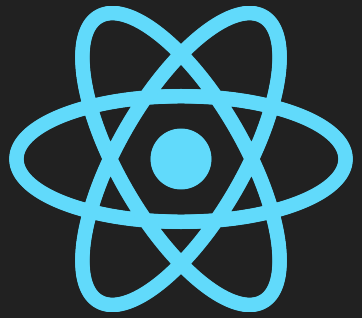

React Native

By Brennan Miller

CPSC575 Presentation 11/20/23

React Native and Why It’s Useful

- Framework for building native apps using JavaScript
- Allows for simultaneous development in iOS (Swift/Obj-C) + Android (Java/Kotlin)
  - Shared codebase means quicker development/easier maintenance
  - No need to use specific languages unless for very specific purposes
  - Can cut labor by eliminating the need for different dev teams
- Behind apps like Facebook, Instagram, Discord, UberEats and more

Expo

Expo is a set of tools that interlinks with the standard React Native CLI

Offers advantages such as

- Android/iOS/Web emulation in one package
- A mobile app for simple transfer for physical devices
- Quick reloads after code changes, making testing easier

Disadvantage: losing access to some native components. Not relevant unless getting super specific.

The Basics: View

- Fundamental component to structure the layout of an application.
  - Container that holds other components
  - Equivalent to layout containers/Viewgroups in Android dev.

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

const ViewBoxesWithColorAndText = () => {
  return (
    <View
      style={{
        flexDirection: 'row',
        height: 100,
        padding: 20,
      }}>
      <View style={{backgroundColor: 'blue', flex: 0.3}} />
      <View style={{backgroundColor: 'red', flex: 0.5}} />
      <Text>Hello World!</Text>
    </View>
  );
};

export default ViewBoxesWithColorAndText;
```

The Basics: Text/Image

- Used to display text/ images in application
  - Equivalent to TextView/ImageView in Android dev.
- If using an image downloaded from the web, dimensions must be specified or that image will not appear

```jsx
const TextInANest = () => {
  const [titleText, setTitleText] = useState("Bird's Nest");
  const bodyText = 'This is not really a bird nest.';

  const onPressTitle = () => {
    setTitleText("Bird's Nest [pressed]");
  };

  return (
    <Text style={styles.baseText}>
      <Text style={styles.titleText} onPress={onPressTitle}>
        {titleText}
        {'\n'}
        {'\n'}
      </Text>
      <Text numberOfLines={5}>{bodyText}</Text>
    </Text>
  );
};

const styles = StyleSheet.create({
  baseText: {
    fontFamily: 'Cochin',
  },
  titleText: {
    fontSize: 20,
    fontWeight: 'bold',
  },
});

export default TextInANest;
```

Bird's Nest

This is not really a bird nest.

The Basics: StyleSheet

- Defines styles of components
  - Equivalent to XML style files in Android dev.
- Advantages of StyleSheets over raw JavaScript
  - Validates is properties, prevents mundane errors like misspellings
  - Minimizes the cost of creating/applying styles

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

const App = () => (
  <View style={styles.container}>
    <Text style={styles.title}>React Native</Text>
  </View>
);

const styles = StyleSheet.create({
  container: {
    flex: 1,
    padding: 24,
    backgroundColor: '#eaeaea',
  },
  title: {
    marginTop: 16,
    paddingVertical: 8,
    borderWidth: 4,
    borderColor: '#20232a',
    borderRadius: 6,
    backgroundColor: '#61dafb',
    color: '#20232a',
    textAlign: 'center',
    fontSize: 30,
    fontWeight: 'bold',
  },
});

export default App;
```

The Basics: Touchables

- Used to create touchable elements that respond to user interaction
  - Somewhat similar to buttons in Android dev. but not entirely
- TouchableWithoutFeedback, TouchableOpacity, TouchableHighlight
- onPress, onLongPress: for user tap/hold input distinction

```
return (
    <TouchableHighlight
        onPress={props.buttonTapHandler}
        style={{...styles.button, ...props.buttonStyles}}
    >
        <Text>{props.buttonText}</Text>
    </TouchableHighlight>
);
```

The Basics: Button

- It’s…well… a button.
- Minimal level of customization.

```
  </Text>
  <Button
    title="Press me"
    color="#f194ff"
    onPress={() => Alert.alert('Button with adjusted color pressed')}
  />
</View>
<Separator />
<View>
  <Text style={styles.title}>
    All interaction for the component are disabled.
  </Text>
  <Button
    title="Press me"
    disabled
    onPress={() => Alert.alert('Cannot press this one')}
  />
</View>
```

Buttons vs Touchables

Key differences:

- Button: high-level abstraction, meaning platform specific styling. Includes all built in accessibility features. Generally only takes ‘title’ as child.
- Touchable: low-level abstraction, no predefined styling. Significantly more customizable. Multiple child components, allowing for more complex structures.

The Basics: Alert

- Launches alert dialog
- By default, looks like OS standard alert with ‘Ok’ button
  - Is customizable
- For iOS only, an alert is available that prompts the user to enter information

```
const createTwoButtonAlert = () =>
  Alert.alert('Alert Title', 'My Alert Msg', [
    {
      text: 'Cancel',
      onPress: () => console.log('Cancel Pressed'),
      style: 'cancel',
    },
    {text: 'OK', onPress: () => console.log('OK Pressed')},
  ]);

const createThreeButtonAlert = () =>
  Alert.alert('Alert Title', 'My Alert Msg', [
    {
      text: 'Ask me later',
      onPress: () => console.log('Ask me later pressed'),
    },
    {
      text: 'Cancel',
      onPress: () => console.log('Cancel Pressed'),
      style: 'cancel',
    },
    {text: 'OK', onPress: () => console.log('OK Pressed')},
  ]);
```

Final Tidbits

- At points where you would need to ascertain which OS a device is using and make changes based on that, use Platform.OS
  - Eg. “if (Platform.OS == ‘ios’) { do thing;}
- Many imports will also require a manual command line import as well, so be aware of that

Used For This Demo

- Node.js Version 15+ (I used curr. version) https://nodejs.org/en/
- Expo Client Mobile App
- I used VS Code as it appears to be industry standard
  - Extensions: React Native Tools by Microsoft, React-Native Snippets by EQuimper
  - Not necessary, but if you are considering using over long period of time, recommended.
- Android Studio for emulation

Creating a Project

https://reactnative.dev/docs/components-and-apis <- API

Install node.js and npm (comes with node), will automatically open terminal and queue up additional download

- Node --version, npm --version to verify installation

“npm install --global expo cli” <- expo installation

“Npx create-expo-app -- template” <- project creation

- Will create menu that allows template selection, pick “Blank”, name app, start coding

Remember to cd into the actual projects before any imports, trying to start etc.

“npm start” to launch Expo