {
    return (
        <button>
            Clicked {this.state.count} times
        </button>
    );
  }
}
```



```
class Button extends React.Component {
  constructor() {
    super();
    this.state = {
      count: 0
    };
    this.updateCount = this.updateCount.bind(this)
  }

  updateCount() {
    this.setState((prevState, props) => {
      return { count: prevState.count + 1 };
    });
  }

  render() {
    return (
      <button onClick={this.updateCount}>
        Clicked {this.state.count} times
      </button>
    );
  }
}
```

DEMO

INITIALIZE STATE

```
this.state = {}
```

CHANGE STATE

```
this.setState()
```

setState()

```
this.setState({ count: 2 });
```

```
this.setState((state) => {
    return {count: state.count + 1};
});
```

setState()

```
constructor() {
    super();
    this.state = {
        count: 0;
    }
}

incrementCount() {
    this.setState({count: this.state.count + 1});
}

handleSomething() {
    this.incrementCount();
    this.incrementCount();
    this.incrementCount();
}
```

```
constructor() {
    super();
    this.state = {
        count: 0;
    }
}

incrementCount() {
    this.setState((state) => {
        return {count: state.count + 1}
    });
}

handleSomething() {
    this.incrementCount();
    this.incrementCount();
    this.incrementCount();
}
```

setState()

STATE
ANTI-PATTENR

```
class MyCompo          exten          mponent {
    constructo          ps
        super(p
        this.sta
            som                    rops.someValue,
        };
    }
}
```

„When should I use a function and when a class?"

~undefined

REACT HOOKS

```jsx
import { useState } from 'react';

const Example = () => {
    // Declare a new state variable, which we'll call "count"
    const [count, setCount] = useState(0);

    return (
        <div>
            <p>You clicked {count} times</p>
            <button onClick={() => setCount(count + 1)}>
                Click me
            </button>
        </div>
    );
}
```

CLASS COMPONENT DIFFERENCES

LIFECYCLE HOOKS

LIFECYCLE HOOKS

MOUNTING

UPDATING

UNMOUNTING

“Faza Render”

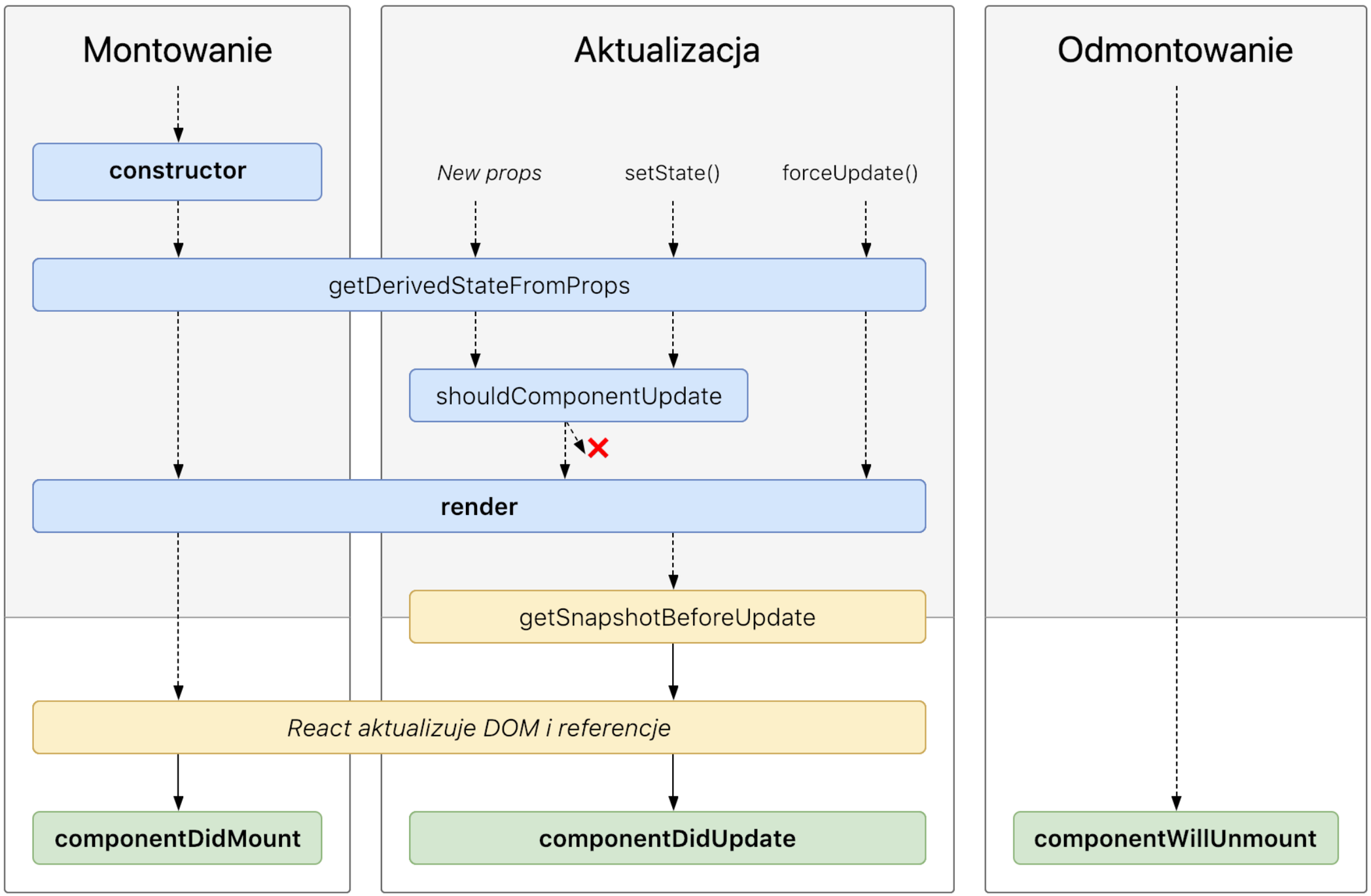
Czysta i bez efektów ubocznych. Może zostać wstrzymana, anulowana lub zre-startowana przez React.

“Faza Pre-commit”

Może czytać DOM.

“Faza Commit”

Może operować na DOM, tworzyć efekty uboczne i kolejkować aktualizacje.



constructor()

initialize state (this.state)

bind methods

static getDerivedStateFromProps()

return an object to update
the state

fired on every render

render()

required

componentDidMount()

load data from a remote endpoint

side-effect

DOM

shouldComponentUpdate()

performance optimization

getSnapshotBeforeUpdate()

componentDidUpdate()

invoked immediately after updating

good to do network requests

componentWillUnmount()

invoked immediately before a component is unmounted and destroyed

cleaning up

BIND

```
class Foo extends Component {
  handleClick() {
    console.log("Click happened", this.state);
  }
  render() {
    return <button onClick={this.handleClick}>Click Me</button>;
  }
}
```

BIND

```
class Foo extends Component {
    constructor(props) {
        super(props);
        this.handleClick = this.handleClick.bind(this);
    }
    handleClick() {
        console.log('Click happened', this.state);
    }
    render() {
        return <button onClick={this.handleClick}>Click Me</button>;
    }
}
```

BIND

```
class Foo extends Component {
  handleClick = () => {
    console.log("Click happened", this.state);
  }
  render() {
    return <button onClick={this.handleClick}>Click Me</button>;
  }
}
```

BIND

```
class Foo extends Component {
  handleClick() {
    console.log("Click happened", this.state);
  };
  render() {
    return (
      <button onClick={this.handleClick.bind(this)}>
         Click Me
      </button>
    );
  }
}
```

BIND

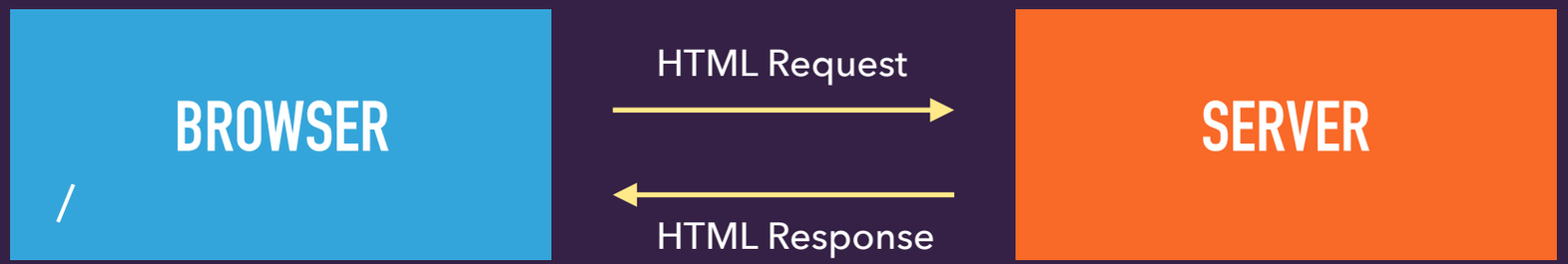
```
class Foo extends Component {
  handleClick() {
    console.log("Click happened", this.state);
  }
  render() {
    return (
      <button onClick={() => this.handleClick()}>
        Click Me
      </button>
    );
  }
}
```

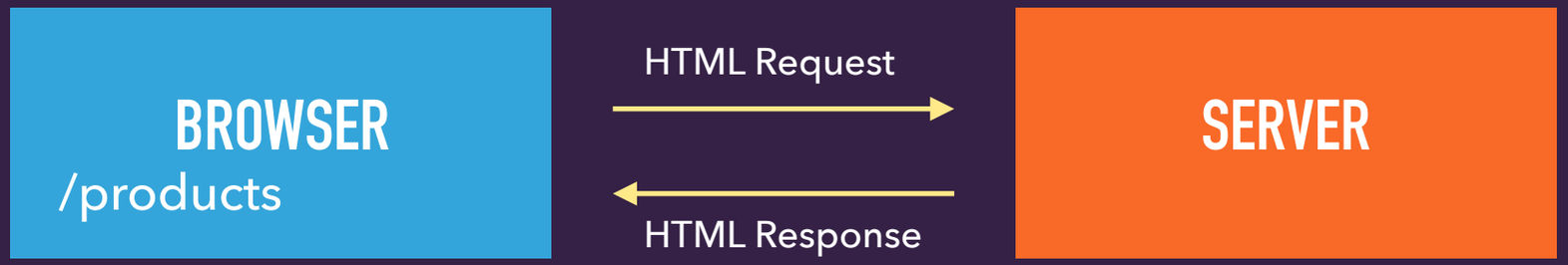
THANKS

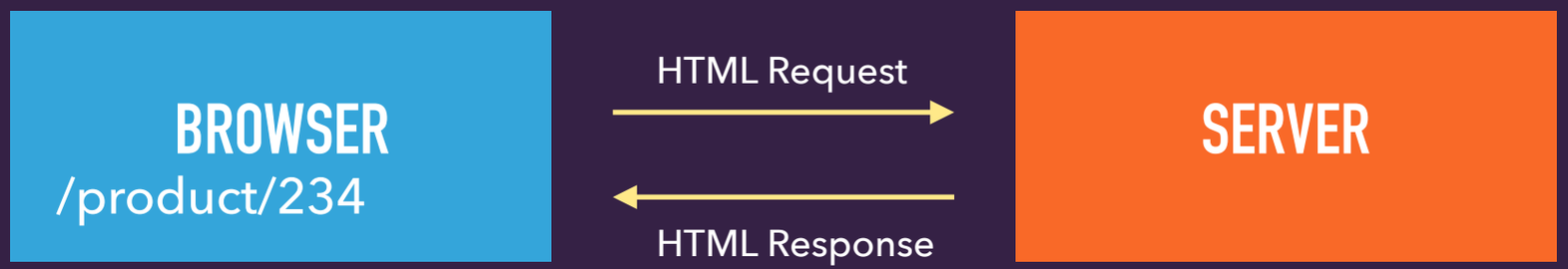
REACT JS

ROUTING

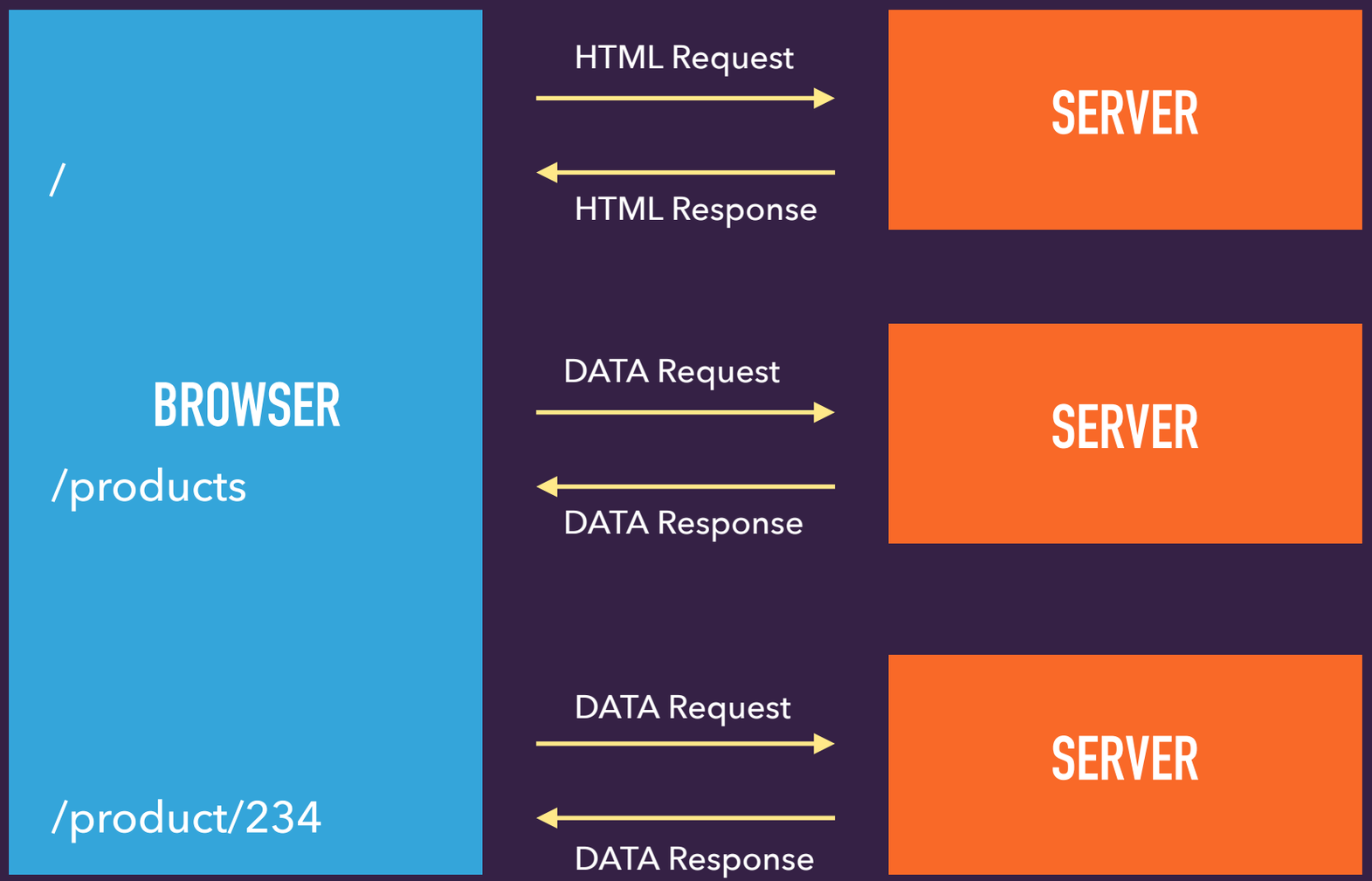
TRADITIONAL WEBSITE







SINGLE PAGE APPLICATION



STATIC ROUTING

DYNAMIC ROUTING

„When we say dynamic routing, we mean routing that takes place as your app is rendering, not in a configuration or convention outside of a running app. That means almost everything is a component in React Router.“

— React Training

localhost:3000

Website

/

footer

localhost:3000/catalog

Website

Catalog Component

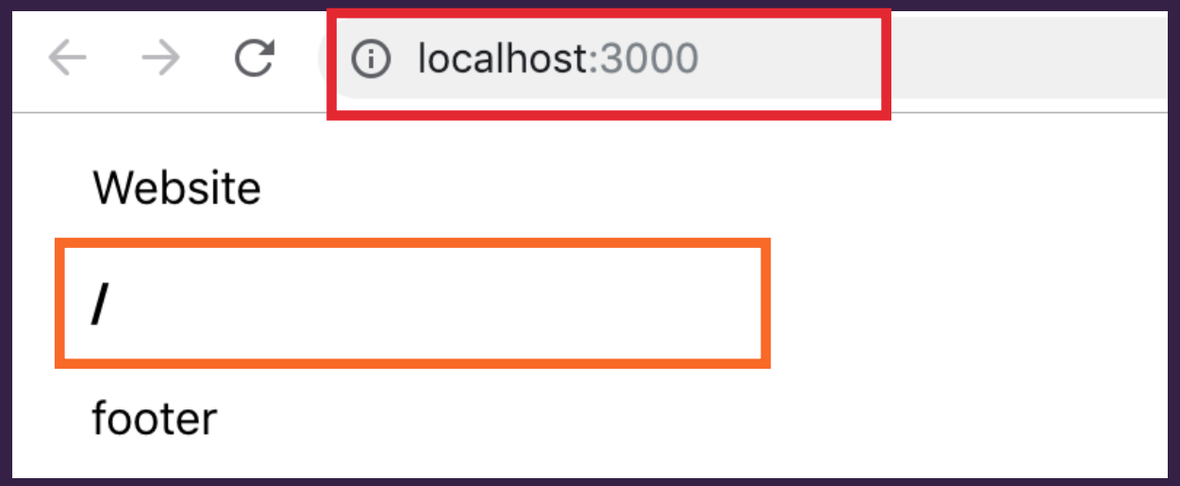
footer

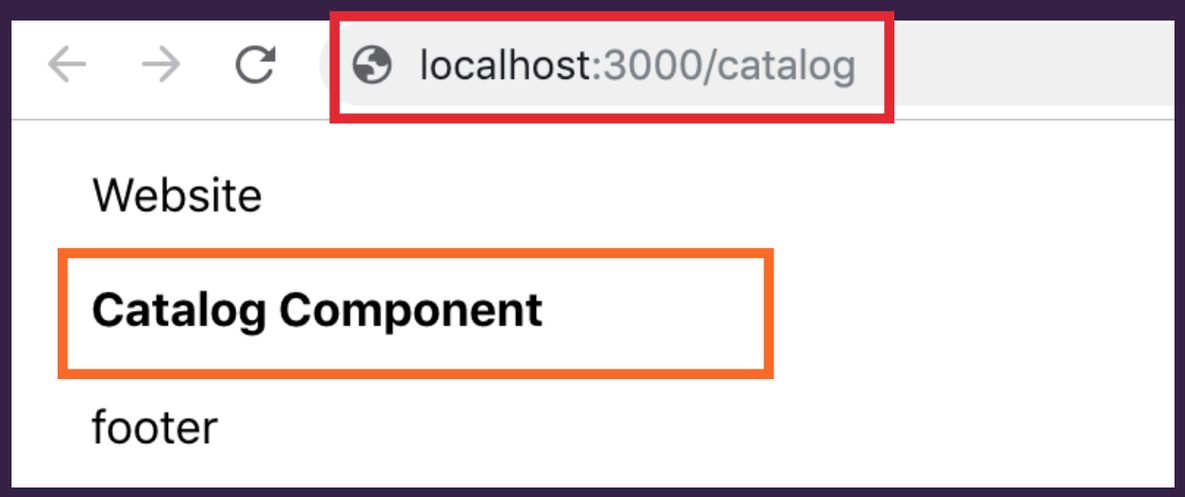
localhost:3000/contact

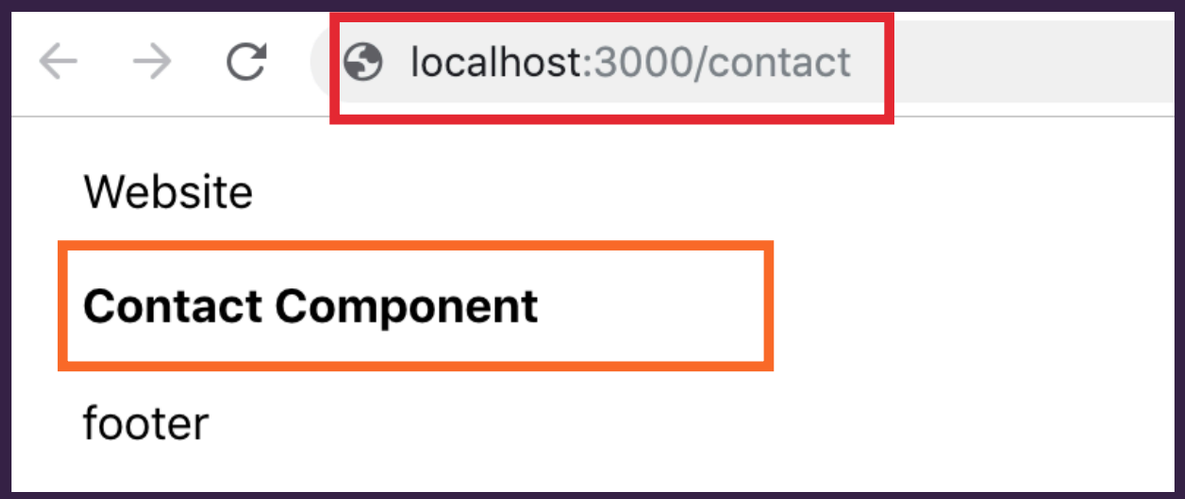
Website

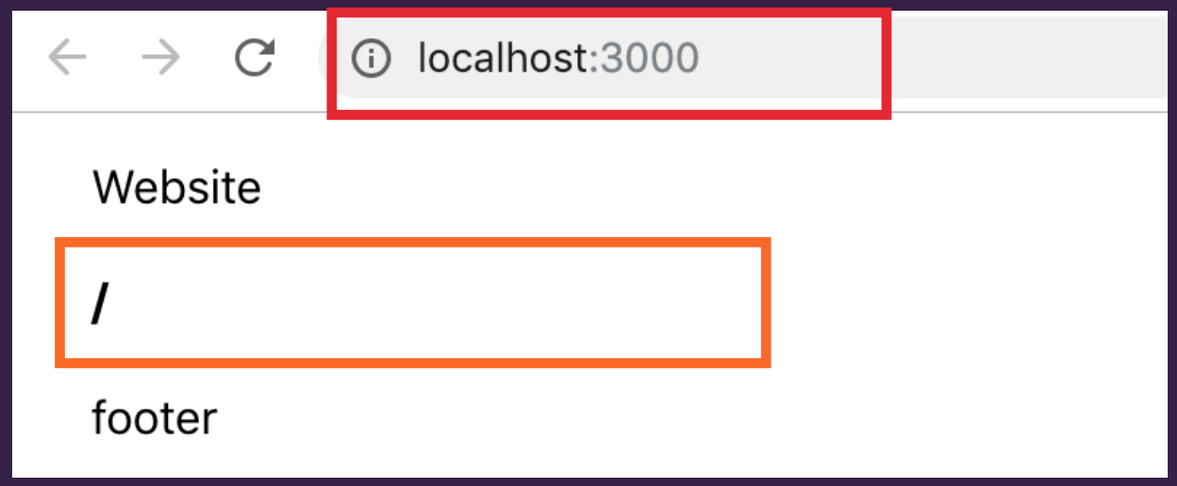
Contact Component

footer

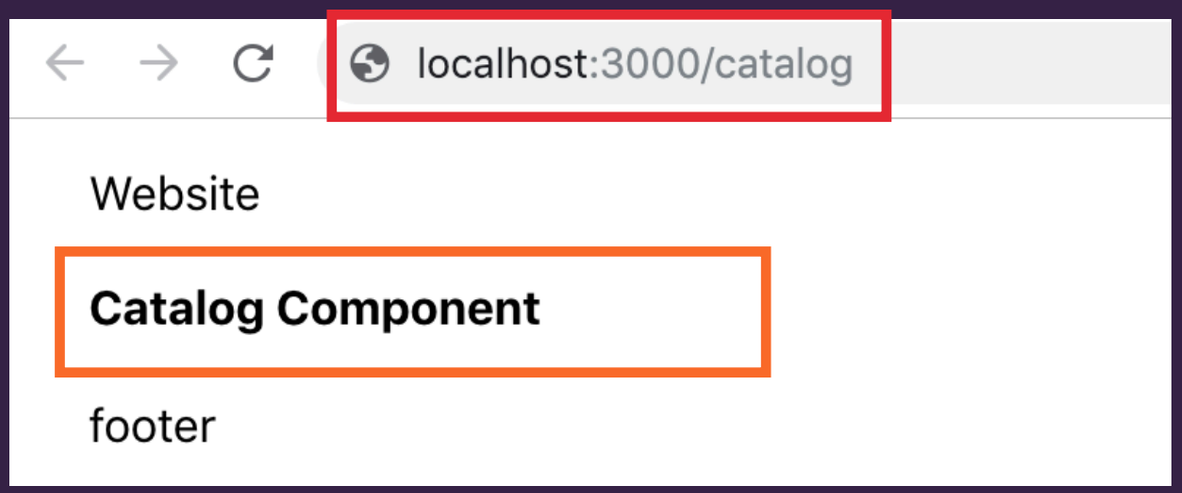

localhost:3000
Website
/
footer


localhost:3000/catalog
Website
Catalog Component
footer

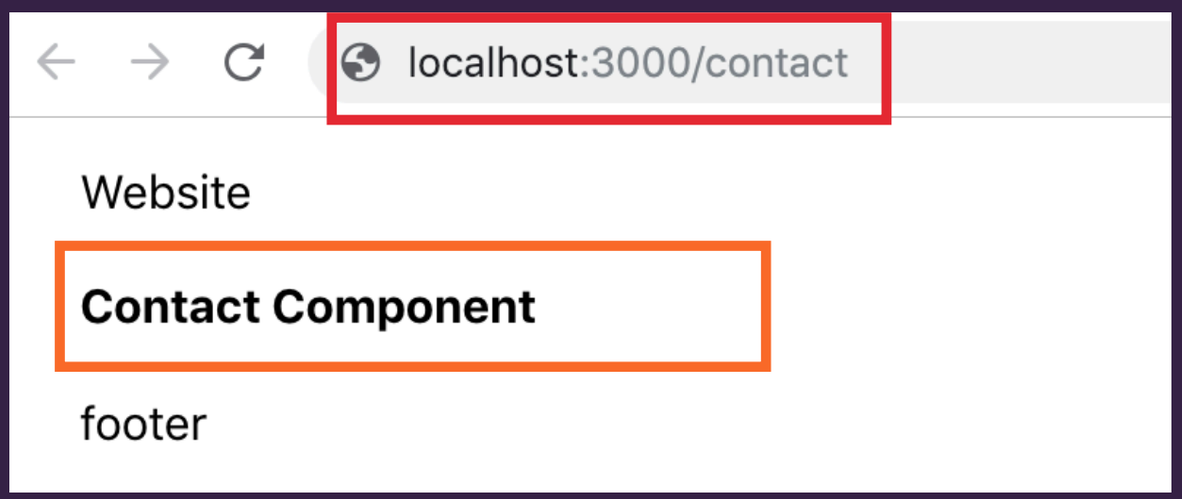

localhost:3000/contact
Website
Contact Component
footer



path: /
component: Home



path: /catalog
component: Catalog



path: /contact
component: Contact

REACT ROUTER V4

PACKAGES

REACT-ROUTER The core of React Router

REACT-ROUTER-DOM DOM bindings for React Router

REACT-ROUTER-NATIVE React Native bindings for React Router

REACT-ROUTER-CONFIG Static route config helpers

REACT ROUTER API

<Router>

```
<Router history={history}>
  <App />
</Router>
```

REACT ROUTER API

<BrowserRouter>

```
<BrowserRouter>
  <App />
</BrowserRouter>
```

REACT ROUTER API

<BrowserRouter>

example.com

example.com/catalog

example.com/contact

REACT ROUTER API

<HashRouter>

```
<HashRouter>
  <App />
</HashRouter>
```

REACT ROUTER API

<HashRouter>

example.com

example.com/#/catalog

example.com/#/contact

REACT ROUTER API

<Route>

```
<Router>
  <div>
    <Route path="/" component={Home}/>
    <Route path="/catalog" component={Catalog}/>
    <Route path="/contact" component={Contact}/>
  </div>
</Router>
```

REACT ROUTER API

<Route>

```
<Route exact path="/" component={Home}/>
<Route strict path=„/catalog/" component={Catalog}/>
<Route sensitive path="/Contact" component={Contact}/>
```

REACT ROUTER API

<Route>

```
<Route path="/" component={Home}/>
<Route path="/" render={() => <Home />}/>
<Route path=„/" children={() => <Home />}/>
```

REACT ROUTER API

<Route>

```
<Router>
  <div>
    <Route path="/" component={Home}/>
  </div>
</Router>


const Home = ({ match, location, history }) => (
  ……
)
```

REACT ROUTER API

<Link>

```
<Link to="/contact">Contact</Link>
```

```
<Link
  to={{
    pathname: "/contact",
    search: „?from=pl",
    hash: "#hash",
    state: { fromDashboard: true }
  }}
/>
```

REACT ROUTER API

<NavLink>

```
<NavLink to="/contact" activeClassName="selected">
  Contact
</NavLink>

<NavLink to=„/contact" activeStyle={{color: "red"}}>
  Contact
</NavLink>
```

REACT ROUTER API

<Redirect>

```
<Redirect to=„/contact"/>

<Redirect
  to={{
    pathname: „/contact"
  }}
/>
```

REACT ROUTER API

<Switch>

```
<Switch>
  <Route path="/" />
  <Route path="/" />
  <Route path="/" />
</Switch>
```

REACT ROUTER API

withRouter

```
class MyComponent extends Component {
  render() {
    const { match, location, history } = this.props;

    return <div>You are now at {location.pathname}</div>;
  }
}

withRouter(MyComponent);
```

DOCUMENTATION

reacttraining.com/react-router/

/catalog => <Catalog />
/contact => <Contact />

```
class App extends Component {
  render() {
    return (
      <div>
        <h2>Website 2019</h2>
        <Catalog />
        <Contact />
      </div>
    );
  }
}
```

npm i **react-router-dom**

```
import {
  BrowserRouter as Router,
  Route
} from "react-router-dom";

class App extends Component {
  render() {
    return (
      <div>
        <h2>Website 2019</h2>
        <Catalog />
        <Contact />
      </div>
    );
  }
}
```

```
import {
  BrowserRouter as Router,
  Route
} from "react-router-dom";

class App extends Component {
  render() {
    return (
      <Router>
        <div>
          <h2>Website 2019</h2>
          <Route path="/catalog" component={Catalog} />
          <Route path="/contact" component={Contact} />
        </div>
      </Router>
    );
  }
}
```

DEMO

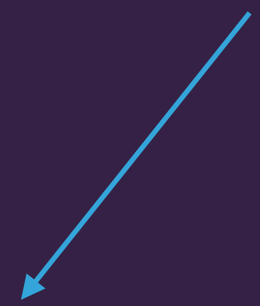
404

```
<Switch>
    <Route exact path=„/" component={Home} />
    <Route path="/catalog" component={Catalog} />
    <Route path="/contact" component={Contact} />
    <Route component={NotFound} />
</Switch>
```

NESTED ROUTERS

url: http://localhost:3000/catalog

match.url: /catalog



```
const Catalog = ({ match }) => (
  <div>
    <h1>I am Catalog Component {match.url}</h1>
    <Route path={`${match.url}/a`} component={CatalogA} />
    <Route path={`${match.url}/b`} component={CatalogB} />
  </div>
);
```

path=/catalog/a

path=/catalog/b

```
<Router>
  <Switch>
    <Route path="/catalog" component={Catalog} />
    <Route path="/contact" component={Contact} />
    <Route component={NotFound} />
  </Switch>
</Router>
```

PARAMETERS ROUTERS

```
const Catalog = ({ match }) => (
  <div>
    <h1>I am Catalog Component, id: {match.params.id}</h1>
  </div>
);
```

```
<Route path="/catalog/:id" component={Catalog} />
```

STYLE REACT COMPONENT

96* Ways to Style React Components in 2019

INLINE CSS

INLINE CSS

```
const helloStyle = {
  color: "red",
  fontSize: "20px"
};

const Hello = () => <h1 style={helloStyle}>Hello</h1>;
```

CLASS NAME

CLASS NAME

Style.css

```css
.red {
  color: red;
  font-size: 20px
}
```

App.js

```jsx
import "./Style.css";

const Hello = () => <h1 className="red">Hello</h1>;
```

CSS MODULES

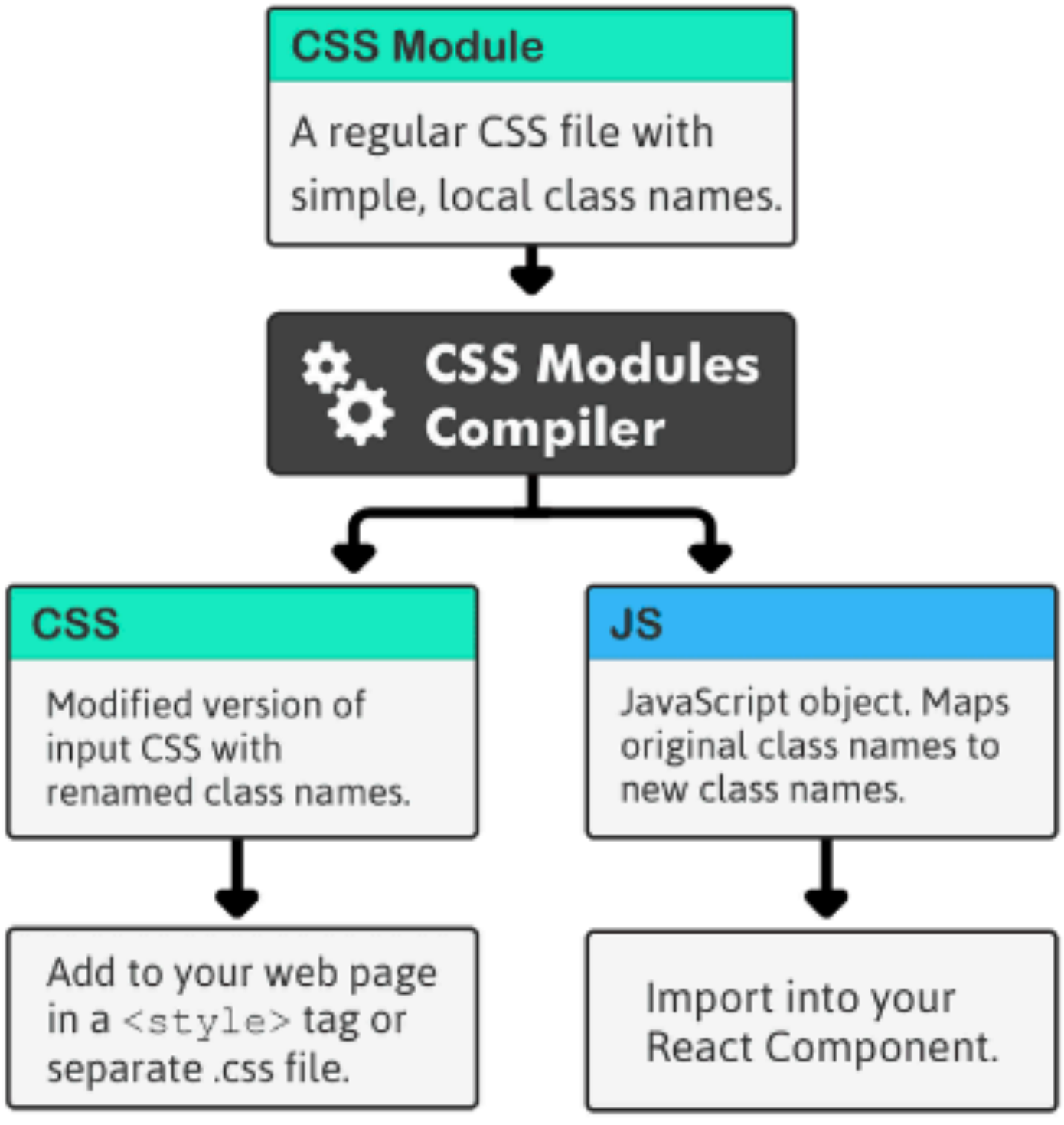
CSS MODULES

Style.module.css

```
.Red {
  color: red;
  font-size: 20px
}
```

App.js

```
import styles from "./Style.module.css";

const Hello = () => <h1 className={styles.Red}>Hello</h1>;
```


CSS Module
A regular CSS file with simple, local class names.
CSS Modules Compiler
CSS
Modified version of input CSS with renamed class names.
Add to your web page in a <style> tag or separate .css file.
JS
JavaScript object. Maps original class names to new class names.
Import into your React Component.

CSS MODULES

Style.module.css

```
.Red {
  color: red;
  font-size: 20px
}
```

```
<style type="text/css">.Style_Red___55GV {
  color: red;
  font-size: 20px
}</style>
```

App.js

```
import styles from "./Style.module.css";
```

console.log(styles);

```
{
  Red: "Style_Red___55GV"
};
```

```
const Hello = () => <h1 className={styles.Red}>Hello</h1>;
```

CSS IN JS

STYLED COMPONENTS

```
import styled from 'styled-components';
```

```
const Hello = styled.h1`
  color: red;
  font-size: 20px
`;
```

STYLED COMPONENTS

```
<style data-styled="" data-styled-version="4.1.3">
  /* sc-component-id: sc-bdVaJa */
  .cSRbkb {
    color:red;
    font-size:20px;
  }
</style>

<h1 class="sc-bdVaJa cSRbkb”>Hello world</h1>


const Hello = styled.h1`
  color: red;
  font-size: 20px
`;
class App extends Component {
  render() {
      return <Hello>Hello world</Hello>;
  }
}
```

STYLED COMPONENTS

```
const Hello = styled.h1`
  color: ${props => props.color};
  font-size: 20px;
`;
```

```
<Hello color="red">Hello world</Hello>;
```

Other libaries...

https://github.com/tuchk4/awesome-css-in-js

REFS

Refs make it possible to access DOM nodes directly within React.

REFS

When use?

Managing focus, text selection, or media playback.

Triggering imperative animations.

Integrating with third-party DOM libraries.

REFS

```
class App extends Component {
  constructor() {
    super();
    this.inputRef = React.createRef();
  }
  render() {
    return (
      <div>
        <input type="text" ref={this.inputRef} />
      </div>
    );
  }
}
```

REFS

```
class App extends Component {
  constructor() {
    super();
    this.inputRef = React.createRef();
  }

  componentDidMount() {
    this.inputRef.current.focus();
  }

  render() {
    return (
      <div>
        <input type="text" ref={this.inputRef} />
      </div>
    );
  }
}
```

SUMMARY

AJAX & APIS

```jsx
class App extends React.Component {
  state = {
    user: null
  };

  render() {
    const { user } = this.state;

    return (
      <div>
        <img src={user.avatar} alt="avatar" />
        Name: {user.name}
        E-mail: {user.email}
      </div>
    );
  }
}
```

```jsx
class App extends React.Component {
  state = {
    user: null
  };

  async componentDidMount() {
    const response = await fetch("https://my.awesome.api/v1/");
    const data = await response.json();
    const [user] = data.results;
    this.setState({ user });
  }

  render() {
    const { user } = this.state;

    return (
      <div>
        <img src={user.avatar} alt="avatar" />
        Name: {user.name}
        E-mail: {user.email}
      </div>
    );
  }
}
```

← → 1 of 2 errors on the page

TypeError: Cannot read property 'avatar' of null

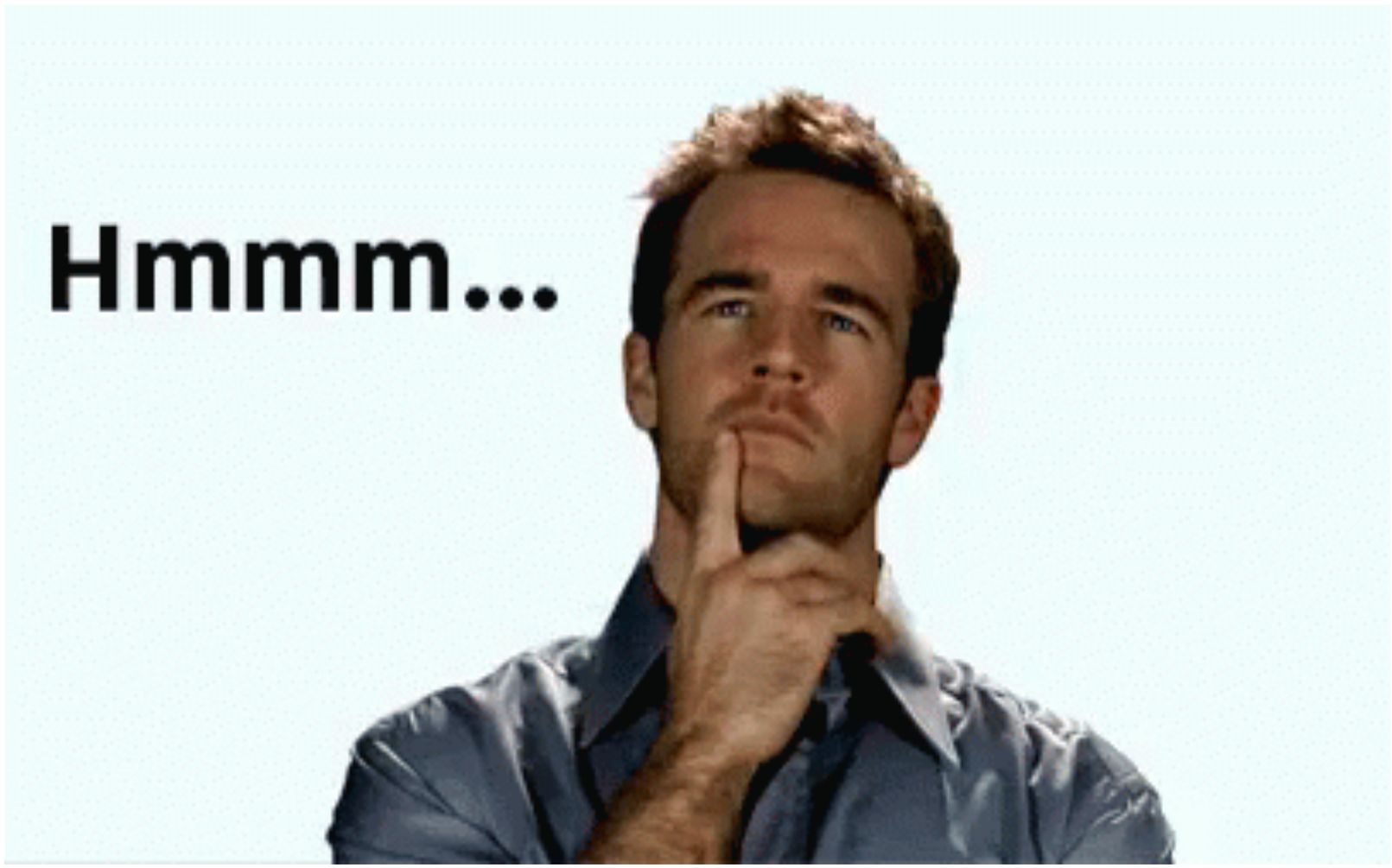
App.render
src/App.js:24

```
  21 |
  22 | return (
  23 |   <div>
> 24 |     <img src={user.avatar} alt="avatar" />
  25 | ^   Name: {user.name}
  26 |     E-mail: {user.email}
  27 |   </div>
```

View compiled

```
class App extends React.Component {
  state = {
    isLoaded: false,
    user: null
  };

  async componentDidMount() {
    ///…pobieranie danych
    this.setState({
      user,
      isLoaded: true
    });
  }

  render() {
    const { user, isLoaded } = this.state;

    if(!isLoaded) {
      return <div>Loading...</div>;
    }

    ///…dalsza czesc render
  }
}
```

ERRORS?



```
class App extends React.Component {
  state = {
    error: null,
    isLoaded: false,
    user: null
  };

  async componentDidMount() {
    try {
      ///…pobieranie danych
      this.setState({
        user,
        isLoaded: true
      });
    } catch (e) {
      this.setState({
        error: e.message,
        isLoaded: true
      });
    }
  }

  render() {
    const { user, isLoaded, error } = this.state;

    if (error) { return <div>{error}</div>; }

    if (!isLoaded) { return <div>Loading…</div>; }

    ///…dalsza czesc render
```

UNCONTROLLED COMPONENTS

UNCONTROLLED COMPONENTS

```
class Form extends Component {
  render() {
    return (
      <div>
        <input type="text" />
      </div>
    );
  }
}
```

UNCONTROLLED COMPONENTS

Form data handled by

DOM

UNCONTROLLED COMPONENTS

```
this.inputRef = React.createRef();
```

```
this.inputRef.current.value;
```

```
<input type="text" ref={this.inputRef}/>
```

INPUT
[TYPE=FILE]

Always uncontrolled

INPUT [TYPE=FILE]

```
this.fileInputRef = React.createRef();
```

```
this.fileInputRef.current.files;
```

```
<input type="file" ref={this.fileInputRef} />
```

CONTROLLED COMPONENTS

CONTROLLED COMPONENTS

Form data handled by

REACT Component

CONTROLLED COMPONENTS

```
<input type="text" value={value} onChange={handleChange} />
```

CONTROLLED COMPONENTS

```
class Form extends React.Component {
  state = {
    name: ""
  };

  handleChange = event => {
    this.setState({ name: event.target.value });
  };

  render() {
    return (
      <input type="text" value={this.state.name} onChange={this.handleChange} />
    );
  }
}
```

CONTROLLED VS UNCONTROLLED

feature	uncontrolled	controlled
one-time value retrieval (e.g. on submit)		
validating on submit		
instant field validation		
conditionally disabling submit button		
enforcing input format		
several inputs for one piece of data		
dynamic inputs		

FORMS

FORMS ARE HARD

FORM STATE

Which field currently has focus?

Which fields are dirty?

Which fields have errors?

Which fields was touched?

Are we currently submitting?

Have we tried to submit and received some errors from server

FINAL FORMS

FINAL FORMS

Zero dependencies

Only peer dependencies: React and Final Form

Opt-in subscriptions - only update on the state you need!

3.0k + 4.8k gzipped

FINAL FORMS

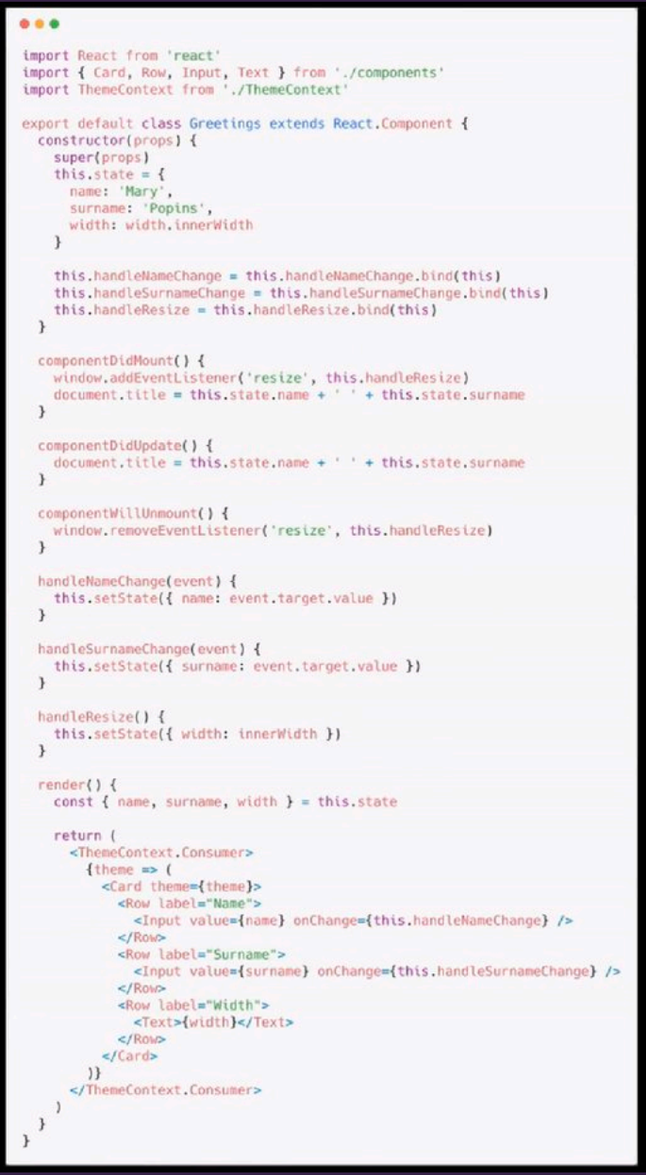
```
class App extends React.Component {
  onSubmit = values => {
    console.log("handleSubmit", values);
  };

  render() {
    return (
      <Form
        onSubmit={this.onSubmit}
        render={({ handleSubmit }) => (
          <form onSubmit={handleSubmit}>
            <Field name="imie" component="input" />
            <Field name="nazwisko" component="input" />
            <input type="submit" value="Wyslij" />
          </form>
        )}
      />
    );
  }
}
```

Hooks

HOOKS

Hooks is a special function that lets you “hook into” React features like React state and lifecycle.

HOOKS



```
import React from 'react'
import { Card, Row, Input, Text } from './components'
import ThemeContext from './ThemeContext'

export default class Greetings extends React.Component {
  constructor(props) {
    super(props)
    this.state = {
      name: 'Mary',
      surname: 'Popins',
      width: width.innerWidth
    }

    this.handleNameChange = this.handleNameChange.bind(this)
    this.handleSurnameChange = this.handleSurnameChange.bind(this)
    this.handleResize = this.handleResize.bind(this)
  }

  componentDidMount() {
    window.addEventListener('resize', this.handleResize)
    document.title = this.state.name + ' ' + this.state.surname
  }

  componentDidUpdate() {
    document.title = this.state.name + ' ' + this.state.surname
  }

  componentWillUnmount() {
    window.removeEventListener('resize', this.handleResize)
  }

  handleNameChange(event) {
    this.setState({ name: event.target.value })
  }

  handleSurnameChange(event) {
    this.setState({ surname: event.target.value })
  }

  handleResize() {
    this.setState({ width: innerWidth })
  }

  render() {
    const { name, surname, width } = this.state

    return (
      <ThemeContext.Consumer>
        {theme => (
          <Card theme={theme}>
            <Row label="Name">
              <Input value={name} onChange={this.handleNameChange} />
            </Row>
            <Row label="Surname">
              <Input value={surname} onChange={this.handleSurnameChange} />
            </Row>
            <Row label="Width">
              <Text>{width}</Text>
            </Row>
          </Card>
        )}
      </ThemeContext.Consumer>
    )
  }
}
```

```
import React from 'react'
import { Card, Row, Input, Text } from './components'
import ThemeContext from './ThemeContext'

export default class Greetings extends React.Component {
  constructor(props) {
    super(props)
    this.state = {
    }

  }

  componentDidMount() {
  }

  componentDidUpdate() {
  }

  componentWillUnmount() {
  }

  render() {
    const { name, surname, width } = this.state

    return (
          <Card theme={theme}>
            <Row label="Name">
              <Input value={name} onChange={this.handleNameChange} />
            </Row>
            <Row label="Surname">
              <Input value={surname} onChange={this.handleSurnameChange} />
            </Row>
            <Row label="Width">
              <Text>{width}</Text>
            </Row>
          </Card>
    )
  }
}
```

```
import React, { useState, useContext, useEffect } from 'react'
import { Card, Row, Input, Text } from './components'
import ThemeContext from './ThemeContext'

export default function Greetings(props) {

  return (
    <Card theme={theme}>
      <Row label="Name">
        <Input {...name} />
      </Row>
      <Row label="Surname">
        <Input {...surname} />
      </Row>
      <Row label="Width">
        <Text>{width}</Text>
      </Row>
    </Card>
  )
}
```

```
import React, { useState, useContext, useEffect } from 'react'
import { Card, Row, Input, Text } from './components'
import ThemeContext from './ThemeContext'

function useFormInput(initialValue) {
  const [value, setValue] = useState(initialValue)

  function handleChange(e) {
    setValue(e.target.value)
  }

  return {
    value,
    onChange: handleChange,
  }
}

function useDocumentTitle(title) {
  useEffect(() => {
    document.title = title
  })
}

function useWindowWidth() {
  const [width, setWidth] = useState(window.innerWidth)
  useEffect(() => {
    const handleResize = () => setWidth(window.innerWidth)
    window.addEventListener('resize', handleResize)
    return () => {
      window.removeEventListener('resize', handleResize)
    }
  })

  return width
}

export default function Greetings(props) {
  const name = useFormInput('Mary')
  const surname = useFormInput('Poppins')
  const theme = useContext(ThemeContext)
  const width = useWindowWidth()

  useDocumentTitle(name.value + ' ' + surname.value)

  return (
    <Card theme={theme}>
      <Row label="Name">
        <Input {...name} />
      </Row>
      <Row label="Surname">
        <Input {...surname} />
      </Row>
      <Row label="Width">
        <Text>{width}</Text>
      </Row>
    </Card>
  )
}
```

HOOKS

Functional Stateless Components

now is

Functional Components

HOOKS

Don’t waist your time to rewriting your existing components overnight but you can start using Hooks in the new ones if you’d like

HOOKS

Don’t call Hooks inside loops, conditions, or nested functions

Only call Hooks at the top level.

Don’t call Hooks from regular JavaScript functions

Only call Hooks from React function components.

HOOKS
useState

useState is a Hook that lets you add React state to function components

HOOKS
useState

```
class App extends React.Component {
    state = {
        count: 0
    }
```

```
const App = () => {
    const [count, setCount] = useState(0);
```

HOOKS
useState

```
this.setState({
  count: 0
});
```

```
setCount(0);
```

HOOKS
useState

```
const App = () => {
  const [count, setCount] = useState(0);

  return (
      <div>
          Count {count}
          <button onClick={() => setCount(count + 1)}>Click</button>
      </div>
  )
};
```

HOOKS
useState

```
const App = () => {
  //multiple state variables!
  const [count, setCount] = useState(0);
  const [name, setName] = useState("");
  const [email, setEmail] = useState("");
```

HOOKS

useState

DEMO

HOOKS
useRef

```jsx
const App = () => {
  const inputRef = useRef(null);
  const onButtonClick = () => {
    inputRef.current.focus();
  };
  return (
    <>
      <input ref={inputRef} type="text" />
      <button onClick={onButtonClick}>Focus the input</button>
    </>
  );
};
```

HOOKS
useContext

Accepts a context object and returns the current context value for that context.

```
const value = useContext(MyContext);
```

HOOKS

useContext

DEMO

HOOKS
useEffect

Accepts a function that contains imperative, possibly effectful code.

```
useEffect(didUpdate);
```

HOOKS
useEffect

```
// side-effects in a React class component
class MyComponent extends Component {
    // setup phase
    componentDidMount() {
        // add listener for feature
    }
    // clean up phase
    componentWillUnmount() {
        // remove listener for feature
    }
}


// side-effects in React function component with React Hooks
MyComponent = () => {
    useEffect(() => {
        // add listener for feature (setup)
        // return function to remove listener for feature (clean up)
    });
}
```

HOOKS
useEffect

```
// side-effects in React function component with React Hooks
MyComponent = () => {
    useEffect(() => {
    });
}
```

```
// side-effects in React function component with React Hooks
MyComponent = () => {
    useEffect(() => {
    }, []);
}
```

```
// side-effects in React function component with React Hooks
MyComponent = () => {
    useEffect(() => {
    }, [name]);
}
```

HOOKS
useEffect

DEMO

HOOKS
useYourImagination

Write custom hook

DEMO

THANKS