React Native - Basics - Intro

- Dr Nick Hayward

A brief intro to the basics of React Native app development.

Contents

Intro

Inherently familiar to React developers, React Native offers a native mobile experience with React JS patterns and structures.

However, instead of creating reusable components for the web, it allows developers to use and create native components for Android and iOS.

As such, the basics of React development are still required for React Native development, including

- components
- JSX
- props
- state
- ...

first app - basic-app

A basic app for React Native will follow a known, prescribed pattern.

We can use the React Native CLI tool to generate a shell app for developing an app.

In a development directory, e.g. `/Development/react-native/`, we can issue the following command to generate project files for an app,

```
react-native init BasicApp
```

This command will call the React Native CLI, which will then initialise a new project named `BasicApp`. This project, and all of the necessary initial files, will be installed to a directory named `BasicApp` in the current working directory.

This command will also output some useful instructions for running an app on iOS and Android. It also details how to open the project files using Xcode.

n.b. this is not a fast process - it may time some time to initialise a new project

how to start an app - iOS on OS X

We can now start our initial project to test that it runs OK on OS X. We'll need to be in the project directory for the app

to load and run. Then, we can issue the following command in the terminal, e.g.

```
react-native run-ios
```

This command will build the project, launch the iOS simulator, and then show the app in a simulator window. For the initial build request, this may take a long time depending upon system performance.

n.b. if the app is not automatically loaded in the simulator window, simply drag the screens in the emulator, and then click on the React Native app icon to launch the app

how to start an app - Android on OS X

Assuming Android has been setup and configured correctly, running an app with Android follows the same pattern as iOS, e.g.

```
react-native run-android
```

Initial run will scan local machine for *symlinks*, starts JS server for development and testing, and then it will need to download and config Gradle for local Android setup. It starts to build and install the app in the CWD.

`basic app` - intro

We can now start to develop a basic app with React Native.

We'll add a basic screen, show a list of items from JSON, and render some images. We'll also respond to user interaction, and connect to a remote resource to show usage of links and external apps.

In particular, we'll consider how the fundamental structures and patterns working in React Native.

app - basic app directory structure

When we generate a new app with React Native CLI, we get a boilerplate structure for the initial app. However, there are a few important files to consider for app development on iOS and Android.

Basic structure is as follows,

```
|-- helloworld
    |__ __tests__
    |__ android
    |__ ios
    |__ node_modules
    |__ App.js
    |__ app.json
    |__ index.js
    |__ package-lock.json
    |__ package.json
    |__ ...
```

These are the main directories and files created as we initialise a new project.

All of the necessary files to build an app with React Native for iOS and Android are located in their respective directories. These are native project directories, and can be imported as native apps into Android Studio and Xcode. However, for most apps it is unlikely a developer will need to modify files in either directory. They can be useful for customisation or deeper config &c., but it's not necessary to modify these files to build most apps.

The `app.json` file includes brief metadata for a generated app. e.g. name, display name, and so on,

As is customary, node modules are located in the `node_modules` directory. Not necessary to modify these files and

directories. Likewise, the `package,json` file is a standard file for Node development, which contains metadata for the React Native app.

app - getting started

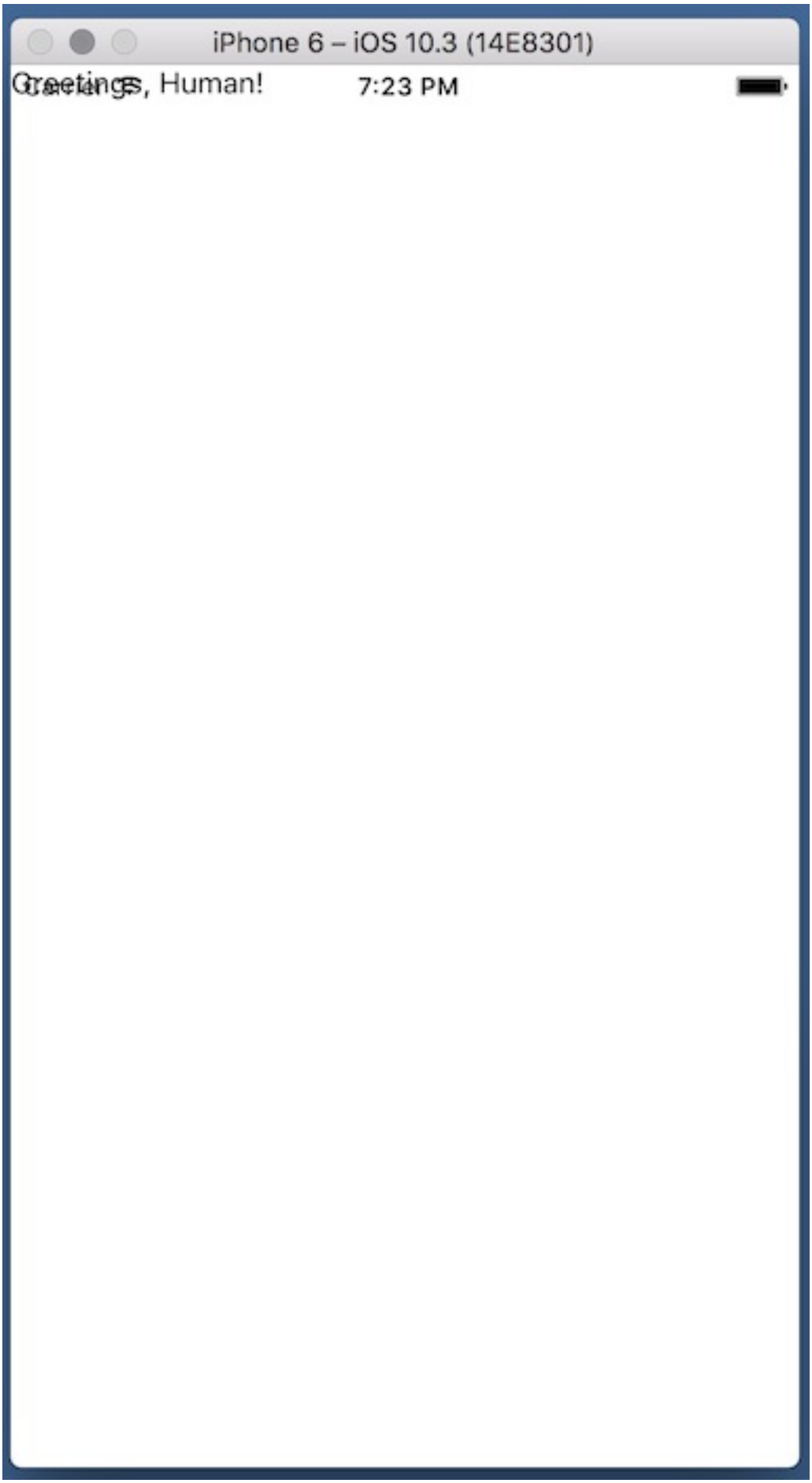
We can start a new `BasicApp` by clearing the biolerplate code from the `App.js` file. Then, we can add a basic component for a home screen message, e.g.

```
// import React, Component module as Component from base React
import React, { Component } from 'react';
// import Text as Text from React Native
import { Text } from 'react-native';

// default export - BasicApp - used when no explicit import reference...
export default class BasicApp extends Component {
  render() {
    return (
      <Text>Greetings, Human!</Text>
    );
  }
}
```

To use this new component within our app, we can register it in the default `index.js` file, e.g.

```
// import AppRegistry as AppRegistry
import { AppRegistry } from 'react-native';
// import App from App.js (.js implied...)
import App from './App';

// register new component as Basic App - pass default from App.js
AppRegistry.registerComponent('BasicApp', () => App);
```

If we now run this app in the iOS simulator, we'll see the following output



There's no styling or app UI structure at the moment, which is why the output text is rendered in the top left corner of the app screen.

This can be fixed by modifying the design and layout for an app.

app - new component

This app includes a new component, `BasicApp`, which we can use to render the Text component as the app starts.

For each new component, the only default requirement is the `render()` function. This function simply returns some JSX for rendering in the app.