Before we begin...

- Open up these slides:
  - https://bit.ly/2I52Ej1

React

GENERAL ASSEMBLY

Learning Objectives

- **Understand** React and its history
- **Identify** and **understand** Components in React
- Effectively **use** props in React
- Effectively **use** state in React
- **Use** event listeners in React
- **Identify** common lifecycle methods and their purpose
- **Use** lifecycle methods

Agenda

- React
  - Props
  - Events
  - State
  - Lifecycle Methods

A quick review

- Classes
- Webpack
- React

Project Time!

React

GENERAL ASSEMBLY

What is React?

- A JavaScript library for building user interfaces
- Include it before your own code:
  - Script
  - NPM
- Built by Facebook
- It is a Front-end JavaScript Framework
  - It changes the way we write code
  - Other frameworks: Vue, Angular, Backbone etc.

The React Ecosystem

- You can go straight ahead and write React code, though it is often written with other technologies:
  - Webpack
  - Babel
  - React (and ReactDOM)
  - JSX (JSX in Depth)
  - ...

What is JSX?

- A syntax extension to JavaScript (not a part of ECMAScript or anything though)
- It's almost like HTML in JavaScript
- JSX produces React "elements"
- React doesn't require JSX, but it is suggested
- You can embed expressions in JSX (like interpolation)
- It just makes our lives easier!

What is JSX?

```
<h1 id="hello">Hello World</h

// Compiles to...

React.createElement(
    "h1",
    { id: "hello" },
    "Hello World"
);
```

What is JSX?

```
<img src="http://fillmurray.com/400/400" id="bill">

// Compiles to...

React.createElement(
     "img",
     { src: "http://fillmurray.com/400/400", id: "bill
     null
);
```

What does React teach?

- Declarative
- Unidirectional Data Flow
- Composition
- Explicit Mutations
- Remember that it is just JavaScript

Imperative vs. Declarative

- Imperative is telling a computer how to do something
- Declarative is telling a computer what to do
  - It relies on the magic
  - Declarative is all about abstraction
  - React Components are always declarative

Advantages

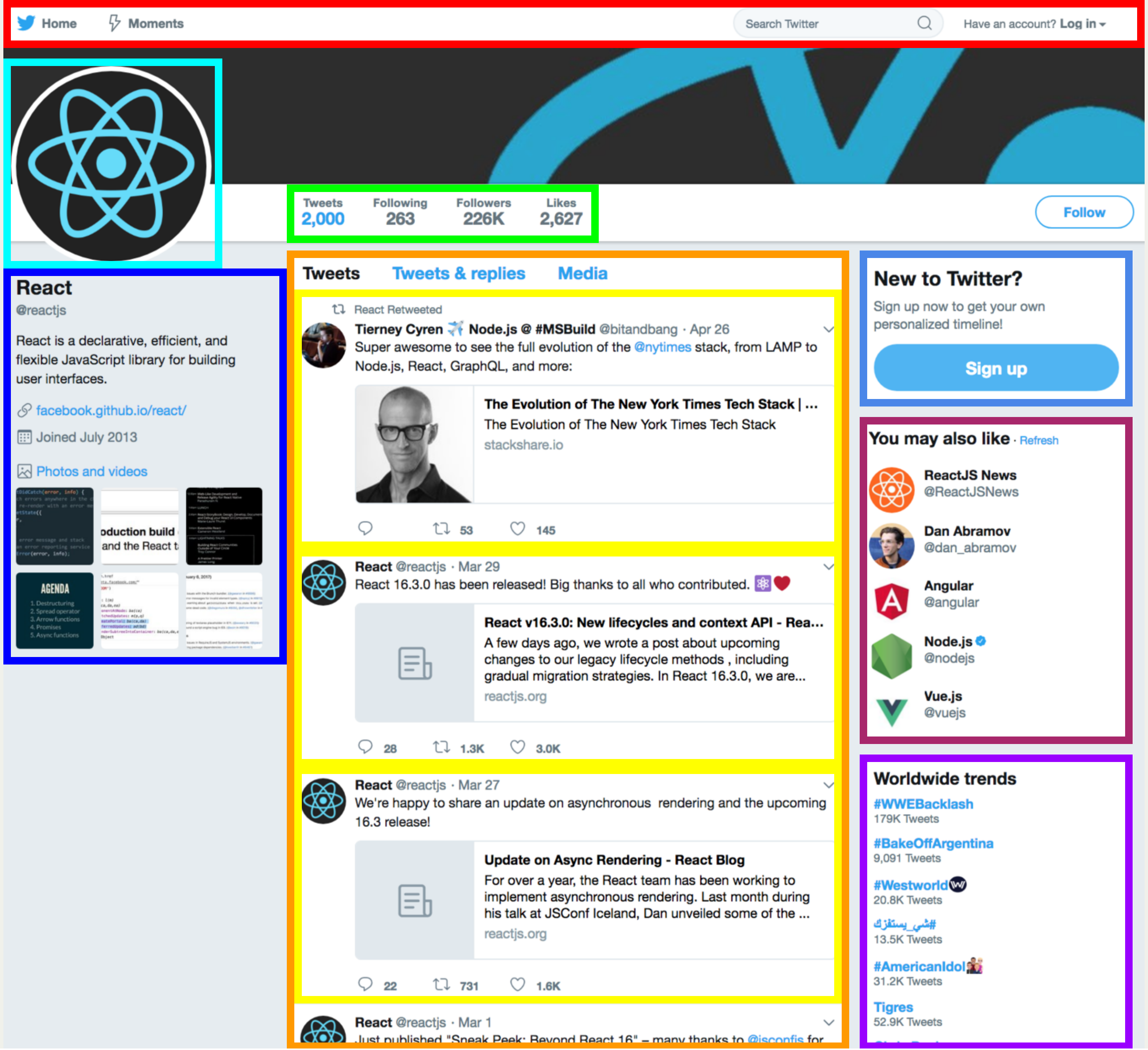
- Really easy to see the structure of your app
- Very good at managing state
- Performant
- Virtual DOM
- Data Binding
- Easy to test
- Isomorphic (can be rendered server-side)
- Agnostic (you can use it with all sorts of other libraries as React is just the view layer)
- Learn once, write everywhere

Disadvantages

- A big library
- Lots of magic
- It is just the view layer
- Typically requires a transformation step
- A steep learning curve
- It changes incredibly regularly

Working with Components

- A component is one of the fundamental parts of React
- Each component represents a small part of a page
  - And each component manages their own state
- You compose your app with lots of different components



Home Moments Search Twitter Have an account? Log in

Tweets 2,000 | Following 263 | Followers 226K | Likes 2,627

Follow

React
@reactjs

React is a declarative, efficient, and flexible JavaScript library for building user interfaces.

facebook.github.io/react/

Joined July 2013

Photos and videos

Tweets | Tweets & replies | Media

React Retweeted
Tierney Cyren ✈️ Node.js @ #MSBuild @bitandbang · Apr 26
Super awesome to see the full evolution of the @nytimes stack, from LAMP to Node.js, React, GraphQL, and more:

The Evolution of The New York Times Tech Stack | ...
The Evolution of The New York Times Tech Stack
stackshare.io

53 145

React @reactjs · Mar 29
React 16.3.0 has been released! Big thanks to all who contributed. ⚛❤

React v16.3.0: New lifecycles and context API - Rea...
A few days ago, we wrote a post about upcoming changes to our legacy lifecycle methods , including gradual migration strategies. In React 16.3.0, we are...
reactjs.org

28 1.3K 3.0K

React @reactjs · Mar 27
We're happy to share an update on asynchronous rendering and the upcoming 16.3 release!

Update on Async Rendering - React Blog
For over a year, the React team has been working to implement asynchronous rendering. Last month during his talk at JSConf Iceland, Dan unveiled some of the ...
reactjs.org

22 731 1.6K

React @reactjs · Mar 1
Just published "Sneak Peek: Beyond React 16" – many thanks to @jsconfis for

New to Twitter?
Sign up now to get your own personalized timeline!
Sign up

You may also like · Refresh

ReactJS News @ReactJSNews
Dan Abramov @dan_abramov
Angular @angular
Node.js @nodejs
Vue.js @vuejs

Worldwide trends

#WWEBacklash
179K Tweets
#BakeOffArgentina
9,091 Tweets
#Westworld
20.8K Tweets
#شي_يستفزك
13.5K Tweets
#AmericanIdol
31.2K Tweets
Tigres
52.9K Tweets

Let's get into it!

Our Hello World!

```
const React = require("react");
const ReactDOM = require("react-dom

ReactDOM.render(
    <h1>Hello World</h1>,
    document.querySelector("#app")
);
```

Our First Component

```
const React = require("react");
const ReactDOM = require("react-dom");

class HelloWorld extends React.Component
  render() {
    return <h1>Hello World</h1>;
  }
}

ReactDOM.render(
    <HelloWorld />,
    document.getElementById("root")
);
```

Interpolation with JSX

```
const React = require("react");
const ReactDOM = require("react-dom");

class FavNumber extends React.Component {
  render() {
    const favNumber = 42;
    return <h1>Favourite Number: {favNumber}!</h1>;
  }
}

ReactDOM.render(<FavNumber />, document.getElementById("roo
```

Props

What are props?

- Props are very similar to parameters with normal functions (they stand for properties)
- They are read-only (immutable)
- You can provide any data you want as props
    - In a class component, they are accessed through ***this.props*** (e.g. this.props.name)
    - The props are also provided to the constructor as the first parameter

Passing Data as Props

```
const React = require("react");
const ReactDOM = require("react-dom");

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

ReactDOM.render(
    <Hello name="Bill" />,
    document.getElementById("root")
);
```

@threequal

Passing Data as Props

```
const React = require("react");
const ReactDOM = require("react-dom");

class UserProfile extends React.Component {
  render() {
    const { firstName, lastName, username, email, imageURL } = this.p
    return (
      <div>
        <h1>{username}</h1>
        <img src={imageURL} />
        <p>Name: {firstName} {lastName}</p>
        <p>Email: {email}</p>
      </div>
    );
  }
}

ReactDOM.render(
  <UserProfile
    firstName="Bill"
    lastName="Murray"
    username="billmuzza"
    email="bill@ga.co"
    imageURL="http://fillmurray.com/200/200"
  />,
  document.querySelector("#root")
);
```

@threequal

Props & Composition

Component Composition

```
const React = require("react");
const ReactDOM = require("react-dom");

class TodoItem extends React.Component {
  render() {
    return <li>{this.props.task}</li>;
  }
}

class TodoList extends React.Component {
  render() {
    return (
      <ul>
        <TodoItem task="Task One" />
        <TodoItem task="Task Two" />
        <TodoItem task="Task Three" />
      </ul>
    );
  }
}

ReactDOM.render(<TodoList />, document.getElementById("roo
```

@threequal

Component Composition

```
const React = require("react");
const ReactDOM = require("react-dom");

class StudentDetails extends React.Component {
  render() {
    return <li>{this.props.name}</li>;
  }
}

class StudentLists extends React.Component {
  render() {
    const students = ["studentOne", "studentTwo", "studentThre
    const studentElems = students.map(student => {
      return <StudentDetails name={student} />;
    });
    return (
      <div>
        <h1>Students</h1>
        <ul>{studentElems}</ul>
      </div>
    );
  }
}

ReactDOM.render(<StudentLists />, document.querySelector("#roo
```

@threequal

Events

Events

```
const React = require("react");
const ReactDOM = require("react-dom");

class Hello extends React.Component {
  render() {
    const { name } = this.props;
    return (
        <h1 onClick={() => alert(`${name} was clicked!`)}>
            Hello {name}!
        </h1>
    )
  }
}

ReactDOM.render(<Hello name="Bill" />, document.getElementById("roo
```

Events

```
const React = require("react");
const ReactDOM = require("react-dom");

class Hello extends React.Component {
  handleClick() {
    alert(`${this.props.name} was clicked!`);
  }
  render() {
    const { name } = this.props;
    return <h1 onClick={() => this.handleClick()}>Hello {name}!</h
  }
}

ReactDOM.render(<Hello name="Bill" />, document.getElementById("roo
```

@threequal

More Events

```
const React = require("react");
const ReactDOM = require("react-dom");

class Hello extends React.Component {
  handleMouseEnter() {
    alert(`The mouse moved over ${this.props.name}`);
  }
  handleRightClick(event) {
    event.preventDefault();
    alert(`${this.props.name} was right-clicked!`);
  }
  handleClick() {
    alert(`${this.props.name} was clicked!`);
  }
  render() {
    const { name } = this.props;
    return (
      <h1
        onMouseEnter={() => this.handleMouseEnter()}
        onContextMenu={e => this.handleRightClick(e)}
        onClick={() => this.handleClick()}
      >
        Hello {name}!
      </h1>
    );
  }
}

ReactDOM.render(<Hello name="Bill" />, document.getElementById("ro
```

@threequal

State

What is state?

- State holds information about the current component (trapped within an instance of the component, can't be accessed from anywhere else)
- It is mutable and is where you add anything that changes (e.g. data and AJAX requests, user inputs)
- It is an object. We can add whatever data we want
- It is initialised in the *constructor*
- It is changed with the *setState* method
    - As soon as you call *setState*, it will re-run the *render method*. It handles the hard work for you!

Click Counter

```
const React = require("react");
const ReactDOM = require("react-dom");

class ClickCounter extends React.Component {
  constructor(props) {
    super(props);
    this.state = { clicks: 0 };
    this.handleClick = this.handleClick.bind(this);
  }
  handleClick() {
    this.setState({ clicks: this.state.clicks + 1 });
  }
  render() {
    return (
      <div>
        <button onClick={this.handleClick}>Click Me!</button>
        <p>Number of clicks: {this.state.clicks}</p>
      </div>
    );
  }
}

ReactDOM.render(<ClickCounter />, document.getElementById("roo
```

@threequal

Date Printer

```
const React = require("react");
const ReactDOM = require("react-dom");

class DatePrinter extends React.Component {
  constructor(props) {
    super(props);
    this.state = { date: new Date() };
    setInterval(() => {
      this.setState({
        date: new Date()
      });
    }, 100);
  }
  render() {
    const { date } = this.state;
    return (
      <p>
        {date.toLocaleDateString()} {date.toLocaleTimeString()
      </p>
    );
  }
}

ReactDOM.render(<DatePrinter />, document.getElementById("root
```

@threequal

Name Printer

```
const React = require("react");
const ReactDOM = require("react-dom");

class InputPrinter extends React.Component {
  constructor(props) {
    super(props);
    this.state = { value: "" };
    this.handleChange = this.handleChange.bind(this);
  }
  handleChange(event) {
    this.setState({ value: event.target.value });
  }
  render() {
    return (
      <div>
        <input
          type="text"
          onChange={this.handleChange}
          value={this.state.value}
          placeholder="Type here!"
        />
        <p>
          You have typed: <strong>{this.state.value}</strong>
        </p>
      </div>
    );
  }
}

ReactDOM.render(<InputPrinter />, document.getElementById("root
```

@threequal

Sign Up Page

```
const React = require("react");
const ReactDOM = require("react-dom");

class SignUpPage extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      name: "",
      email: "",
      imageURL: ""
    };
    this.createAccount = this.createAccount.bind(this);
    this.updateName = this.updateName.bind(this);
    this.updateEmail = this.updateEmail.bind(this);
    this.updateImage = this.updateImage.bind(this);
  }
  createAccount(e) {
    e.preventDefault();
    console.log(this.state);
```

@threequal

Conditional Rendering

Login / Logout

```
const React = require("react");
const ReactDOM = require("react-dom");

class LoginControl extends React.Component {
  constructor(props) {
    super(props);
    this.state = { loggedIn: true };
    this.logIn = this.logIn.bind(this);
    this.logOut = this.logOut.bind(this);
  }
  logOut() {
    this.setState({ loggedIn: false });
  }
  logIn() {
    this.setState({ loggedIn: true });
  }
  render() {
    if (this.state.loggedIn) {
      return <button onClick={this.logOut}>Log Out!</button>;
    }
    return <button onClick={this.logIn}>Log In!</button>;
  }
}

ReactDOM.render(<LoginControl />, document.getElementById("roo
```

@threequal

Ron Swanson Jokes

```
const React = require("react");
const ReactDOM = require("react-dom");

class RonSwansonJokes extends React.Component {
  constructor(props) {
    super(props);
    this.state = { data: null, error: null };
    fetch("//ron-swanson-quotes.herokuapp.com/v2/quotes")
      .then(r => r.json())
      .then(data => {
        this.setState({ data: data[0] });
      });
  }
  render() {
    const text = this.state.data || "Loading...";
    return (
      <div>
        <h3>{text}</h3>
      </div>
    );
  }
}

ReactDOM.render(<RonSwansonJokes />, document.getElementById("root
```

@threequal

Ron Swanson Jokes (More)

```
const React = require("react");
const ReactDOM = require("react-dom");

class RonSwansonJokes extends React.Component {
  constructor(props) {
    super(props);
    this.state = { data: null };
    this.getJoke = this.getJoke.bind(this);
    this.handleClick = this.handleClick.bind(this);
    this.getJoke();
  }
  handleClick() {
    this.setState({ data: null });
    this.getJoke();
  }
  getJoke() {
    fetch("//ron-swanson-quotes.herokuapp.com/v2/quotes")
      .then(r => r.json())
      .then(data => {
        this.setState({ data: data[0] });
      });
  }
  render() {
    const text = this.state.data || "Loading...";
    return (
      <div>
        <h3>{text}</h3>
        <button onClick={this.handleClick}>Get another!</button>
      </div>
    );
  }
}

ReactDOM.render(<RonSwansonJokes />, document.getElementById("root
```

Lifting State Up

Todo App

```
const React = require("react");
const ReactDOM = require("react-dom");

class TodoView extends React.Component {
  render() {
    return <li>{this.props.todo}</li>;
  }
}

class TodoInputView extends React.Component {
  constructor(props) {
    super(props);
    this.state = { text: "" };
    this.handleChange = this.handleChange.bind(this);
    this.handleSubmit = this.handleSubmit.bind(this);
  }
```

Edit Profile Page

```
import React, { Component } from "react";
import { render } from "react-dom";

class EditProfileForm extends Component {
  render() {
    const {
      name,
      email,
      imageURL,
      updateName,
      updateEmail,
      updateImage
    } = this.props;
    return (
      <form>
        <input
```

@threequal

Lifecycle Methods

What are lifecycle methods?

- Lifecycle methods represent the life of a React component
  - From the birth (instantiation)
  - To the death (unmounting)
- They signify important moments
  - When it gets added to the page, when it gets updated, when state changes etc.

What are lifecycle methods?

- They are broken down into four main phases:
  - Initialisation
  - Mounting
  - Updating
  - Unmounting

“Render Phase”

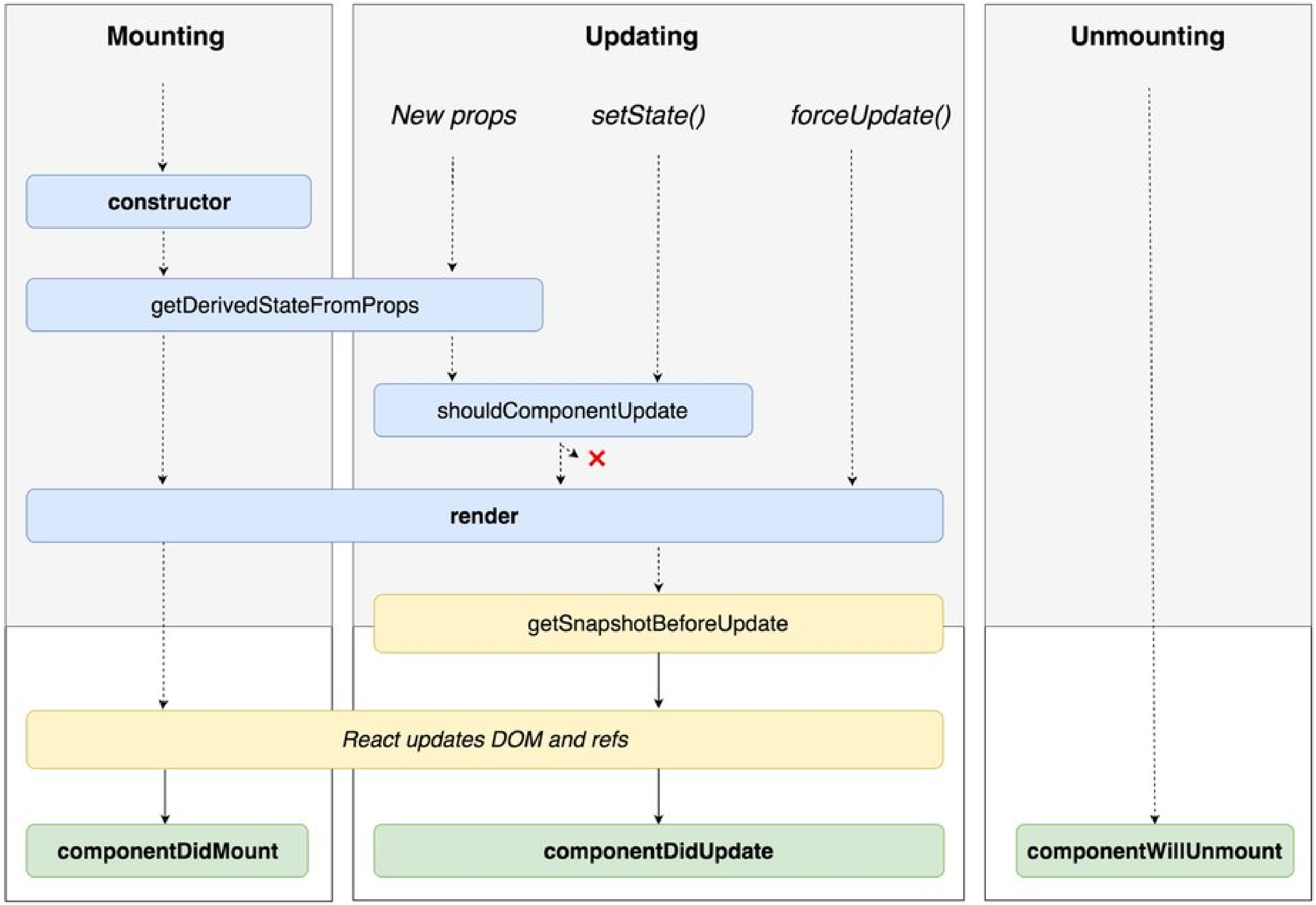
Pure and has no side effects. May be paused, aborted or restarted by React.

“Pre-Commit Phase”

Can read the DOM.

“Commit Phase”

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



@threequal

Resources

- ReactJS Website
- Egghead: Beginner's Guide to React
- React for Beginners
- Codecademy
- Cabin: Learn React
- React Armory: Learn React
- The Road to Learn React
- SurviveJS: React

Homework

- It's Project Time!
- Do some React Tutorials
- Add Babel and Webpack to previous homework!
- Read up on ES2015
-
  - Translate some of your previous code into it!
- Finish all exercises from class
- Upload your homework to GitHub
- Prepare for next lesson

Homework (Extra)

- Go through some tasks in Exercism
- Get into JavaScript30
- Go through The Modern JavaScript Tutorial
- Read Exploring ES6
- Read Eloquent JavaScript
- Read Speaking JavaScript

What's next?

- More React
  - Events
  - State
  - Lifecycle Methods

Questions?

Feedback time!

Lesson 15: ***React***

https://ga.co/js05syd

Thanks!