

REACT.JS



PRZEMYSŁAW WISZOWATY

HELLO!



SOFTware HUT

TENDERHUT GROUP

meet.js

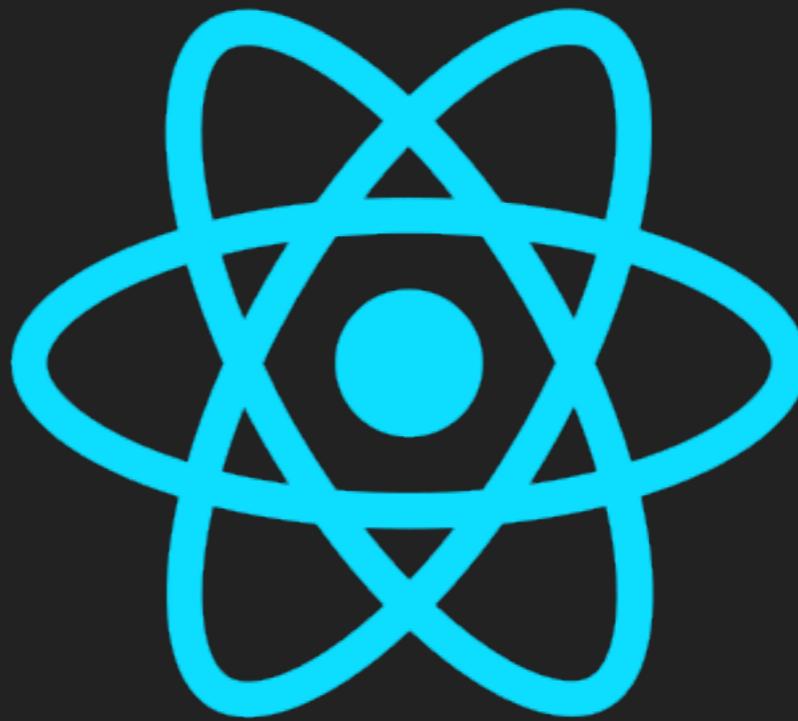
białystok



REACT IS EVERYWHERE

„Learn Once, Write Anywhere”

FRONTEND
BACKEND
MOBILE



A back to the 2000s

N.R. (OG)
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://

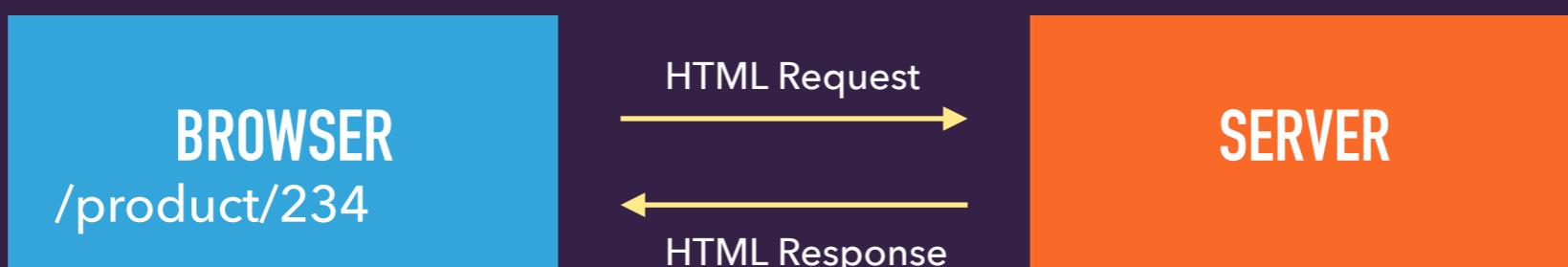
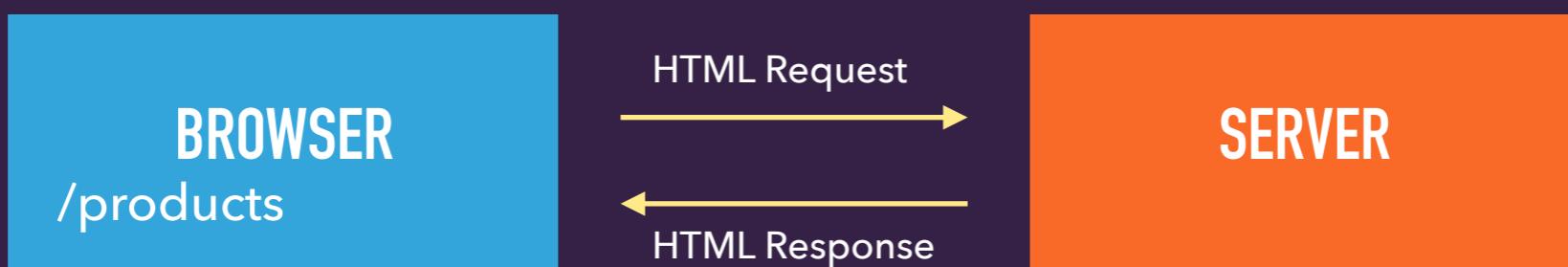
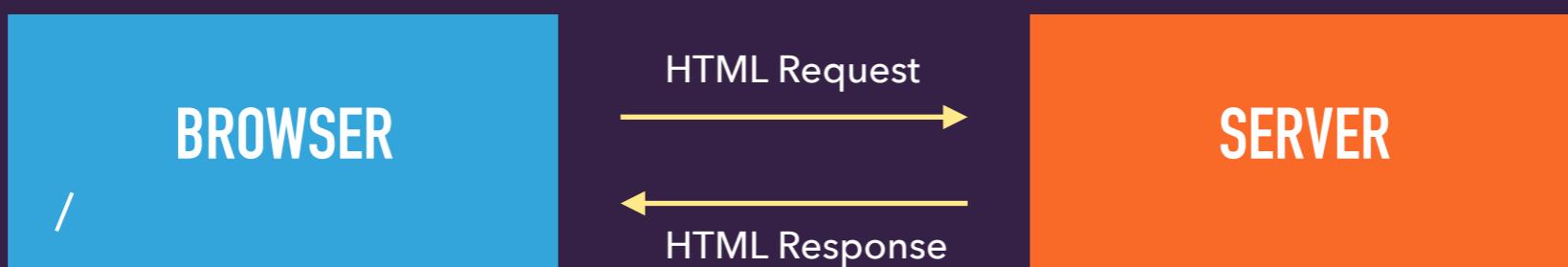
Some pages may work very slowly or not at all. Many webapps are just too advanced for Geocities.

Turn your sound up for the full effect.

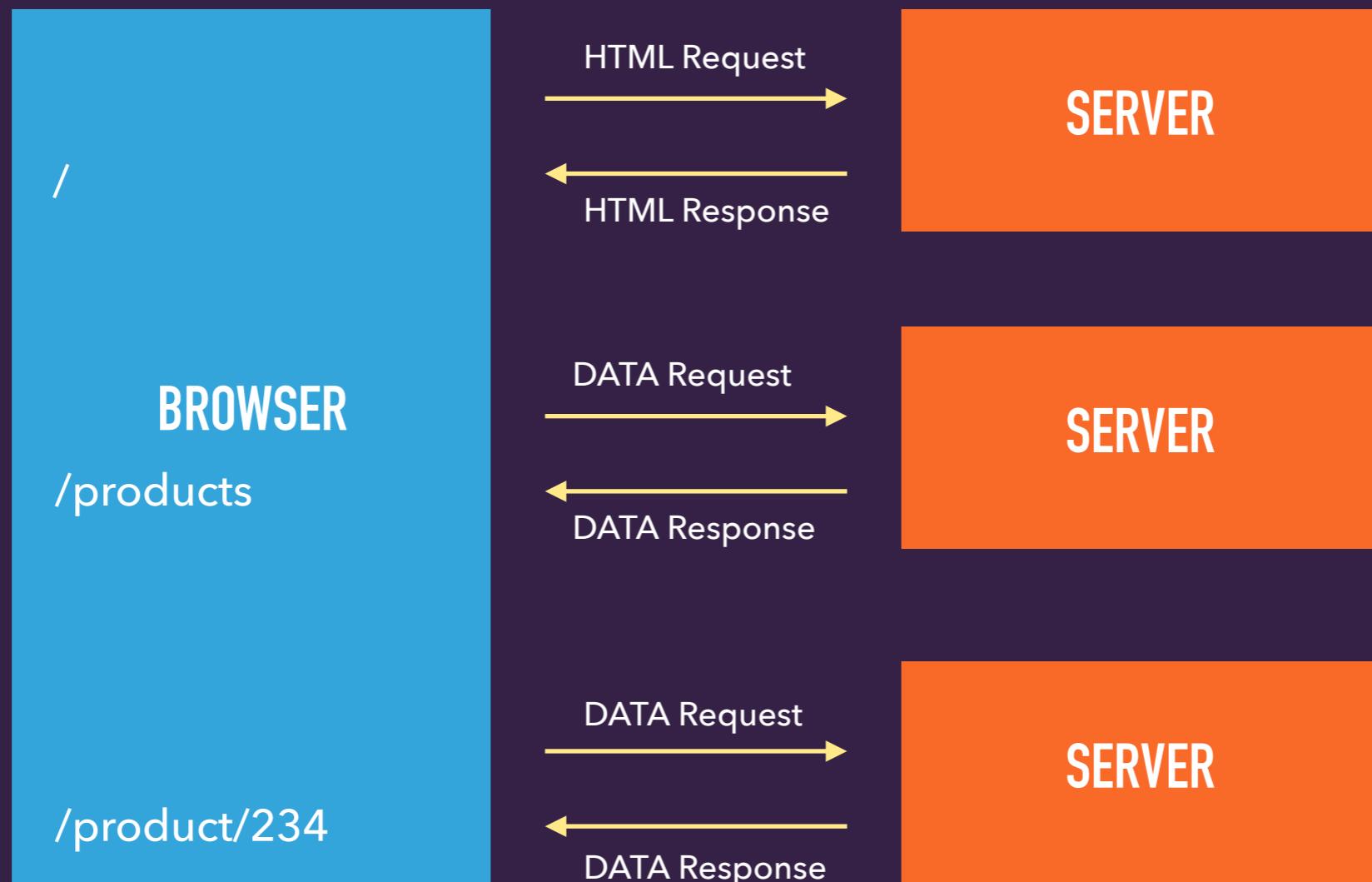
Created by [Mike Lacher](#)



TRADITIONAL WEBSITE



SINGLE PAGE APPLICATION





414 x 736



Elements

Console

Sources

Network

Performance

Memory

Application

Security

Audits

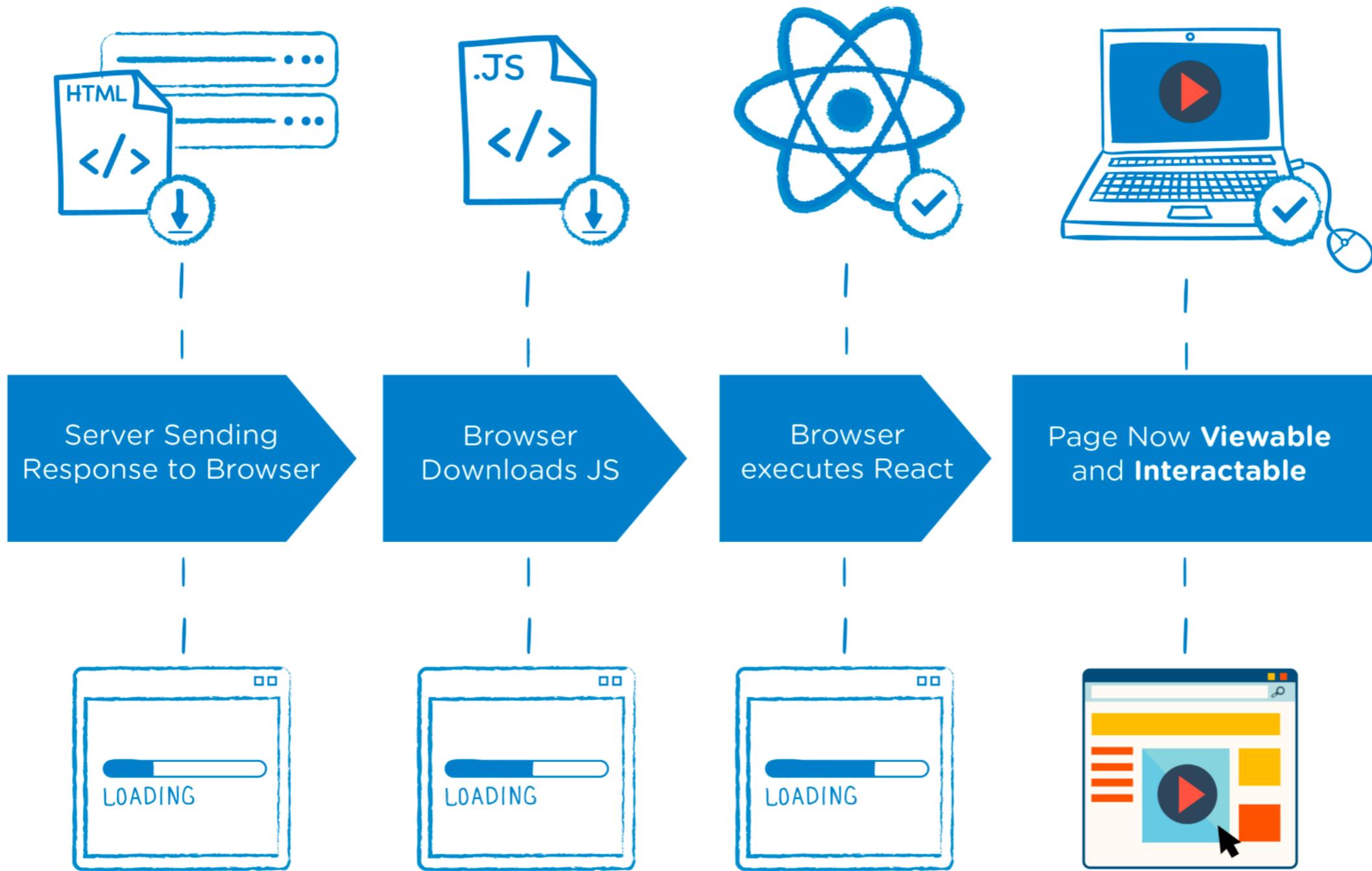


(index) :formatted x

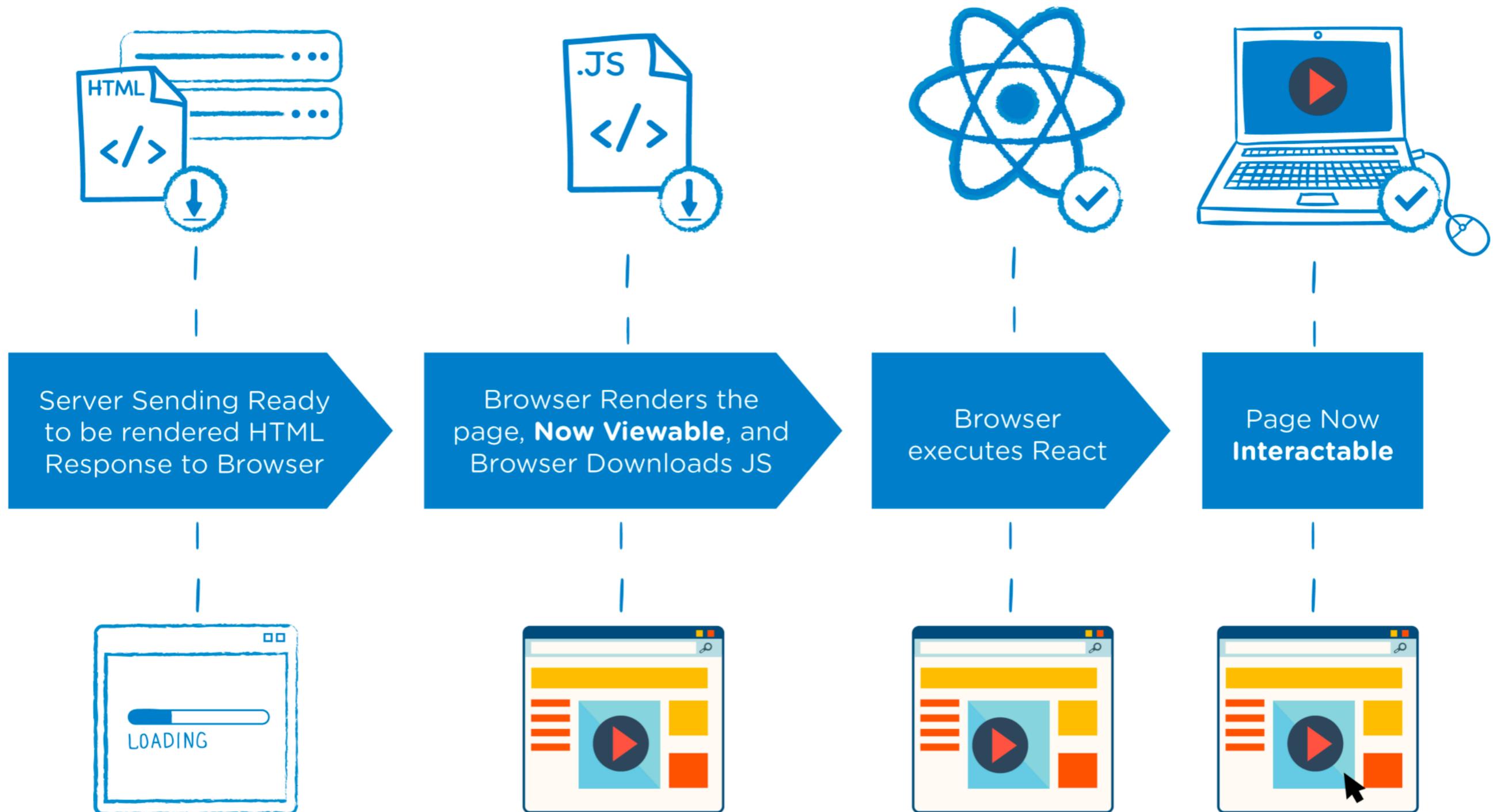
```
1 <!DOCTYPE html>
2 <html lang="en" prefix="og: http://ogp.me/ns#>
3   <head>
4     <meta charset="utf-8">
5     <meta name="viewport" content="width=device-width,initial-scale=1,shrink-to-fit=no">
6     <link rel="apple-touch-icon" sizes="180x180" href="/icons/apple-touch-icon.png?v=694AzX47gQ">
7     <link rel="icon" type="image/png" sizes="32x32" href="/icons/favicon-32x32.png?v=694AzX47gQ">
8     <link rel="icon" type="image/png" sizes="16x16" href="/icons/favicon-16x16.png?v=694AzX47gQ">
9     <link rel="mask-icon" href="/icons/safari-pinned-tab.svg?v=694AzX47gQ" color="#5bbad5">
10    <link rel="shortcut icon" href="/icons/favicon.ico?v=694AzX47gQ">
11    <meta name="msapplication-TileColor" content="#2d89ef">
12    <meta name="msapplication-config" content="/icons/browserconfig.xml?v=694AzX47gQ">
13    <meta name="theme-color" content="#ffffff">
14    <meta name="google-site-verification" content="NCJvpBpTbp9pUJdDk7t6dhIis0j4kZJEphgntRU-TbM"/>
15    <link rel="manifest" href="/manifest.json">
16    <title>TenderHut</title>
17    <link href="/static/css/main.5ca042e5.css" rel="stylesheet">
18  </head>
19  <body>
20    <noscript>You need to enable JavaScript to run this app.</noscript>
21    <div id="app-root"></div>
22    <div id="modal-root"></div>
23    <script type="text/javascript" src="/static/js/main.dc65024f.js"></script>
24  </body>
25</html>
```



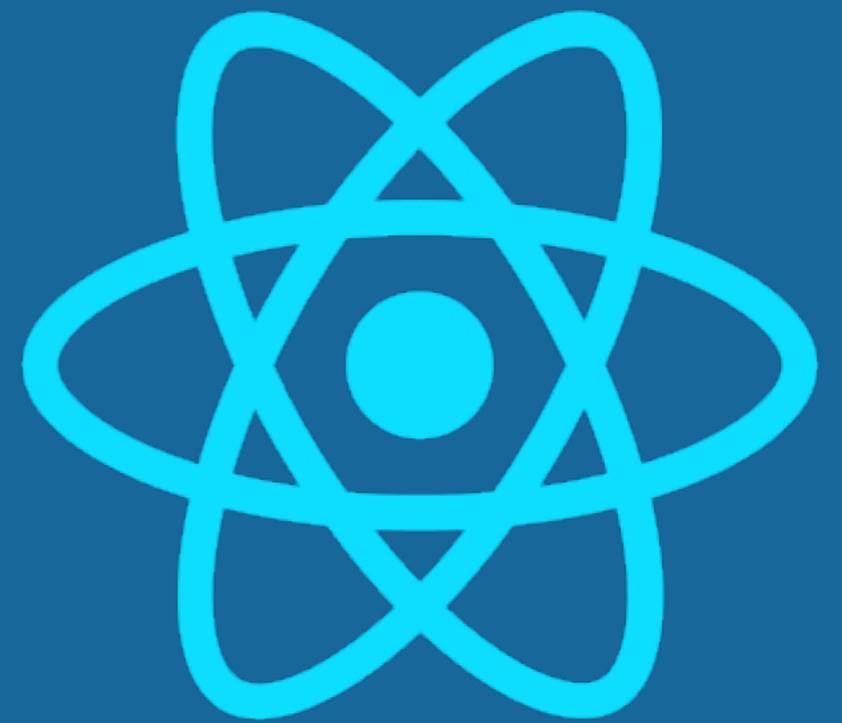
CSR



SSR



WHAT IS REACT?

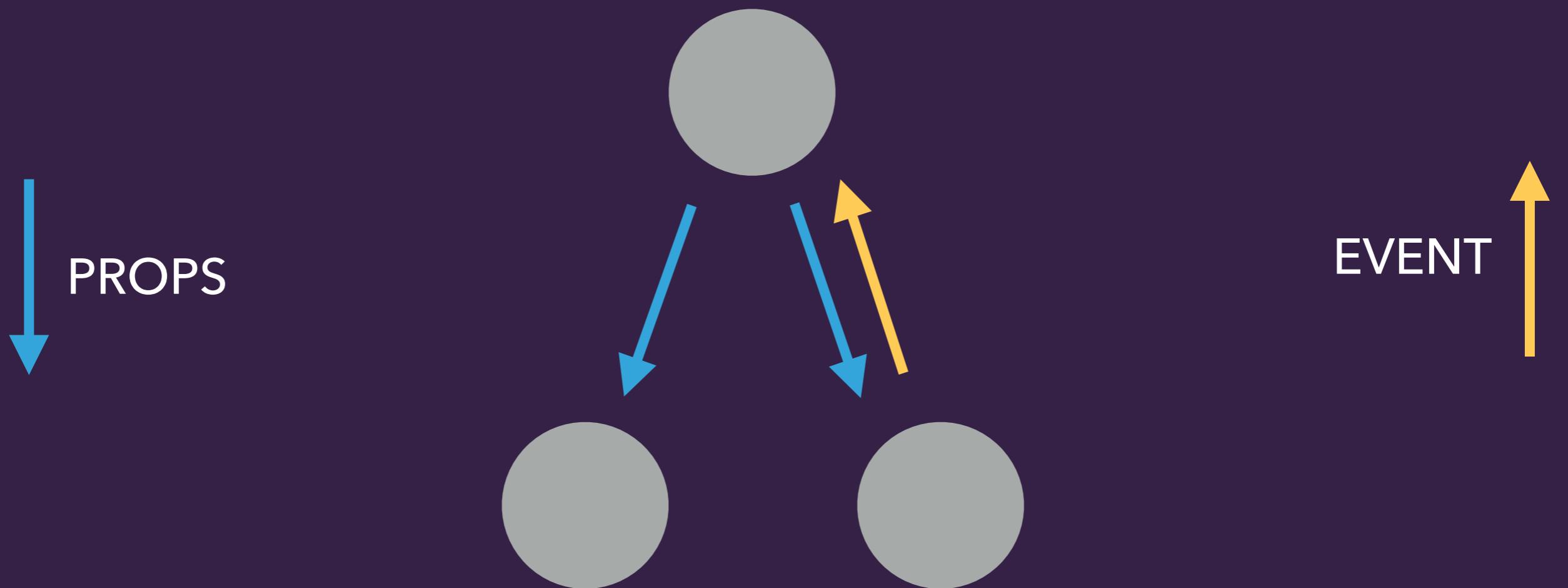


**LIBRARY
NOT
~~FRAMEWORK~~**

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

CLASS

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

FUNCTIONAL

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

FUNCTIONAL

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

FUNCTIONAL

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

FUNCTIONAL

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

<Welcome />

**EVERYTHING
IS A COMPONENT**

CardComponent

CardHeaderComponent

COMPONENT...

COMPONENT EVERYWHERE

makeameme.org

CardBodyComponent

UserPhotoComponent

```
1 import React from 'react';
2
3 const FacebookComponent = () => (
4   <CardComponent>
5     <CardHeaderComponent>
6       <SelectorComponent />
7       <SelectorComponent />
8       <SelectorComponent />
9     </CardHeaderComponent>
10
11    <CardBodyComponent>
12      <UserPhotoComponent />
13    </CardBodyComponent>
14
15    <CardFooterComponent />
16  </CardComponent>
17);
18
19 export default FacebookComponent;
20
```

JSX

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

JSX

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

JS

**REACT DOESN'T
REQUIRE USING JSX**

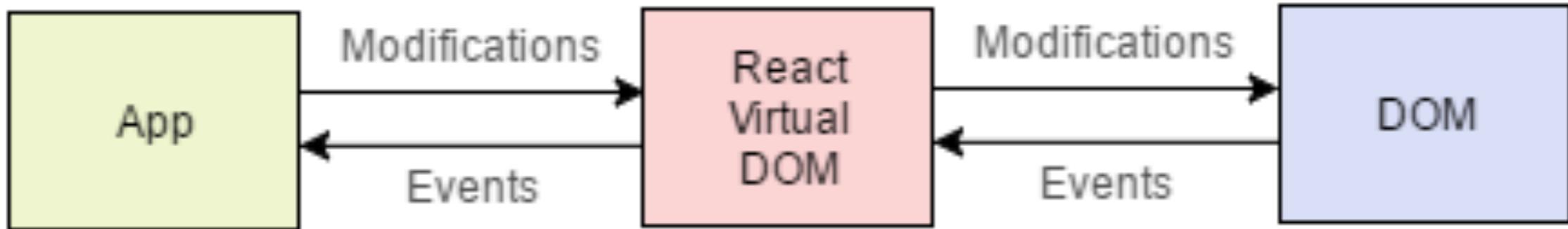
**VIRTUAL
DOM**

IT'S FAST
IT'S PURE
IT WORKS

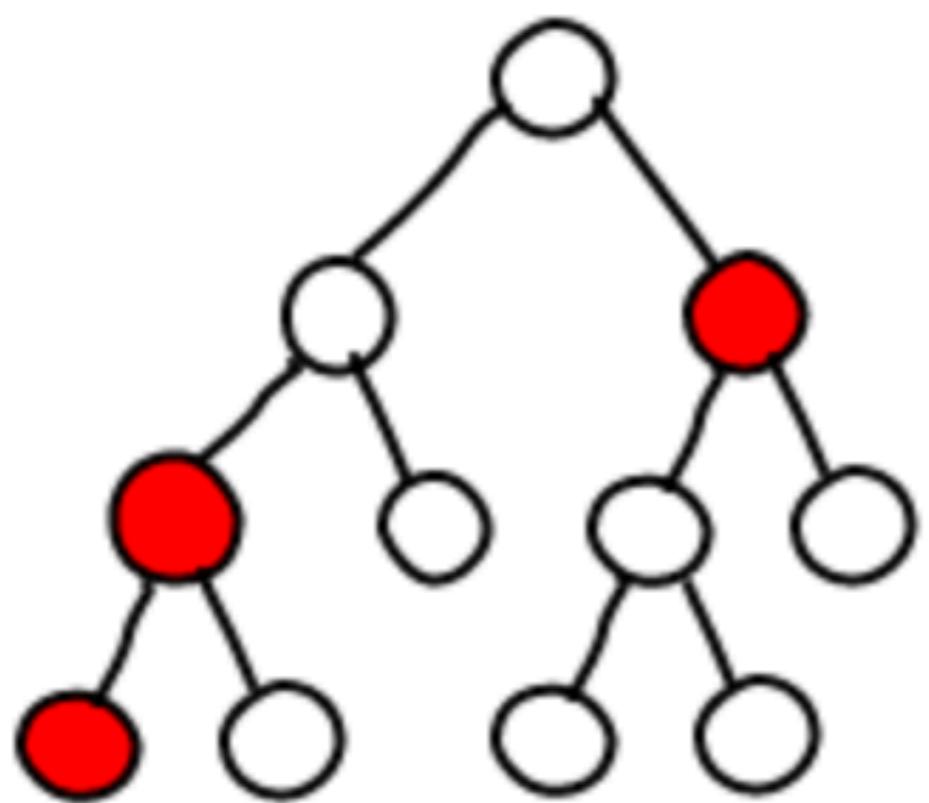
Traditional Web Application



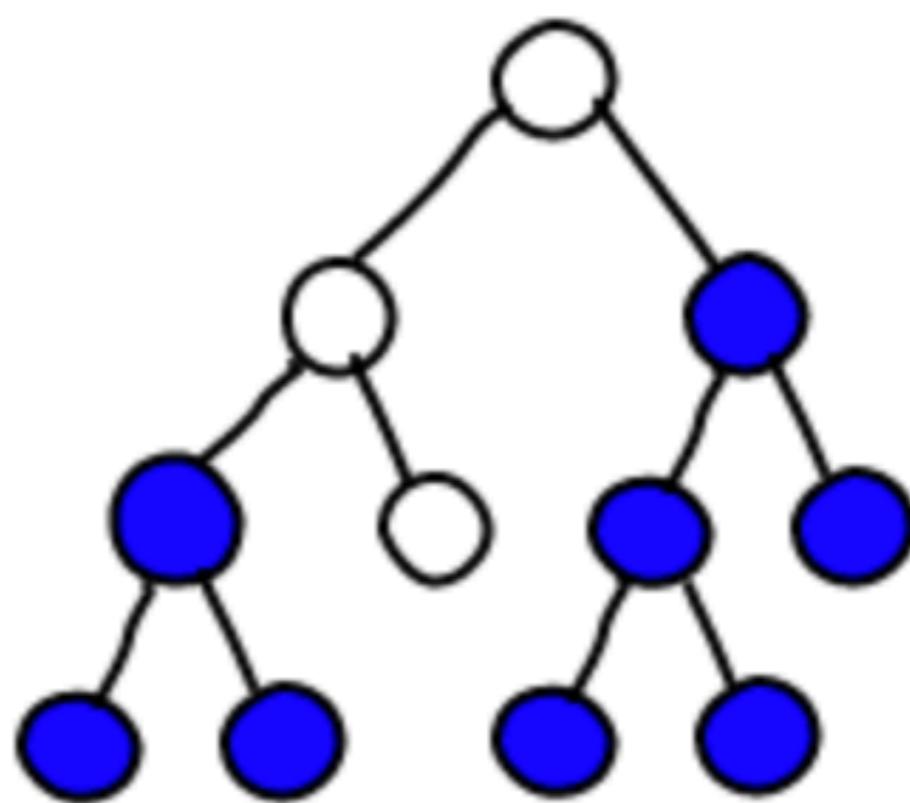
React.js



Dirty



Re-rendered



Hello, world!

It is 12:26:46 PM.

```
Console  Sources  Network  Timeline
▼<div id="root">
  ▼<div data-reactroot>
    <h1>Hello, world!</h1>
    ▼<h2>
      <!-- react-text: 4 -->
      "It is "
      <!-- /react-text -->
      <!-- react-text: 5 -->
      "12:26:46 PM"
      <!-- /react-text -->
      <!-- react-text: 6 -->
      "."
      <!-- /react-text -->
    </h2>
  </div>
</div>
```

PROPS

**PROPS =
PRIMITIVE VALUES,
REACT ELEMENTS
FUNCTIONS**

```
<Welcome />
```

```
props: {}
```

```
<Welcome name="Jack" />
```

```
props: {  
    name: "Jack"  
}
```

```
<Welcome>Jack</Welcome>
```

```
props: {  
    children: "Jack"  
}
```

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



PROPS DEFAULT IS TRUE

```
<MyComponent dark />
```

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

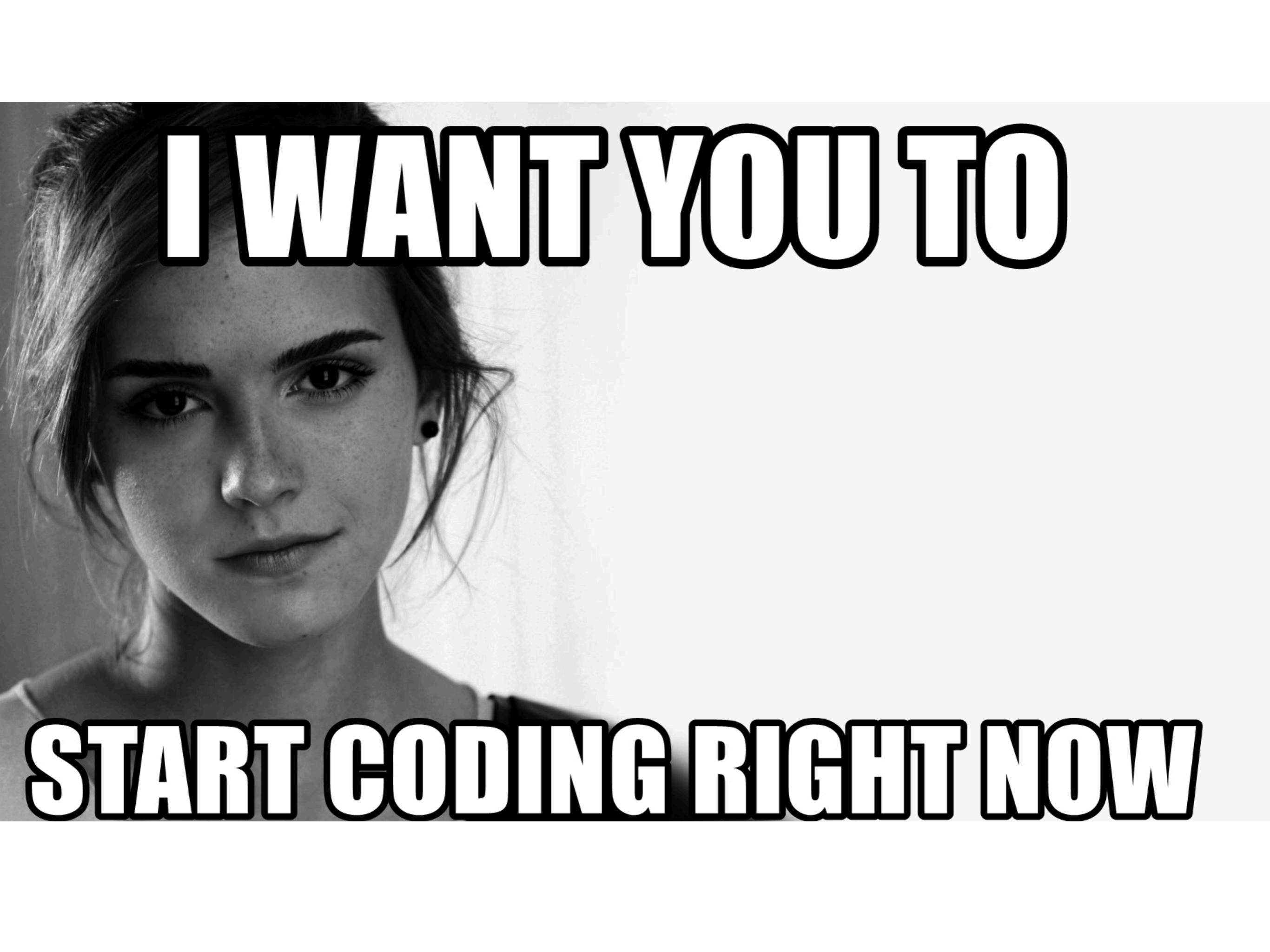
PROPS SPREAD ATTRIBUTES

```
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."  
/>
```

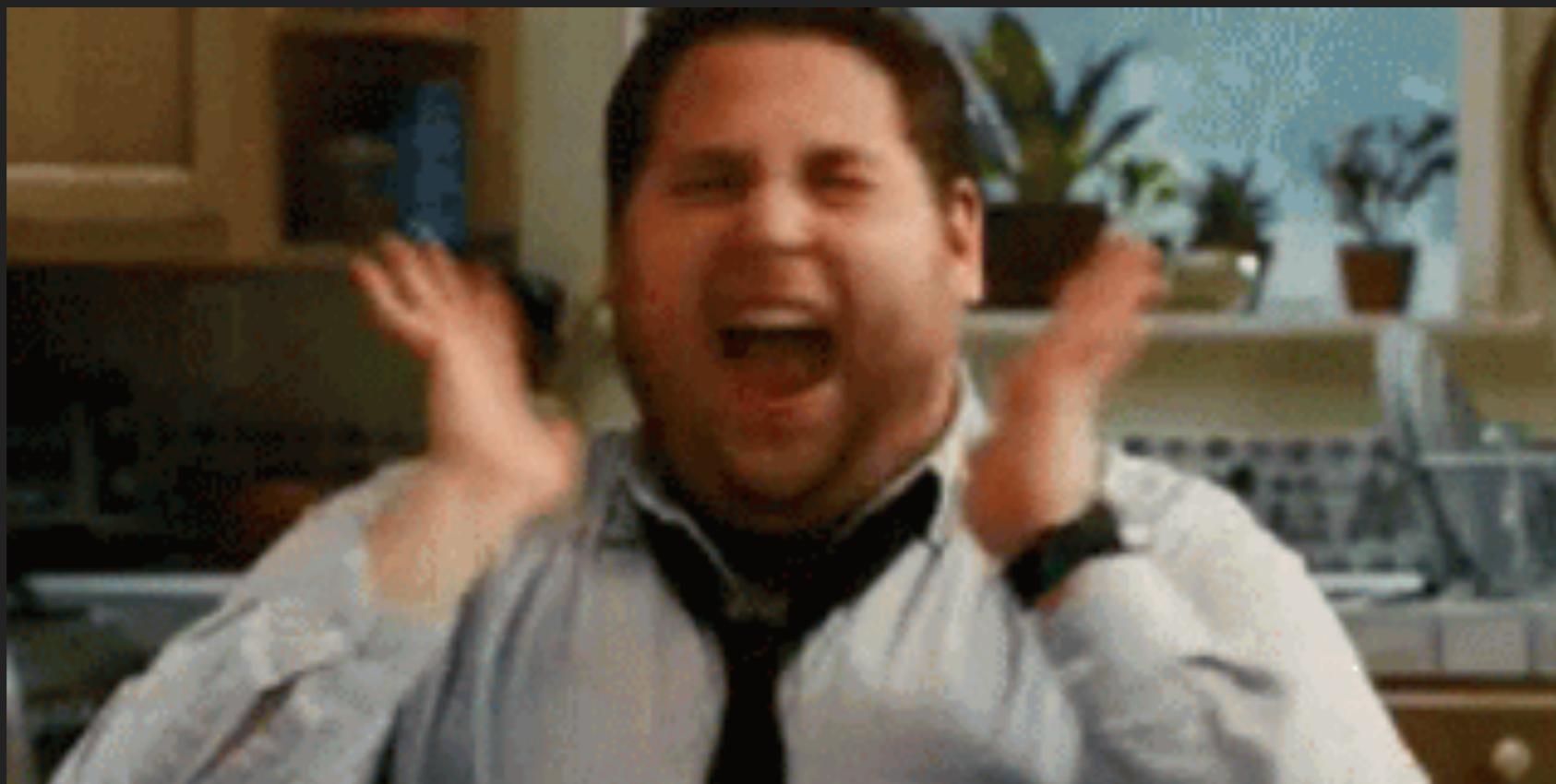


I WANT YOU TO

START CODING RIGHT NOW

**CREATE
REACTAPP**

npx create-react-app myapp

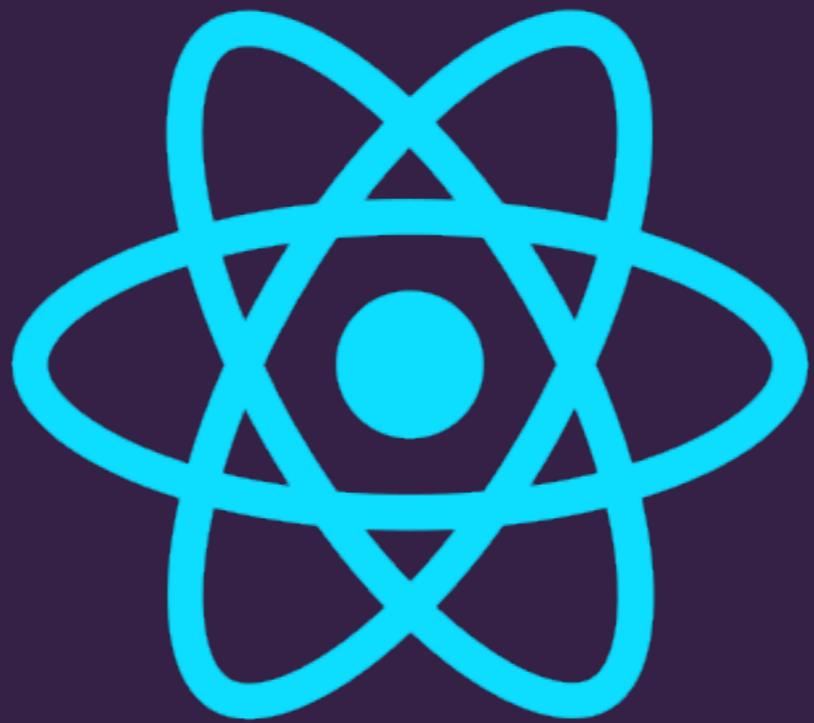


DEMO

REACT DEV TOOLS

DEMO

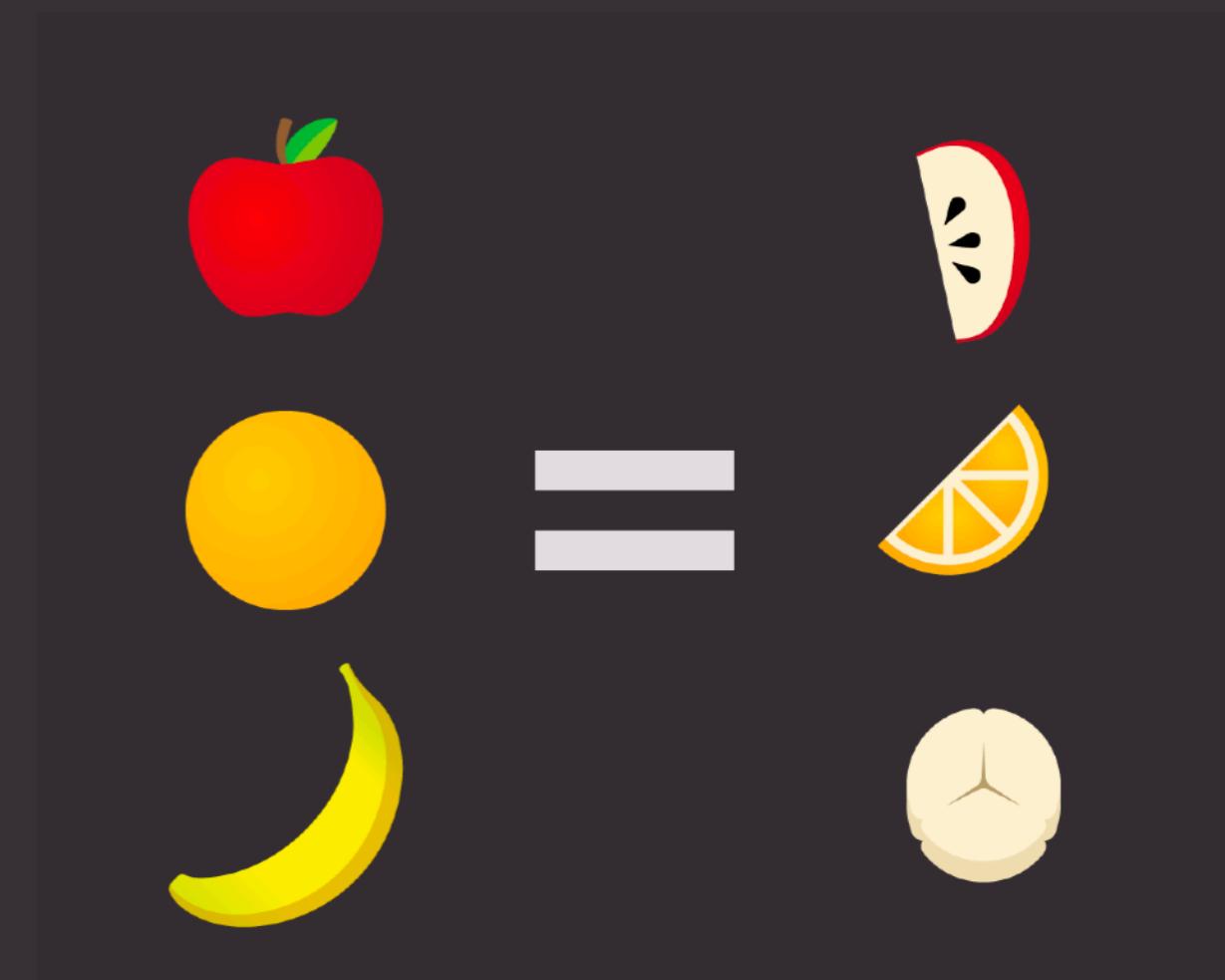
THANKS



REACT.JS

FUNCTIONAL PROGRAMMING

MAP



MAP

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

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

```
> newArray[2,,,]
```

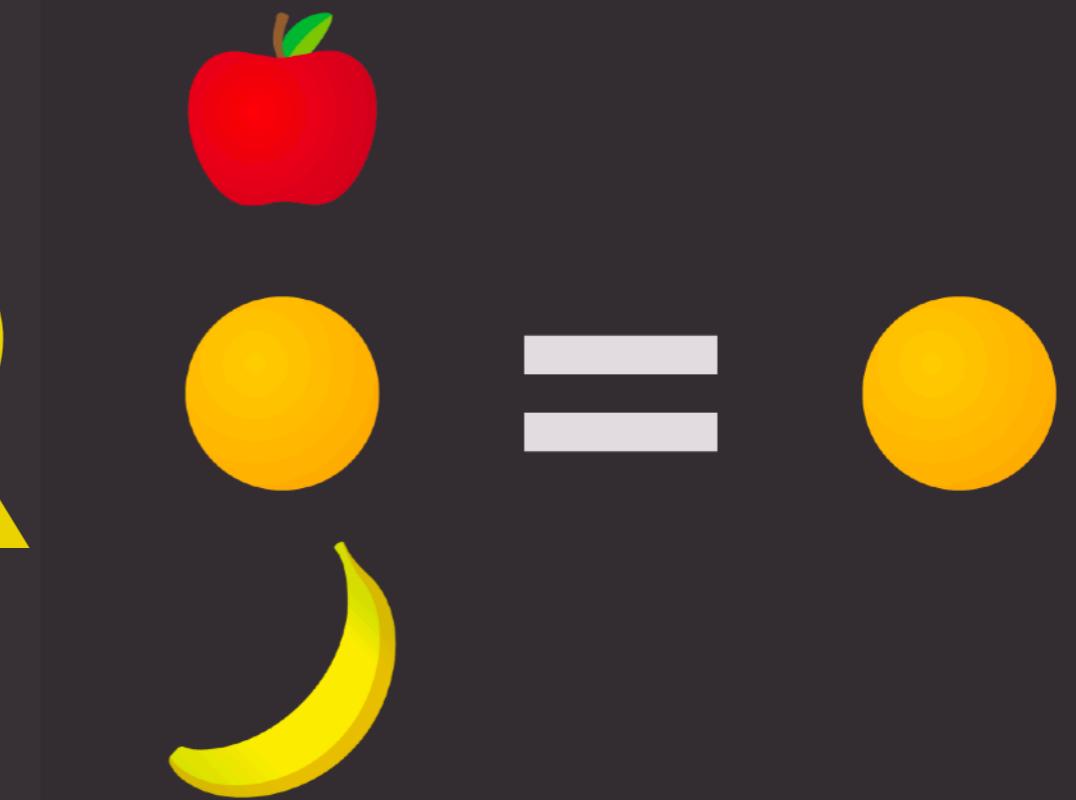
```
> newArray[2,3,,,]
```

```
> 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.filter(element => element > 3);
```

```
> newArray[]
```

```
> newArray[]
```

```
> newArray[]
```

```
> newArray[4]
```

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



```
[ 🐂 , 🥔 , 🐓 , 🌽 ].map(cook) => [ 🍔 , 🍟 , 🍗 , 🍹 ]  
[ 🍔 , 🍟 , 🍗 , 🍹 ].filter(isVegetarian) => [ 🍟 , 🍹 ]  
[ 🍔 , 🍟 , 🍗 , 🍹 ].reduce(eat) => 💩
```

RENDER MULTIPLE COMPONENT

RENDER MULTIPLE COMPONENT

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

```
const App = () => (
  <div>
    {[{"name": "Asia"}, {"name": "Paweł"}, {"name": "Gosia"}, {"name": "Darek"}].map(item =>
      <h4>{item}</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**

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 null;
  }
  return <Post {...post} />;
};
```

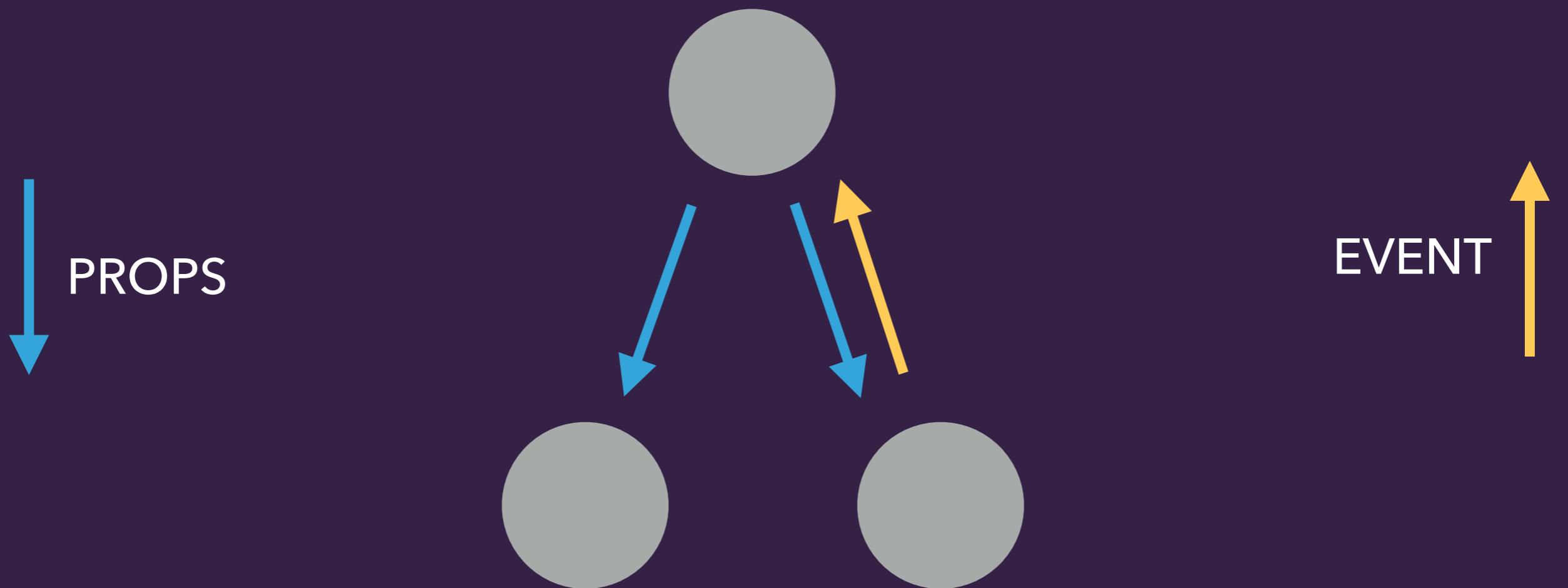
```
const App = () => {
  return post ? <Post {...post} /> : null;
};
```

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 = () => (
  <button onClick={() => alert("Click Button")}>
    Click Me!
  </button>
);
```

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

```
<Hello name={[]} />
```

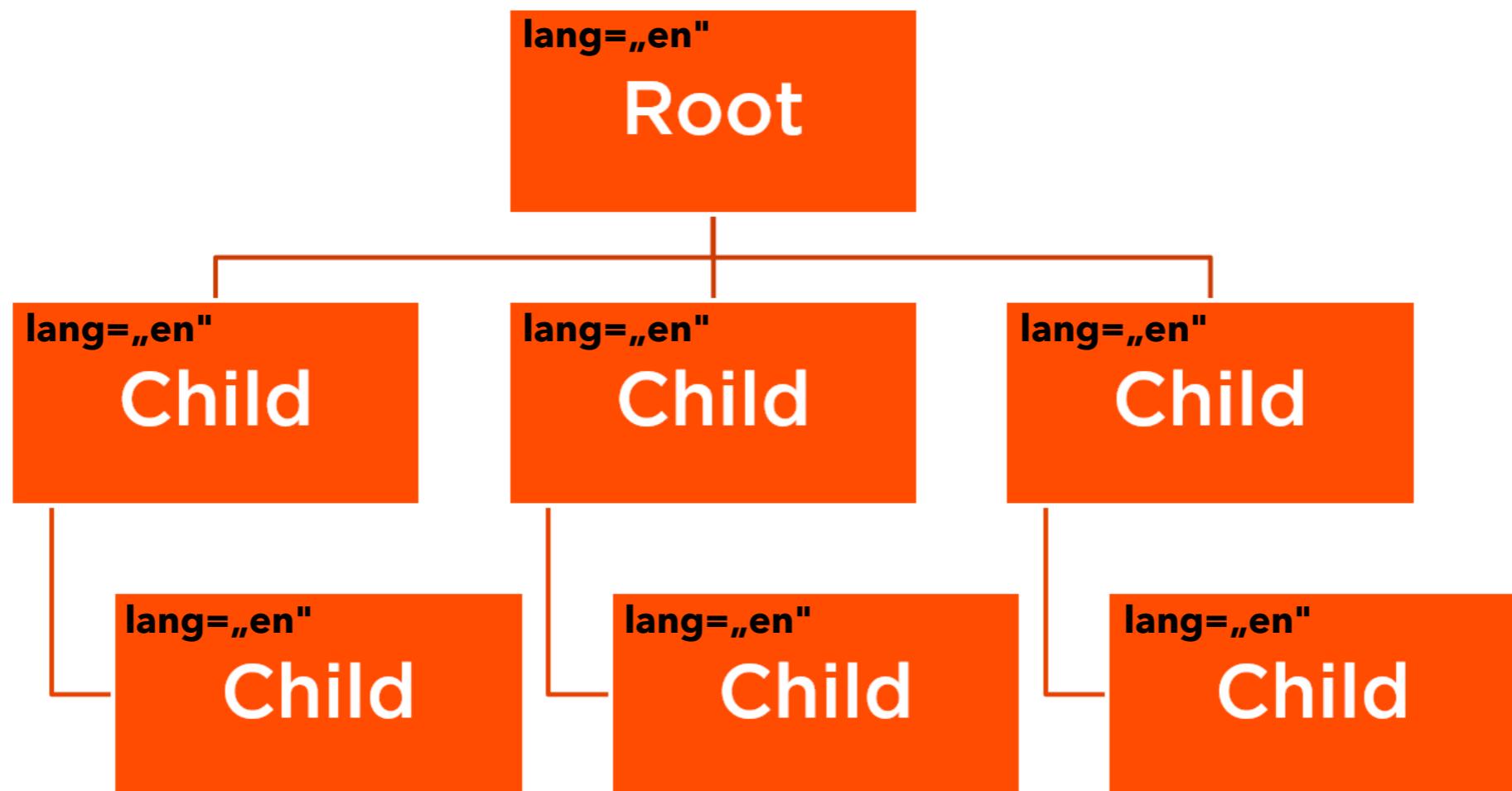
- ✖ 22:52:11.651 ► Warning: Failed prop type: Invalid prop `name` of type `array` supplied to `Hello`, expected `string`.
in Hello (at App.js:6)
in App (at src/index.js:7)

PROPTYPES

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

CONTEXT API

**CONTEXT
WHEN?**



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 />
```

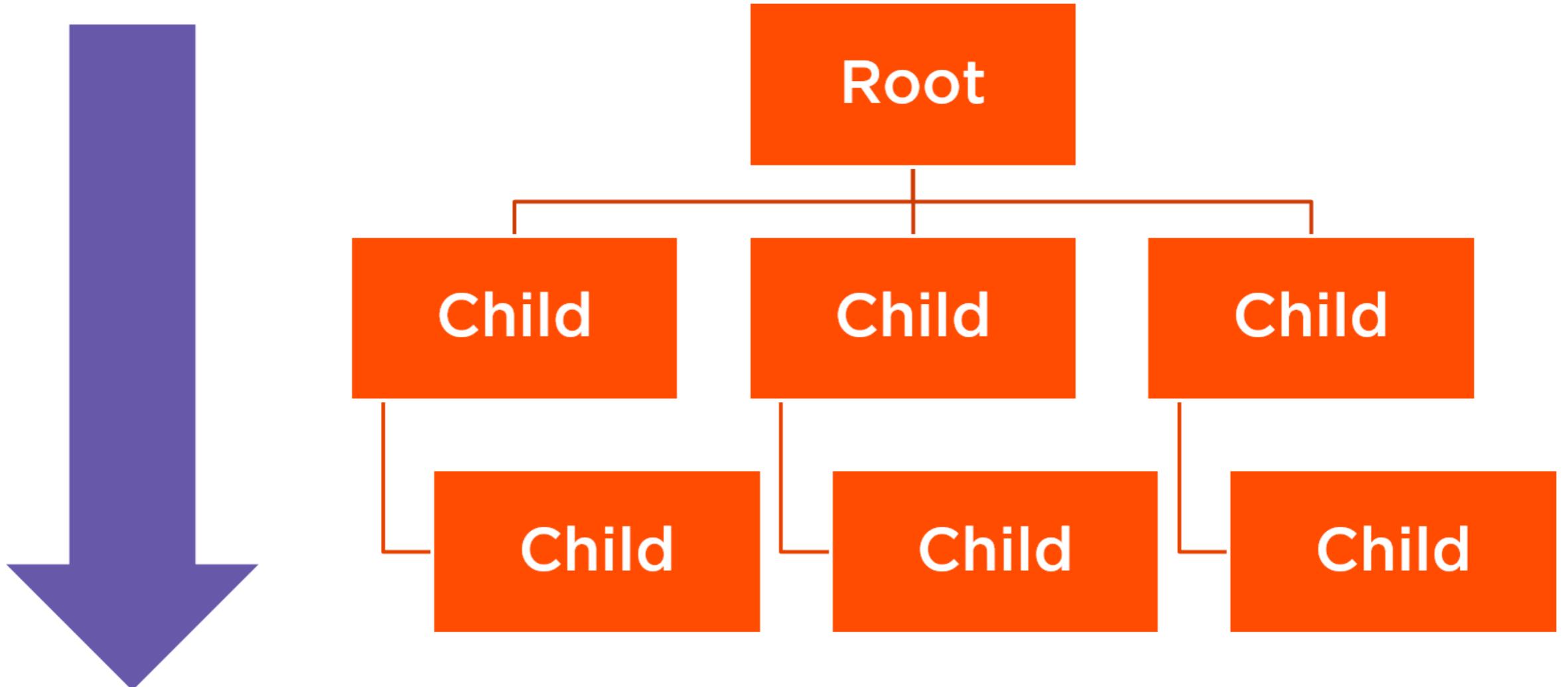
```
<Context.Consumer />
```

<PROVIDER>

<CONSUMER>

Language

Context

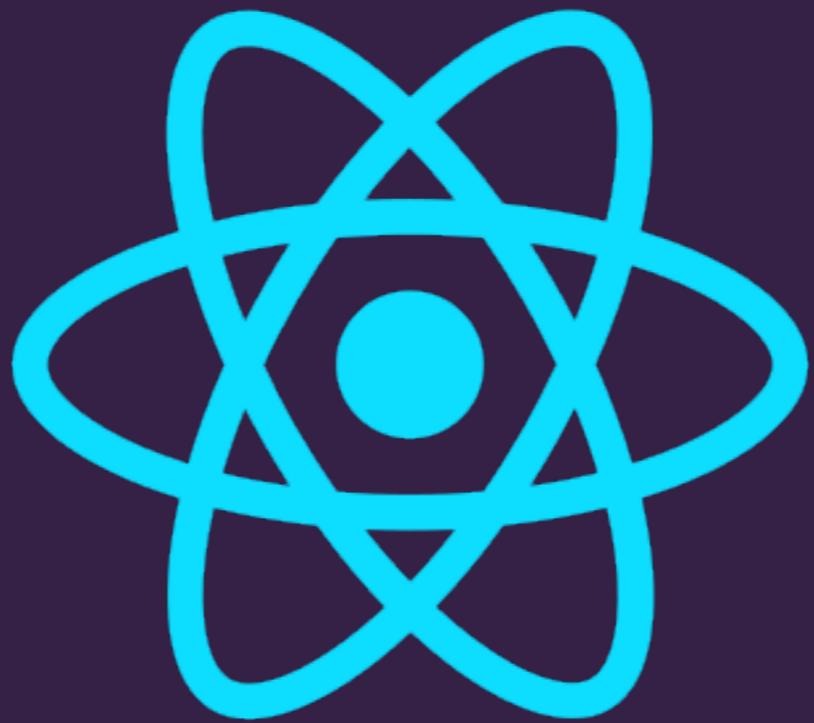


```
const LanguageContext = React.createContext();
```

```
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>
);
```

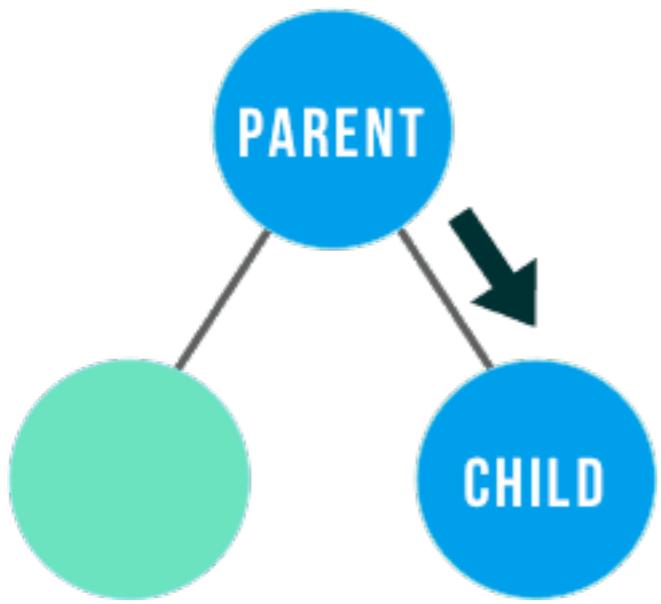
DEMO

THANKS

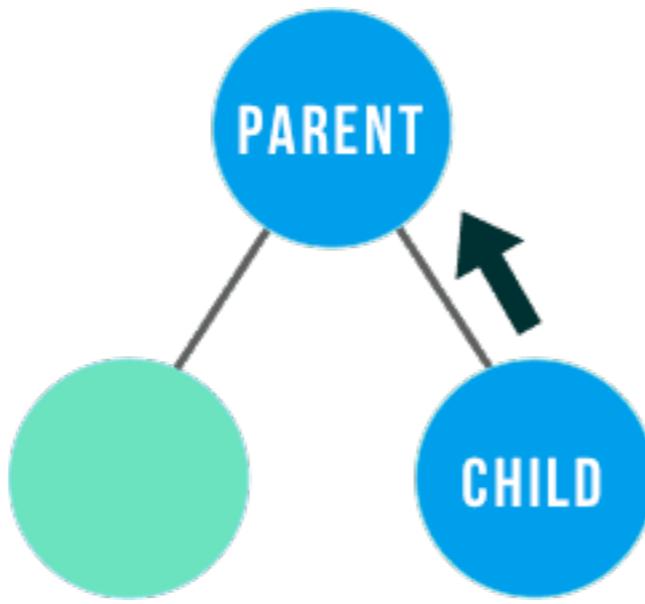


REACT.JS

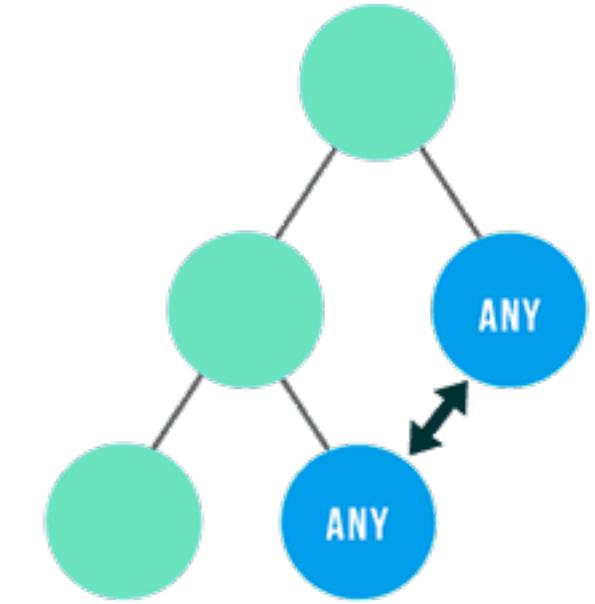
SUMMARY



PROPS



EVENTS



CONTEXT

HOC
**HIGH ORDER
COMPONENT**

HOC

```
component => componentOnSteroids;
```

HOC

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

HOC

```
const App = () => (
  <>
    <ComponentA name="Kasia" />
    <ComponentB name="Asia" />
  </>
);
```

HOC

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

```
const ComponentALang = withLanguage(ComponentA);
const ComponentBLang = withLanguage(ComponentB);
```

```
const App = () => (
  <>
    <ComponentALang name="Kasia" />
    <ComponentBLang name="Asia" />
  </>
);
```

HOC

DEMO

HOC

DON'T MUTATE THE ORIGINAL COMPONENT.

USE COMPOSITION.

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

react-fns

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

HOC BUT...

Not very intuitive to implement

Lots of code

Wrapper hell

```
▼ <Route>
  ▼ <main>
    ▼ <TransitionGroup component={null}>
      ▼ <CSSTransition key=".0" timeout={0} in={true}>
        ▼ <Transition timeout={0} in={true} exit={true} enter={true} mountOnEnter={false} unmountOnExit={false} appear={false}>
          ▼ <div className="page-wrap">
            ▼ <Route>
              ▼ <Switch>
                ▼ <AuthorisedRoute exact={true} path="/">
                  ▼ <Connect(RoutePrivate) path="/" exact={true}>
                    ▼ <RoutePrivate path="/" exact={true} token="eyJraWQiOiJjZGVFei1qTzFNNmhhsT3BCUzM5VjJkQWRocU9oZ1..." isRefreshing={false} to="/login">
                      ▼ <Route path="/" exact={true} to="/login">
                        ▼ <withRouter(ScrollToTopRoute)>
                          ▼ <Route>
                            ▼ <ScrollToTopRoute>
                              ▼ <LoadableComponent>
                                ▼ <withRouter(Connect(Dashboard))>
                                  ▼ <Route>
                                    ▼ <Connect(Dashboard)>
                                      ▼ <Dashboard>
                                        ▼ <Section home={true} color="yellow-dark">
                                          ▼ <div className="section section--yellow-dark section--home">
                                            ▼ <div className="section_inner">
                                              ▼ <Wrapper>
                                                ▼ <Container tag="div">
                                                  ▼ <div className="container">
                                                    ▼ <Row className="justify-content-center" tag="div">
                                                      ▼ <div className="justify-content-center row">
                                                        ▼ <Col md={15} xl={10} tag="div">
                                                          ▼ <div className="col-md-15 col-xl-10">
                                                            <h1>Welcome 123_test,</h1>
                                                          ▶ <Connect(QuickActionContainer)>...</Connect(QuickActionContainer)>
```





There's something you should know.

COMPONENT BASIC LIFECYCLE

MOUNTING



UPDATING



UNMOUNTING



HOOKS

HOOKS

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

HOOKS

The image shows four versions of a React component, each with a different background color (blue, purple, orange, and yellow) and a corresponding screenshot of the component's interface.

Version 1 (Blue): Class-based component using lifecycle methods and event listeners.

```
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: 'Poppins',
      width: window.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: window.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>
    )
  }
}
```

Version 2 (Purple): Class-based component using 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: 'Poppins',
      width: window.innerWidth
    }
  }

  componentDidMount() {
    this.setState({ name: 'Mary' })
    this.setState({ surname: 'Poppins' })
    this.setState({ width: window.innerWidth })
  }

  componentDidUpdate() {
    this.setState({ name: 'Mary' })
    this.setState({ surname: 'Poppins' })
    this.setState({ width: window.innerWidth })
  }

  componentWillUnmount() {
    this.setState({ name: 'Mary' })
    this.setState({ surname: 'Poppins' })
    this.setState({ width: window.innerWidth })
  }

  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>
    )
  }
}
```

Version 3 (Orange): Functional component using useState, useContext, and useEffect.

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

export default function Greetings(props) {
  const [name, setName] = useState('Mary')
  const [surname, setSurname] = useState('Poppins')
  const theme = useContext(ThemeContext)
  const width = useState(window.innerWidth)

  useEffect(() => {
    document.title = name + ' ' + surname
  })

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

function handleNameChange(e) {
  setName(e.target.value)
}

function handleSurnameChange(e) {
  setSurname(e.target.value)
}
```

Version 4 (Yellow): Functional component using useState, useContext, useEffect, and a utility hook.

```
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

CURRE useState

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

current value

update function

Initial value

HOOKS

useState

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

```
  return (  
    <div>  
      <h2>Counter: {count}</h2>  
    </div>  
  );  
};
```

HOOKS

useState

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

  return (
    <div>
      <h2>Counter: {count}</h2>
      <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

useContext

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

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

HOOKS

useContext

DEMO

HOOKS

useEffect

Accepts a function that contains imperative, possibly effectful code.

```
useEffect(() => {  
  // add listener for feature (setup)  
  // return function to remove listener for feature (clean up)  
});
```

HOOKS

useEffect

```
useEffect(() => {  
  window.document.title = `Counter: ${count}`  
});
```

HOOKS

useEffect

```
useEffect(() => {  
});
```

Always

```
useEffect(() => {  
}, []);
```

Only Once

```
useEffect(() => {  
}, [count]);
```

When deps change

HOOKS

useEffect

```
useEffect(() => {  
  ...  
  return () => {  
    console.log('Component will unmount')  
  }  
});
```

HOOKS

useEffect

DEMO

HOOKS

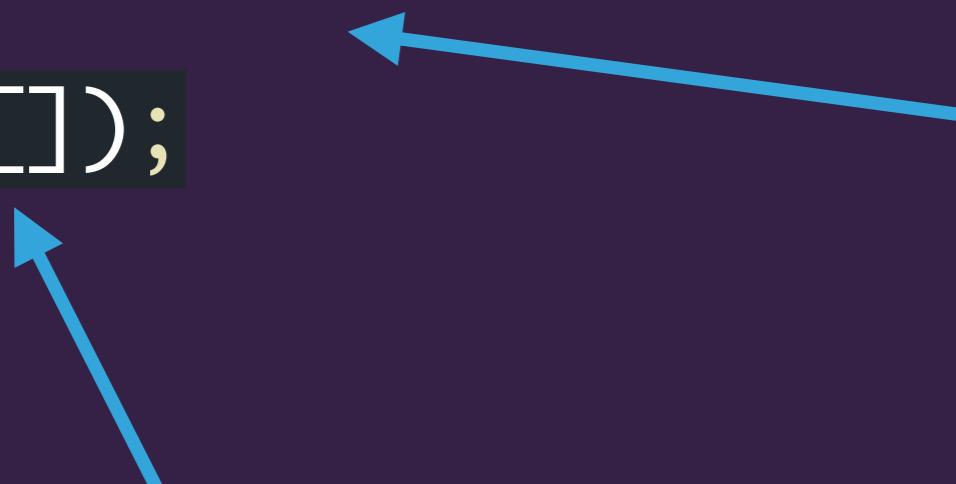
useMemo

```
useMemo(() => {
```

```
}, []);
```

factory

dependency



HOOKS

useMemo

```
const memoizedValue = useMemo(  
  () => {  
    return computeExpensiveValue(a, b);  
  },  
  [a, b]  
);
```

HOOKS

useMemo

DEMO

HOOKS

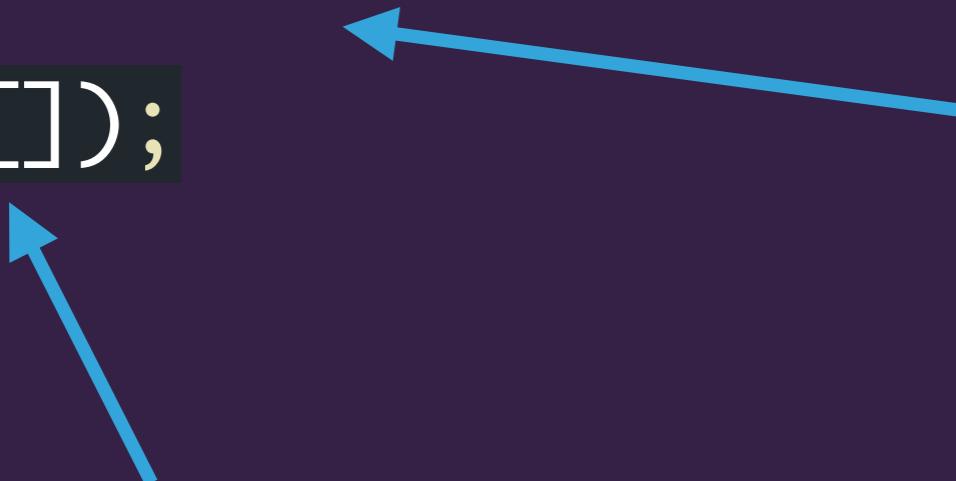
useCallback

```
useCallback(() => {
```

```
}, []);
```

callback

dependency



HOOKS

useCallback

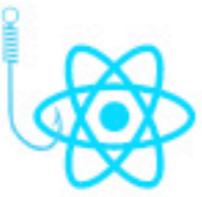
```
const removeFromCart = () => removeItem(product.id);
```

```
return (
  <Button onClick={removeFromCart}>Delete</Button>
);
```

HOOKS

useCallback

```
const removeFromCart = useCallback(  
  () => {  
    removeItem(product.id);  
  },  
  [product.id]  
);  
  
return (  
  <Button onClick={removeFromCart}>Delete</Button>  
);
```



React Hooks Lifecycle

version: latest, 16.9.x, 16.8.x

"Render phase"

Pure and has no side effects. May be paused, aborted or restarted by React

Mounting

`function () {}`

`useMemo()`

`return ()`

"Commit phase"

Can work with DOM, run side effects, schedule updates.

Updating

`useState()`, `useReducer()`, `useContext()`

`useCallback()`

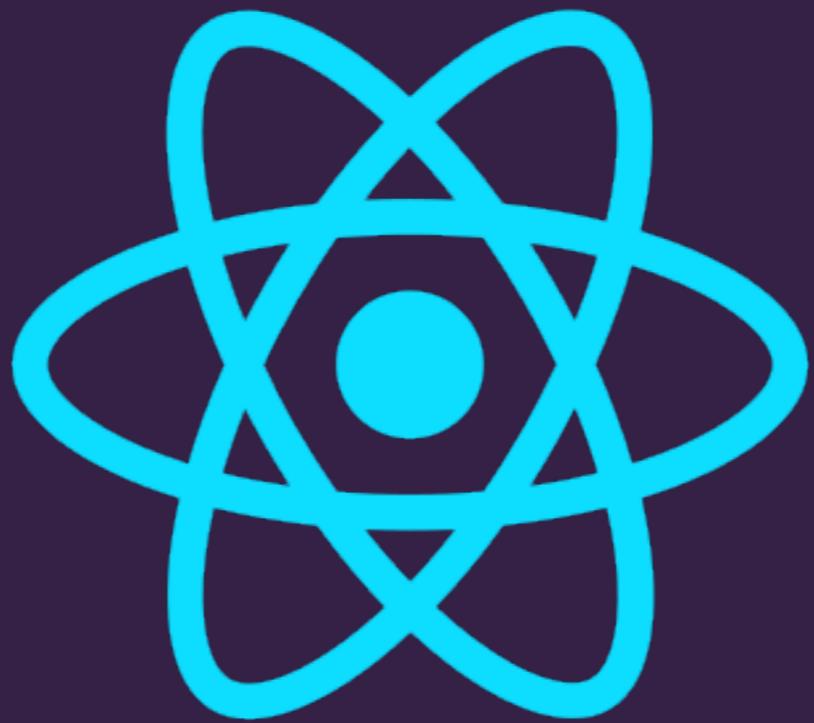


React updates DOM and refs

`useEffect()`

`useLayoutEffect()`

Unmounting



REACT.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.

HOOKS

useRef

```
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

useRef

DEMO

HOOKS

useYourImagination

Write custom hook

HOOKS

useYourImagination

useCounter

HOOKS

useCounter

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

```
  useEffect(  
    () => {  
      window.document.title = `Count ${count}`;  
    },  
    [count]  
  );
```

```
  const increase = () => setCount(count + 1);
```

```
  return [count, increase];  
};
```

HOOKS

useCounter

```
const App = () => {
  const [count, increase] = useCounter();
  return (
    <div>
      <h2>Count {count}</h2>
      <button onClick={() => increase()}>Click</button>
    </div>
  );
};
```



**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

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

App.js

```
import "./Style.css";  
  
const Hello = () => <h1 className="red">Hello</h1>;
```

CSS MODULES

CSS MODULES

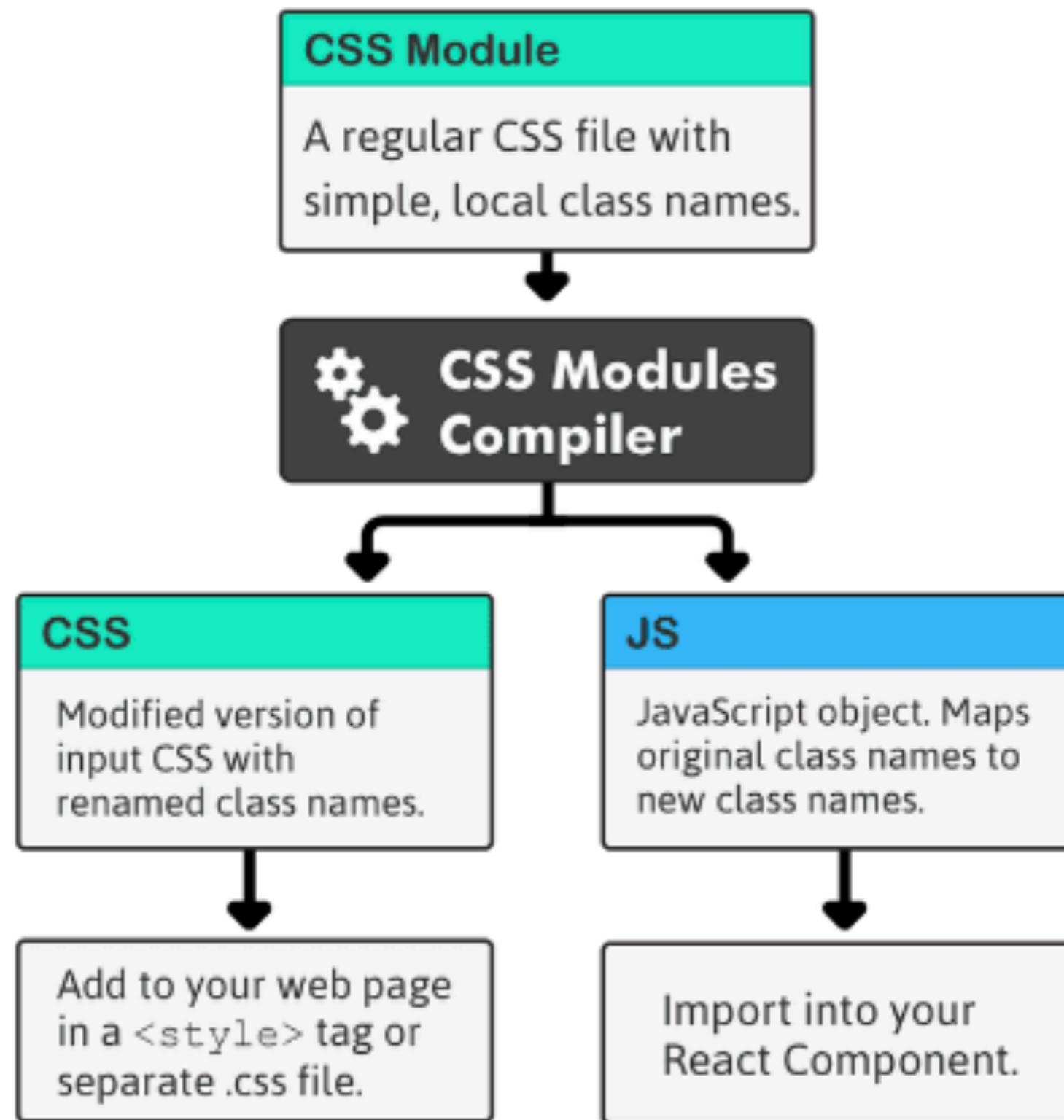
Style.module.css

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

App.js

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

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



CSS 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  
`;
```

```
const App = () => <Hello>Hello world</Hello>;
```

STYLED COMPONENTS

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

```
const App = () => <Hello>Hello world</Hello>;
```

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

```

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

STYLED COMPONENTS

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

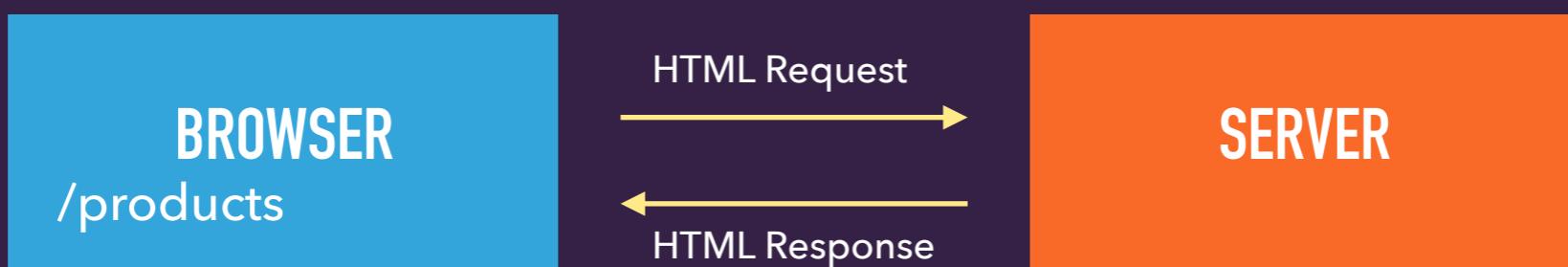
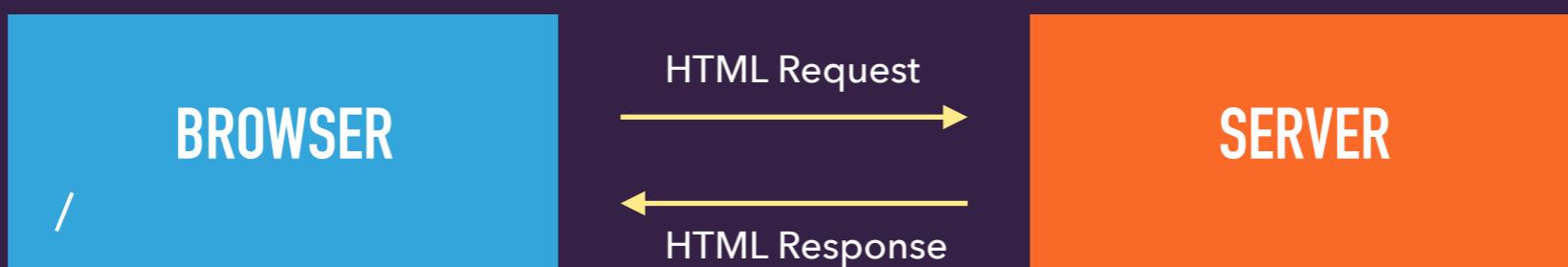
```
<Hello color="red">Hello world</Hello>;
```

Other libraries...

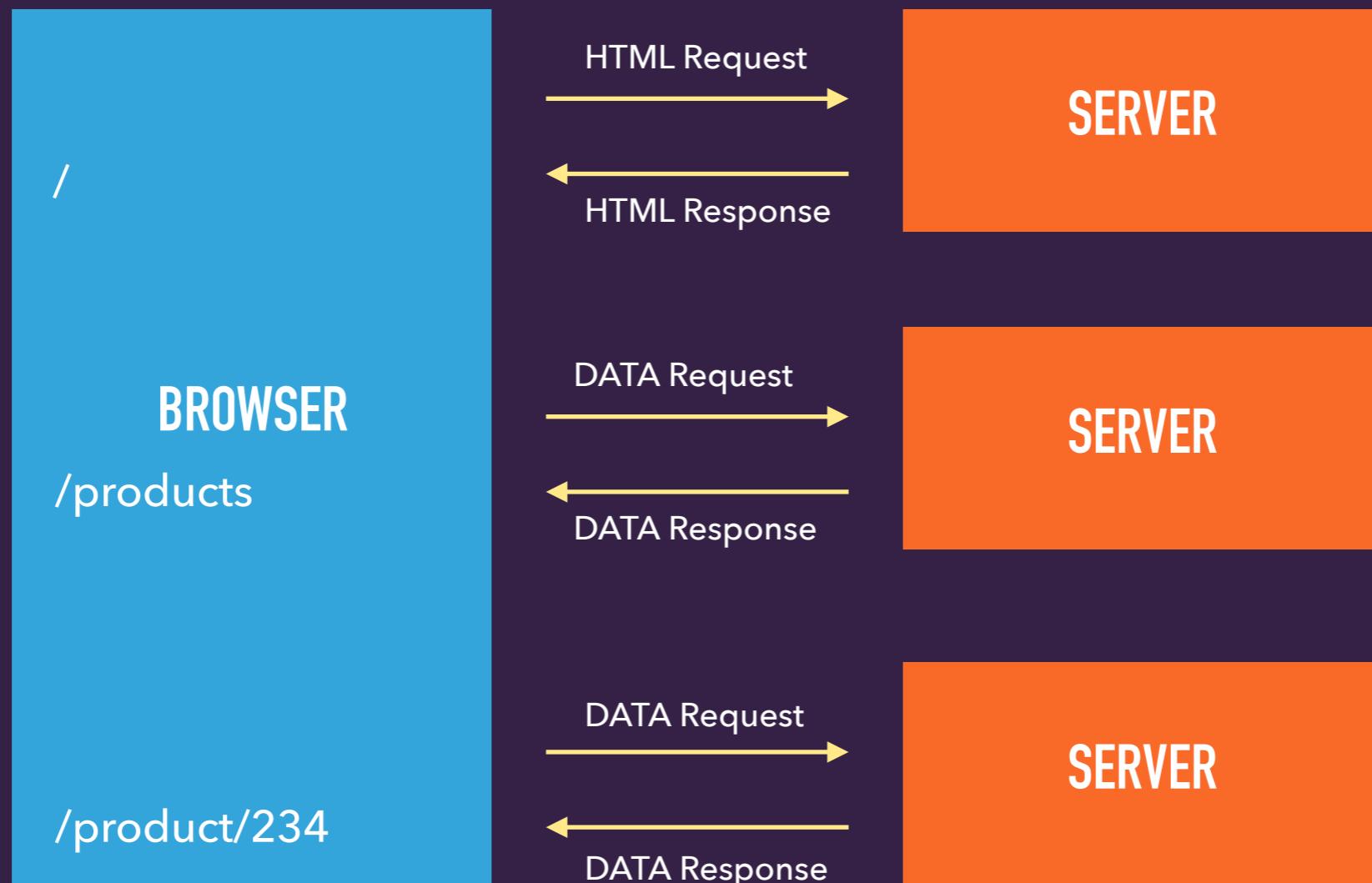
<https://github.com/tuchk4/awesome-css-in-js>

ROUTING

TRADITIONAL WEBSITE



SINGLE PAGE APPLICATION

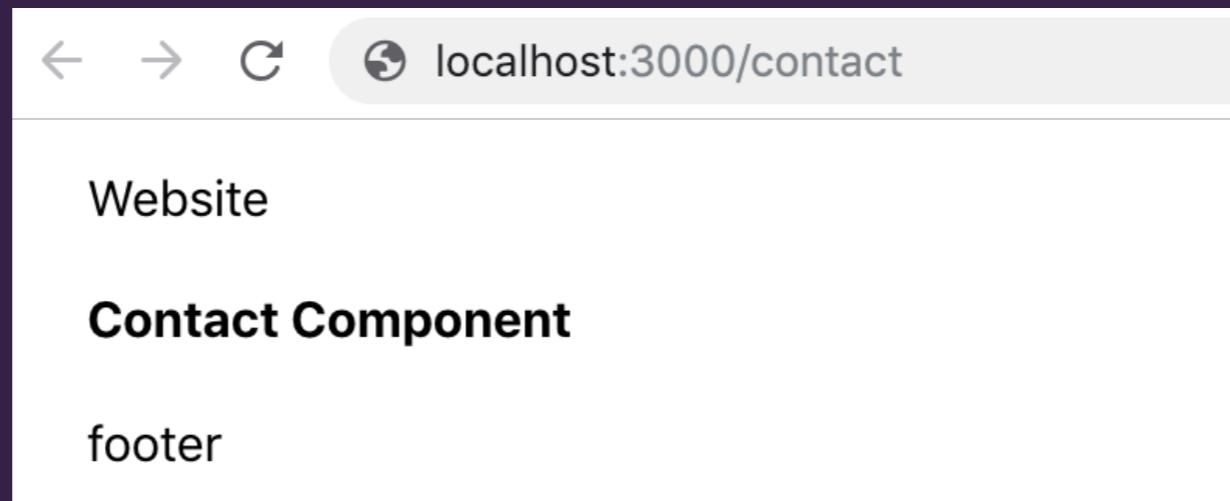
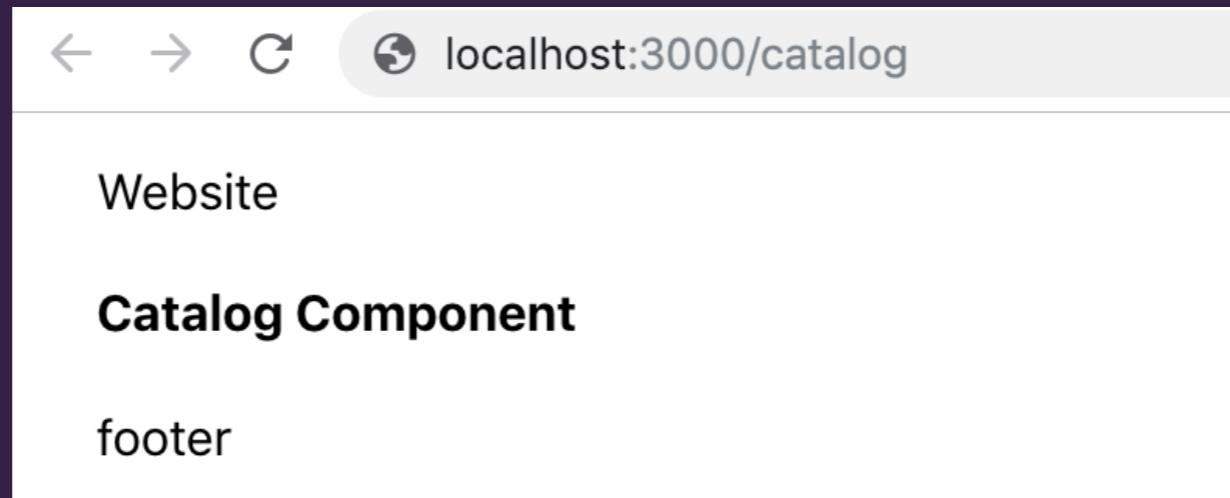
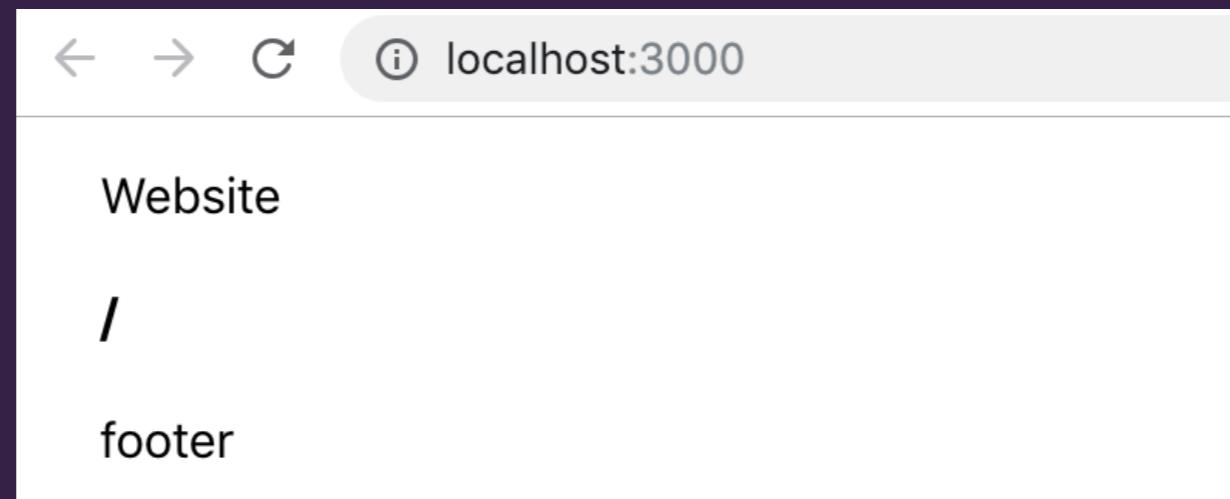


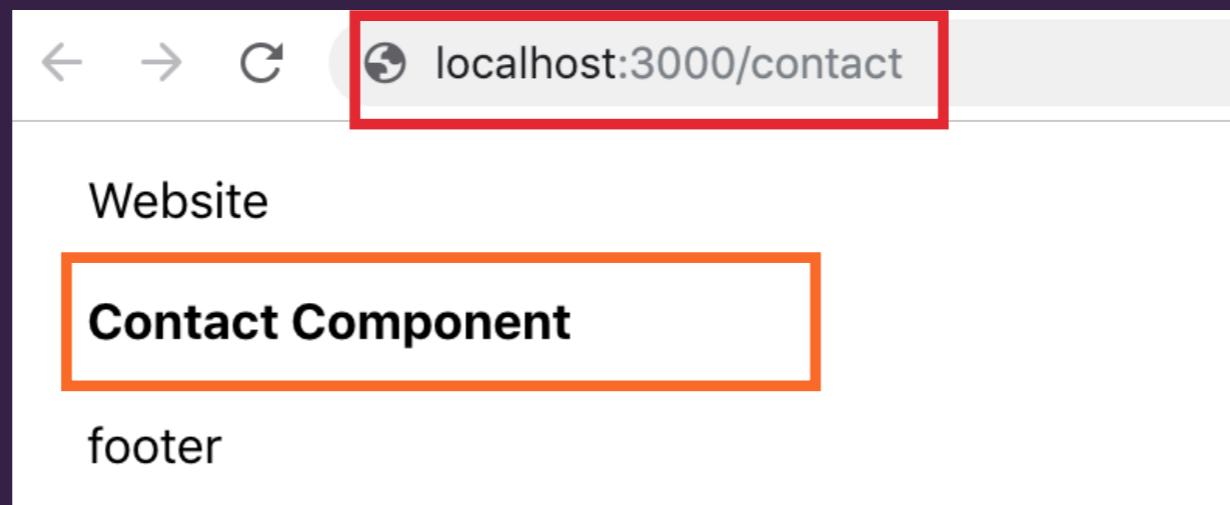
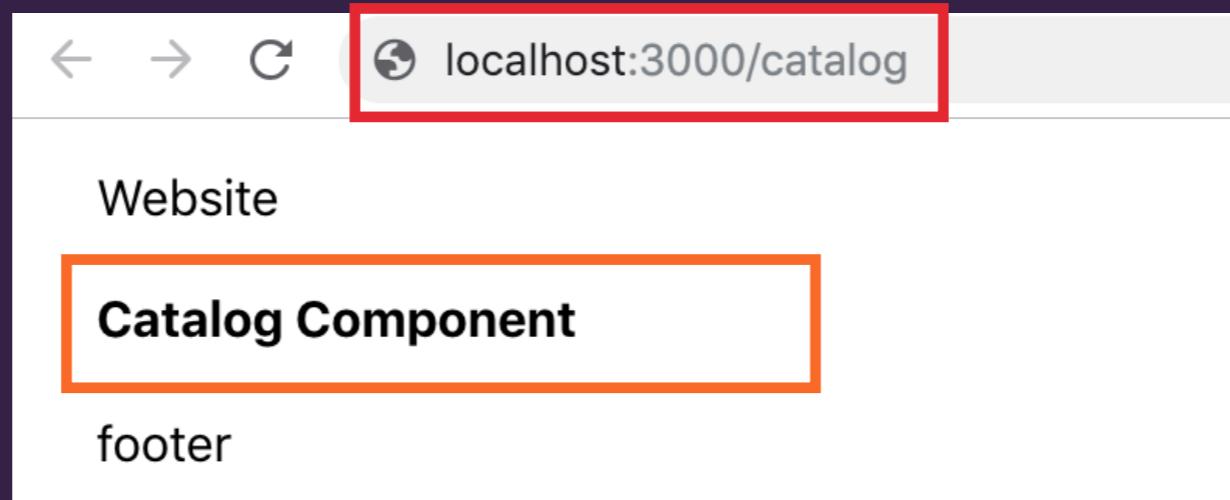
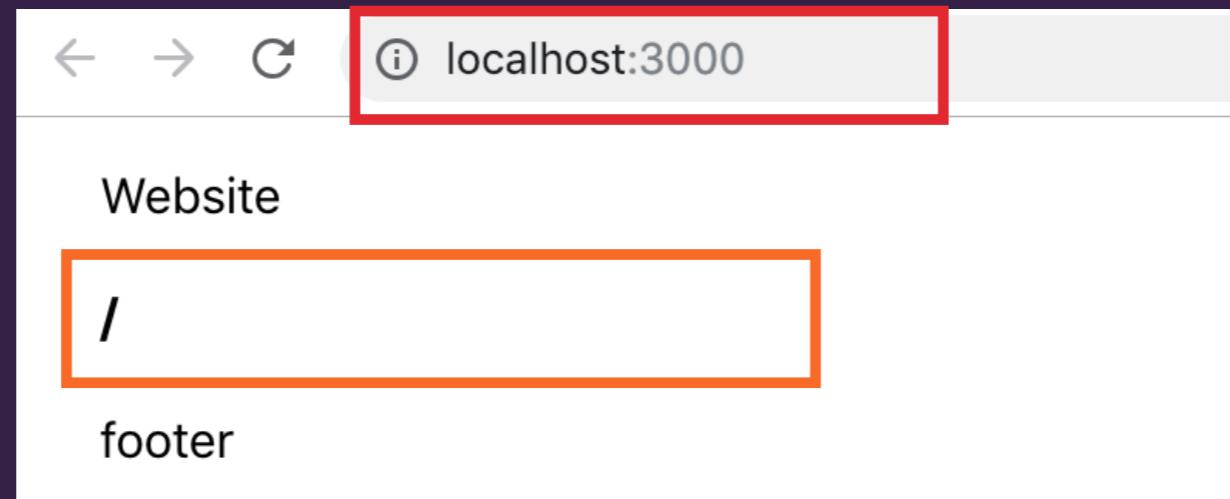
STATIC ROUTING

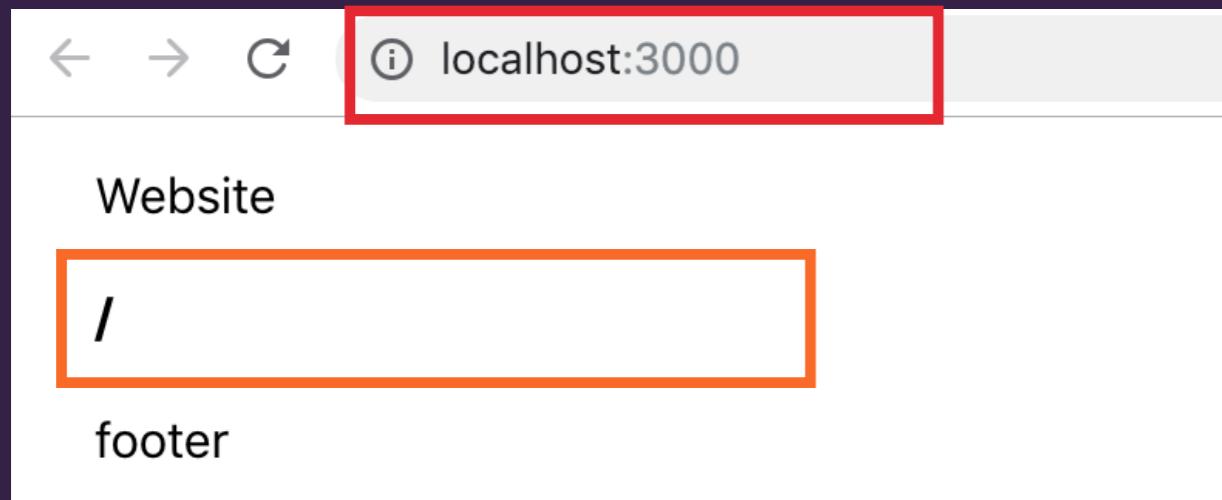
DYNAMIC ROUTING

„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.”

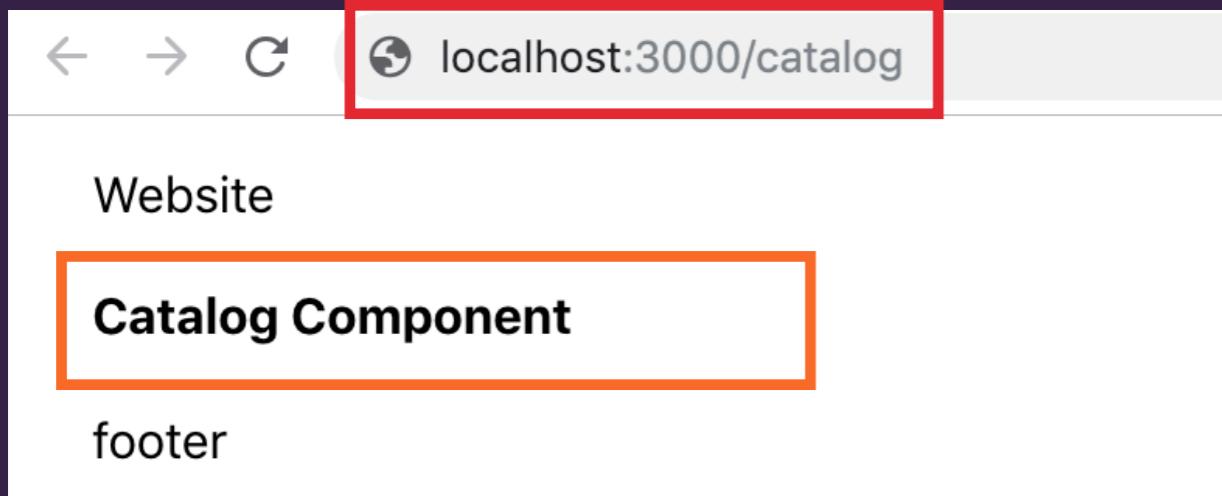
– React Training



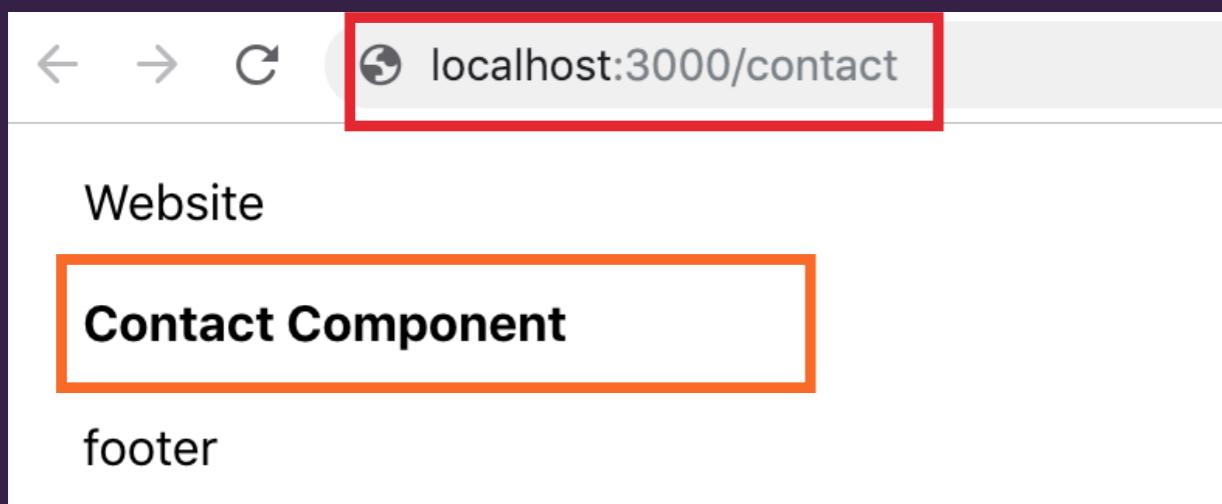




path: /
component: Home



path: /catalog
component: Catalog



path: /contact
component: Contact

REACT ROUTER V5

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 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

<MemoryRouter>

```
<MemoryRouter>
  <App />
</MemoryRouter>
```

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 />} />
```

PARAMETERS ROUTERS

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

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

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

{match, location, history} = props

REACT ROUTER API HOOKS

`useHistory`

REACT ROUTER API HOOKS

```
import { useHistory } from "react-router-dom";\n\nfunction HomeButton() {\n  const history = useHistory();\n\n  function handleClick() {\n    history.push("/home");\n  }\n\n  return (\n    <button type="button" onClick={handleClick}>\n      Go home\n    </button>\n  );\n}
```

REACT ROUTER API HOOKS

`useLocation`

REACT ROUTER API HOOKS

```
const location = useLocation();
```

```
useEffect(() => {
  //send information to google analytics
  console.log('You are here:', location.pathname)
}, [location]);
```

REACT ROUTER API HOOKS

useParams

REACT ROUTER API HOOKS

```
function BlogPost() {  
  const { postId } = useParams();  
  return <div>Now showing post {postId}</div>;  
}
```

DOCUMENTATION

reacttraining.com/react-router/

```
/catalog => <Catalog />
```

```
/contact => <Contact />
```

```
function App() {  
  return (  
    <div>  
      <h2>Website 2019</h2>  
      <Catalog />  
      <Contact />  
    </div>  
  );  
}
```

`npm i react-router-dom`

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

```
function App() {  
  return (  
    <div>  
      <h2>Website 2019</h2>  
      <Catalog />  
      <Contact />  
    </div>  
  );  
}
```

```
import {  
  BrowserRouter as Router,  
  Route  
} from "react-router-dom";  
  
function App() {  
  return (  
    <Router>  
      <div>  
        <h2>Website 2019</h2>  
        <Route path="/catalog" component={Catalog} />  
        <Route path="/contact" component={Contact} />  
      </div>  
    </Router>  
  );  
}  
export default App;
```

DEMO

404

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

NESTED ROUTERS

```
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>
);

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

url: http://localhost:3000/catalog
match.url: /catalog

path=/catalog/a
path=/catalog/b

**CLASS
COMPONENT**

CLASS COMPONENT

```
class Welcome extends React.Component
```

CLASS COMPONENT

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

```
const Welcome = props => {  
  return <h1>Hello, {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;
})(React.Component);
```

CLASS COMPONENT



**CLASS
COMPONENT
STATE**

STATE VS PROPS

**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

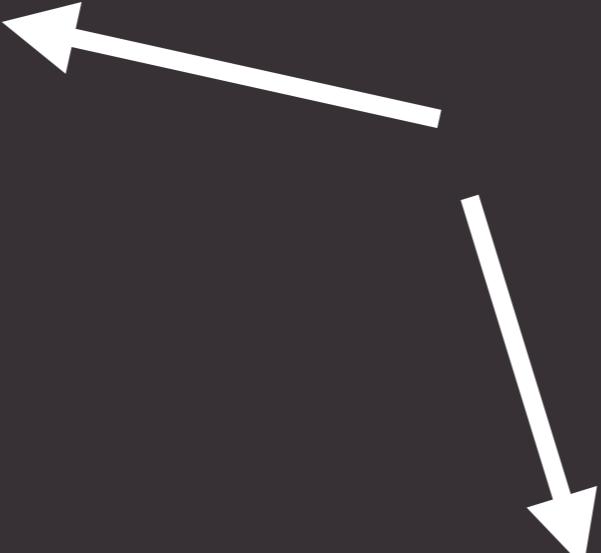
```
class Button extends React.Component {  
  constructor() {  
    super();  
    this.state = {  
      count: 0,  
    };  
  }  
}
```

```
  render() {  
    ....  
  }  
}
```



CLASS COMPONENT STATE

```
class Button extends React.Component {  
  constructor() {  
    super();  
    this.state = {  
      count: 0  
    };  
  }  
  
  render() {  
    return (  
      <button>  
        Clicked {this.state.count} times  
      </button>  
    );  
  }  
}
```

A diagram illustrating the flow of state from the constructor to the render method. A white arrow points from the 'count' variable declaration in the constructor up to the 'count' prop being used in the render method's JSX.

```
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()`

ASYNCHRONOUS

```
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()`

TRIGGER RENDER

REACT HOOKS

```
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>
  );
}
```

STATE ANTI-PATTERN



```
class MyComponent extends Component {
  constructor(props) {
    super(props);
    this.state = {
      someValue: props.someValue,
    };
  }
}
```

CLASS COMPONENT DIFFERENCES

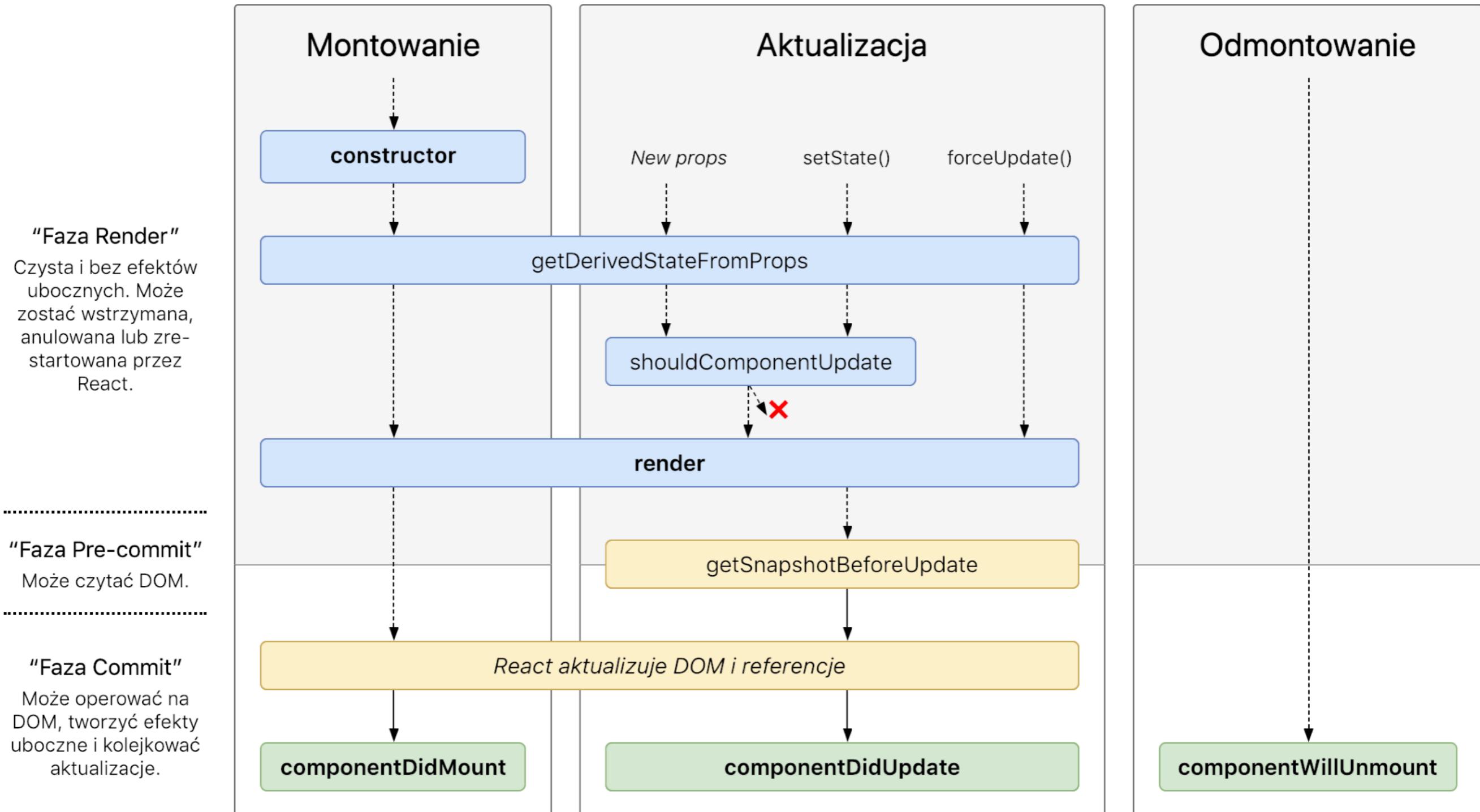
LIFECYCLE HOOKS

LIFECYCLE HOOKS

MOUNTING

UPDATING

UNMOUNTING



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

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

~undefined

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>  
    );  
  }  
}
```

AJAX & APIS

```
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  
  };
```

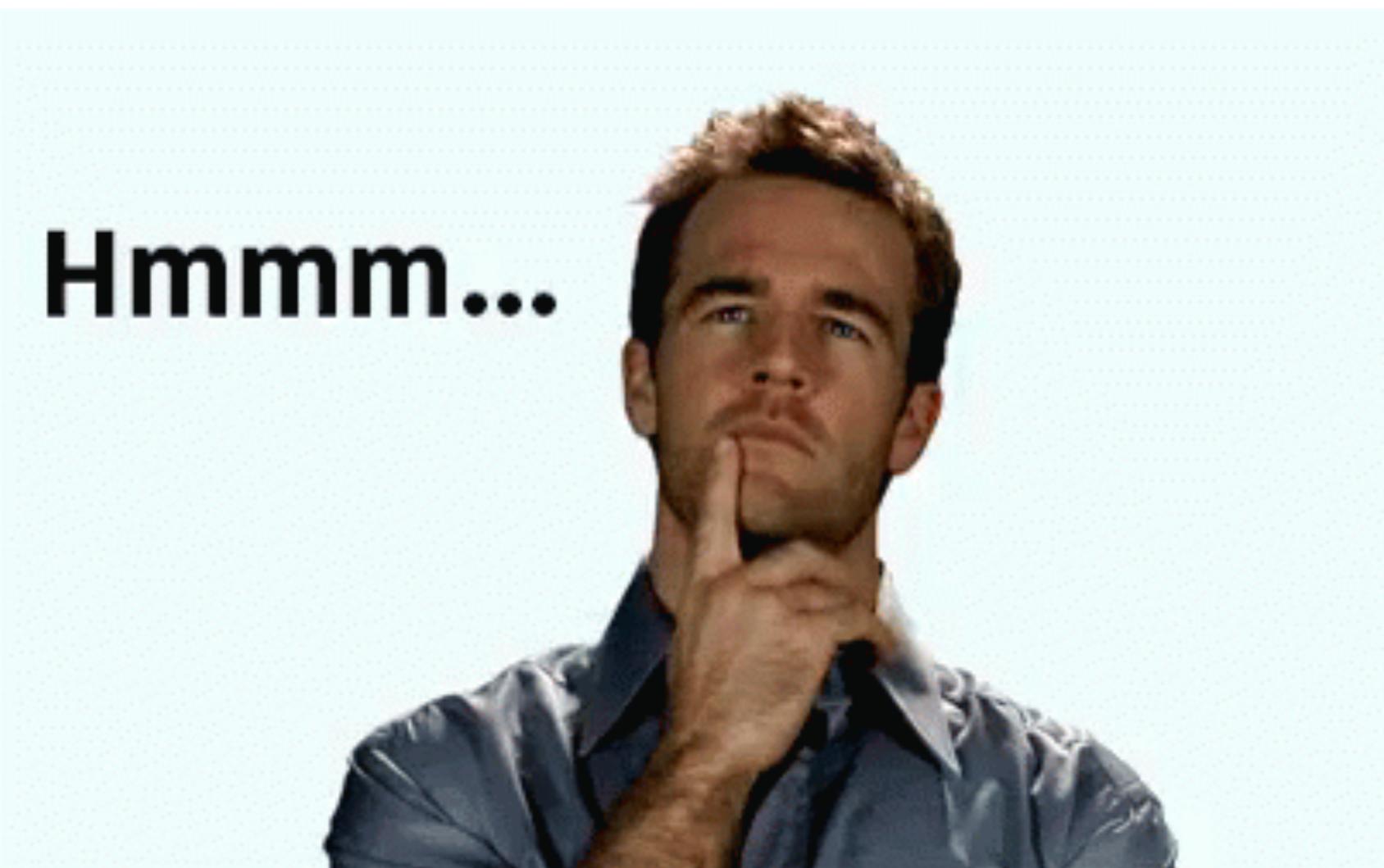
```
async componentDidMount() {  
  ///...pobieranie danych  
  this.setState({  
    user,  
    isLoaded: true  
  });  
}
```

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

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

```
  ///...dalsza część 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,  
      });  
    }  
  }  
  
  render() {  
    const { user, isLoaded, error } = this.state;  
  
    if (error) { return <div>{error}</div>; }  
  
    if (!isLoaded) { return <div>Loading...</div>; }  
  
    ///...dalsza część 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

```
// Create ref
this.inputRef = React.createRef();
```

```
// Read value
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

SIMPLE FORM

```
<form>
  <input type="text" id="login" />
  <input type="password" id="password" />
  <input type="submit" value="Log in" />
</form>
```

SIMPLE FORM

```
<form onSubmit={handleSubmit}>
  <input type="text" value={login} onChange={onChangeLogin} />
  <input type="password" value={password} onChange={onChangePassword} />
  <input type="submit" value="Log in" />
</form>
```

SIMPLE FORM

```
const [login, setLogin] = useState('')
const [password, setPassword] = useState('')

const handleSubmit = (e) => {
  e.preventDefault()
  console.log(`Login: ${login}, Password: ${password}`)
}

const onChangeLogin = (e) => {
  setLogin(e.target.value)
}

const onChangePassword = (e) => {
  setPassword(e.target.value)
}

<form onSubmit={handleSubmit}>
  <input type="text" value={login} onChange={onChangeLogin} />
  <input type="password" value={password} onChange={onChangePassword} />
  <input type="submit" value="Log in" />
</form>
```

SIMPLE FORM

```
const [formData, setFormData] = useState({  
  login: '',  
  password: '', ←  
})  
  
const handleSubmit = (e) => {  
  e.preventDefault()  
  console.log(`Login: ${login}, Password: ${password}`)  
}  
  
const onChange = ({ target: { name, value } }) => {  
  setFormData({  
    ...formData,  
    [name]: value,  
  })  
}  
const { login, password } = formData  
  
<form onSubmit={handleSubmit}> ←  
  <input type="text" name="login" value={login} onChange={onChange} />  
  <input type="password" name="password" value={password} onChange={onChange} />  
  <input type="submit" value="Log in" />  
</form>
```

DEMO

VALIDATION?

SIMPLE FORM



```
const [errors, setErrors] = useState({  
  login: '',  
  password: '',  
})
```

```
const { login, password } = formData
```

```
<form onSubmit={handleSubmit}>  
  {errors.login && <span>{errors.login}</span>}  
  <input type="text" name="Login" value={login} onChange={onChange} />  
  {errors.password && <span>{errors.password}</span>}  
  <input type="password" name="password" value={password} onChange={onChange}/>  
  <input type="submit" value="Log in" />  
</form>
```

SIMPLE FORM

```
const [formData, setFormData] = useState({  
  login: '',  
  password: '',  
})  
const handleSubmit = (e) => {  
  ???  
}  
const onChange = ({ target: { name, value } }) => {  
  ???  
}  
const { login, password } = formData  
  
<form onSubmit={handleSubmit}>  
  <input type="text" name="login" value={login} onChange={onChange} />  
  <input type="password" name="password" value={password} onChange={onChange} />  
  <input type="submit" value="Log in" />  
</form>
```

FORMIK

„Build forms in React, without the tears.”

<https://jaredpalmer.com/formik/>

FORMIK

```
import { useFormik } from 'formik'

const { values, handleChange, handleSubmit } = useFormik({
  initialValues: {
    login: '',
    password: '',
  },
  onSubmit: (values) => {
    console.log(`Login: ${values.login}, Password: ${values.password}`)
  },
})
```

```
<form onSubmit={handleSubmit}>
  <input type="text" name="login" value={values.login} onChange={handleChange} />
  <input type="password" name="password" value={values.password} onChange={handleChange} />
  <input type="submit" value="Log in" />
</form>
```

FORMIK

getFieldProps(name)

```
{  
  name: "name",  
  onBlur: f (),  
  onChange: f (),  
  value: ""  
}
```

```
<input type="text" {...formik.getFieldProps('login')} />  
<input type="password" {...formik.getFieldProps('password')} />
```

FORMIK

```
import { Formik, Field, Form } from 'formik'
```

```
<Formik
  initialValues={{ login: '', password: '' }}
  onSubmit={(values) => {
    console.log(`Login: ${values.login}, Password: ${values.password}`)
  }}
>
  <Form>
    <Field name="login" type="text" />
    <Field name="password" type="password" />
    <button type="submit">Submit</button>
  </Form>
</Formik>
```

FORMIK DEMO

VALIDATION?

FORMIK

Field-Level

```
const required = (value) => {
  let error
  if (!value) {
    error = 'Required'
  }
  return error
}
```

```
<ErrorMessage name="login" />
<Field name="login" type="text" validate={required} />
```



FORMIK

Form-Level

```
const validate = (values) => {
  const errors = {}
  if (!values.login) {
    errors.login = 'Required'
  }
  if (!values.password) {
    errors.password = 'Required'
  }
  return errors
}
```

```
<Formik
  validate={validate}
  initialValues={{ login: '', password: '' }}>
```



FORMIK YUP

```
import * as Yup from 'yup'
```

```
const LoginSchema = Yup.object().shape({
  login: Yup.string().required('Required'),
  password: Yup.string().required('Required'),
})
```



```
<Formik
  validationSchema={LoginSchema}
  initialValues={{ login: '', password: '' }}
```

DEMO

GLOBAL STATE?
WHY?

CATALOG

Search

Clear

search...

Manufacturer

- All
- Apple
- Dell

Cart

Viewed



\$2099.99

iMac 27 Retina



\$2099.99

iMac 27 Retina



\$2099.99

iMac 27 Retina



\$2099.99

iMac 27 Retina



\$2099.99

iMac 27 Retina



\$2099.99

iMac 27 Retina



\$2099.99

iMac 27 Retina



\$2099.99

iMac 27 Retina



\$2099.99

iMac 27 Retina



\$2099.99

iMac 27 Retina



\$2099.99

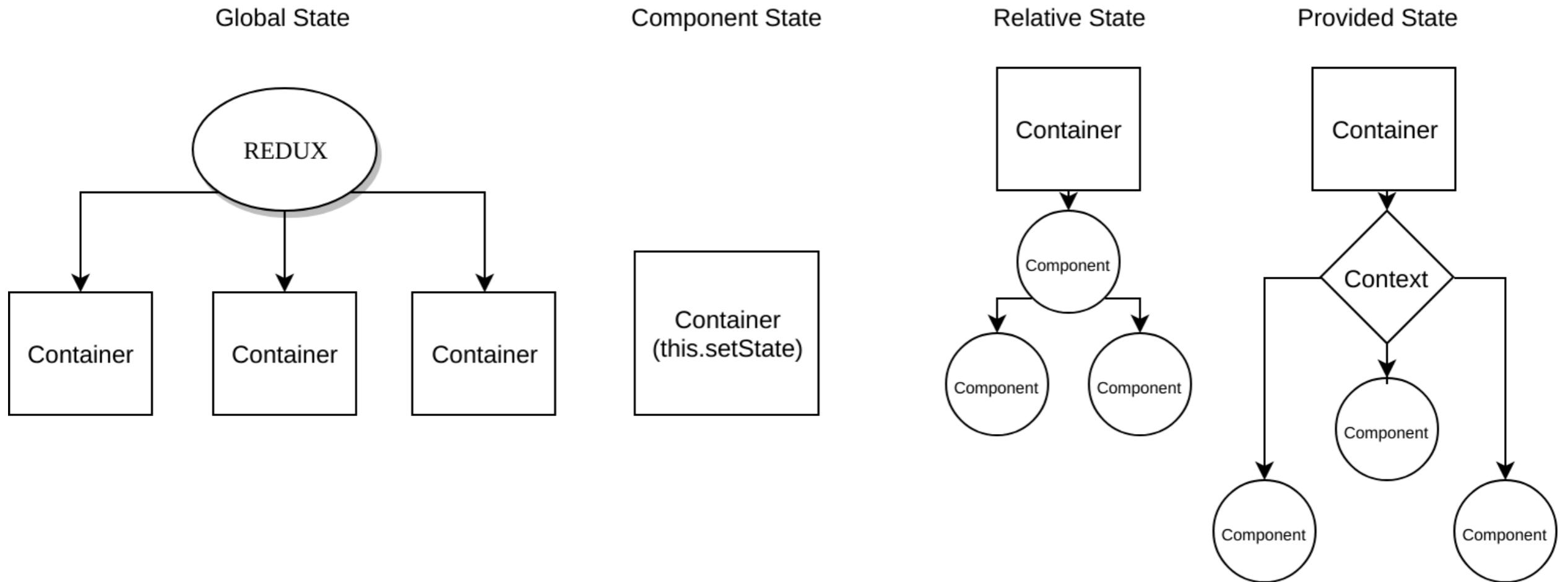
iMac 27 Retina



\$2099.99

iMac 27 Retina

REACT STATE MANAGEMENT PATTERNS



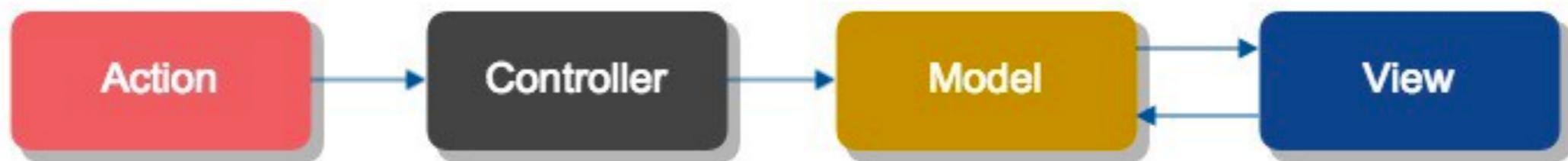
REDUX?



BUT FIRST...

MVC

MODEL, VIEW, CONTROLLER



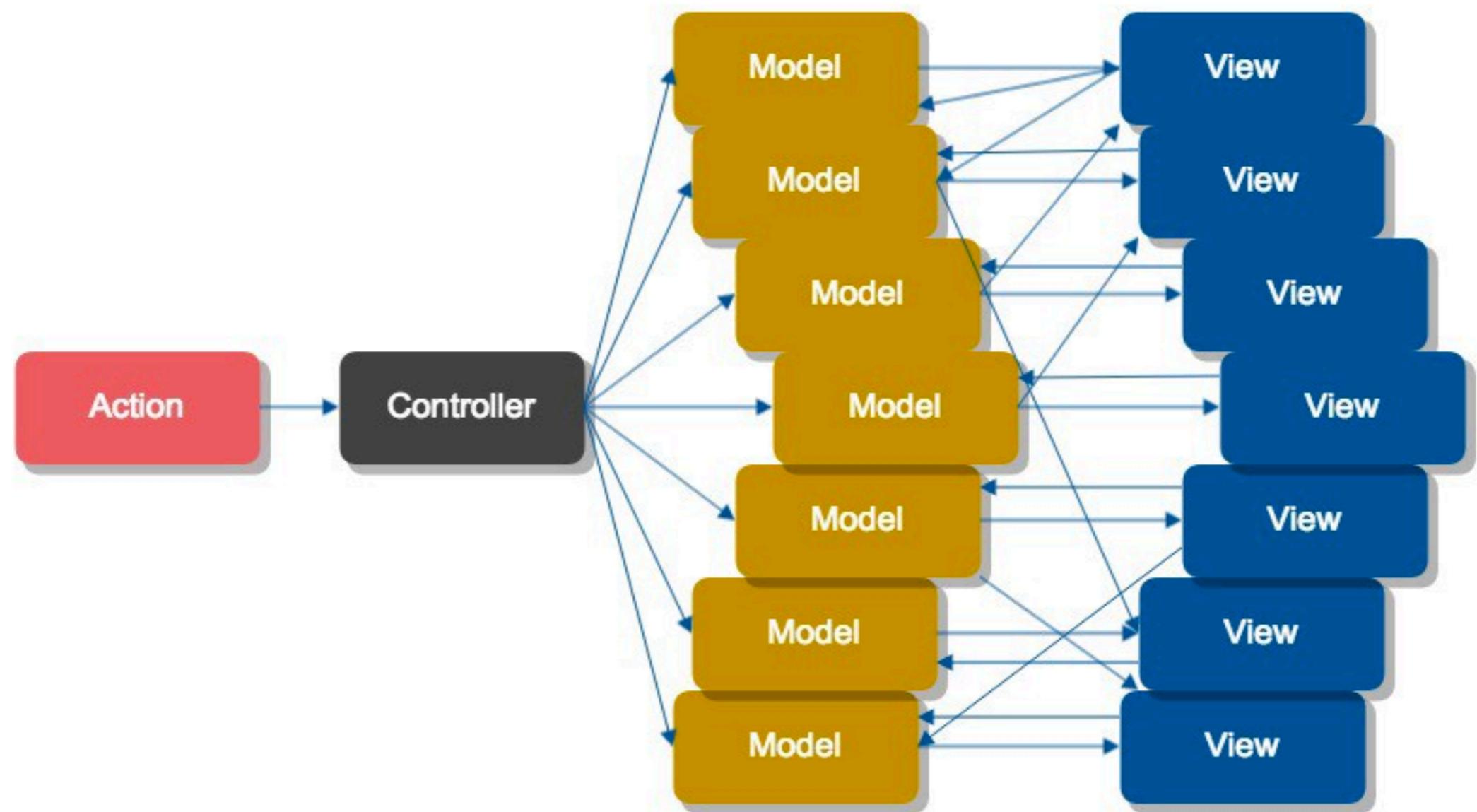
Model: manages the behavior and data of the application domain

View: represents the display of the model in the UI

Controller: takes user input, manipulates the model and causes the view to update

**IN SERVER SIDE, MVC IS GOOD,
BUT IN CLIENT...**

“MVC DOESN’T SCALE”



FACEBOOK

STORY







Search for people, places and things



Jing Home



83 →

Inbox Other

More

Jing Aling

+ New Message

Actions



Search



Jing Aling
let's do it!

1:12pm



Tom:

1:10pm
2 new



...



...



...



...



...



...

Conversation started today



Jing Chen
hey is it time to grab dinner again?

1:10pm



Jing Aling
let's do it!

1:12pm

Write a reply...

Add Files

Add Photos

Press Enter to send

Jing Aling



2 minutes ago

hey is it time to grab dinner again?



let's do it!



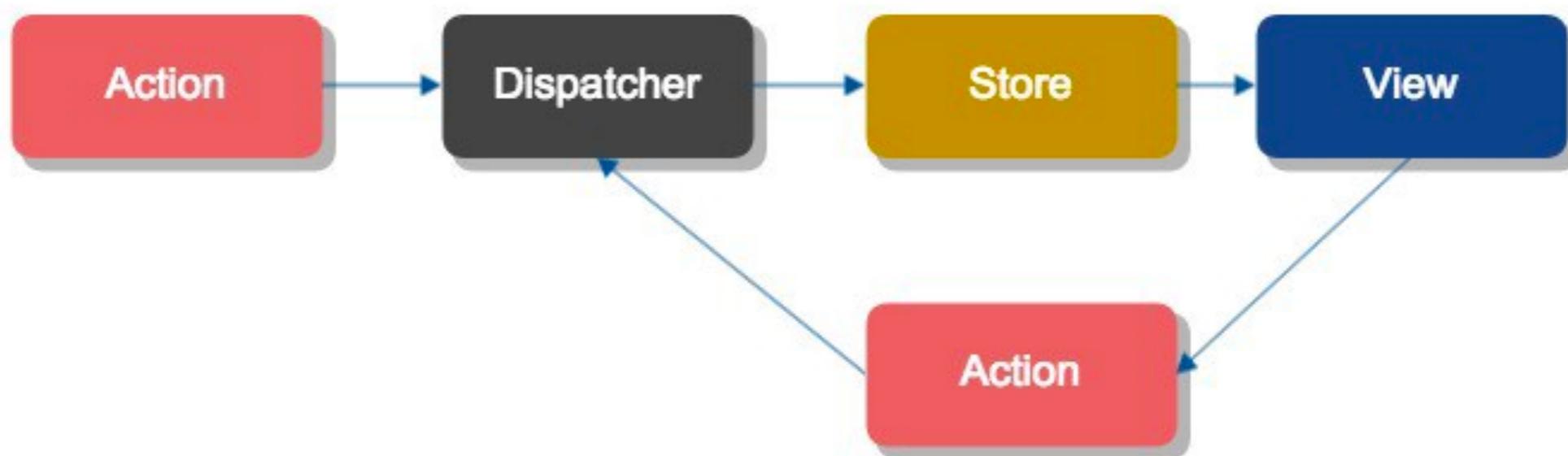
FLUX

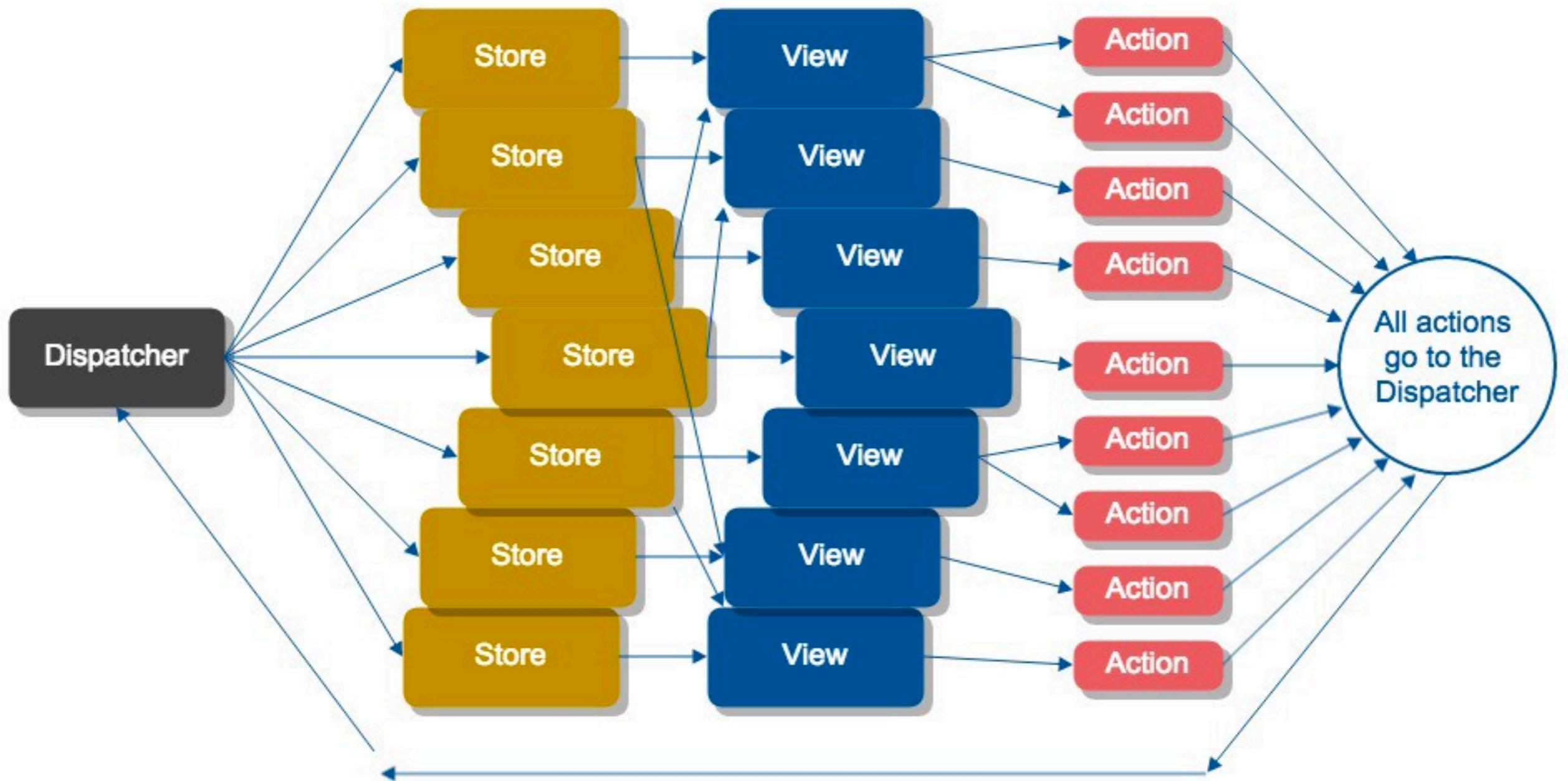
FLUX

**architecture for creating data layers in JavaScript applications
designed at Facebook along with the React**

Focused on

**explicit and understandable update paths
tracing changes during development simpler
bugs easier to track down and fix**





Flux keeps things predictable.

REDUX

A predictable state container for JavaScript apps.



A predictable state container for JavaScript apps.

[GET STARTED](#)



Predictable

Redux helps you write applications that **behave consistently**, run in different environments (client, server, and native), and are **easy to test**.



Centralized

Centralizing your application's state and logic enables powerful capabilities like **undo/redo**, **state persistence**, and much more.



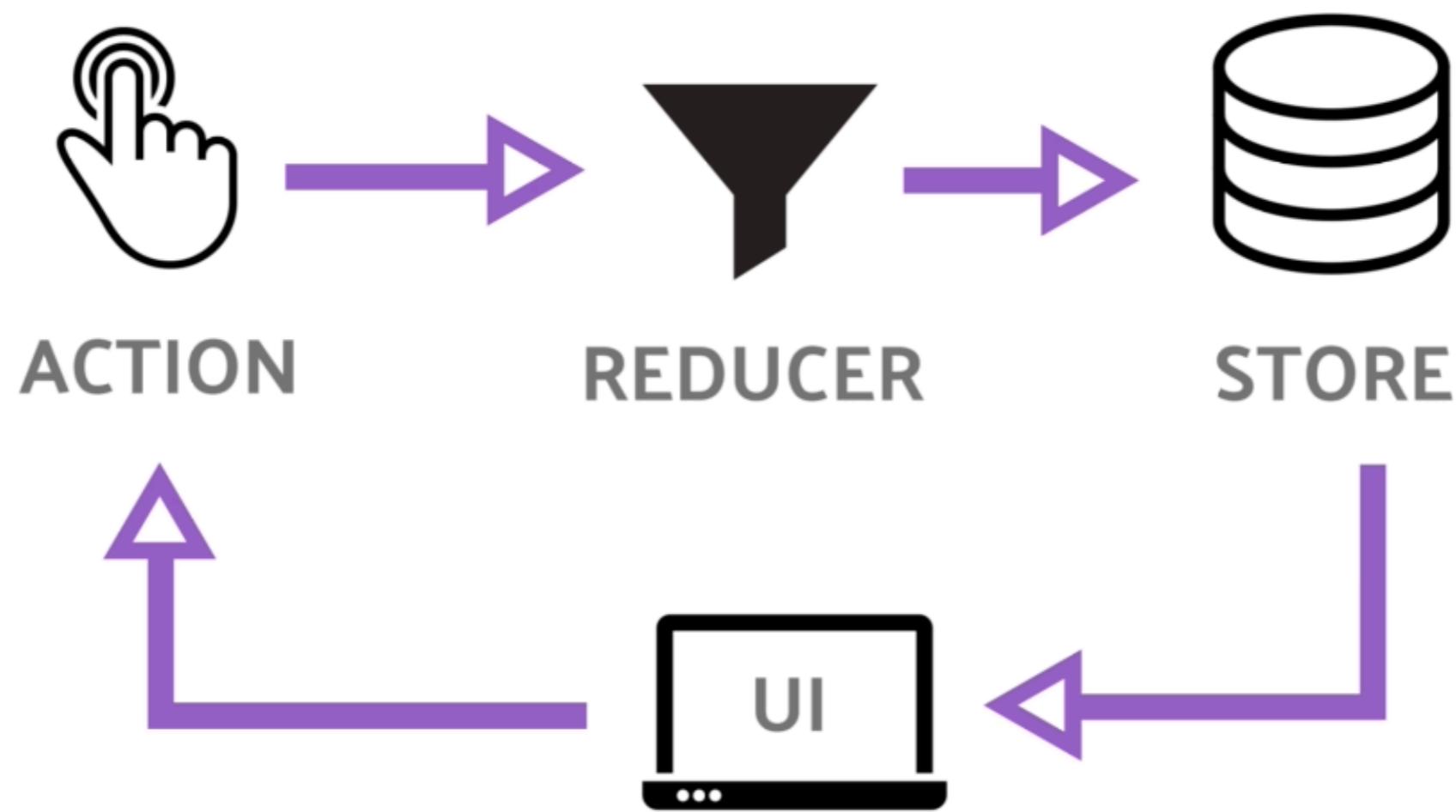
Debuggable

The Redux DevTools make it easy to trace **when, where, why, and how your application's state changed**. Redux's architecture lets you log changes, use "time-travel debugging", and even send complete error reports to a server.



Flexible

Redux **works with any UI layer**, and has a **large ecosystem of addons** to fit your needs.





The Redux principle #1.

ONE application STATE OBJECT
managed by ONE STORE



The Redux principle #2.

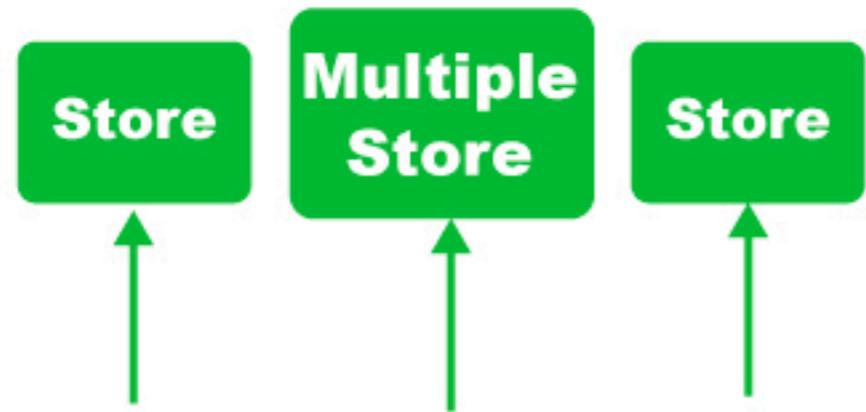
The only way to change the state is to emit an action, an object describing what happened.



The Redux principle #3.

To specify how the state tree is transformed by actions, you write pure reducers.

Flux



multiple source of truth

singleton dispatcher

mutable state



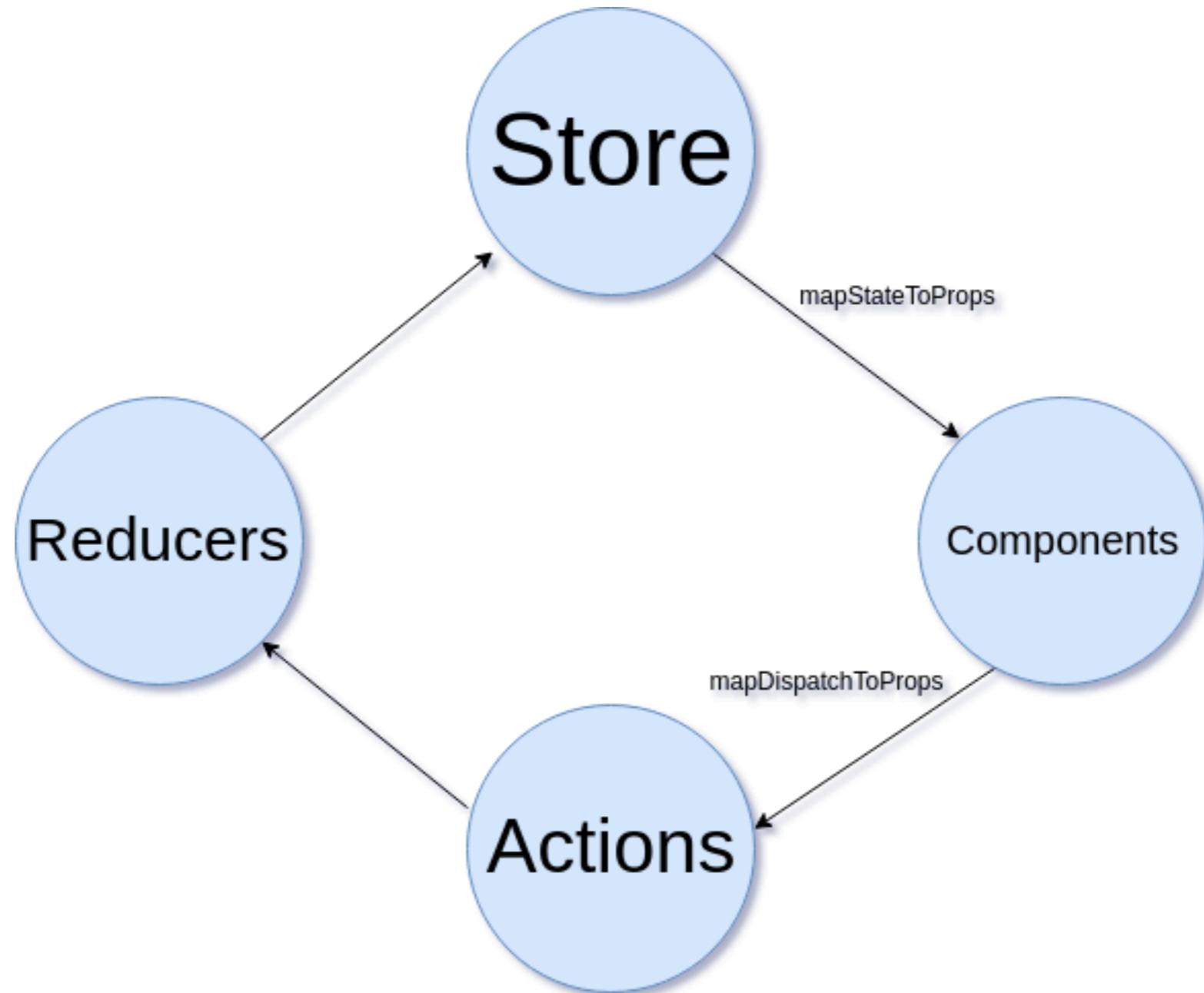
Redux



one source of truth

no dispatcher

immutable state



STORE

STORE

```
import { createStore } from "redux";  
  
const store = createStore(  
  reducer,  
  { counter: 0 }, /* optional */  
  enhancer, /* optional */  
);
```

ACTIONS

payloads of information that send data from your application to your store.

ACTIONS

```
store.dispatch( { type: "INCREMENT" } )
```

```
store.dispatch( { type: "DECREMENT" } )
```

```
store.dispatch( { type: "OTHER_ACTION", payloads } )
```

BEST PRACTICES

```
// actionTypes.js
```

```
export const INCREMENT = 'INCREMENT';
export const DECREMENT = 'DECREMENT';
```

REDUCER

**Specify how the application's state changes in response
to actions sent to the store**

REDUCER

```
const reducer = (state, action) => {  
  
  switch (action.type) {  
    case „ACTION_NAME“:  
      return {...}; // newCalculatedState  
    default:  
      return newState;  
  }  
};
```

REDUCER

```
const reducer = (state, action) => {
  switch (action.type) {
    case "INCREMENT":
      return { ...state, counter: state.counter + 1 };
    case "DECREMENT":
      return { ...state, counter: state.counter - 1 };
    default:
      return state;
  }
};
```

REDUCER

No side effects.

No API calls.

No mutations.

Just a calculation.

YOU SHOULD NEVER DO

Mutate its arguments;

Perform side effects like API calls and routing transitions;

Call non-pure functions, e.g. Date.now() or Math.random().

REACT-REDUX

Official React bindings for Redux

<PROVIDER>

makes the Redux store available to the rest of your app

<PROVIDER>

```
import { Provider } from "react-redux";
```

```
const App = () => (
  <Provider store={store}>
    <Counter />
  </Provider>
);
```

CONNECT()

connect your component to the store.

CONNECT()

```
function connect(  
  mapStateToProps?,  
  mapDispatchToProps?,  
  mergeProps?,  
  options?  
)
```

CONNECT()

```
mapStateToProps?: (state, ownProps?) => Object
```

```
const mapStateToProps = state => {
  return { counter: state.counter };
};
```

```
const mapStateToProps = (state, ownProps) => {
  return { item: state.items[ownProps.itemId] };
};
```

CONNECT()

```
mapDispatchToProps?: Object | (dispatch, ownProps?) => Object
```

```
const mapDispatchToProps = dispatch => {
  return {
    onIncrement: () => dispatch({ type: "INCREMENT" }),
    onDecrement: () => dispatch({ type: "DECREMENT" })
  };
};
```

CONNECT()

```
mapDispatchToProps?: Object | (dispatch, ownProps?) => Object
```

```
const onIncrement = () => ({ type: "INCREMENT" });
const onDecrement = () => ({ type: "DECREMENT" });
```

```
const mapDispatchToProps = {
  onIncrement,
  onDecrement
};
```

CONNECT()

```
mergeProps?: (stateProps, dispatchProps, ownProps) => Object
```

Returns default

```
{ ...ownProps, ...stateProps, ...dispatchProps }
```

CONNECT() EXAMPLES

```
connect(mapStateToProps, mapDispatchToProps)(Counter);
```

```
connect(mapStateToProps)(Counter);
```

```
connect(null, mapDispatchToProps)(Counter);
```

COMBINE REDUCERS

COMBINE REDUCERS

```
{  
  books: {  
    // ... array of books  
  },  
  authors: {  
    // ... array of authors  
  },  
  ....  
}
```

COMBINE REDUCERS

```
createStore(reducer, { books: [], authors: [] });
```

COMBINE REDUCERS

```
const reducer = (state, action) => {
  switch (action.type) {
    case ADD_BOOK:
      return {
        ...state,
        books: [...state.books, action.book]
      };
    case REMOVE_BOOK:
      return {
        ...state,
        books: state.books.filter(book => book.id !== action.id)
      };
    default:
      return state;
  }
};
```

COMBINE REDUCERS

```
const reducer = (state, action) => {
  switch (action.type) {
    case ADD_BOOK:
      return {
        ...state,
        books: [...state.books, action.book]
      };
    case REMOVE_BOOK:
      return {
        ...state,
        books: state.books.filter(book => book.id !== action.id)
      };
    case ADD_AUTHOR:
      return {
        ...state,
        authors: [...state.authors, action.author]
      };
    case REMOVE_AUTHOR:
      return {
        ...state,
        authors: state.authors.filter(author => author.id !== action.id)
      };
    default:
      return state;
  }
};
```

COMBINE REDUCERS

```
const reducer = (state, action) => {
  switch (action.type) {
    case ADD_BOOK:
      return {
        ...state,
        books: [...state.books, action.book]
      };
    case REMOVE_BOOK:
      return {
        ...state,
        books: state.books.filter(book => book.id !== action.id)
      };
    case ADD_AUTHOR:
      return {
        ...state,
        authors: [...state.authors, action.author]
      };
    case REMOVE_AUTHOR:
      return {
        ...state,
        authors: state.authors.filter(author => author.id !== action.id)
      };
    default:
      return state;
  }
};
```

COMBINE REDUCERS

```
const bookReducer = (state = [], action) => {
  switch (action.type) {
    case ADD_BOOK:
      return [...state, action.book];
    case REMOVE_BOOK:
      return state.filter(book => book.id !== action.id);
    default:
      return state;
  }
};
```

COMBINE REDUCERS

```
const authorReducer = (state = [], action) => {
  switch (action.type) {
    case ADD_AUTHOR:
      return [...state, action.author];

    case REMOVE_AUTHOR:
      return state.filter(author => author.id !== action.id);

    default:
      return state;
  }
};
```

COMBINE REDUCERS

```
createStore(reducer, { books: [], authors: [] });
```

COMBINE REDUCERS

```
import { createStore, combineReducers } from "redux";
```

```
const rootReducer = combineReducers({  
  books: bookReducer,  
  authors: authorReducer  
});
```

```
createStore(rootReducer);
```

MIDDLEWARE

„Redux middleware provides a third-party extension point between dispatching an action, and the moment it reaches the reducer.”

LOGGING

▼ ADD_TODO

```
 ❶ dispatching: Object {type: "ADD_TODO", text: "Use Redux"}  
  next state: ► Object {visibilityFilter: "SHOW_ALL", todos: Array[1]}
```

▼ ADD_TODO

```
 ❶ dispatching: Object {type: "ADD_TODO", text: "Learn about middleware"}  
  next state: ► Object {visibilityFilter: "SHOW_ALL", todos: Array[2]}
```

▼ COMPLETE_TODO

```
 ❶ dispatching: Object {type: "COMPLETE_TODO", index: 0}  
  next state: ► Object {visibilityFilter: "SHOW_ALL", todos: Array[2]}
```

▼ SET_VISIBILITY_FILTER

```
 ❶ dispatching: Object {type: "SET_VISIBILITY_FILTER", filter: "SHOW_COMPLETED"}  
  next state: ► Object {visibilityFilter: "SHOW_COMPLETED", todos: Array[2]}
```

MIDDLEWARE

Logging manually

```
const action = { type: "ADD_TODO", text: "Use Redux" };
```

```
console.log("dispatching", action);
store.dispatch(action);
console.log("next state", store.getState());
```

MIDDLEWARE

Applying the Middleware

```
const logger = store => next => action => {
  console.log("dispatching", action);
  let result = next(action);
  console.log("next state", store.getState());
  return result;
};
```

```
const store = createStore(rootReducer, applyMiddleware(logger));
```

REDUX DEVTOOLS EXTENSION

<http://extension.remotedev.io/>

REDUCER

No side effects.

No API calls.

No mutations.

Just a calculation.

REDUX THUNK

Thunk middleware for Redux.

WHAT'S A THUNK?!

```
// calculation of 1 + 2 is immediate
// x === 3
let x = 1 + 2;
```

```
// calculation of 1 + 2 is delayed
// foo can be called later to perform the calculation
// foo is a thunk!
let foo = () => 1 + 2;
```

REDUX THUNK

```
npm install redux-thunk
```

```
import { createStore, applyMiddleware } from 'redux';
import thunk from 'redux-thunk';
import rootReducer from './reducers/index';
```

```
// Note: this API requires redux@>=3.1.0
const store = createStore(
  rootReducer,
  applyMiddleware(thunk)
);
```

REDUX THUNK

```
export function addBook(book) {  
  return { type: ADD_BOOK, book };  
}
```

```
export function addBook(book) {  
  return function(dispatch) {  
    setTimeout(() => dispatch({ type: ADD_BOOK, book }), 5000);  
  };  
}
```

REDUX THUNK

```
export function getDataRequested() {  
  return {  
    type: 'GET_DATA_REQUESTED'  
  };  
}  
};
```

```
export function getDataDone(data) {  
  return {  
    type: 'GET_DATA_DONE',  
    payload: data  
  };  
};
```

```
export function getDataFailed(error) {  
  return {  
    type: 'GET_DATA_FAILED',  
    payload: error  
  };  
};
```

```
export function getData() {  
  return dispatch => {  
    // set state to "loading"  
    dispatch(getDataRequested());  
  
    fetch('https://api')  
      .then(response => response.json())  
      .then(data => {  
        // set state for success  
        dispatch(getDataDone(data));  
      })  
      .catch(error => {  
        // set state for error  
        dispatch(getDataFailed(error));  
      })  
  };  
};
```

SELECTORS

SELECTORS

```
const DisplayBooks = ({books}) => books.map(book => <h2>{book.name}</h2>);
```

```
const mapStateToProps = state => {
  return {
    books: state.items.books
  }
}
```

SELECTORS

```
const getBooks = state => state.items.books;
```

```
const DisplayBooks = ({books}) => books.map(book => <h2>{book.name}</h2>);
```

```
const mapStateToProps = state => {
  return {
    books: getBooks(state)
  }
}
```

SELECTORS

Reselect

<https://github.com/reduxjs/reselect>

REACT-REDUX

HOOKS

useStore()

```
const store = useStore()
```

```
console.log('State: ', store.getState())
```

useSelector()

```
const Counter = () => {
  const count = useSelector((state) => state);

  return <h2>Counter: {count}</h2>;
};
```

useDispatch()

```
const Counter = () => {
  const dispatch = useDispatch();
  const increment = () => dispatch({ type: "INCREMENT" });
  return <button onClick={increment}>increment</button>;
};
```



Redux Toolkit

The official, opinionated, batteries-included toolset for efficient Redux development

[Get Started](#)

Simple

Includes utilities to simplify common use cases like **store setup**, **creating reducers**, **immutable update logic**, and more.



Opinionated

Provides **good defaults for store setup out of the box**, and includes **the most commonly used Redux addons built-in**.



Powerful

Takes inspiration from libraries like Immer and Autodux to let you write "**mutative immutable update logic**", and even **create entire "slices" of state automatically**.



Effective

Lets you focus on the core logic your app needs, so you can **do more work with less code**.

Other Libraries from the Redux Team

[Redux ↗](#)

A predictable state container for JavaScript applications

[React-Redux ↗](#)

Official React bindings for Redux

THANKS