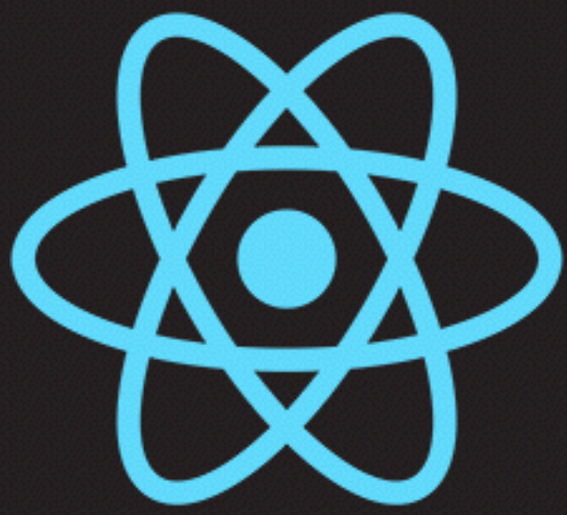
COLLEGIO TECHNOLOGIES INC.

REACT NATIVE: INTRODUCTION



WHAT IS REACT NATIVE?

- React Native is a framework that is built on top of React to allow you to build out mobile apps.
- A lot of frameworks in the past built out their apps via hybrid applications such as Ionic, but these methods had some major drawbacks:
  - They built out “hybrid”, or “HTML5” apps. These apps are basically a web interface that is serving custom HTML via JavaScript.
- React Native uses fundamental native UI components to develop Android and iOS apps, developing an app that is indistinguishable from a natively built app.
  - In fact, you can combine Swift and Java components into your apps if you needed a part of your app to interact with native code.

INSTALLING REACT NATIVE (OSX)

- If you are planning on working with React Native on OSX, we suggest working in the iOS environment, as opposed to Android.
- You will need the following in order to run a React Native application on OSX:
  - **XCode** - required for iOS development. Comes with a simulator!
  - **Node.js** (can be downloaded with Homebrew or via downloading at nodejs.org)
  - **watchman** - used for watching for changes
  - **The React Native CLI** - used for generating and compiling React Native projects
    - **npm install -g react-native-cli**
    - **yarn global add react-native-cli**

INSTALLING REACT NATIVE (WINDOWS)

- If you are using Windows as your Operating System, you are limited to testing your application on Android.
- You will need to install the following to work with React Native:
  - **Java SDK -** http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
  - **Node.js -** nodejs.org
  - **Python -** https://www.python.org/downloads/
  - **Android Studio -** https://developer.android.com/studio/index.html
  - **The React Native CLI** - used for generating and compiling React Native projects
    - **npm install -g react-native-cli**
    - **yarn global add react-native-cli**
- **Note:** There are tons of resources and installation videos for all of these on the web!

CREATING AND LAUNCHING A NEW REACT NATIVE PROJECT

- Once the React Native CLI is installed, a new project can be created within the command line prompt by moving to the directory, and typing the following command:
  - **react-native init <project-name>**

- Once the project is created, it is then ready to be run inside of a simulator:
  - **react-native run-ios (Mac)**
  - **react-native run-android (Windows)**

THE REACT NATIVE PROJECT STRUCTURE

- After you initialize a new React Native Project, you will notice that there are a lot of folders and files that look familiar:
  - **.babelrc**
  - **/node_modules**
  - **package.json**

- However, there are a few files missing that we are used to working with as well, such as the **app.js** file. For React Native projects, the default starting point is the **index.js** file located in the project root.

REACT VS. REACT NATIVE

- Like with our React web project which required use of both the React and ReactDOM libraries in order to work, React Native projects work in a similar manner.
  - The major difference is, instead of using '**react-dom**', we use '**react-native**'.

- The React Library knows how the components should interact with each other and other core React functionality, while the React Native Library is what renders the component to the device screen.

- Another major difference is in how we will display our elements: There are no native React Native elements available by default inside of your application, so you will need to import the library often within your code. For this reason, you can make use of importing **named imports** into your code.

EXAMPLE: BASIC REACT NATIVE SCRIPT (INDEX.JS)

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

// A very simple Stateless Functional Component
const App = () => (
    <Text>Hey There!</Text>
);

// Render the component
AppRegistry.registerComponent('freeAgentTracker', () => App);
```

JSX IN REACT NATIVE

- We are still using JSX inside of our React Native code, as you could tell from the previous example.
  - However, the major difference to note is that, instead of using HTML elements inside of our JSX to display elements on the screen, we are instead using containers for Native elements.

- We saw our first example of a React Native element when we used the **Text** element, which, as the name suggests, displays text to the screen.

- Let's review some of the common elements that you'll be using in React Native:

COMMON REACT NATIVE DISPLAY COMPONENTS

- **Text** - displays a text element to the screen.
- **TextInput** - displays text input
- **View** - used as a container for other display elements. Will be used to group elements together. Consider it like your React Native ** ** element.
- **Image** - used to display an image.
- **Button** - displays a button.
- **FlatList** - displays a simple list.

THE IMAGE CONTAINER

- The **Image** container is set up in a similar fashion to an **<img>** tag in HTML; however, there are some differences:
  - You declare the URI of the image location inside of the **source** attribute, that accepts an object with a **uri** attribute.
  - In addition to declaring the source, you must also declare the image size in the component style. If you do not declare image dimensions, the image will be rendered at 0x0 pixels.

```
<Image
    style={styles.avatar}
    source={{ uri: props.player.avatar_image }}
/>
```

```
const styles = {
    avatar: {
        height: 50,
        width: 50
    }
};
```

SCROLLABLE COMPONENTS USING SCROLLVIEW

- React Native components do not allow for scrolling by default. This means that if your component happens to be taller than the current screen size, you will be unable to scroll through the view.

- You can enable scrolling on a view element by using another React Native component: **ScrollView**.

  - The **ScrollView** component can be used in place of a view to enable scrolling.

  - **Please Note:** In order to use ScrollView, make sure to set the **flex: 1** style to the parent component container.

BUTTONS IN REACT

- In addition to the standard **Button** component, there are also several other options for creating components with optional effects. Some useful button components are:
- **Touchable Highlight** - touching the button changes the color temporarily.
- **Touchable Opacity** - touching the button adds a quick fade effect.

```
<TouchableOpacity onPress={props.onPressAction}>
    <Text>
        {props.children}
    </Text>
</TouchableOpacity>
```

REACT PROJECT STRUCTURE

- We will be setting up our project structure in a similar manner to previous projects, beginning with storing all of our code inside of a **/src** folder.
  - Inside of the **/src** folder, create a **/components** folder for all of your components.
  - We'll be adding additional folders here as we need them.

- As with our React projects, we can use **ES6 importing and exporting** to only load in the components that we need.

STYLING IN REACT NATIVE

- Although we won't be covering styling in React Native until the next lesson, it is good to know that, just like when we need to display React Native elements, we are not able to use a lot of the common CSS rules that we're used to.
  - There is a great list of available CSS properties located at https://github.com/vhpoet/react-native-styling-cheat-sheet

- React Native components use Flexbox to set layout. Flexbox is a way of structuring child elements within their parent elements. The basic Flexbox rules we'll use for alignment are:
  - **justifyContent: 'flex-start'** - vertical top
  - **justifyContent: 'center'** - vertical center
  - **justifyContent: 'flex-end'** - vertical bottom
  - **alignItems: 'flex-start'** - horizontal left
  - **alignItems: 'center'** - horizontal center
  - **alignItems: 'flex-end'** - horizontal right

EXAMPLE: BASIC STYLING OF A HEADER COMPONENT

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

const Header = (props) => {

    const { viewStyle, textStyle } = styles;

    return (
        <View style={viewStyle}>
            <Text style={textStyle}>{props.title}</Text>
        </View>
    );
};

const styles = {
    viewStyle: {
        backgroundColor: '#A5A5A5',
        justifyContent: 'center',
        alignItems: 'center',
        height: 60,
        paddingTop: 15,
        shadowColor: '#000',
        shadowOffset: { width: 0, height: 2 },
        shadowOpacity: 0.2,
        elevation: 2,
        position: 'relative'
    },
    textStyle: {
        fontSize: 20
    }
};

export default Header;
```

LINKING IN REACT NATIVE

- Linking in React Native is the process of communicating with other applications on the device, such as the web browser or email.
  - https://facebook.github.io/react-native/docs/linking.html

- You can use the **Linking.openURL()** function to open a URL in the browser.

```
<TouchableOpacity onPress={() => Linking.openURL(props.theUrl)}>
    <Text>
        Go to Webpage
    </Text>
</TouchableOpacity>
```

EXERCISE: YOUR FIRST REACT NATIVE APP!

- Let's take advantage of our first bits of React Native knowledge to create our first Native App. We'll create a spin on the Free Agent Tracker:
  - Instead of adding and managing free agents, we're going to build an app that will allow us to view the list of current free agents, and reach out to them to join our team!

- The app will require the following:
  - A listings view containing all of the available free agents.
  - An individual card view to display a singular free agent containing:
    - The free agent photo, name, sport, and message.
    - The option to email the individual to see if they want to join your team.

RESOURCES

- **Java SDK:** http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
- **Node.js:** https://nodejs.org/en/download/
- **Python:** https://www.python.org/downloads/
- **Android Studio:** https://developer.android.com/studio/index.html
- **Yarn:** https://yarnpkg.com/en/docs/install
- **Babel.js:**
  - **Official Site:** https://babeljs.io/
  - **Babel.js Live Demo:** https://babeljs.io/repl/
  - **Babel.js Plugins:** babeljs.io/docs/plugins
- **Github Links:** https://github.com/
- **React Native:**
  - **RN Styling CheatSheet:** https://github.com/vhpoet/react-native-styling-cheat-sheet
  - **Official Site:** https://facebook.github.io/react-native
  - **RN Docs:** https://facebook.github.io/react-native/docs/getting-started.html

DISCLAIMER

- Copyright 2018 Collegio Technologies Inc.
- All content within this presentation has been compiled by Collegio Technologies Inc. All material has been sourced through open source resources and source code and as such conforms to Canadian copyright laws and regulations. All rights reserved.