Mobile Development (CM3050)

Couurse Notes

Felipe Balbi

October 12, 2021

Contents

Week 1

Key Concepts

- Understand the limitations and advantages of different platforms
- Discuss the elements of apps you enjoy
- Understand the course structure

1.006 Getting started on this module

React Native is an Open Source framework created by Facebook which lets us JavaScript to produce native code for different operating systems. Some of the benefits are:

1. Single language for multiple platforms
2. Ease and flexibility of JavaScript
3. Near-native performance
4. Heavily used in industry

1.103 The pathway from developer to consumer

The process to moving an app from developer to consumer is roughly laid out below.

1. Ideation

   The idea for an app has to come from somewhere

2. Storyboarding

   Drawing an idea is usually the simplest form of conveying that idea

3. Prototyping

   Helps us understanding challenges that may arise from implementing the idea

4. Feedback and Production

   Using our prototype, we can gather feedback to understand if the app fulfills its purpose

5. Testing

   Automated testing and manual testing is an important part of guaranteeing stability and quality of the app

6. Approval and Release

   After testing, apps go through an approval process by e.g. Google and Apple before they can be released to end users

7. On sale

   Apps are finally available in the App Store and can be purchased by anyone

1.201 Multiple codebases

There are two major mobile OSes:

- iOS ($\approx 26.5\%$ market share)
- Android ($\approx 72.9\%$ market share)

This means that developers usually have to write the same app twice — once for iOS with Switch of Objective-C, and once for Android using Kotlin or Java —; needless to say this creates the possibility of minor incompatibilities between the two versions of the app.

The alternative to writing two apps is to write a single app in a common language that can be compiled for both platforms.

Native Apps Written specifically for a single OS

Hybriid Apps Written in a common language

1.203 Native apps vs hybrid apps

React Native, while using a common language for all platforms, is much closer to a native application. During compilation of the project, React Native renderas our views with native code on the target platform thus giving us the best of both worlds.

In table 1 we list pros and cons of Native Apps. Similarly, table 2 shows the same comparison for Hybrid Apps.

Table 1: Native Apps Pros & Cons

Pros	**Cons**
Fast	Lengthy and Complex to build
Device-level APIs	No cross-compatibility
Native GUI Elements	

Table 2: Hybrid Apps Pros & Cons

Pros	**Cons**
Cross-compatibility	Slower perforance
Larger market reach	No device-level APIs
Faster to write	

1.301 What is React Native?

React Native is a framework written in JavaScript that allows us to build code for multiple platforms.

1.302 What is Expo?

Expo helps manage a React Native project.

1.305 Javascript arrow notation

A traditional JavaScript function looks like this:

```
function (a) {
  return a * 2;
}
```

We can convert it Arrow Notation like so:

```
(a) => {
  return a * 2;
}
```

Single line functions can be further simplified by removing braces and the `return` keyword:

```
(a) => a * 2;
```

Next, the parenthesis around `(a)` can also be removed:

```
a => return a * 2;
```

Functions without arguments or with more than one argument must keep parenthesis around arguments:

```
() => 42;
(a, b) => a**2 + 2*a*b + b**2;
```

1.306 Creating a new React Native project and overview of the file structure

Create a new project with:

```
$ expo init --npm
```

Choose a name for the application and choose the `blank` template. Hitting `ENTER` will create a new directory with the name of the application and all dependencies already installed. When it completes, we can change into the new application's directory and run:

```
$ expo start
```

Week 2

Key Concepts

- Understand the limitations and advantages of different platforms
- Discuss the elements of apps you enjoy
- Understand the course structure

1.401 JSX and props

JSX is a key part of React Native. It allows us to create components and use them as if they were regular HTML tags. Component names must start with a capital letter.

Props can be passed in an HTML `attribute=value` format.

1.501 Testing on simulators and devices

Testing the application on a device while under development can help us uncover issues early. The simulator is another option which works well.

1.601 Marketplaces and conforming to the rules

Reasons for rejection

1. App doesn't work
2. Use of copyrighted material
3. Breach of safety guidelines
4. Safeguarding issues
5. Circumventing the app stores
6. Low-quality design

Week 3

Key Concepts

- Understand the need for wireframing
- Discuss the link between psychology and design decisions
- Understand and identify different design styles and replicate them using code

2.001 Skeuomorphism, minimalism and neumorphism

Skeuomorphism emulating the look of physical objects in your application

Minimalism prioritization of essential elements

Neumorphism the middle-ground between skeuomorphism and minimalism

2.003 Colour palettes and mood psychology

Blue calmness, serenity, stability, reliability

Red & Yellow speed, immediacy, increase appetite

White simplicity, calmness, professionalism

2.006 Design ideology

- Apple Human Interface Guidelines
- Google Material Design Guidelines

2.101 What are dark patterns?

Dark Patterns is the term given to design choices that are crafted to make you do things you don't want to do.

2.103 Dark patterns

- Dark Patterns website

2.201 Introduction to wireframing

Wireframes allow us to design layouts in order to demonstrate an idea and check if it's likely to be practical.

2.203 Creating a wireframe for a shopping app

Before wireframing, it's a good to create a User Flow Diagram. After we can start working on our wireframe.

Week 4

Key Concepts

- Understand the need for wireframing
- Discuss the link between psychology and design decisions
- Understand and identify different design styles and replicate them using code

2.401 How to style elements in React Native

Styles, in `React`, are created with `StyleSheet.create()` method. We can use them by setting the `style` property of the element to the relevant object. Like so:

```
import { StatusBar } from 'expo-status-bar';
import React from 'react';
import { StyleSheet, Text, View } from 'react-native';

export default function App() {
  return (
      <View style={styles.container}>
        <Text>Open up App.js to start working on your app!</Text>
        <StatusBar style="auto" />
      </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#fff',
    alignItems: 'center',
    justifyContent: 'center',
  }
});
```

In this case, the `<Text>` and `<StatusBar>` elements are children of the `<View>` element and will, therefore, *inherit* its styling.

2.501 What is responsive design?

With so many different models of devices in the market, each with a different pixel density, screen sizes, and resolution, designing a single app that works on all form factors is a rather large task.

Responsive design is a technique where the application's UI adapts to the screen dimensions. We do this by building apps designed for proportions, rather than specific sizes.

In ReactNative, Flexbox is used to provide a consistent layout on different screens.

2.504 React Native flex

- React Documentation: Flexbox

2.505 Apple developer guidelines

- Apple Developer Guidelines: Adaptivity And Layout

2.601 Design with accessibility in mind

When building an application, four categories should be kept in mind:

1. Vision

   Blindness, color blindness and all forms of vision impairment

2. Hearing

   Hearing loss and other hearing disabilities

3. Physical and motor

   People with physical and motors impairment, may experience difficulties holding or manipulating their devices

4. Literacy and learning

   These include difficulty speaking, reading, managing complexity and maintaining attention or focus.

ReactNative wraps Android and iOS specific accessibility features.

`accessible` **Prop** When true, indicates the view is an accessibility element.

`accessibilityLabel` **Prop** Allows us to assign descriptive label to an element.

`accessibilityHint` **Prop** Describes what will happen when the user performs the action on the element.

Accessibility Actions Allow assistive technologies to programmatically perform actions.

2.603 Accessibility in computational technology

- Antona, M. and C. Stephanidis (eds) Universal Access in Human-Computer Interaction. Design Approaches and Supporting Technologies. 14th International Conference, UAHCI 2020, Held as Part of the 22nd HCI International Conference, HCII 2020. (Springer International Publishing, 2020).