LECTURE 11

INTRODUCTION TO REACT NATIVE

DANIEL RYS
JAN VÁCLAVÍK

OVERVIEW

- React
- JSX
- React Native
- Compare React Native and Ionic
- Installation
- Props vs. state
- Life cycles (initial, update)
- Styles, Flexbox

REACT IN GENERAL

WHAT IS REACT

- **JS library for building UI**
- **No MVC** framework like Angular, Ember, etc.
- Released 2013 by Facebook
- Goes against some practices devs were used to

WHAT IS REACT

Only the View layer

- **JS library for building UI**
- **No MVC** framework like Angular, Ember, etc.
- Released 2013 by Facebook
- Goes against some practices devs were used to

WHAT IS REACT

Only the View layer

- **JS library for building UI**
- **No MVC** framework like Angular, Ember, etc.
- Released 2013 by Facebook
- Goes against some practices devs were used to

Template X Logic separation

REACT FEATURES

COMPOSABLE COMPONENTS

- Building **reusable components**
- **Separation of concerns**
- Good for large apps

ONE-WAY DATA BINDING

- Value can be updated only from the component itself

```
render() {
    return <input value={this.state.value} onChange={this.handleChange} />
}
handleChange(e) {
    this.setState({value: e.target.value});
}
```

VIRTUAL DOM

- HTML diff which generates minimal necessary DOM change
- Refreshes only changed elements
- (No DOM in React Native)

http://bit.ly/1TjoJup

JSX

- Briefer **Javascript syntax extension** looks like XML
- Compiled to JS
- Every HTML element is a “function” (component)
- React doesn’t use HTML templates
- Other frameworks include **code into HTML**,
  React includes **“HTML” into the code**

COMPILING JSX TO JS

```
1 var Nav;
2 // Input (JSX):
3 var app = <Nav color="blue" />;
4 // Output (JS):
5 var app = React.createElement(Nav, {color:"blue"});
```

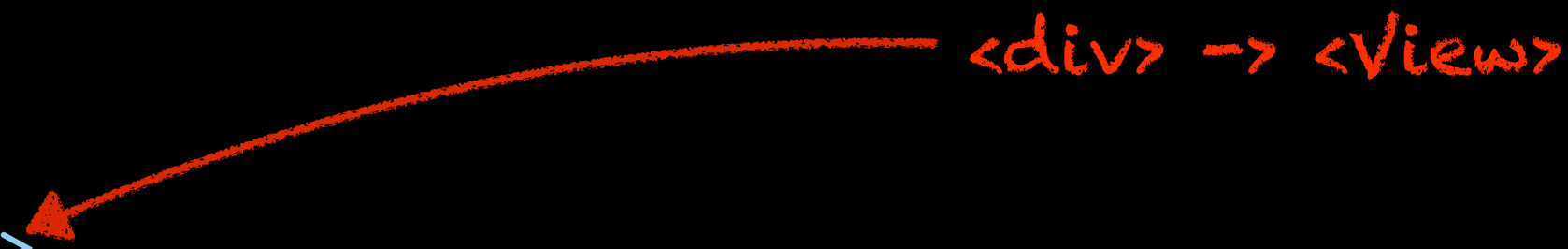
MORE COMPLICATED JSX IN REACT

```
1  <div>
2      <a onClick={() =>
           this.showMovieDetail(movie)}>
3          <div>
4              <img src={"http://image.tmdb.org/t/p/
                   w500"+poster_path}/>
5              <div>
6                  <span>{title}</span>
7                  <span>{vote_average} / 10</span>
8              </div>
9          </div>
10     </a>
11     <div style={styles.separator}/>
12 </div>
```

MORE COMPLICATED JSX IN REACT NATIVE

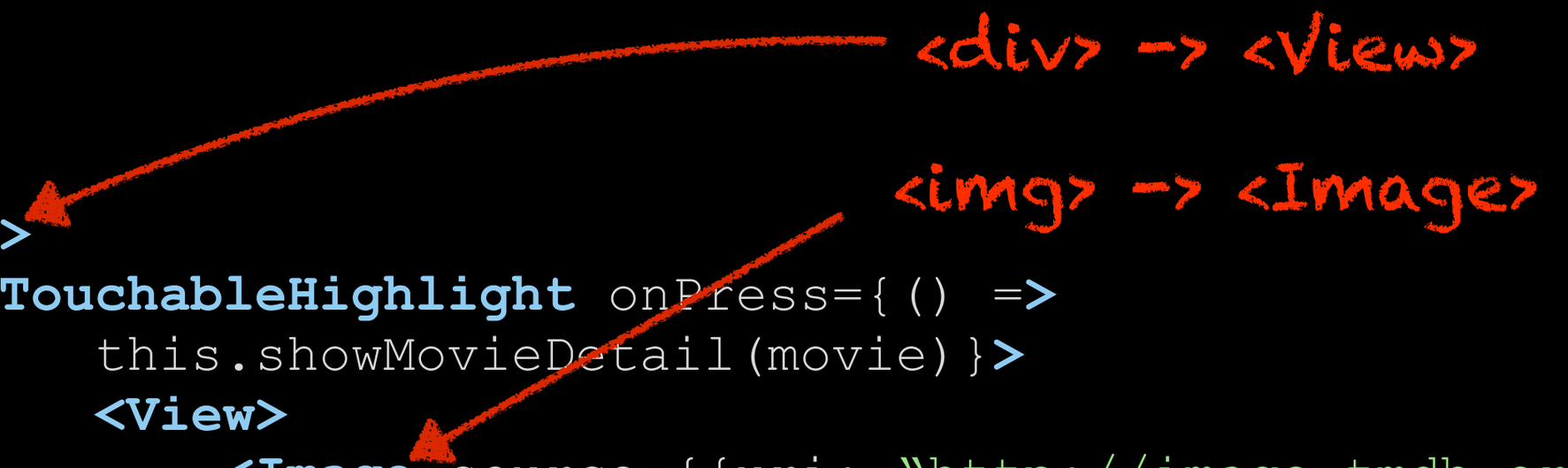
```
1 <View>
2     <TouchableHighlight onPress={() =>
          this.showMovieDetail(movie)}>
3         <View>
4             <Image source={{uri: "http://image.tmdb.org/t/p/
                  w500"+poster_path}}/>
5             <View>
6                 <Text>{title}</Text>
7                 <Text>{vote_average} / 10</Text>
8             </View>
9         </View>
10    </TouchableHighlight>
11    <View style={styles.separator}/>
12 </View>
```

MORE COMPLICATED JSX IN REACT NATIVE




```
1  <View>
2      <TouchableHighlight onPress={() =>
           this.showMovieDetail(movie)}>
3          <View>
4              <Image source={{uri: "http://image.tmdb.org/t/p/
                   w500"+poster_path}}/>
5              <View>
6                  <Text>{title}</Text>
7                  <Text>{vote_average} / 10</Text>
8              </View>
9          </View>
10     </TouchableHighlight>
11     <View style={styles.separator}/>
12 </View>
```

MORE COMPLICATED JSX IN REACT NATIVE




```
1  <View>
2      <TouchableHighlight onPress={() =>
           this.showMovieDetail(movie)}>
3          <View>
4              <Image source={{uri: "http://image.tmdb.org/t/p/
                   w500"+poster_path}}/>
5              <View>
6                  <Text>{title}</Text>
7                  <Text>{vote_average} / 10</Text>
8              </View>
9          </View>
10     </TouchableHighlight>
11     <View style={styles.separator}/>
12 </View>
```

MORE COMPLICATED JSX IN REACT NATIVE


```
1  <View>
2      <TouchableHighlight onPress={() =>
           this.showMovieDetail(movie)}>
3          <View>
4              <Image source={{uri: "http://image.tmdb.org/t/p/
                   w500"+poster_path}}/>
5              <View>
6                  <Text>{title}</Text>
7                  <Text>{vote_average} / 10</Text>
8              </View>
9          </View>
10     </TouchableHighlight>
11     <View style={styles.separator}/>
12 </View>
```

JSX: HTML TAGS VS REACT COMPONENTS

```
var myDivElement = <div className="foo" />;
ReactDOM.render(myDivElement, document.getElementById('example'));
```

JSX: HTML TAGS VS REACT COMPONENTS

```
var myDivElement = <div className="foo" />;
ReactDOM.render(myDivElement, document.getElementById('example'));
```

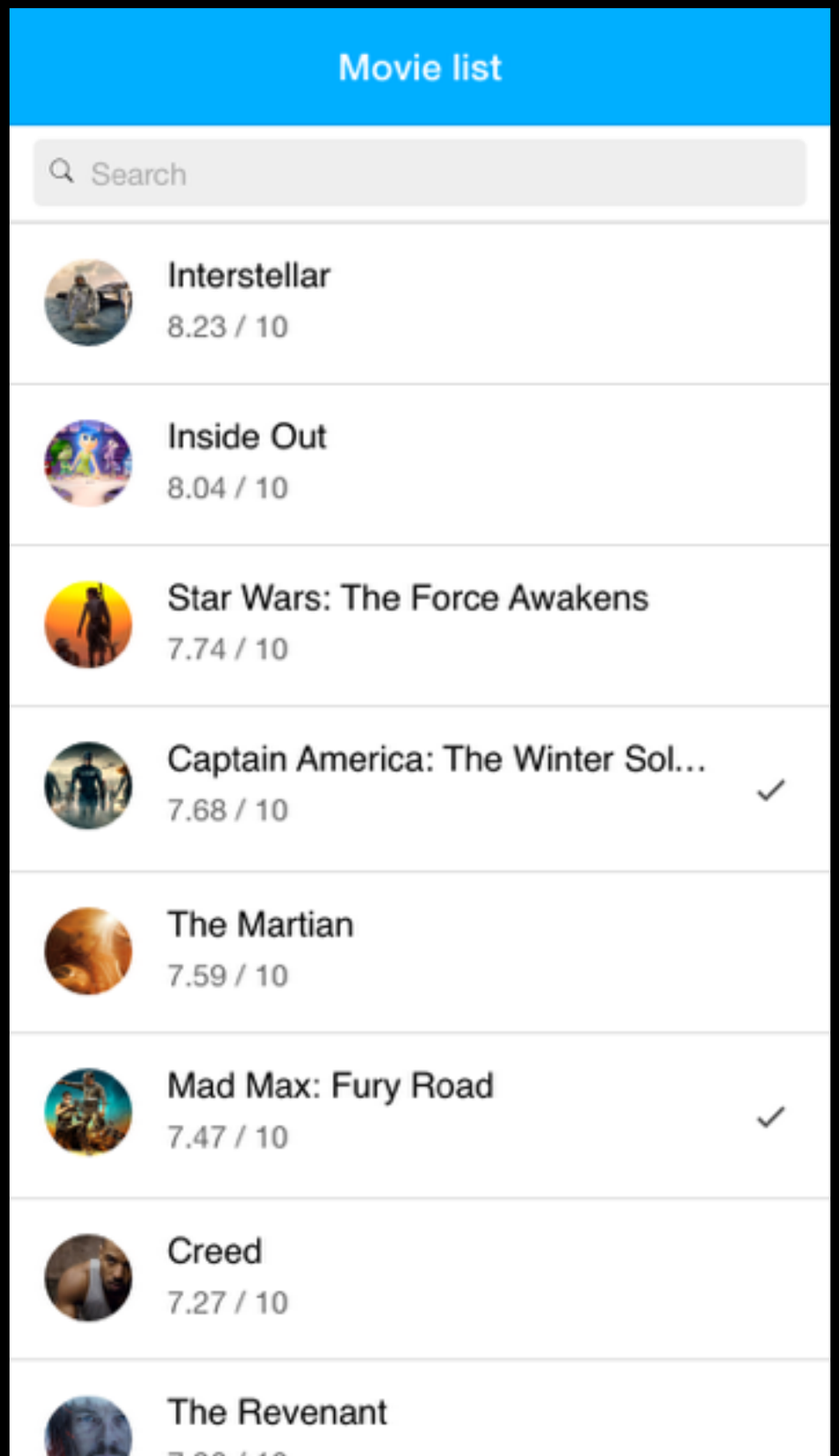
```
var MyComponent = React.createClass({/*...*/});
var myElement = <MyComponent someProperty={true} />;
ReactDOM.render(myElement, document.getElementById('example'));
```

SERVER RENDERING

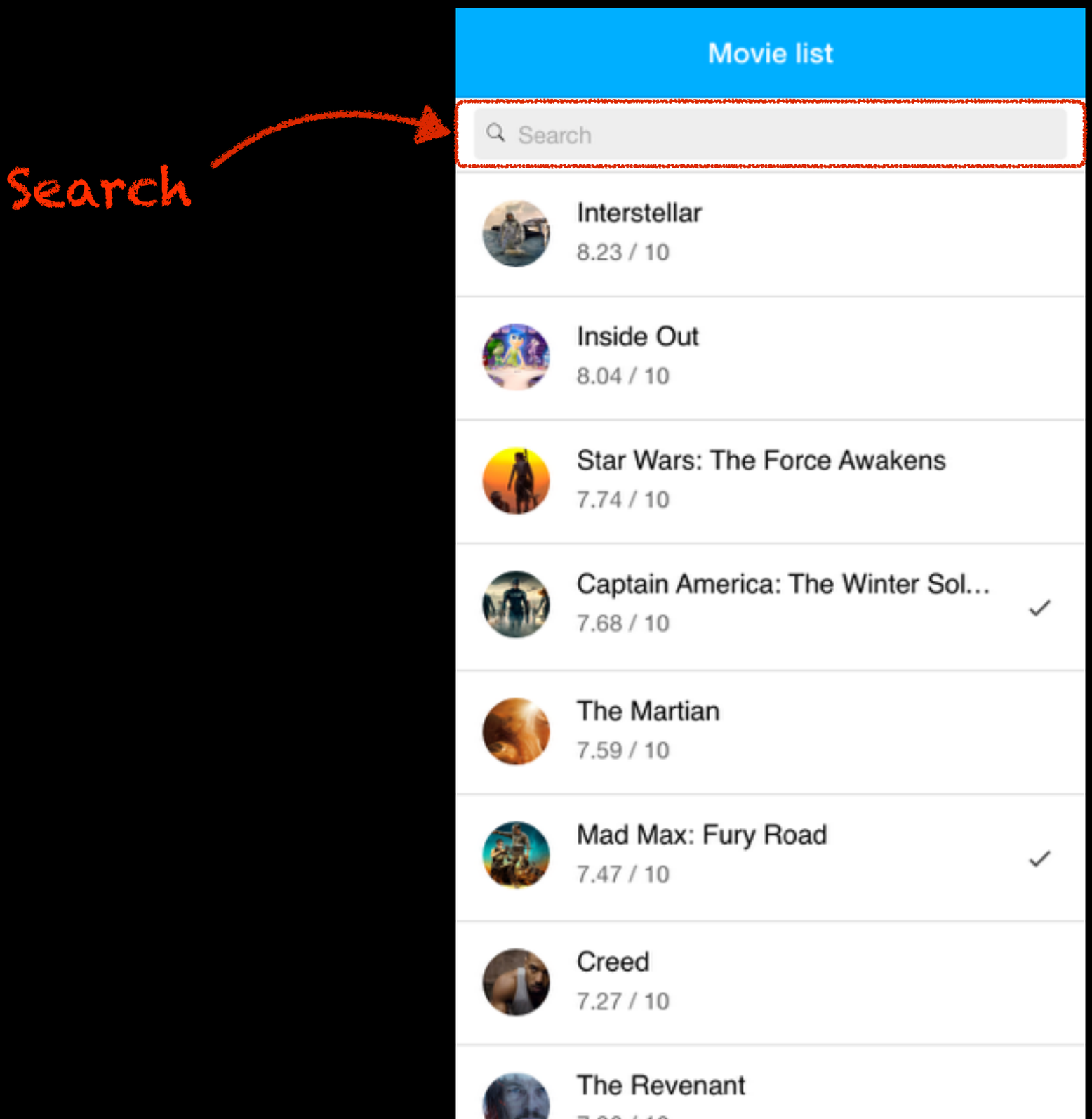
- Code sharing
- Better performance
- SEO friendly

EVERYTHING IS A COMPONENT

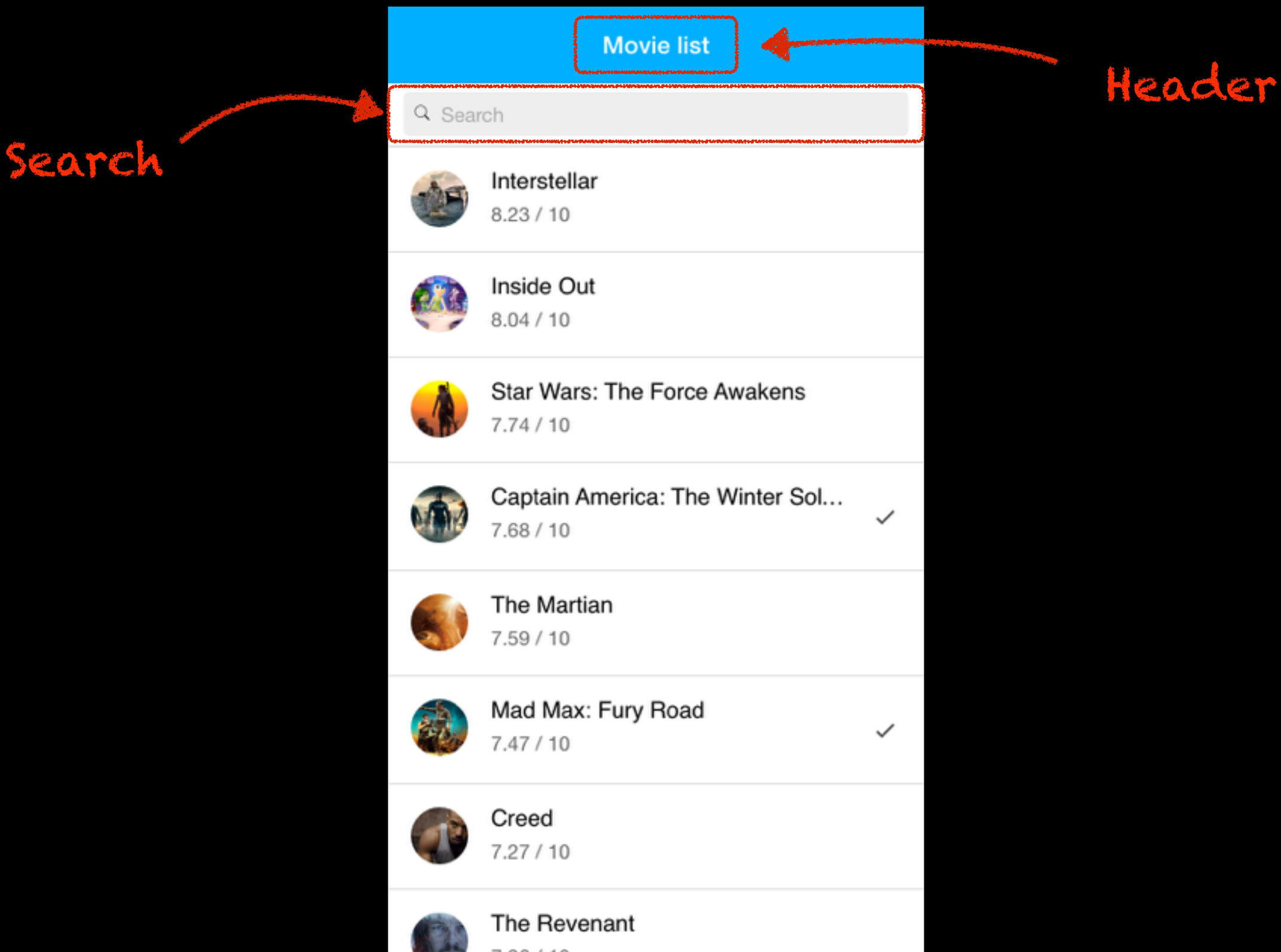
EVERYTHING IS A COMPONENT



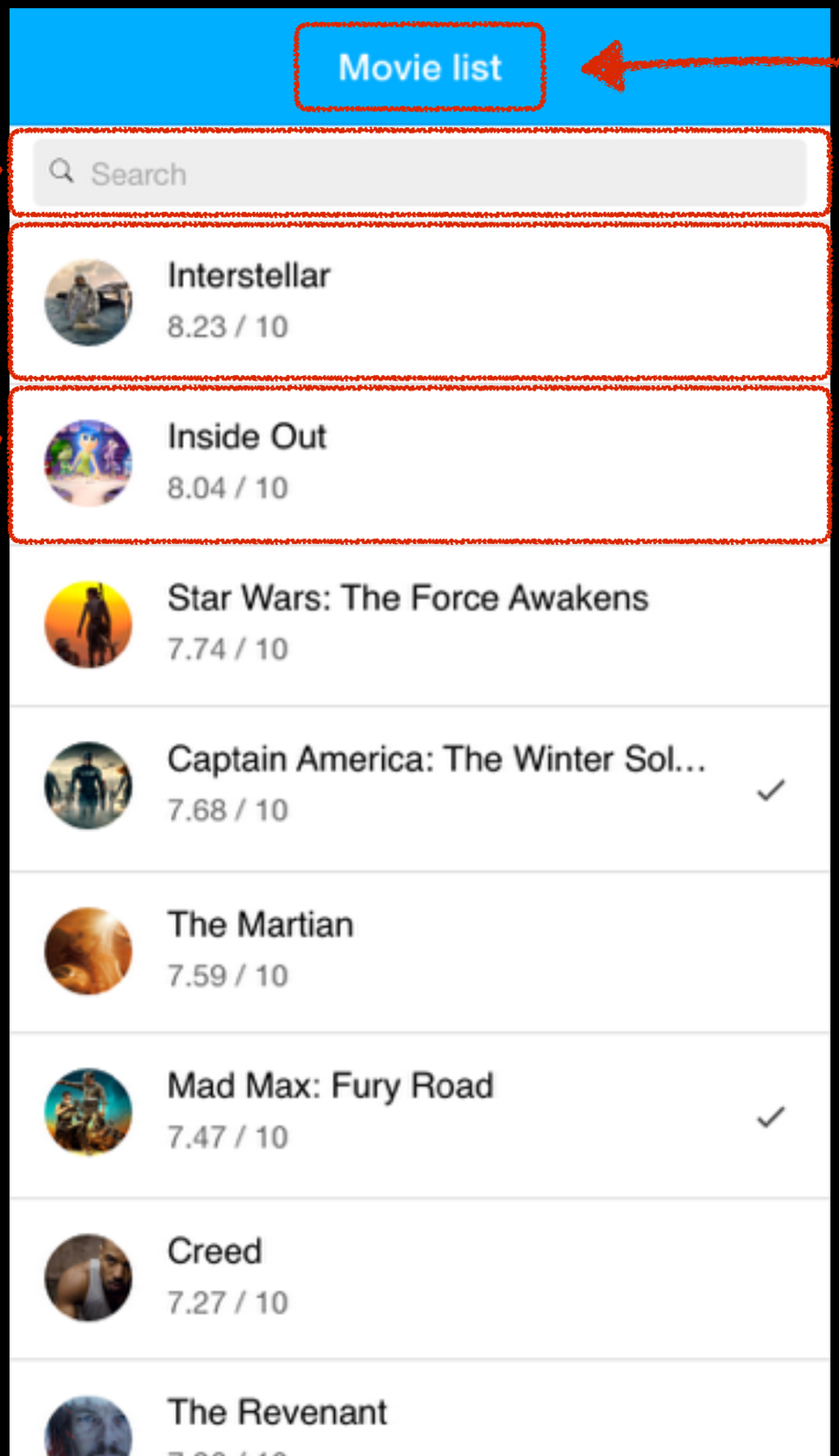
EVERYTHING IS A COMPONENT



EVERYTHING IS A COMPONENT



EVERYTHING IS A COMPONENT

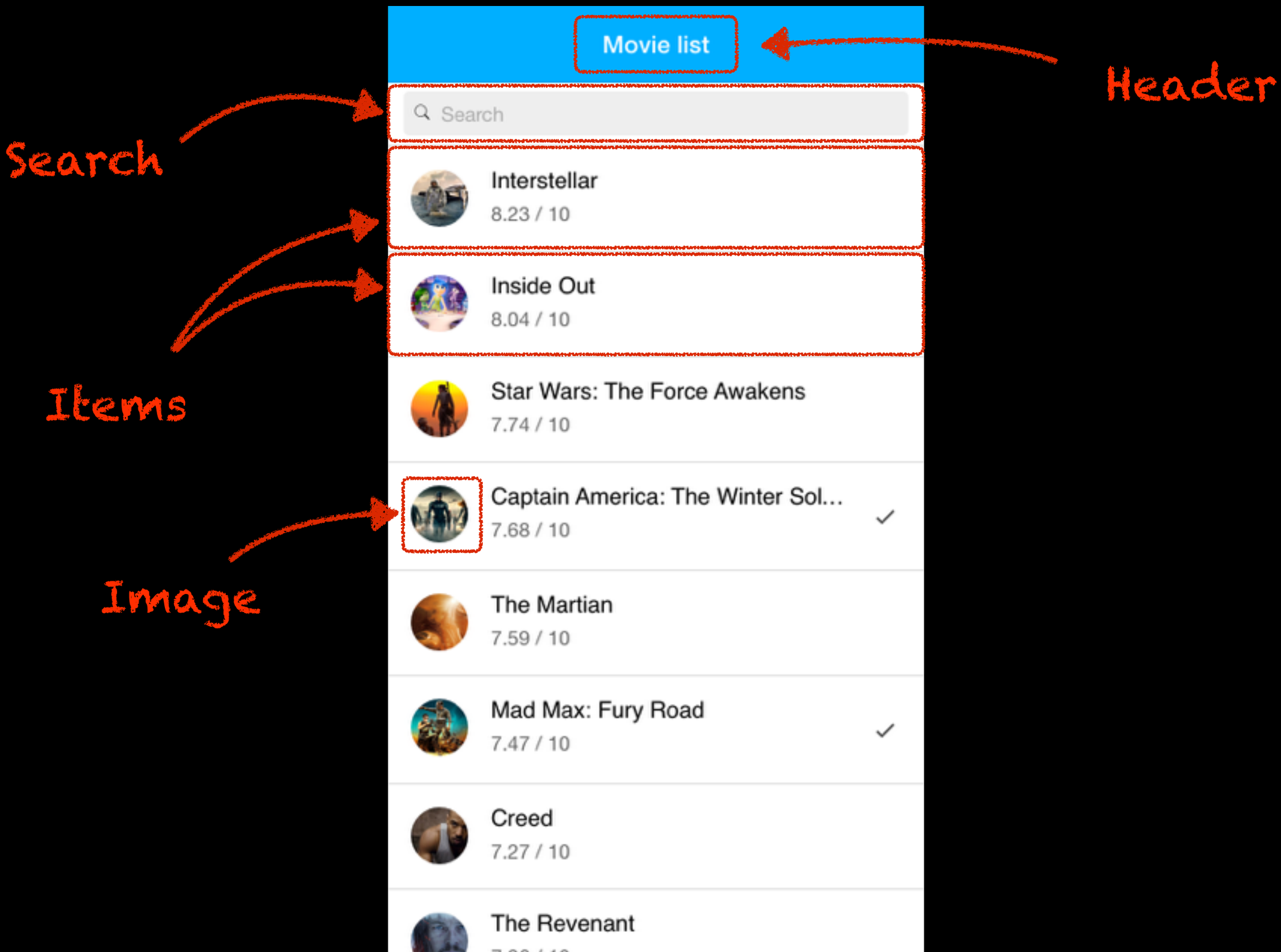


Header

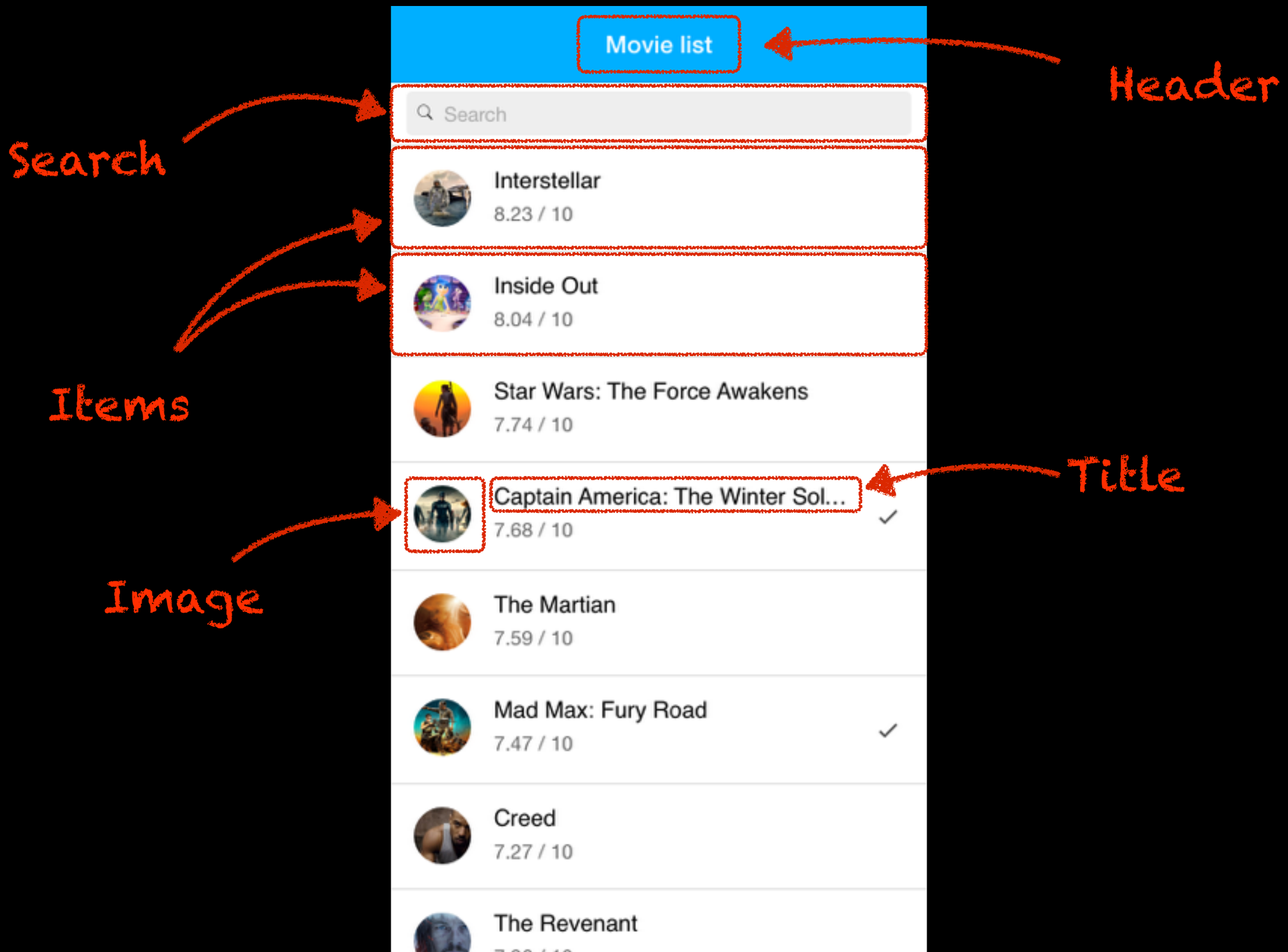
Search

Items

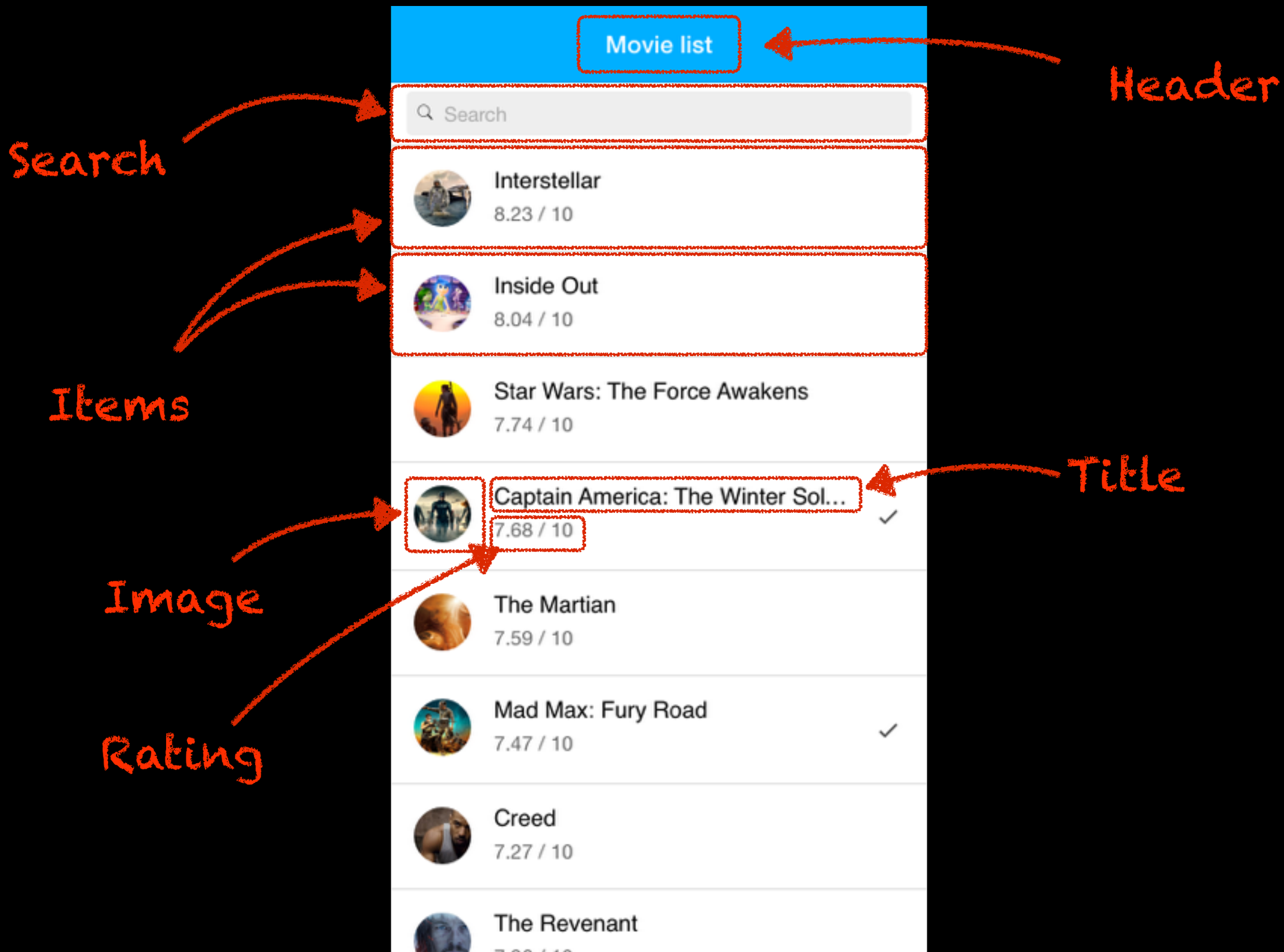
EVERYTHING IS A COMPONENT



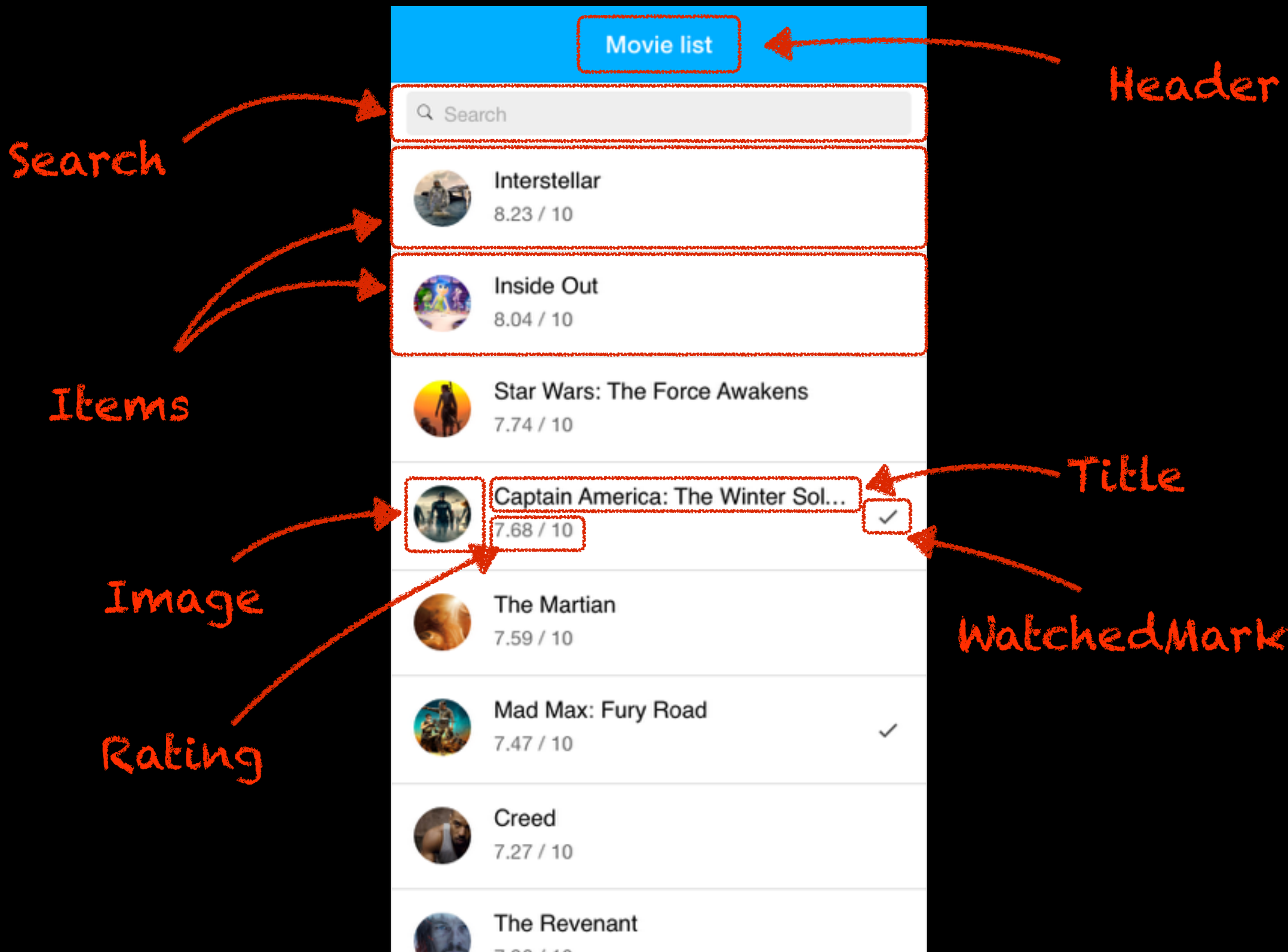
EVERYTHING IS A COMPONENT



EVERYTHING IS A COMPONENT



EVERYTHING IS A COMPONENT



WHAT IS REACT NATIVE?

- **Framework for building native mobile apps using React**

- Supports **Android and iOS**
  (plan Windows universal platform in the future)

- Based on Javascript and ReactJS

WHAT IS REACT NATIVE?

- **Framework for building native mobile apps using React**

- Supports **Android and iOS**
  (plan Windows universal platform in the future)

- Based on Javascript and R

WHY REACT NATIVE

SAME SKILL SET ON EACH PLATFORM

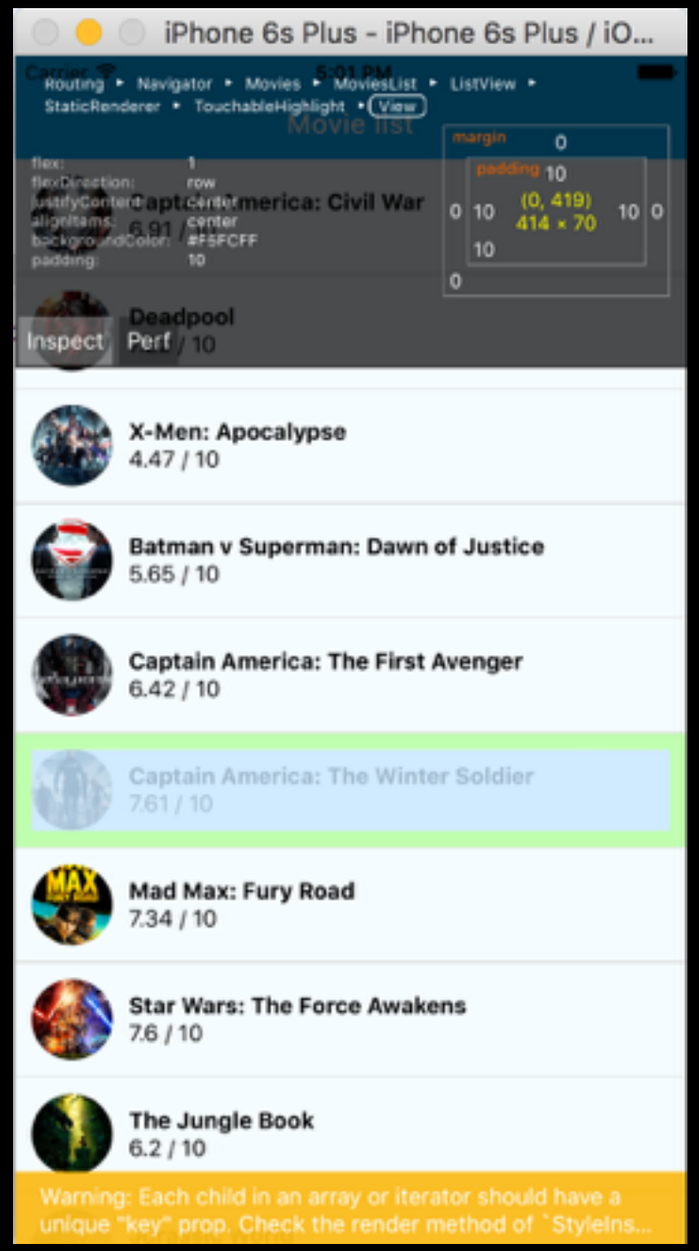
- ~70–80 % reuse of app code
- Javascript (React)
- Using “CSS” and “HTML” (looks like that)
- One team can develop app for more platforms

JAVASCRIPT WRAPS AROUND OBJECTIVE-C/JAVA

- We can add custom native components
- Many native components for frequently used features (mainly for iOS)

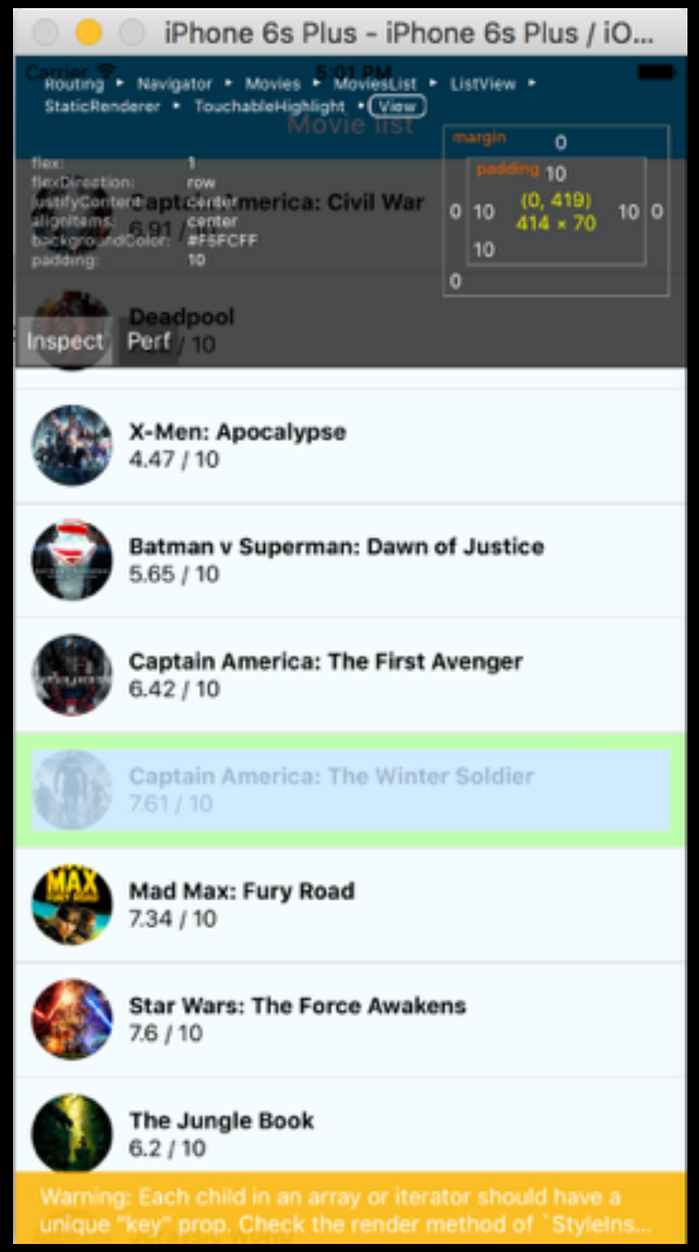
EASY TO DEBUG

- Can use Chrome dev tools
- Or Inspector in the app
- Performance monitor



EASY TO DEBUG

- Can use Chrome dev tools
- Or Inspector in the app
- Performance monitor



PRODUCES NATIVE APPS

- Native feeling without additional work
- Components are always up to date with current OS version

STRONG COMMUNITY

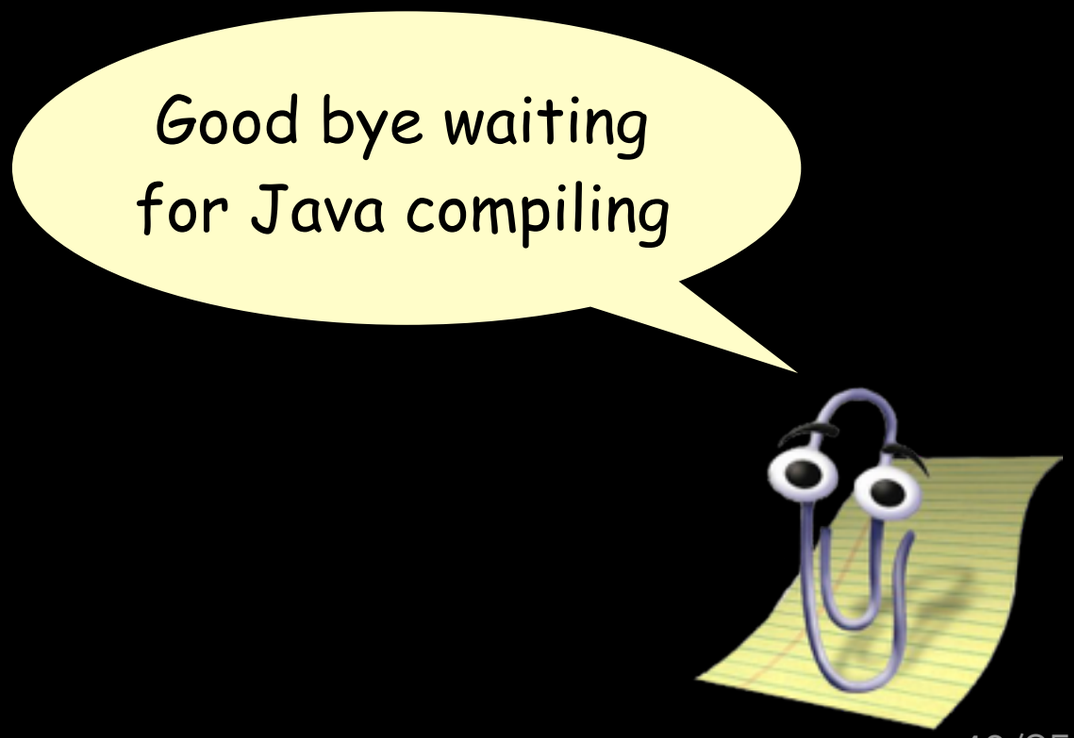
- 32 242 stars on GitHub (May 2016)
- Developed by Facebook
- Many contributors

HOT RELOAD

- **Updates code instantly** during development
- **No page reload**, modifying updated component only
- Keep component state
- Superb for styling the app

HOT RELOAD

- **Updates code instantly** during development
- **No page reload**, modifying updated component only
- Keep component state
- Superb for styling the app



JAVASCRIPT IN SEPARATE THREAD

- React Native runs the **JavaScript code in a separate thread**
- **User interface does not block**
- Animations should be silky and smooth

DOWNSIDES OF REACT NATIVE

- **Lack of documentation**, examples, tutorials
- **Poor Android support**
- Some components are missing (especially on Android)
- Memory issues with PNG images on Android
- **Slower development than hybrid**
- Styling components from the scratch
- Hard to identify problems during install and development
  (you will probably notice that)

DOWNSIDES OF REACT NATIVE

- **Lack of documentation**, examples, tutorials
- **Poor Android support**
- Some components are missing (especially on Android)
- Memory issues with PNG images on Android
- **Slower development than hybrid**
- Styling components from the scratch
- Hard to identify problems during install and development (you will probably notice that)

COMPARE REACT NATIVE AND IONIC

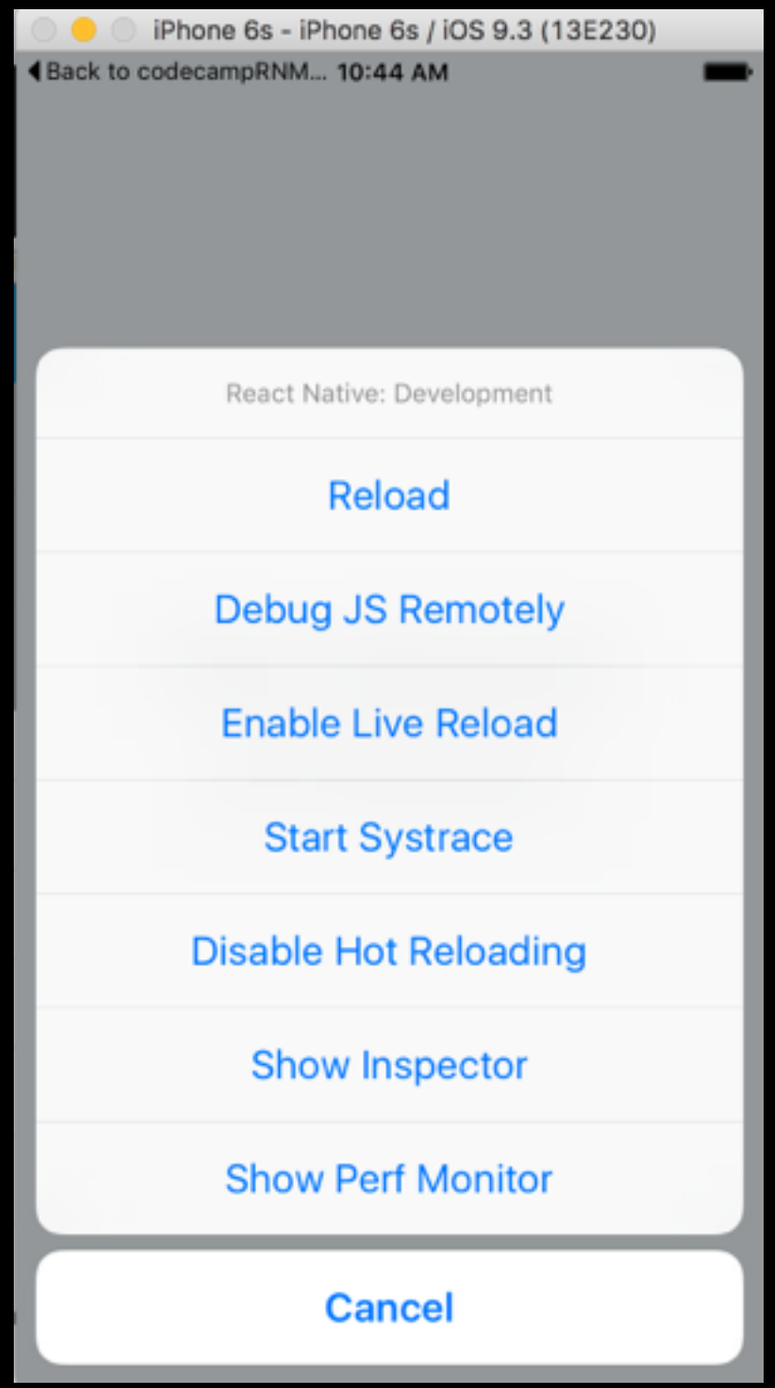
	ionic	React Native
Philosophy	**Write once**, run anywhere	**Learn once**, write anywhere
Mobile apps	Hybrid	Native
Platforms	iOS, Android, WP10, mobile web, (desktop)	iOS, Android
Ready to deploy	Yes	Not yet
Initial complexity	Low	High

INSTALL REACT NATIVE

```
$ npm install -g react-native-cli
$ react-native init AwesomeProject
```

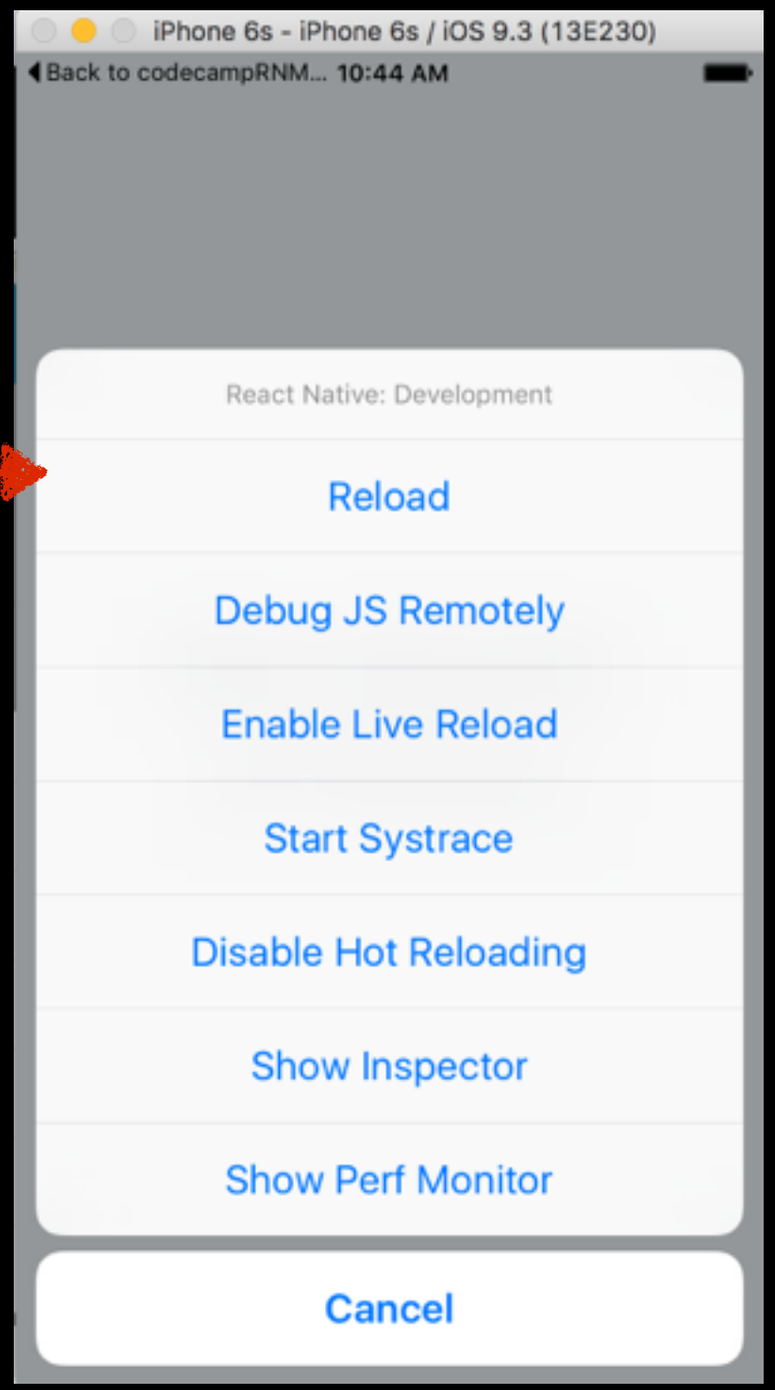
DEBUGGING

- **CMD + D**: Development menu
- **CMD + R**: Reload
- Live reload, hot reload
- Performance monitor



DEBUGGING

- **CMD + D**: Development menu
- **CMD + R**: Reload
- Live reload, hot reload
- Performance monitor

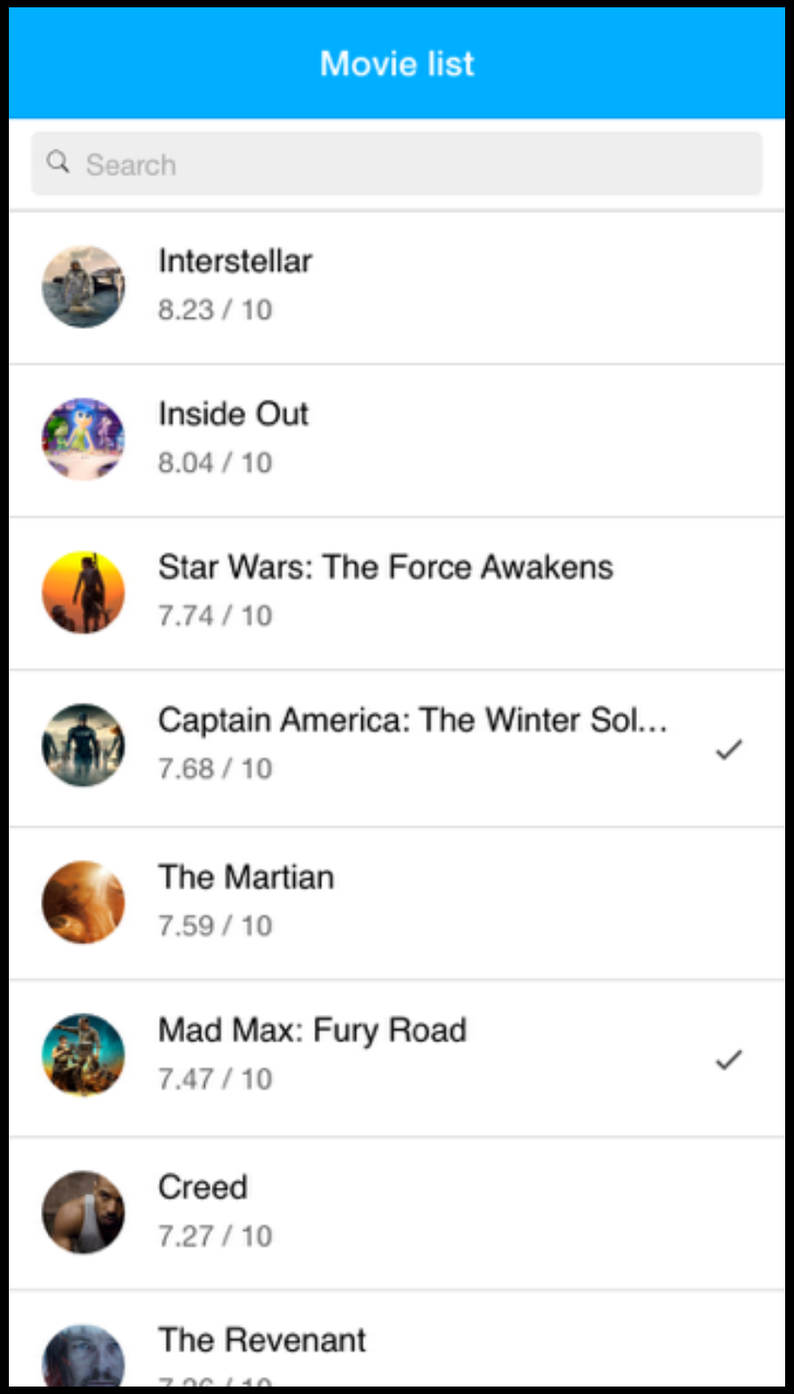


PROPS VS. STATE

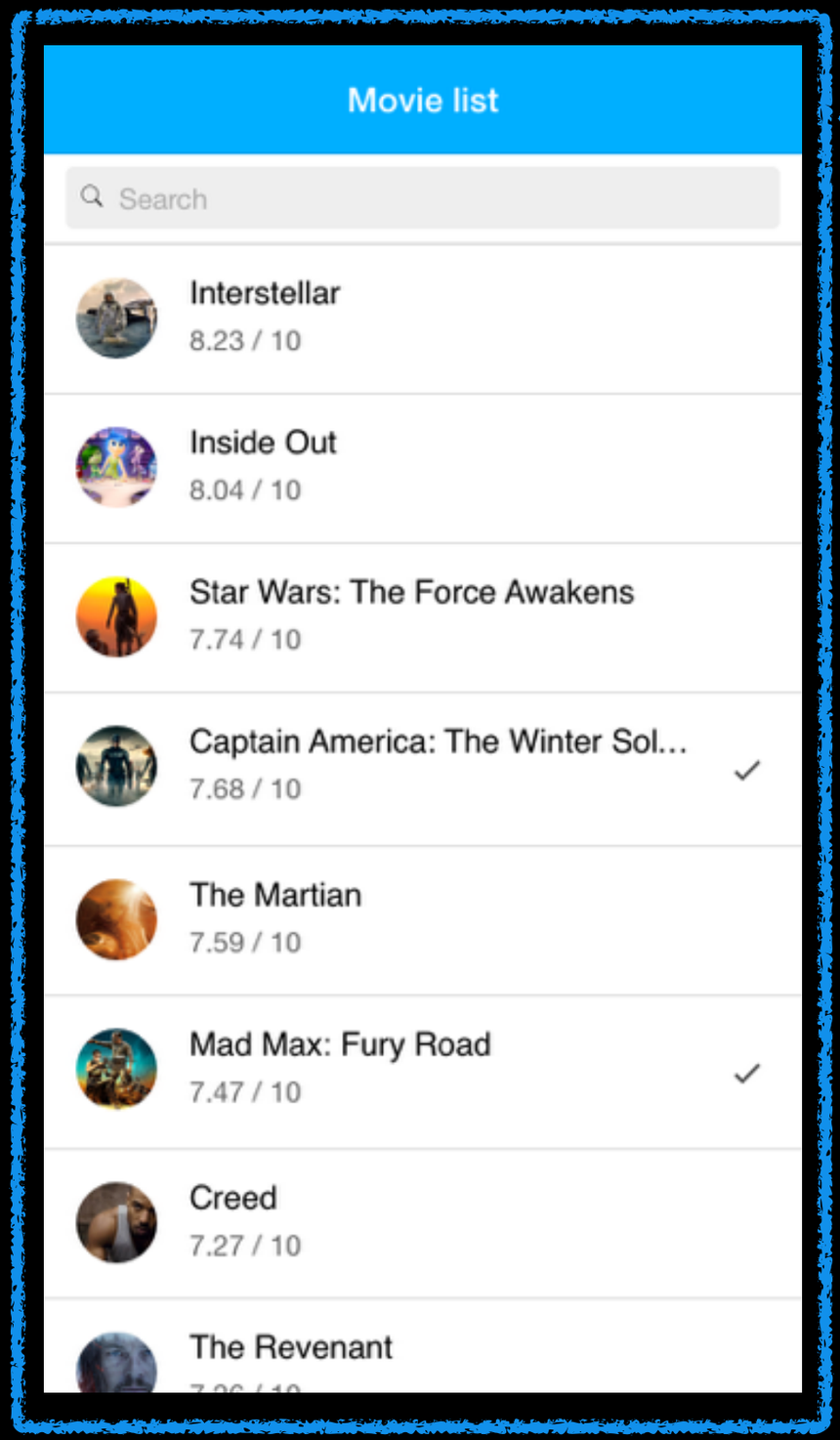
PROPS VS. STATE

- **STATE**: Local component state for storing data
- **PROPS**: For reading data passed between components

EXAMPLE – 2 COMPONENTS

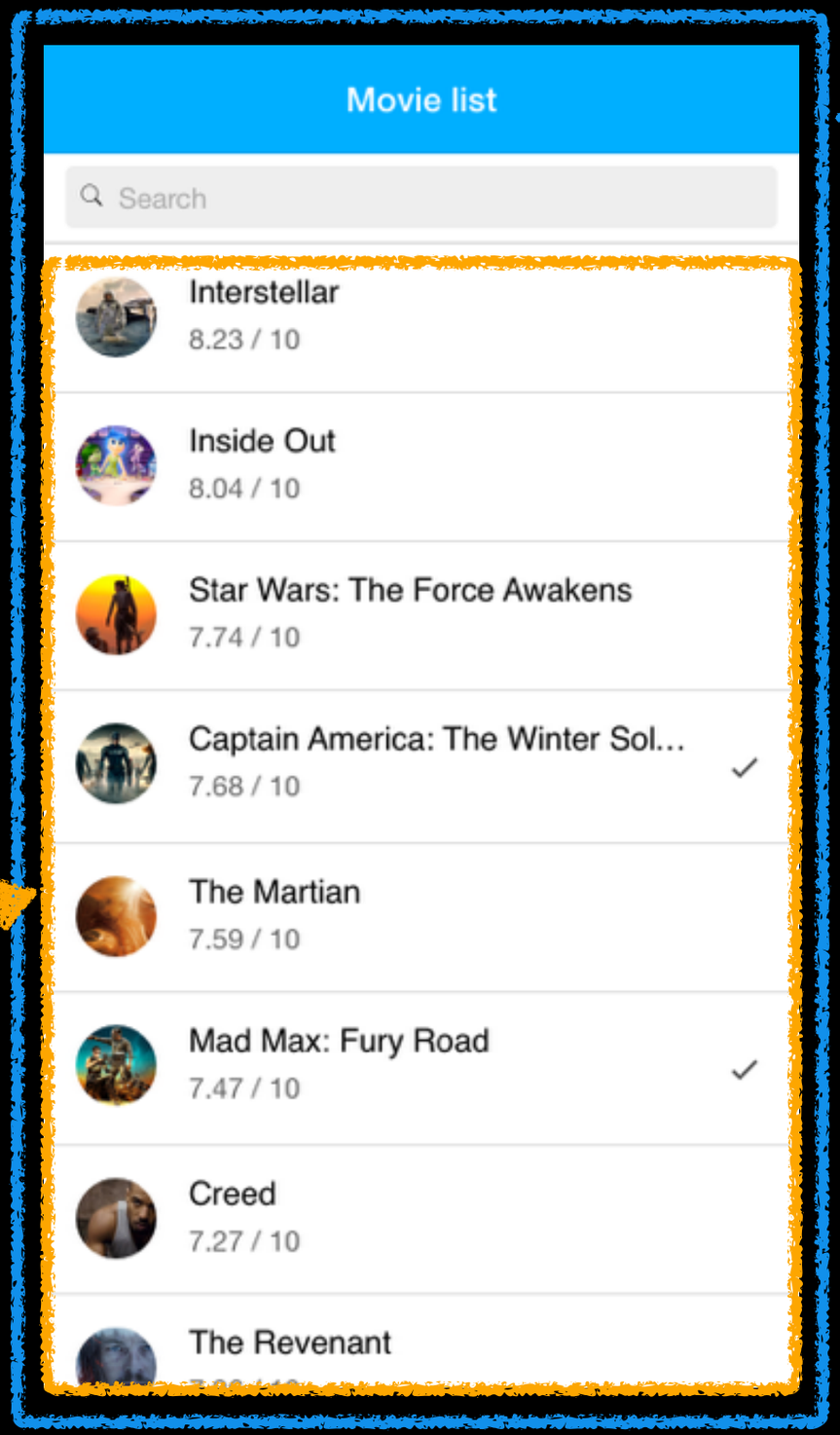


EXAMPLE – 2 COMPONENTS



Movies (parent)

EXAMPLE – 2 COMPONENTS



Movies (parent)

MovieList (child)

Component Movies (parent)

```
this.setState({
    data: getDataFromApi()
})
```

Component Movies (parent)

```
this.setState({
    data: getDataFromApi()
})
```

Component Movie List (child)

Receive **SOME DATA** for ListView

Component Movies (parent)

```
this.setState({
    data: getDataFromApi()
})
```

```
<MovieList />
```

Component Movie List (child)

Receive **SOME DATA** for ListView

Component Movies (parent)

```
this.setState({
    data: getDataFromApi()
})
```

Call component MovieList

```
<MovieList />
```

Component Movie List (child)

Receive **SOME DATA** for ListView

Component Movies (parent)

```
this.setState({
    data: getDataFromApi()
})
```

```
<MovieList movies="this.state.data" />
```

Component Movie List (child)

Receive **SOME DATA** for ListView

Component Movies (parent)

```
this.setState({
    data: getDataFromApi()
})
```

Pass data to MovieList component

```
<MovieList movies="this.state.data" />
```

Component Movie List (child)

Receive **SOME DATA** for ListView

Component Movies (parent)

```
this.setState({
    data: getDataFromApi()
})
```

```
<MovieList movies="this.state.data" />
```


Component Movies List (child)

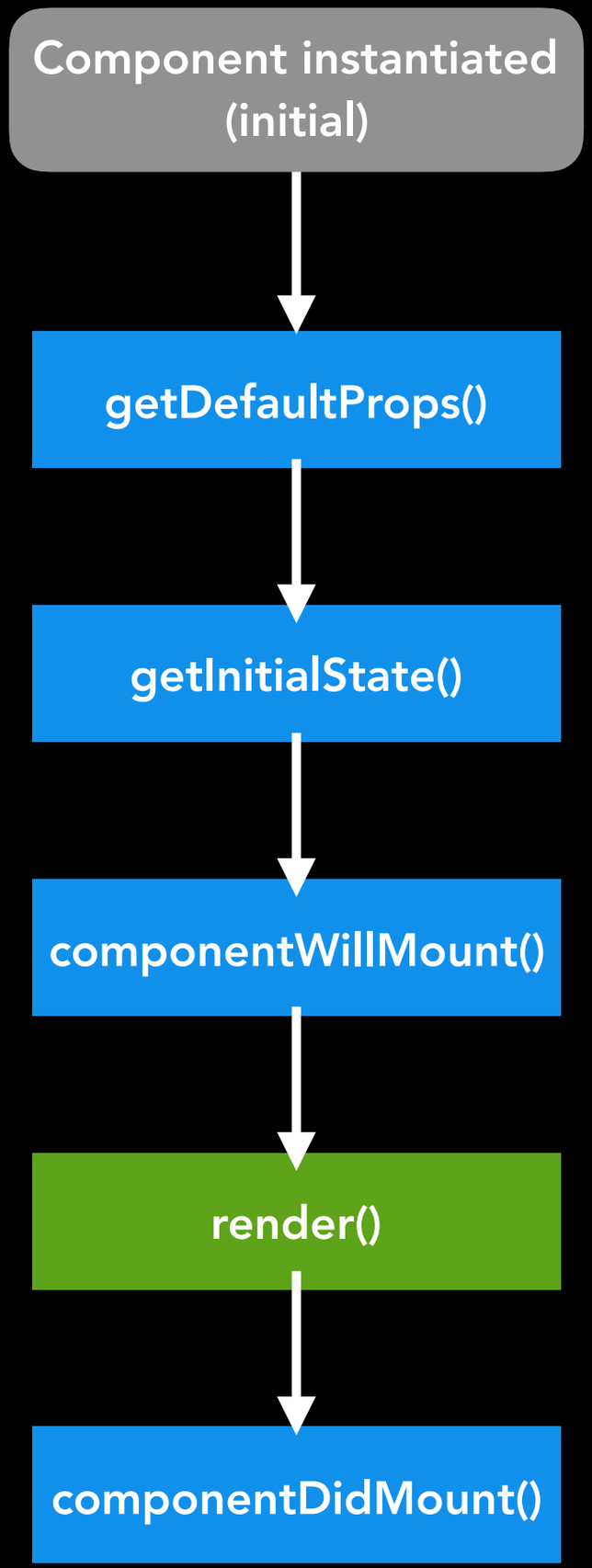
Receive **`this.props.movies`** for ListView

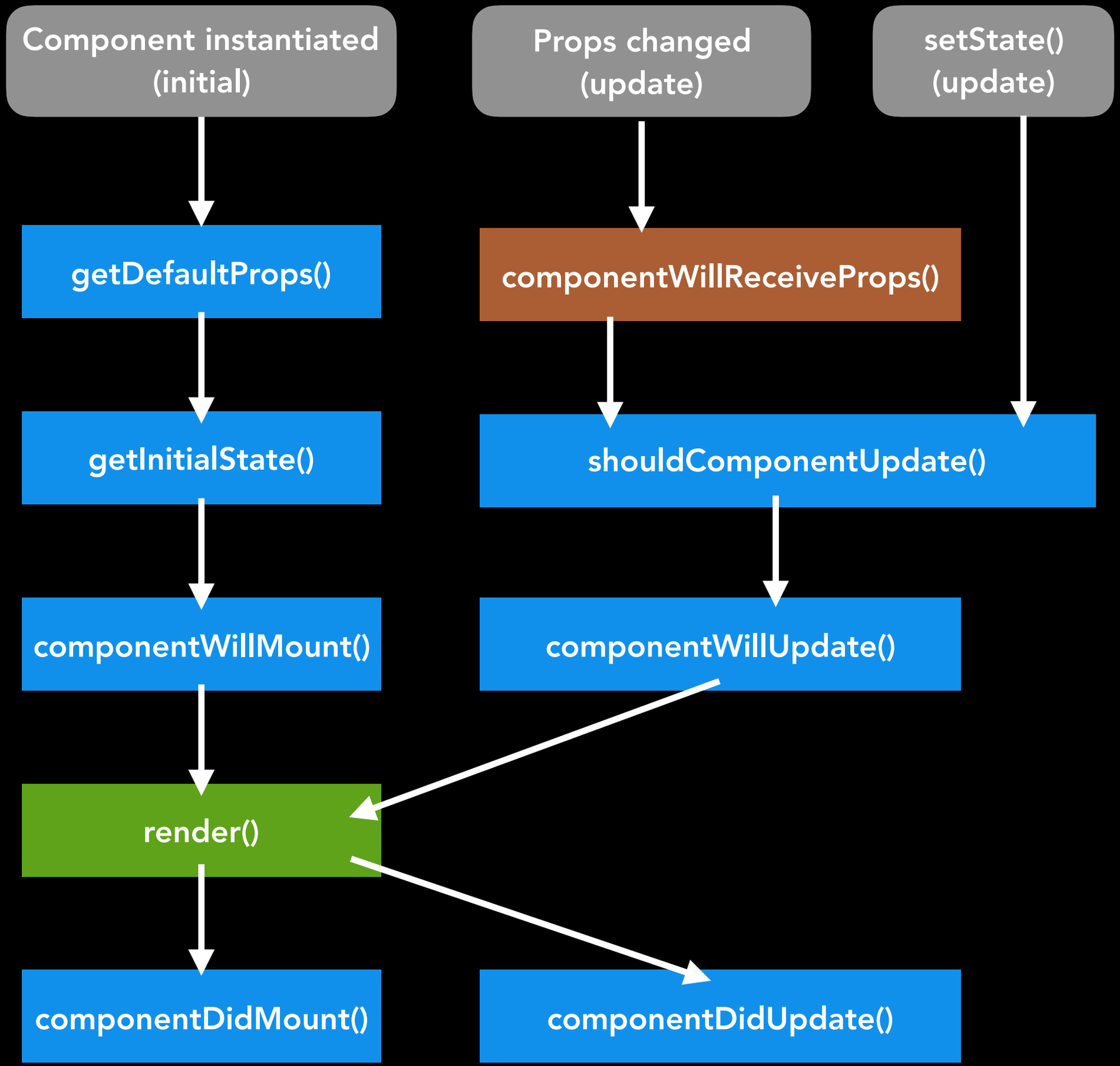
Component Movies (parent)

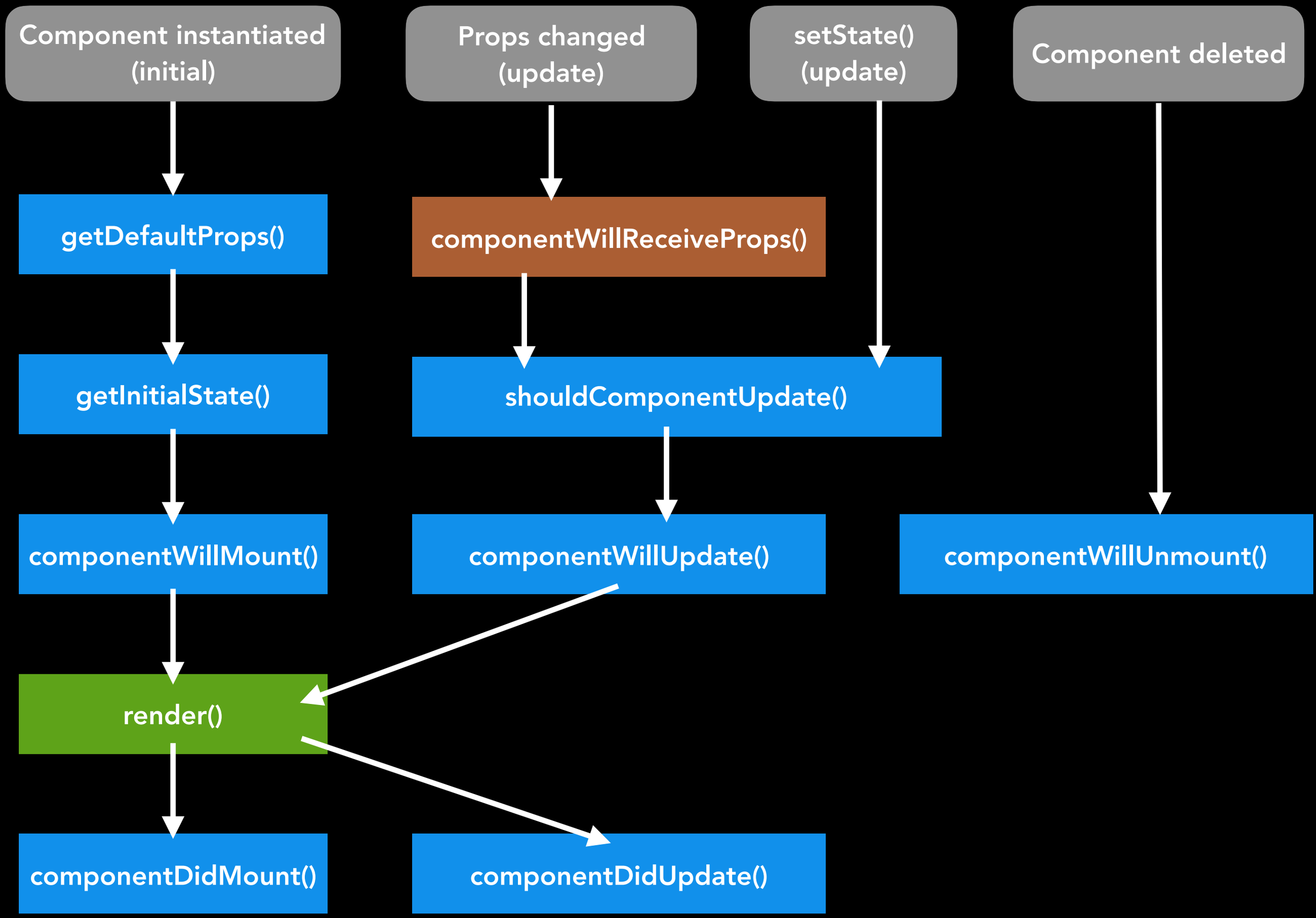
```
this.setState({
    data: getDataFromApi()
})
```

When state is changed, render function is called

LIFE CYCLE


Component instantiated
(initial)
getDefaultProps()
getInitialState()
componentWillMount()
render()
componentDidMount()

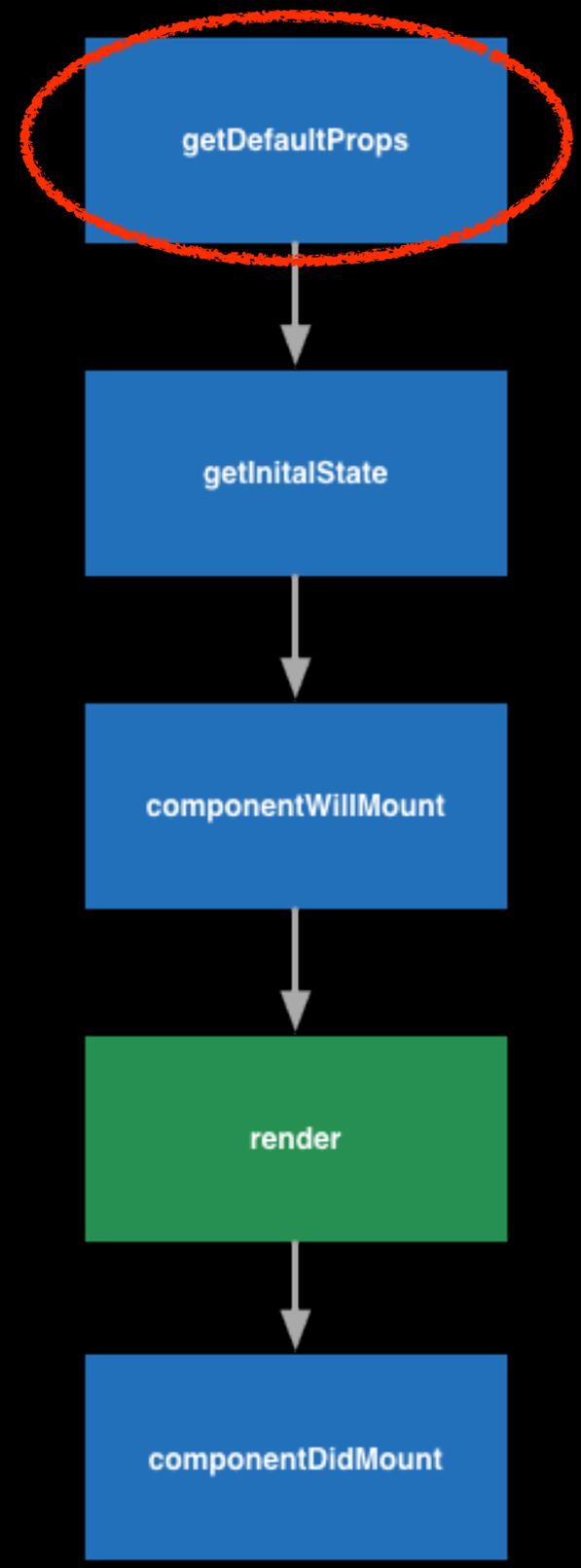

Component instantiated (initial)
Props changed (update)
setState() (update)
getDefaultProps()
componentWillReceiveProps()
getInitialState()
shouldComponentUpdate()
componentWillMount()
componentWillUpdate()
render()
componentDidMount()
componentDidUpdate()


Component instantiated (initial)
Props changed (update)
setState() (update)
Component deleted
getDefaultProps()
componentWillReceiveProps()
getInitialState()
shouldComponentUpdate()
componentWillMount()
componentWillUpdate()
componentWillUnmount()
render()
componentDidMount()
componentDidUpdate()

INITIAL LIFE CYCLE

INITIAL LIFE CYCLE

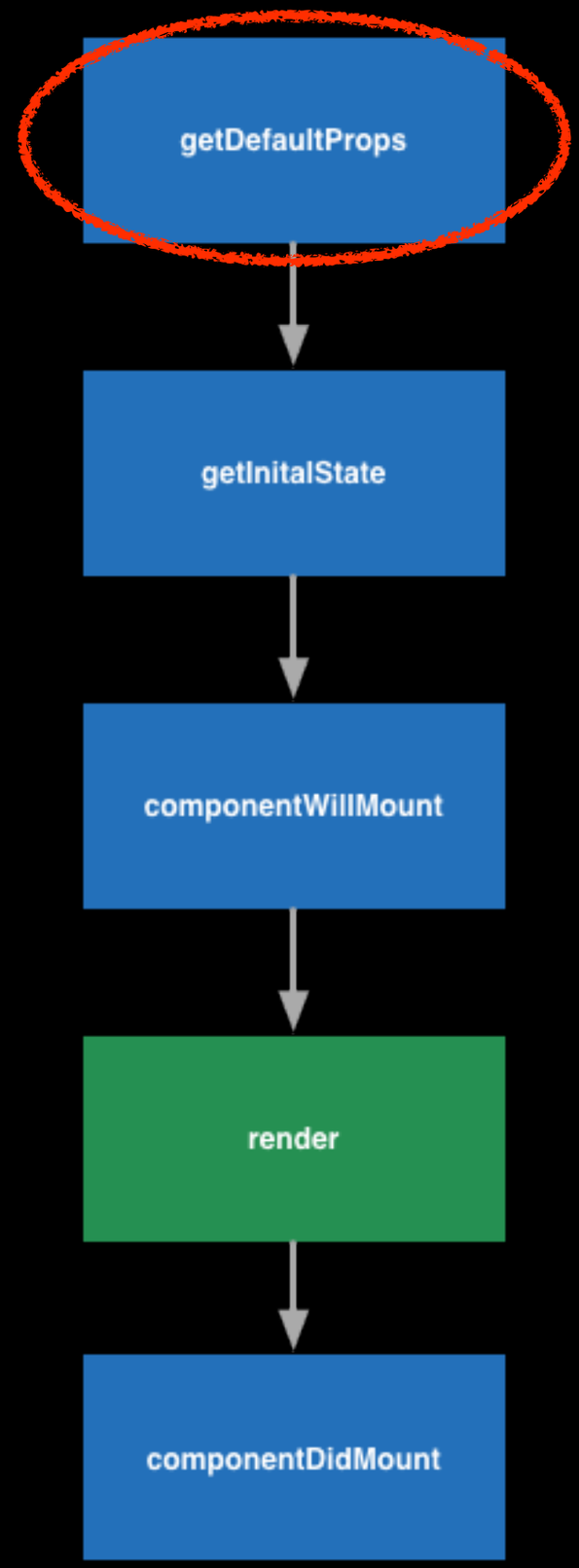
For defining **default props** which can be accessed via **this.props**



INITIAL LIFE CYCLE

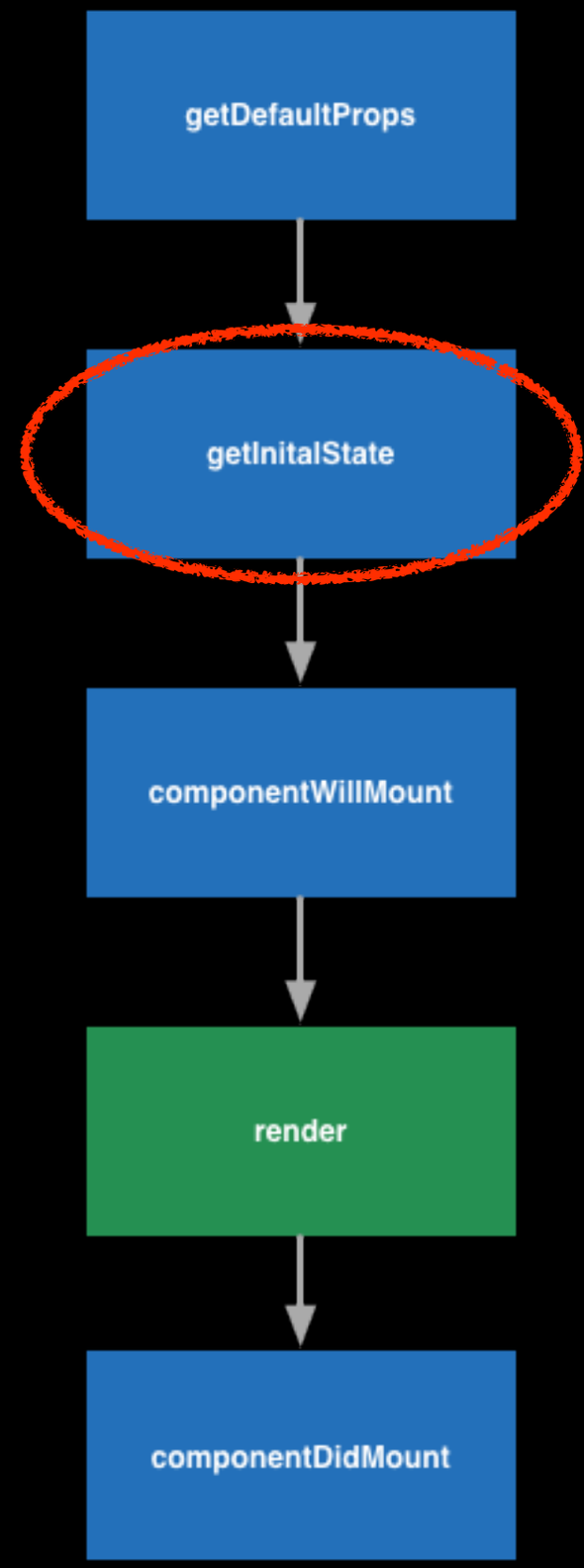
For defining **default props** which can be accessed via **this.props**

Interface between 2 components, remember?



INITIAL LIFE CYCLE

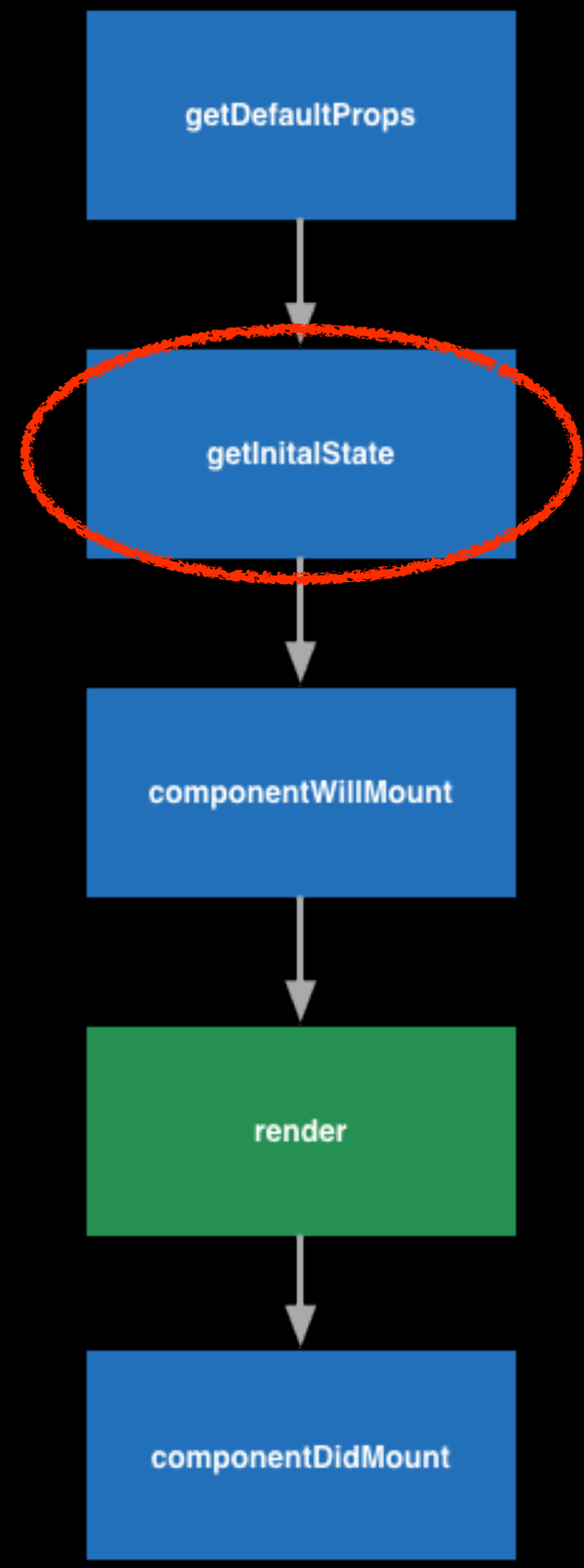
For defining **initial state** which can be accessed via **this.state**



INITIAL LIFE CYCLE

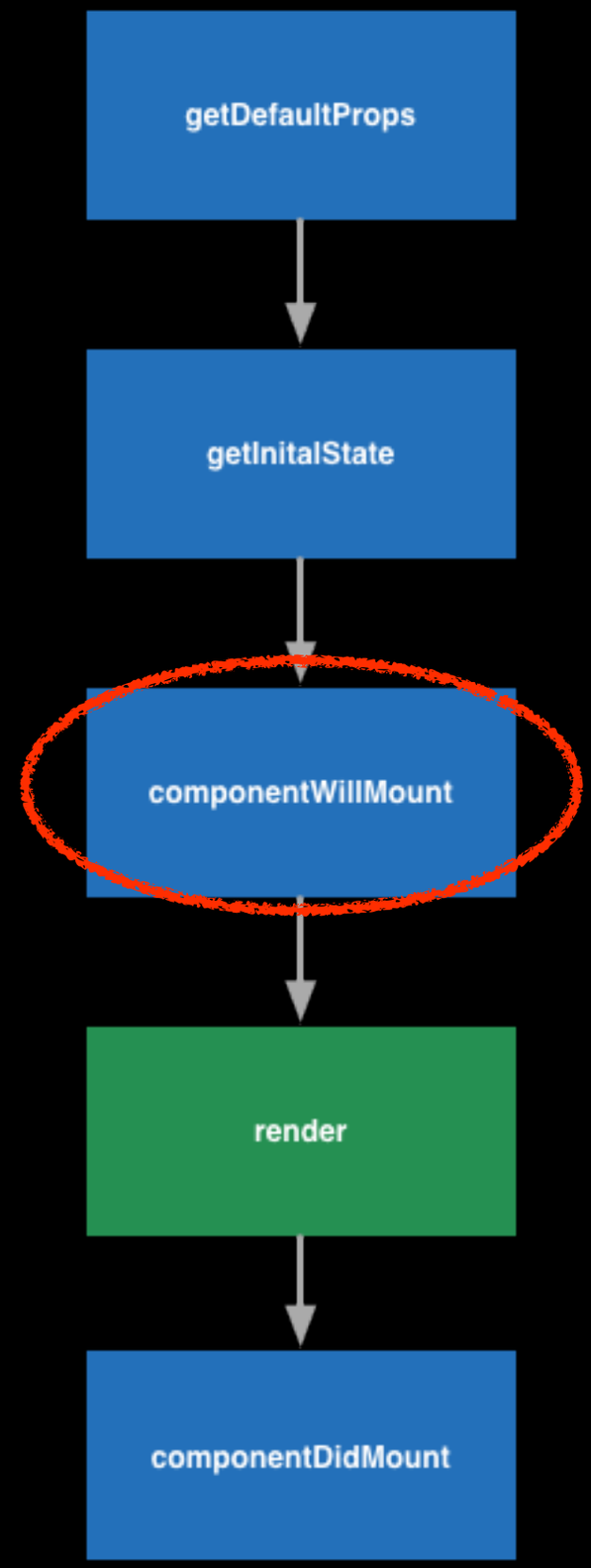
For defining **initial state** which can be accessed via **this.state**

Internal state only
accessible by component

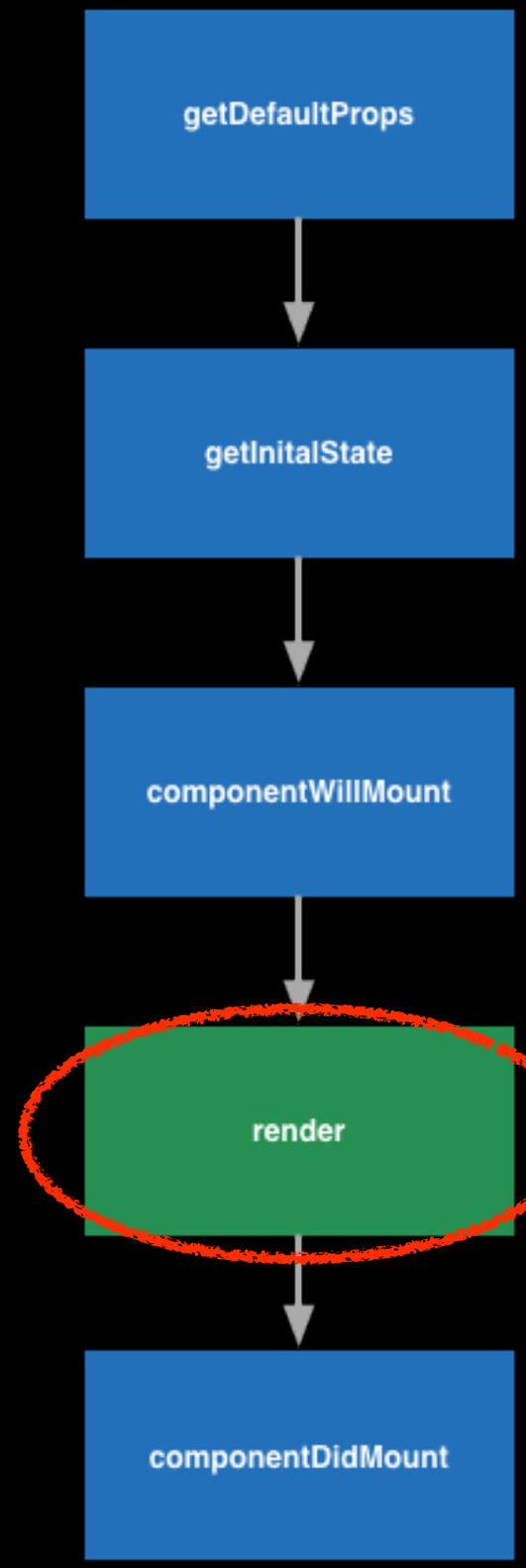


INITIAL LIFE CYCLE

Called before render. Any state changes will not cause render

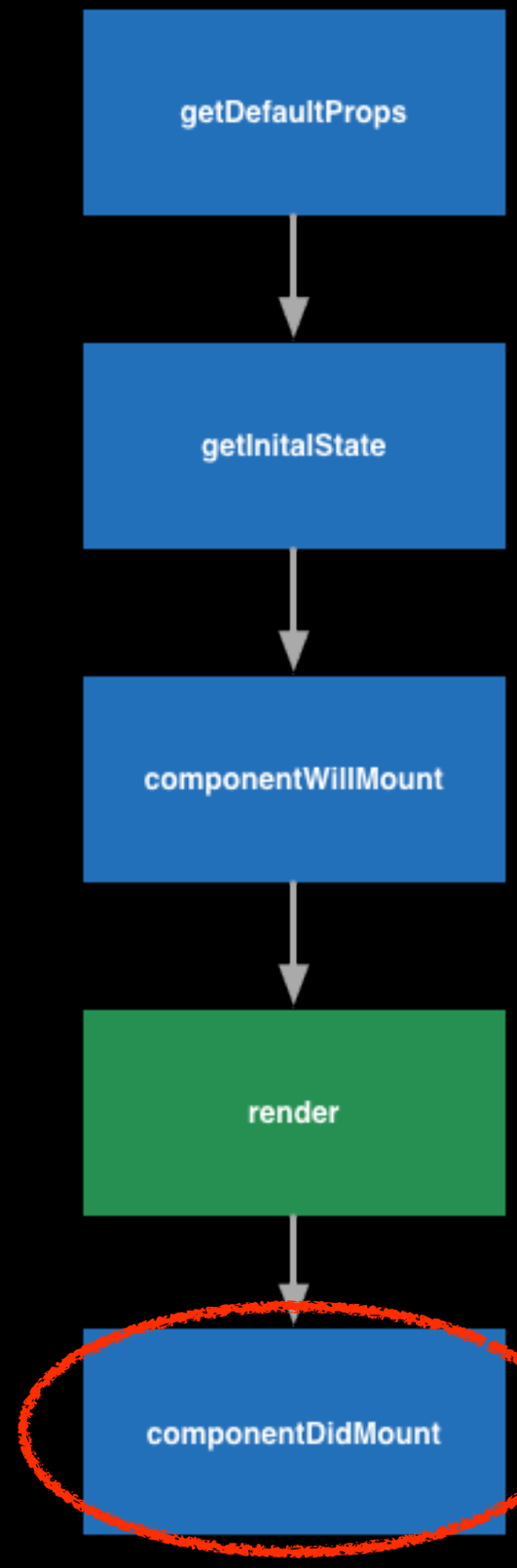


INITIAL LIFE CYCLE



Returns **React element with logic**

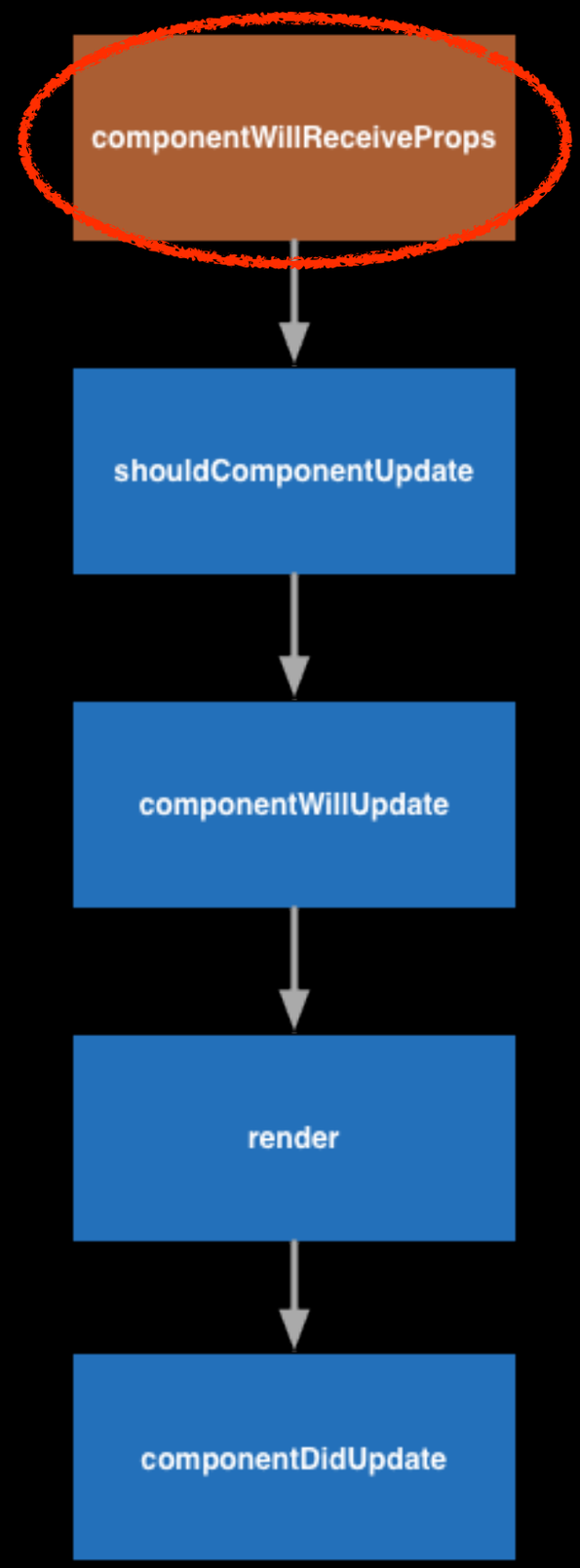
INITIAL LIFE CYCLE



Can access DOM (component tree) and every interaction with DOM should be done here (not in render)

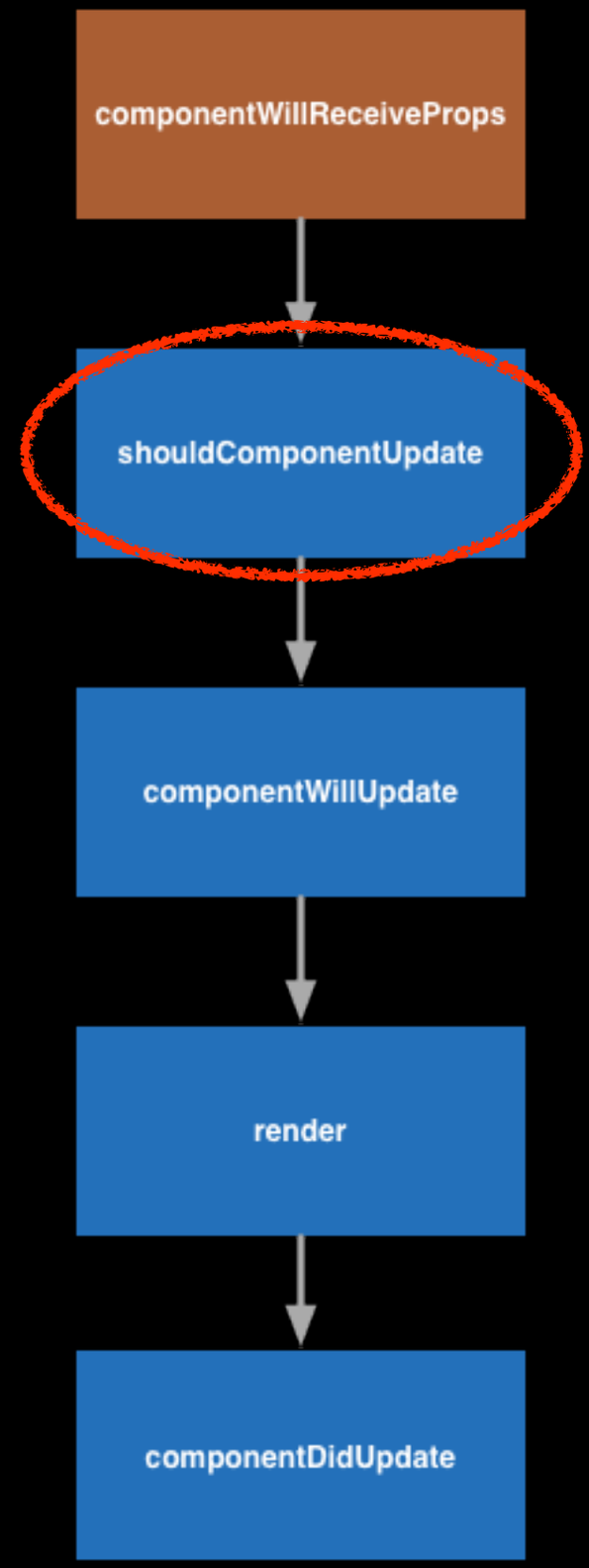
UPDATE LIFE CYCLE

UPDATE LIFE CYCLE



Call only **if props has changed**.
State can be changed without re-render

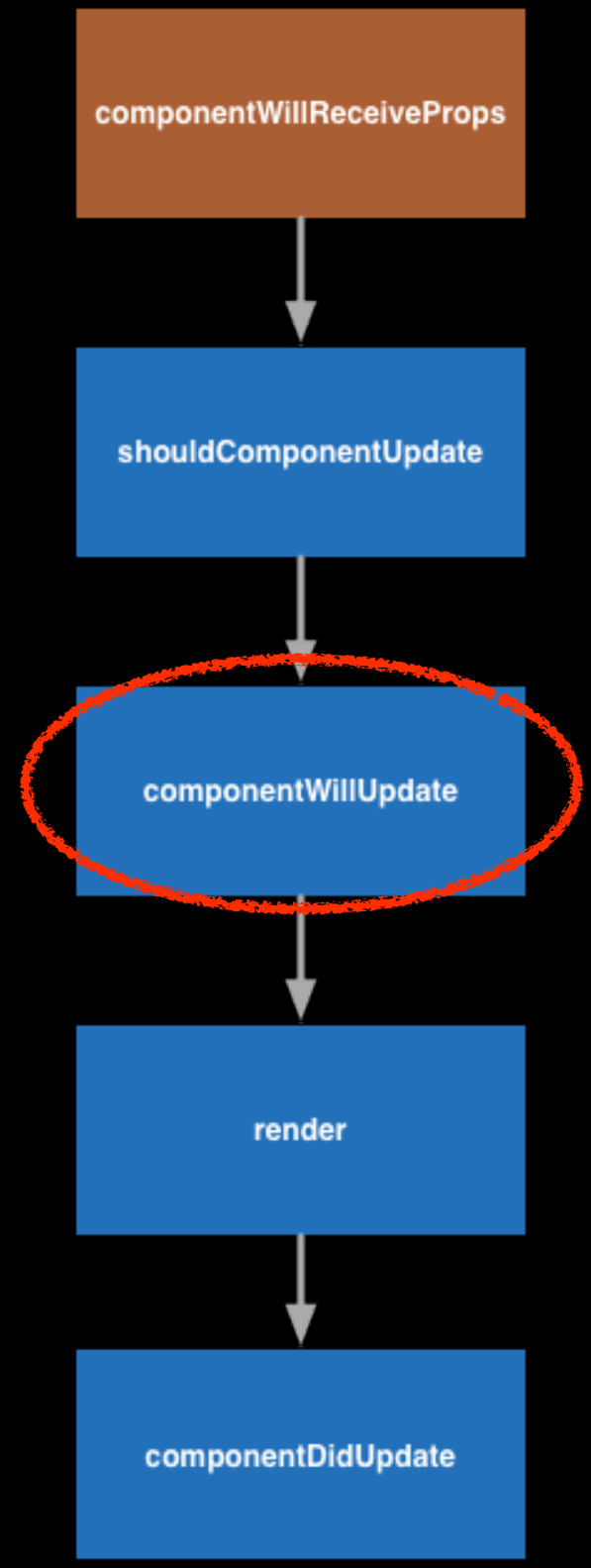
UPDATE LIFE CYCLE



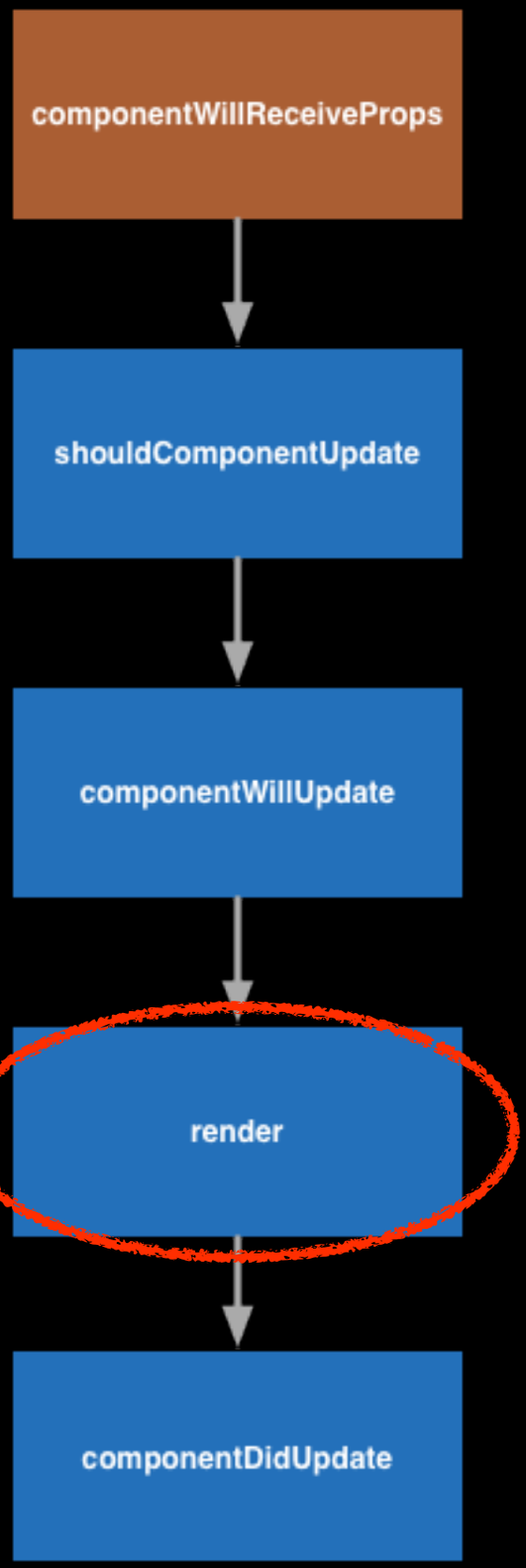
Decide if component should be re-rendered. Returns true/false. Has access to current and new state and props

UPDATE LIFE CYCLE

Call if shouldComponentUpdate returns true



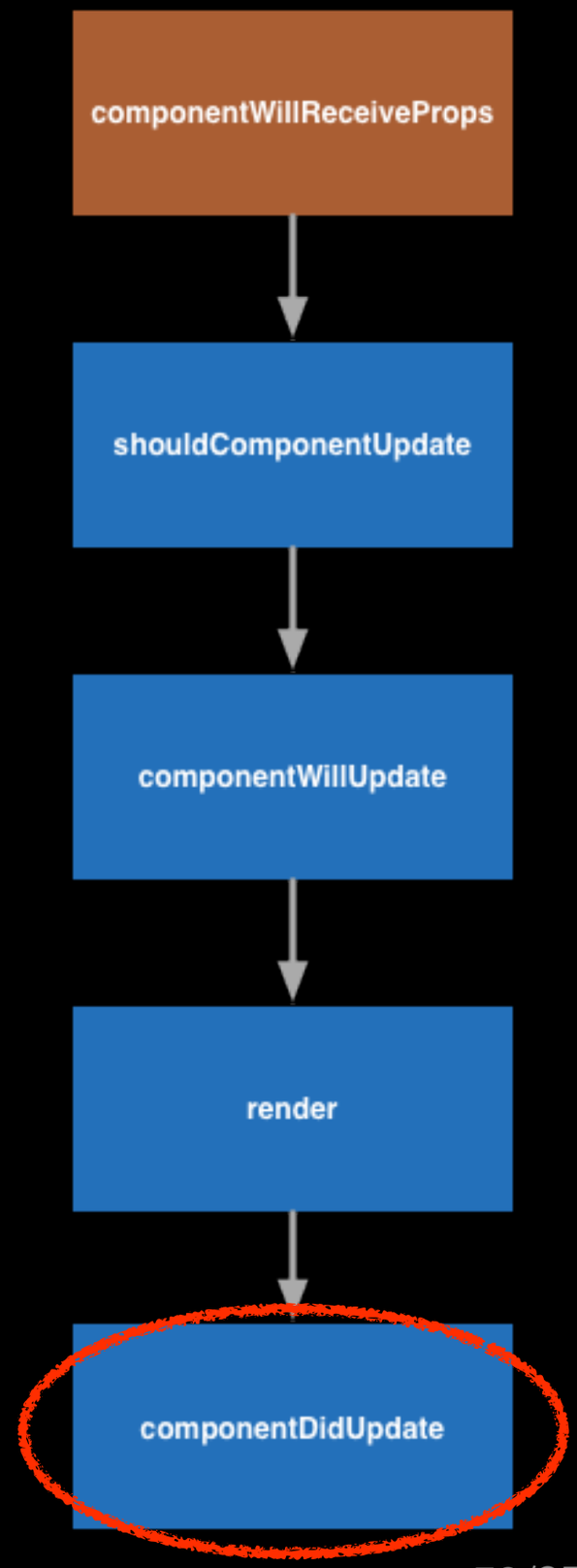
UPDATE LIFE CYCLE



Returns **React element with logic**

UPDATE LIFE CYCLE

Can access DOM (component tree) and every interaction with DOM should be done here (not in render)



REGISTERING COMPONENT

```
import React, { Component } from 'react';
import {
  AppRegistry,
  Text,
  View
} from 'react-native';

class Movies extends Component {
  render() {
    return (
      <View>
        <Text>Hello</Text>
      </View>
    );
  }
}

AppRegistry.registerComponent('Movies', () => Movies);
```

REGISTERING COMPONENT

```
import React, { Component } from 'react';
import {
  AppRegistry,
  Text,
  View
} from 'react-native';

class Movies extends Component {
  render() {
    return (
      <View>
        <Text>Hello</Text>
      </View>
    );
  }
}

AppRegistry.registerComponent('Movies', () => Movies);
```

Dependency injection

REGISTERING COMPONENT

```
import React, { Component } from 'react';
import {
  AppRegistry,
  Text,
  View
} from 'react-native';

class Movies extends Component {
  render() {
    return (
      <View>
        <Text>Hello</Text>
      </View>
    );
  }
}

AppRegistry.registerComponent('Movies', () => Movies);
```

Dependency injection

Register component

EVENTS

- **FORM**: onChange, onSubmit,…
- **MOUSE**: onClick, onDoubleClick, onDrag, onDrop,…
- **KEYBOARD**: onKeyPress,…

https://facebook.github.io/react/docs/events.html

EVENTS FORM

```
var MovieAdd = React.createClass({
  render: function() {
    return (
    <form onSubmit={this._handleSubmit}>
      <input ref="title" />
    </form>
    );
  },
  _handleSubmit: function(e) {
    console.log(this.refs.title)
  }
});
```

COMPONENT

- Every component needs function **render**
- Render function **have to return JSX (or boolean)**
- Components should be reusable and independent

STYLE SHEETS

- **No CSS in React, but similar attributes**
- Styles **declared in JS**
- Should be outside of render function – no re-creating in every render

https://facebook.github.io/react-native/docs/style.html

STYLE SHEETS IMPORT

```
import React, { Component } from 'react';
import {
  AppRegistry,
  StyleSheet,
  Text,
  View
} from 'react-native';
```

STYLE SHEETS EXAMPLE

```
<Text style={styles.title}>Hello clippy</Text>
```

```
var styles = StyleSheet.create({
    title: {
        padding: 10,
        fontSize: 15,
        color: '#656565'
    },
});
```

STYLE SHEETS EXAMPLE

```
<Text style={styles.title}>Hello clippy</Text>
```

```
var styles = StyleSheet.create({
    title: {
        padding: 10,
        fontSize: 15,
        color: '#656565'
    },
});
```

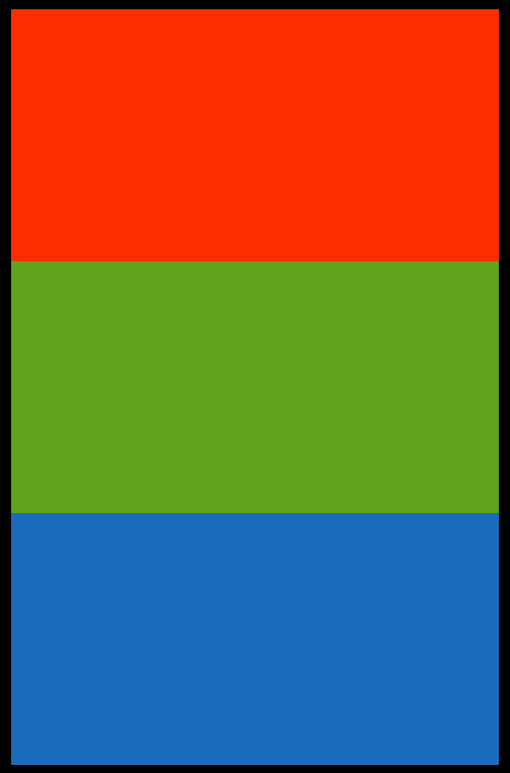
FLEXBOX

- CSS layout type
- **Similar use** as in CSS, **different implementation**
- Used in React Native

http://bit.ly/1TjoJup
https://css-tricks.com/snippets/css/a-guide-to-flexbox/

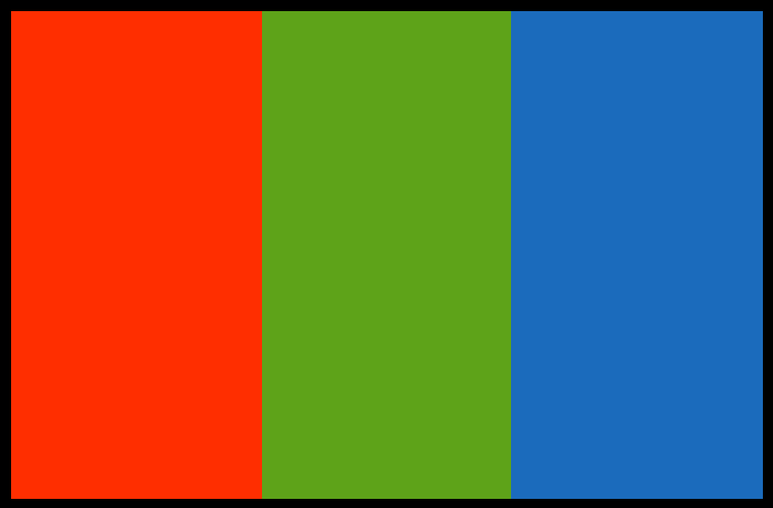
FLEXBOX COLUMN EXAMPLE

```
1 class HelloWorldView extends React.Component {
2     render() {
3         return (
4             <View style={{flexDirection: 'column', flex: 1}}>
5                 <View style={{flex: 1, backgroundColor: 'red'}}></View>
6                 <View style={{flex: 1, backgroundColor: 'green'}}></View>
7                 <View style={{flex: 1, backgroundColor: 'blue'}}></View>
8             </View>
9         )
10     }
11 }
```



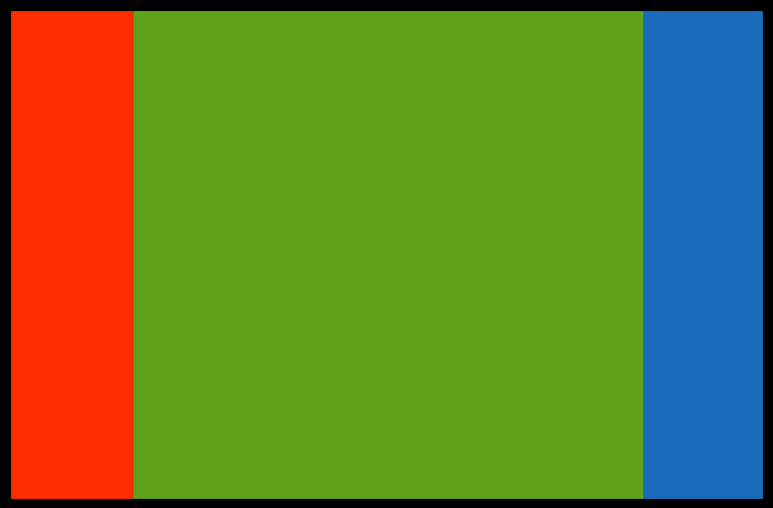
FLEXBOX ROW EXAMPLE

```
1 class HelloWorldView extends React.Component {
2     render() {
3         return (
4             <View style={{flexDirection: 'row', flex: 1}}>
5                 <View style={{flex: 1, backgroundColor: 'red'}}></View>
6                 <View style={{flex: 1, backgroundColor: 'green'}}></View>
7                 <View style={{flex: 1, backgroundColor: 'blue'}}></View>
8             </View>
9         )
10     }
11 }
```

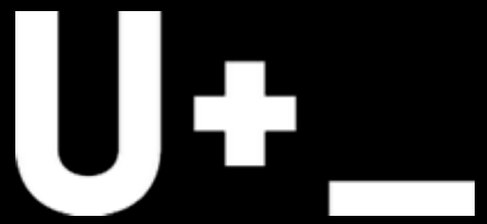


FLEXBOX ROW EXAMPLE

```
1 class HelloWorldView extends React.Component {
2     render() {
3         return (
4             <View style={{flexDirection: 'row', flex: 1}}>
5                 <View style={{flex: 1, backgroundColor: 'red'}}></View>
6                 <View style={{flex: 4, backgroundColor: 'green'}}></View>
7                 <View style={{flex: 1, backgroundColor: 'blue'}}></View>
8             </View>
9         )
10     }
11 }
```



QUESTIONS?



JAN VÁCLAVÍK

@janvaclavik

DANIEL RYS

@danielrys

WWW.USERTECHNOLOGIES.COM