

# React Native Quickly

SoCal Code Camp

2+3 December 2017

# Troy Miles

- Troy Miles
- Nearly 40 years of experience
- Programmer, speaker & author
- rockncoder@gmail.com
- @therockncoder
- lynda.com Author



# Check Versions

app command		my version
git	git --version	2.15.0
node.js	node -v	8.6.0
npm	npm --v	5.5.1
react		16.0.0
react-native	react-native -v	0.50.4

“Learn once, write anywhere: Build mobile apps  
with React”

# React Native

- “Deliver native Android, iOS, and Windows apps, using existing skills, teams, and code.”
- Released: March 27, 2015
- 51k+ GitHub Stars

# Installation

- Node.js (& npm)
- Android SDK
- React Native

All of our code will run on both PC & Mac. But iOS can only be developed on a Mac. So we will only build code for iOS.

# Android Development

- Install Java 8
- Install Android Studio
- Install Android SDK 23 (Marshmallow)
- Install Intel HAXM (x86 Android Emulator)



# Mac Installation

- `brew install node`
- `brew install watchman*`
- `npm i -g react-native-cli`

# Windows Installation

- `choco install nodejs.install`
- `choco install python2`
- `choco install jdk8`

# React-Native

- `react-native init <app name>`
- `cd <app name>`
- `react-native run-ios`

```
Europa:desktop troymiles$ react-native init ToDo
This will walk you through creating a new React Native project in /Users/troymiles/Desktop/ToDo
Using yarn v1.3.2
Installing react-native...
yarn add v1.3.2
info No lockfile found.
[1/4] 🔍 Resolving packages...
warning react-native > connect@2.30.2: connect 2.x series is deprecated
[2/4] 📦 Fetching packages...
[3/4] 🔗 Linking dependencies...
warning " > react-native@0.50.4" has unmet peer dependency "react@16.0.0".
[4/4] 🏗 Building fresh packages...
success Saved lockfile.
success Saved 504 new dependencies.
├─ abbrev@1.1.1
├─ absolute-path@0.0.0
├─ accepts@1.2.13
├─ ajv@5.5.1
├─ ansi-escapes@3.0.0
├─ ansi-regex@2.1.1
├─ ansi-styles@2.2.1
├─ ansi@0.3.1
├─ anymatch@1.3.2
├─ aproba@1.2.0
├─ are-we-there-yet@1.1.4
├─ arr-diff@2.0.0
├─ arr-flatten@1.1.0
├─ array-differ@1.0.0
├─ array-filter@0.0.1
├─ array-map@0.0.0
├─ array-reduce@0.0.0
├─ array-uniq@1.0.3
├─ array-unique@0.2.1
├─ art@0.10.1
├─ asap@2.0.6
├─ asn1@0.2.3
├─ assert-plus@1.0.0
├─ async@2.6.0
├─ asynckit@0.4.0
├─ aws-sign2@0.7.0
```

```
— longest@1.0.1
— natural-compare@1.4.0
— nwmatcher@1.4.3
— optionator@0.8.2
— p-cancelable@0.3.0
— parse5@1.5.1
— path-parse@1.0.5
— pinkie-promise@2.0.1
— pinkie@2.0.4
— prelude-ls@1.1.2
— pretty-format@21.2.1
— react-test-renderer@16.0.0
— resolve@1.1.7
— right-align@0.1.3
— sprintf-js@1.0.3
— string-length@2.0.0
— strip-bom@2.0.0
— symbol-tree@3.2.2
— test-exclude@4.1.1
— tr46@0.0.3
— type-check@0.3.2
— uglify-js@2.8.29
— uglify-to-browserify@1.0.2
— webidl-conversions@4.0.2
— whatwg-encoding@1.0.3
— whatwg-url@4.8.0
— window-size@0.1.0
— xml-name-validator@2.0.1
✚ Done in 5.64s.
```

**To run your app on iOS:**

```
cd /Users/troymiles/Desktop/ToDo
react-native run-ios
- or -
Open ios/ToDo.xcodeproj in Xcode
Hit the Run button
```

**To run your app on Android:**

```
cd /Users/troymiles/Desktop/ToDo
Have an Android emulator running (quickest way to get started), or a device connected
react-native run-android
```

Europa:desktop troymiles\$ █



```
7  import React, { Component } from 'react';
8  import {
9    Platform,
10   StyleSheet,
11   Text,
12   View
13 } from 'react-native';
14
15  const instructions = Platform.select({
16    ios: 'Press Cmd+R to reload,\n' +
17        'Cmd+D or shake for dev menu',
18    android: 'Double tap R on your keyboard to reload,\n' +
19        'Shake or press menu button for dev menu',
20  });
21
22  export default class App extends Component<{}> {
23    render() {
24      return (
25        <View style={styles.container}>
26          <Text style={styles.welcome}>
27            Welcome to React Native!
28          </Text>
29          <Text style={styles.instructions}>
30            To get started, edit App.js
31          </Text>
32          <Text style={styles.instructions}>
33            {instructions}
34          </Text>
35        </View>
36      );
37    }
38  }
```

```
40  const styles = StyleSheet.create({
41    container: {
42      flex: 1,
43      justifyContent: 'center',
44      alignItems: 'center',
45      backgroundColor: '#F5FCFF',
46    },
47    welcome: {
48      fontSize: 20,
49      textAlign: 'center',
50      margin: 10,
51    },
52    instructions: {
53      textAlign: 'center',
54      color: '#333333',
55      marginBottom: 5,
56    },
57  });
```

# React Native + NativeBase

- `rnpm install native-base`
- `npm i -S moment react-redux redux`
- `npm install`



# babel-preset-react-native ERROR

- npm uninstall babel-preset-react-native
- npm install babel-preset-react-native@2.1.0

```
TransformError: /Users/troymiles/  
Documents/MyWorkshops/DEL-RN/nbt/  
index.ios.js: Unexpected token ) (While  
processing preset: "/Users/troymiles/  
Documents/MyWorkshops/DEL-RN/nbt/  
node_modules/babel-preset-react-native/  
index.js")
```

```
RCTFatal
```

```
__28-[RCTCxxBridge handleError:]_block_i  
nvoke
```

```
_dispatch_call_block_and_release
```

```
_dispatch_client_callout
```

```
_dispatch_main_queue_callback_4CF
```

```
__CFRUNLOOP_IS_SERVICING_THE_MAIN_DISPAT
```

# Mac Fix

- Close packager window
- `watchman watch-del-all`
- `rm -rf node_modules && npm install`
- `react-native run-android`

# PC Fix

- Close packager window
- Delete the node\_modules directory
- npm install
- react-native run-android

# react-native CLI

Command	Purpose
init [app name]	creates a react-native app
run-android [options]	builds app, starts on Android
run-ios [options]	builds app, starts on iOS
start [options]	starts webserver
new-library [options]	generates a native library bridge
bundle [options]	builds offline javascript bundle
unbundle [options]	builds unbundle javascript

# react-native CLI

Command	Purpose
eject [options]	takes the shackles off
link [options] <packageName>	links all native dependencies
unlink [options] <packageName>	unlink native dependency
install [options] <packageName>	install+link native dependencies
uninstall [options] <packageName>	uninstall+unlink native
upgrade [options]	upgrade app's template files
log-android [options]	starts adb logcat
log-ios [options]	starts iOS device syslog tail

# Development Environment

- Command/Terminal window for CLI Commands
- The Packager window
- iOS simulator or Android emulator
- Chrome browser
- Your IDE

Scanning folders for symlinks in /Users/troymiles/Desktop/ToDo/node\_modules (13ms)

Running Metro Bundler on port 8081.

Keep Metro Bundler running while developing on any JS projects. Feel free to close this tab and run your own Metro Bundler instance if you prefer.

<https://github.com/facebook/react-native>

Looking for JS files in  
/Users/troymiles/Desktop/ToDo

Metro Bundler ready.

Loading dependency graph, done.

Bundling `index.js` [development, non-minified] 100.0% (478/478), done.

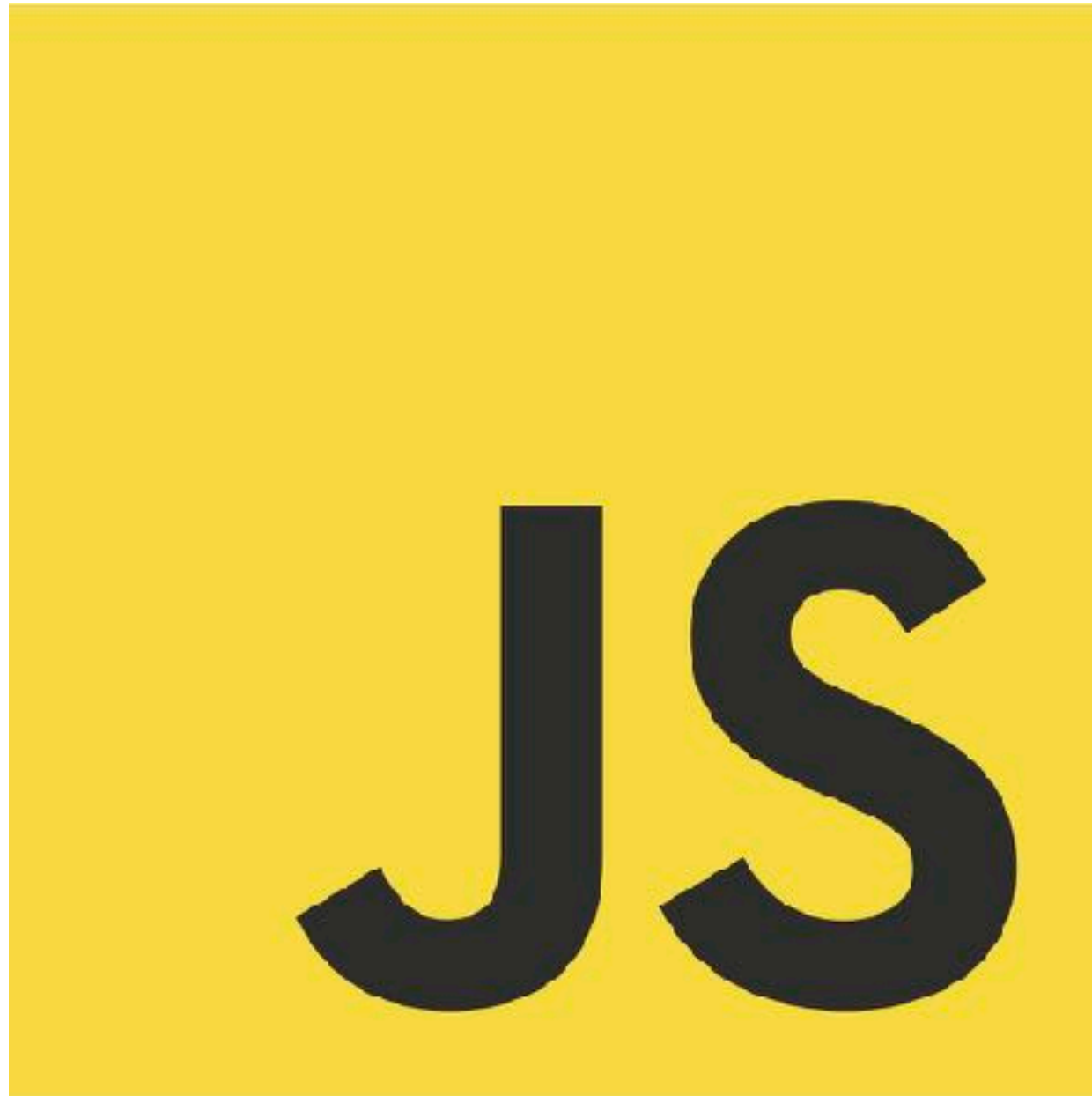
# In-App Developer Menu

Command	Purpose
Command-M or Alt-M	Show Developer Menu
Reload	Reloads the code
Debug JS Remotely	Debugging via Chrome
Enable Live Reload	Automatically updates on save
Enable Hot Reloading	Automatically updates on save, with state
Toggle Inspector	Shows view construction
Show Perf Monitor	Shows performance information



# Application Root Directory

- All of the commands, for all of the tools are designed work on the application root directory
- If used anywhere else bad things will happen
- be sure you are in the app root
- double check that you are in the app root



JavaScript

# ECMAScript Versions

Version	Date
ES1	June 1997
ES2	June 1998
ES3	December 1999
ES4	DOA 2006
ES5	December 2009
ES2015 / ES6	June 2015
ES2016 / ES7	2016

# Collection Operators

- `.isArray()`
- `.every()`
- `.forEach()`
- `.indexOf()`
- `.lastIndexOf()`
- `.some()`
- `.map()`
- `.reduce()`
- `.filter()`

# map

```
let junk = [1, 2, 3, 4, 'Alpha', 5, {name: 'Jason'}];  
let letters = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K'];  
let nums = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20];  
console.log(nums);  
  
// map iterates over all of the elements and returns a new array with the same  
// number of elements  
let nums2 = nums.map((elem) => elem * 2);  
console.log(nums2);  
/// [2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40]
```

# filter

```
let junk = [1, 2, 3, 4, 'Alpha', 5, {name: 'Jason'}];  
let letters = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K'];  
let nums = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20];  
console.log(nums);  
  
// filter iterates over the array and returns a new array with only the elements  
// that pass the test  
let nums3 = nums.filter((elem) => !(elem % 2));  
console.log(nums3);  
/// [1, 3, 5, 7, 9, 11, 13, 15, 17, 19]
```

# reduce

```
let junk = [1, 2, 3, 4, 'Alpha', 5, {name: 'Jason'}];  
let letters = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K'];  
let nums = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20];  
console.log(nums);  
  
// reduce iterates over the array passing the previous value and the current  
// element it is up to you what the reduction does, let's concatenate the strings  
let letters2 = letters.reduce((previous, current) => previous + current);  
console.log(letters2);  
/// ABCDEFGHIJK  
  
// reduceRight does the same but goes from right to left  
let letters3 = letters.reduceRight((previous, current) => previous + current);  
console.log(letters3);  
/// KJIHGFEDCBA
```

# let

- let allows us to create a block scoped variables
- they live and die within their curly braces
- var is considered deprecated
- best practice is to use let instead of var



# let

```
// let allows us to create block scoped variables
// they live and die within the curly braces
let val = 2;
console.info(`val = ${val}`);
{
    let val = 59;
    console.info(`val = ${val}`);
}
console.info(`val = ${val}`);
```

# const

- const creates a variable that can't be changed
- best practice is to make any variable that should not change a constant
- does not apply to object properties or array elements

# const

```
const name = 'Troy';  
console.info(`My name is ${name}`);  
// the line below triggers a type error  
name = 'Miles';
```

# Template strings

- Defined by using opening & closing back ticks
- Templates defined by `${JavaScript value}`
- The value can be any simple JavaScript expression
- Allows multi-line strings (return is pass thru)

# Template strings

```
let state = 'California';
let city = 'Long Beach';
console.info(`This weekend's workshop is in ${city}, ${state}.`);

// template strings can run simple expressions like addition
let cup_coffee = 4.5;
let cup_tea = 2.5;
console.info(`coffee: ${cup_coffee} + tea: ${cup_tea} = ${cup_coffee + cup_tea}.`);

// they can allow us to create multi-line strings
console.info(`This is line #1.
this is line #2.`);
```

# Arrow functions

- Succinct syntax
- Doesn't bind its own *this*, *arguments*, or *super*
- Facilitate a more functional style of coding
- Can't be used as constructors

# Arrow functions

- When only one parameter, parenthesis optional
- When zero or more than one parameter, parenthesis required

# Arrow function

```
let anon_func = function (num1, num2) {  
  return num1 + num2;  
};  
console.info(`Anonymous func: ${anon_func(1, 2)}`);  
  
let arrow_func = (num1, num2) => num1 + num2;  
console.info(`Arrow func: ${arrow_func(3, 4)}`);
```



# this

- this is handled different in arrow functions
- In anonymous function this is bound to the global object
- In arrow function this is what it was in the outer scope

# Destructuring

- Maps the data on the right side of the equals sign to the variables on the left
- The data type decides the way values are mapped
- It is either object or array destructuring

# Object Destructuring

```
16// this is a demo of the power of destructuring
17// we have two objects with the same 3 properties
18 const binary = {kb: 1024, mb: 1048576, gb: 1073741824};
19 const digital = {kb: 1000, mb: 1000000, gb: 10000000000};
20// We use a ternary statement to choose which object
21// assign properties based on their property names
22 const {kb, mb, gb} = (useBinary) ? binary : digital;
```

# Array Destructuring

```
5
6  let [param1, bob, key] = ['first', 'second', '3rd'];
7  console.info(`param1 = ${param1}, bob = ${bob}, key = ${key}`);
8  // param1 = first, bob = second, key = 3rd
```

# Spread syntax

- Expands an expression in places where multiple arguments, elements, or variables are expected

# The spread operator

```
11
12  // the spread operator
13  const myArray = ['Bob', 'Sue', 'Fido'];
14  function printFamily(person1, person2, pet) {
15    console.info(`Person 1: ${person1}, Person 2: ${person2}, and their pet: ${pet}`);
16  }
17  printFamily(...myArray);
18  // Person 1: Bob, Person 2: Sue, and their pet: Fido
19
```

# Generator Function\*

- function\* is called a generator
- A generator is a function which can be exited and re-entered
- statements are executed until a yield is encountered
- the generator then pauses execution until the yield returns
- if a return statement is encountered, it finishes the generator

# Yield

- yield is a keyword which pauses the execution of a generator function
- yield can return a value to the generator



# Class

- Syntactic sugar over JavaScript use of function constructors
- JavaScript uses proto-typical inheritance
- If a class **extends** another, it must include **super()** as first instruction in its **constructor**
- Only create a constructor if it does something

# Class

- Syntactic sugar over JavaScript use of function constructors
- JavaScript uses proto-typical inheritance
- If a class **extends** another, it must include **super()** as first instruction in its **constructor**
- Only create a constructor if it does something

# import

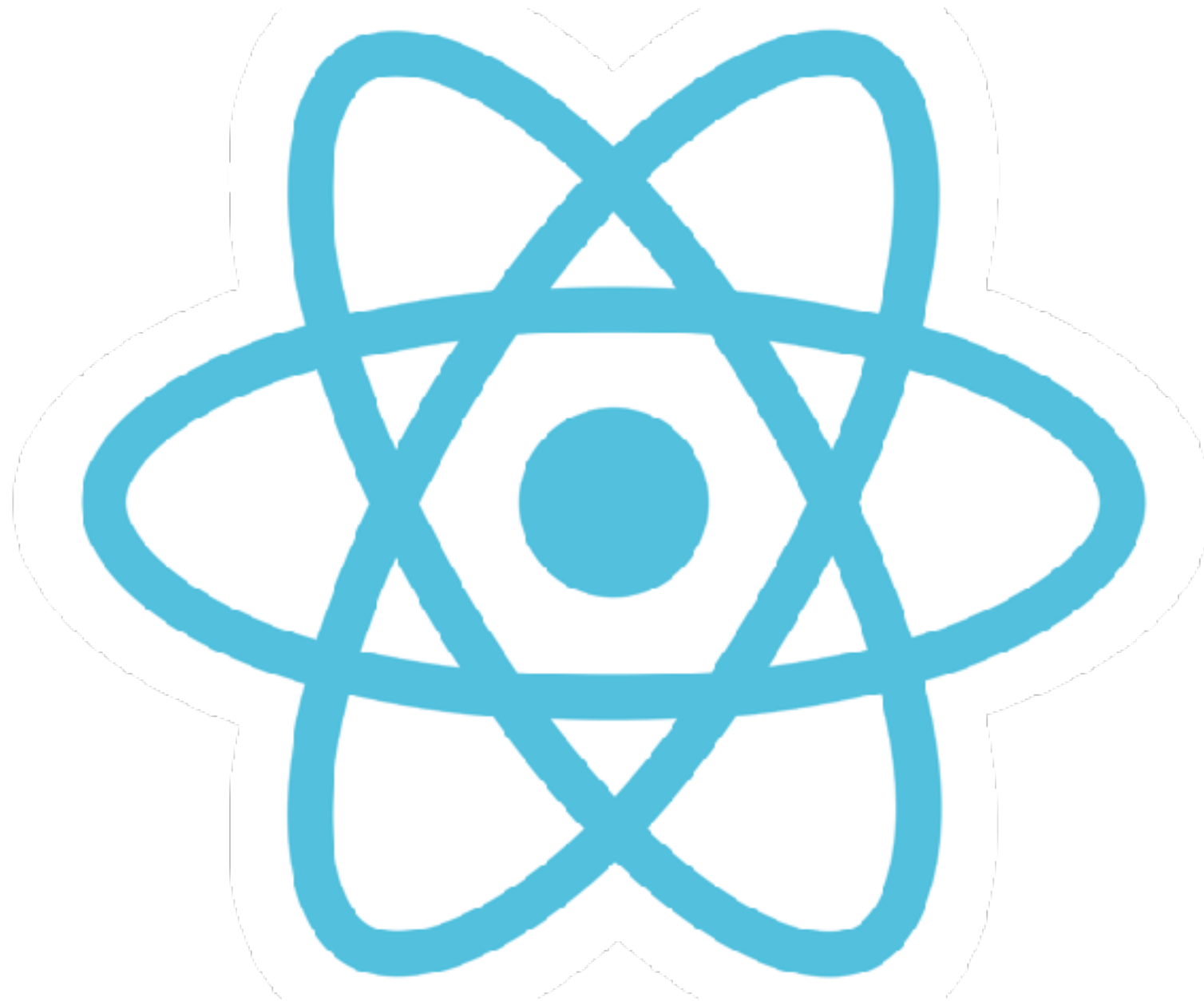
- Imports functions, objects, or primitives from other files
- `import <name> from "<module name>";`
- `import {name } from "<module name>";`
- `import * as Greetings from "<module name>";`
- relative path indicates not an npm package

# export

- export <var a>
- export {a, b};

# export default

- only one per file
- common pattern for libraries
- `const Greetings = {sayHi, sayBye};`
- `export default Greetings;`
- `export default {sayHi, sayBye};`



React

# React

- A JavaScript library for building user interfaces
- Created by Facebook & Instagram
- Initial release March 2013
- Current version 15.6.1
- Next major version 16.0.0, will have breaking changes

# React

- Virtual DOM
- One-way data flow
- JSX - JavaScript eXtension allows in HTML generation
- Component-based



# React API

- The use of JSX is optional in React
- You can use the React “API” instead
- *The `createClass` method is deprecated and will be removed in React 16*

# Component

- Fundamental building block of React
- Can be created with a JS Class or Function
- The render method is mandatory

# Class Component

```
3
4  class Square extends React.Component {
5      render() {
6          return (
7              <button
8                  className="square"
9                  onClick={() => this.props.onClick()}>
10                 {this.props.value}
11             </button>
12         );
13     }
14 }
15
```

# Functional Component

```
15
16  function Square(props) {
17    return (
18      <button
19        className="square"
20        onClick={() => props.onClick()}>
21        {props.value}
22      </button>
23    );
24  }
```

# Props

- Props are read-only objects
- Passed to component via attributes from parent
- Access via *this.props*

# State

- State is internal to the component
- It is mutable
- Access via *this.state*
- Initialize in the constructor
- Should only modified via setState

# PropTypes

- Use the npm package “prop-types” instead
- `import PropTypes from 'prop-types';`
- *React.PropTypes is deprecated*

# PropTypes

- PropTypes allow you to declare what properties your component expects
- React validates property at runtime
- Using propTypes is optional



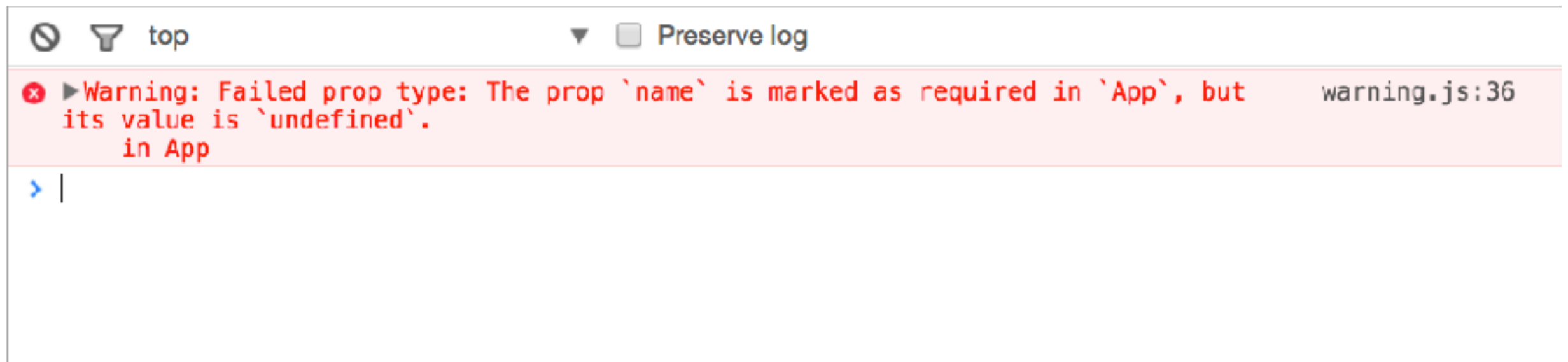
# Some PropTypes

Component	Command
PropTypes.array	an optional array
PropTypes.bool	an optional bool
PropTypes.element	An optional React element
PropTypes.func	An optional function
PropTypes.node	Anything that can be rendered
PropTypes.number	An optional number
PropTypes.object	An optional object
PropTypes.string	An optional string
PropTypes.symbol	An optional Symbol

# PropType in code

```
1
2 import React from 'react';
3 import ReactDOM from 'react-dom';
4 import PropTypes from 'prop-types';
5
6 class App extends React.Component {
7   render () {
8     return React.DOM.span(null, `My name is ${this.props.name}`);
9   }
10 }
11
12 App.propTypes = {
13   name: PropTypes.string.isRequired
14 };
15
16 ReactDOM.render(
17   React.createElement(App, {}),
18   document.getElementById('root')
19 );
```

# When property is missing



The screenshot shows a web browser's developer console. At the top, there is a toolbar with a close button, a filter icon, and the text 'top'. To the right of this is a dropdown arrow and a checkbox labeled 'Preserve log'. Below the toolbar, a red warning message is displayed on a light red background. The message reads: 'Warning: Failed prop type: The prop `name` is marked as required in `App`, but its value is `undefined`.' followed by 'in App' on a new line. The file path 'warning.js:36' is shown to the right of the message. Below the warning, there is a blue arrow icon and a vertical line, indicating the current position in the log.

```
top ▼ ☐ Preserve log  
⚠ Warning: Failed prop type: The prop `name` is marked as required in `App`, but  
  its value is `undefined`. warning.js:36  
  in App  
> |
```

# React Components

- Can be created two ways:
  - Using JavaScript
  - Using JSX

# Components via JS

- React's API contains method createElement()
- Takes 3 parameters: type, props, children

```
render() {  
  return React.createElement('h1', null, "Hello there.");  
}
```

# JSX

- **J**avaScript **S**yntax **E**Xtension
- A mixture of JavaScript and HTML syntax
- Compiles to JavaScript
- Is optional but preferred over using JavaScript
- And easier to read

# JSX Attributes

- Can assign either a string or an expression
- Strings can be either single or double quotes
- Expressions must be enclosed with curly braces

# Boolean Attributes

- HTML has a few boolean attributes, if present they are true
- Some of these include: checked, selected, disabled
- In JSX,
- `<Component disabled={true / false} />`



# Forbidden Attributes

- Two attributes are JavaScript keywords
- JavaScript keywords can't be used in JSX
  - `class` -> `className`
  - `for` -> `htmlFor`

# JSX Spread Syntax

- a shortcut to passing props to a component
- uses ES2015 spread operator
- `<Component {...object} />`

# JSX Spread Syntax

```
return (  
  <Timer  
    id={this.props.id}  
    amount={this.props.amount}  
    elapsed={this.props.elapsed}  
    runningSince={this.props.runningSince}  
    onStartClick={this.props.onStartClick}  
    onStopClick={this.props.onStopClick}  
    onResetClick={this.props.onResetClick}  
    onSetClick={this.handleSetClick}  
  />  
);
```

# JSX Spread Syntax

```
return (  
  <Timer  
    {...this.props}  
    onSetClick={this.handleClick}  
  />  
);
```

# JSX Debugging Tip

- Assign component to a variable
- `console.log` the variable

# JSX Debugging Tip

```
const timerComp = (  
  <Timer troy='miles'  
    {...this.props}  
    onSetClick={this.handleClick}  
  />  
);  
console.log(timerComp);
```

# JSX Debugging Tip

Navigated to <http://localhost:3000/>

[timer-dashboard.js:183](#)

► Object {*\$\$typeof*: Symbol(react.element), *key*: null, *ref*: null, *props*: Object, *type*: function...}

> |

# JSX Debugging Tip

```
Navigated to http://localhost:3000/  
timer-dashboard.js:183  
▼ Object {$$typeof: Symbol(react.element), key: null, ref: null, props: Object, type: function...} ⓘ  
  $$typeof: Symbol(react.element)  
  key: null  
  ▼ props: Object  
    amount: "15"  
    elapsed: 0  
    id: 1  
    ▶ onResetClick: function (timerId)  
    ▶ onSetClick: function ()  
    ▶ onStartClick: function (timerId)  
    ▶ onStopClick: function (timerId)  
    runningSince: undefined  
    troy: "miles"  
    ▶ __proto__: Object  
  ref: null  
  ▶ type: function Timer()  
  ▶ _owner: ReactCompositeComponentWrapper  
  ▶ _store: Object  
  ▶ _self: EditableTimer  
  ▶ _source: Object  
  ▶ __proto__: Object  
>
```



# Lifecycle Events

Event	When
componentWillMount	invoked once before rendering
componentDidMount	invoked after component loaded
componentWillReceiveProps	invoked when receiving new props
shouldComponentUpdate	asks if component should update
componentWillUpdate	invoked before rendering new props
componentDidUpdate	invoked after rendered new props
componentWillUnmount	invoked before component removed

# The Bridge

# Styles

# Flexbox

# NativeBase

# Navigation

# Ajax with Fetch

# Debugging



- App.js is where the code begins
- index.js registers the component

```
— __tests__
— android
— ios
— node_modules
— .babelrc
— .buckconfig
— .flowconfig
— .gitattributes
— .gitignore
— .watchmanconfig
— App.js
— app.json
— index.js
— package.json
— yarn.lock

4 directories, 11 files
```



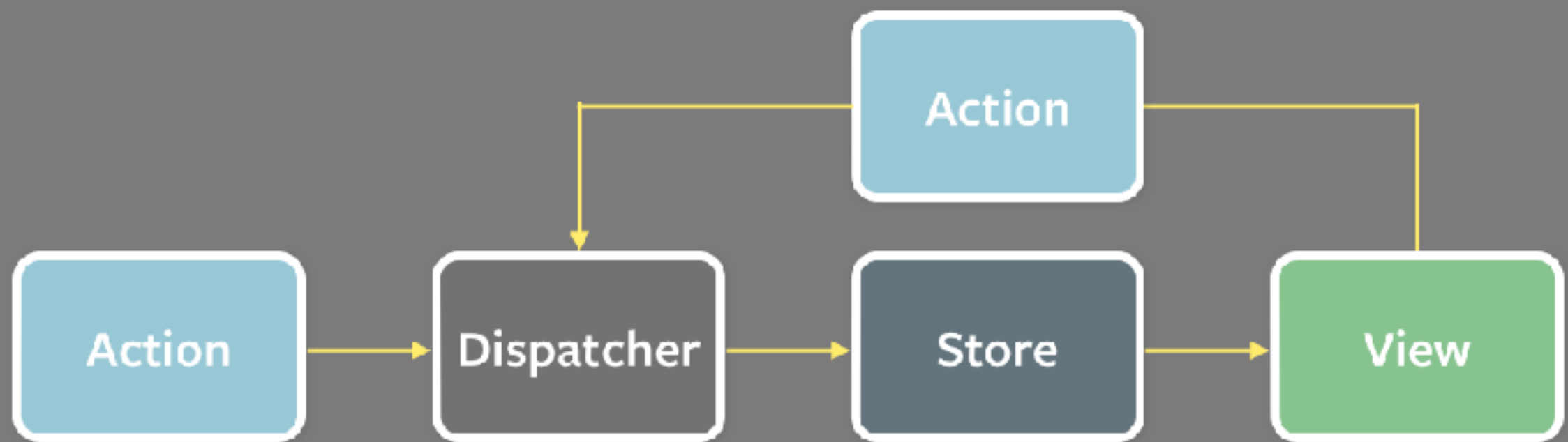
Flux

# Flux

- Application architecture for building user interfaces
- A pattern for managing data flow in your app
- One way data flow
- 4 main parts: Dispatcher, Store, Action, & View

# The 4 main parts

- Dispatcher: receives actions & dispatches them to stores
- Store: holds the data of an app
- Action: define app's internal API
- View: displays the data from stores





Redux

# Redux

- A predictable state container for JS apps
- Works well with React Native
- An alternative to and inspired by Flux
- Single store for the entire app
- Makes it easier to hot-load your app
- Created by Dan Abramov

# 3 Principles

- Single source of truth
- State is read-only
- Changes are made with pure functions



# Actions

- Actions are payloads of information that send data from your application to your store.
- They are sent using `store.dispatch()`
- They are JavaScript objects
- Typically have a `type` property which defines their action
- The type is normally a string

# Action Creators

- Functions that create actions
- This makes them both portable and easy to test
- (In Flux, the creator usually triggers a dispatch)

# Reducers

- Take the current state and an action and return the new state
- It is important for reducers to stay pure

# A Reducer Shouldn't

- Mutate its arguments
- Perform side effects like API calls or routing transitions
- Call non-pure functions like `Date.now()`

# Store

- Holds app state
- Allows access to the state via `getState()`
- Allows state to be updated via `dispatch(action)`
- Registers listeners via `subscribe(listener)`

# Redux Data Lifecycle

- Call `store.dispatch(action)`
- Redux store calls the reducer function
- The root reducer combines the output of multiple reducers into a single state tree
- The redux store saves the complete state tree
- (`component.setState(newState)` called)

# Middleware

- Redux is optimized for a synchronous workflow
- Middleware occurs between the dispatcher and the reducer
- Can be used for: logging, optimistic updating, etc.

# React-Redux

- Binds Redux to React
- Redux middleware
- `npm i -S react-redux`



# context

- Similar to props except doesn't have to be explicitly passed
- Any child can have access to context no matter how far down it is
- Libraries use context to create provider components

# connect()

- Separates the concerns of the container and presentational components
- Takes 2 arguments
  - a function that maps app state to a JS object
  - a function that maps the store's dispatch to props
- returns a new function

# Binders

- `mapStateToProps(state)`
  - Maps state values to the component's props
- `mapDispatchToProps`
  - Binds dispatch actions of the component to the store

# Component Types

	<b>Presentational</b>	<b>Container</b>
Purpose	How things look	How things work
Redux Aware	No	Yes
Read data	Read from props	Subscribe to Redux State
Change data	Invoke callbacks	Dispatch Redux actions
How written	By hand	Generated by react-redux

# System Components

Component
ActivityIndicator
Button
DatePickerIOS
DrawerLayoutAndroid
FlatList
Image
KeyboardAvoidingView

# System Components

Component
ListView
Modal
NavigatorIOS
Picker
PickerIOS
ProgressBarAndroid
ProgressViewIOS

# System Components

Component
RefreshControl
ScrollView
SectionList
SegmentedControlIOS
Slider
SnapshotViewIOS
StatusBar

# System Components

Component
Switch
TabBarIOS
TabBarIOS.Item
Text
TextInput
ToolbarAndroid
TouchableHighlight



# System Components

Component
TouchableNativeFeedback
TouchableOpacity
TouchableWithoutFeedback
View
ViewPagerAndroid
VirtualizedList
WebViewAPIs

# Platform Differences

- Not all components work with all devices
- Some are for iOS only or Android only
- Those that end in IOS are for iOS
- Those that end in Android are for Android

# Hot Links

- <https://github.com/rockncoder>
- <https://facebook.github.io/react-native/>
- <https://nativebase.io/>
- <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>

# Summary

- React Native allows you to make mobile apps in JS
- While easier than native development, it is not easy