VIETNAM NATIONAL UNIVERSITY - HO CHI MINH
UNIVERSITY OF SCIENCE

Coursera - React Native of Meta

Module 1

Student:

23127438 - Dang Truong Nguyen

Academic Supervisors:

Bui Duy Dang

Huynh Lam Hai Dang

Tran Trung Kien

Semester: I
Academic Year: 2025 - 2026

Ho Chi Minh City, October 23, 2025

TABLE OF CONTENTS

TABLE OF CONTENTS 1

LIST OF FIGURES 4

1 Course Introduction 5

1.1 Video: Introduction to the course . . . 5

1.1.1 Perspective . . . 5

1.1.2 Video: Learning Object . . . 5

1.2 What is React Native . . . 5

1.2.1 Video: What is React Native? . . . 5

1.2.2 Native application . . . 6

1.3 Video: Native, cross-platform & hybrid developer roles . . . 7

1.3.1 Native development . . . 7

1.3.2 Video: Cross-platform Application . . . 7

1.3.3 Video: Hybrid app - another category of cross-platform apps . . . 7

1.4 Video: Meet a cross-platform Developer . . . 8

1.5 Dialog: Bridging the Gap: Your React Foundation for Mobile Apps . . . 9

1.6 Reading: Course Syllabus . . . 9

1.7 Reading: How to be successful in this course . . . 10

1.8 Video: React Native with Expo . . . 10

1.8.1 Required materials . . . 10

1.8.2 Create a sample React Native App . . . 11

2 Introduction to React Native 12

2.1 Video: How is React Native used in the real world? . . . 12
2.2 Video: React Native Code . . . 13
2.2.1 Sample code: . . . 13
2.3 Reading: Benefits of React Native . . . 14
2.4 Video: What is Expo . . . 14
2.4.1 What is Expo? . . . 14
2.4.2 Over the Air (OTA) . . . 14
2.4.3 API for Expo . . . 15
2.4.4 Some caution about Expo . . . 15
2.5 Reading: Building React Native apps with Expo . . . 15
2.6 Knowledge check: Introduction to React Native . . . 16

3 React Native Components 17

3.1 Video: What are React Native Components . . . 17
3.1.1 Basic Concept . . . 17
3.1.2 Categorization . . . 17
3.2 Video: Building a components . . . 18
3.3 Reading: Exploring building a components . . . 19
3.4 Reading: Exercise: Your first React Native components . . . 19
3.5 Self review: Your first React Native components . . . 20
3.6 Knowledge checking: React Native Components . . . 20

4 Views, Text and Scrollable components 21

4.1 Video: What are View and Text components? . . . 21
4.1.1 View . . . 21
4.1.2 Text . . . 21
4.2 Video: Using the View and Text Components . . . 21
4.3 Reading: Exercise: Build a React Native Screen . . . 22

4.4 Self Review: Build a React Native screen . . . 22
4.5 Video: What is the ScrollView component? . . . 23
4.5.1 Problem . . . 23
4.5.2 ScrollView . . . 23
4.6 Video: Using the ScrollView component . . . 23
4.7 Reading: Exercise: Build a scrollable component . . . 24
4.8 Practice Assignment: Self review: Build a scrollable component . . . 24
4.9 Practive Assignment: Knowledge check: Views, Text and Scrollable components 25

5 Styling components 26

5.1 Video: StyleSheet API . . . 26
5.2 Video: Practical Styling . . . 27
5.3 Reading: Styling using StyleSheet . . . 29
5.4 Reading: Exercise: Style a component . . . 29
5.5 Self review: Style a component . . . 33
5.6 Video: Module summary: Introduction to React Native . . . 33
5.7 Module quiz: Introduction to React Native . . . 35

LIST OF FIGURES

Figure 1.1 My dialog about React Foundation with coach 9

Figure 1.2 Sample React Native 11

Figure 2.1 Knowledge check: Introduction to React Native 16

Figure 3.1 React Native Components 18

Figure 3.2 Exercise: Your first React Native components 19

Figure 3.3 Self review: Your first React Native components 20

Figure 3.4 Knowledge checking: React Native Components 20

Figure 4.1 Exercise: Build a React Native Screen 22

Figure 4.2 Self Review: Build a React Native Screen 22

Figure 4.3 Scroll View practice 23

Figure 4.4 Exercise: Build a scrollable component 24

Figure 4.5 Self review: Build a scrollable component 24

Figure 4.6 Knowledge check: Views, Text and Scrollable components 25

Figure 5.1 Self review: Style a component 33

Figure 5.2 Module quiz: Introduction to React Native 35

CHAPTER 1

Course Introduction

1.1. Video: Introduction to the course

1.1.1. Perspective

- Developing a mobile app for a client - Little Lemon, a restaurant served in a casual environment.

1.1.2. Video: Learning Object

- React Native.
- Basic Components.
- Core Components: View, Text, Scroll View.
- FlatList, SectionList, TextInput.
- Pressable, images, hooks.
- Navigating: Stack, Tab, Drawer.
- Graded assessment.

1.2. What is React Native

1.2.1. Video: What is React Native?

- Open-source JS Lib based on React that is used to build cross-platform native mobile apps.
- Created by Meta, used for Facebook app.
- Can be used to build:

- iOS
- Android
- Windows
- TV Apps

1.2.2. Native application

- **Native apps:** the app was designed for that specific device and OS.
- Compiling process: Javascript → Native code → Device processor.
- Contact with cross-platform through API (Application Programming Interfaces).
- Sample code for React Native:

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

const WelcomeApp = () => {
  return (
    <View
      styles={{
        flex: 1,
        justifyContent: 'center',
        alignItems: 'center'
      }}
    >
      <Text>Welcome to React Native</Text>
    </View>
  )
}
```

- **Practive Question:** To be proficient in React Native, which programming languages should u also be proficient in? → Javascript, HTML, CSS.

1.3. Video: Native, cross-platform & hybrid developer roles

1.3.1. Native development

- Platform: iOS, Android, Windows.
- iOS:
  - Languages: Swift, Objective-C.
  - A macbook is required to build an iOS app.
- Android:
  - Languages: Java, Kotlin (Not exclusive to Android)

1.3.2. Video: Cross-platform Application

- Apps built with a single codebase for multiple platforms.
- Cross-platform developer: programmer who develop cross-platform application.
- Other framework beside React Native: Flutter, Xamarin.
- React Native is still more popular due to its large support libraries.

1.3.3. Video: Hybrid app - another category of cross-platform apps

- Hybrid developer: programmer who develop hybrid application.
- Hybrid app do not have access to built-in native features of the devices.
- Integrate native features to the app.
- Render the graphic through the browers
- **Practive Question:** You develop apps in React Basic that use the same codebase for multiple mobile platforms. What kind of mobile developer does this make you? → Cross-platform.

1.4. Video: Meet a cross-platform Developer

- Eric Hartzog: SE at Meta.
- What is rewarding about being a cross-platform dev?
  - You have an idea for building an app on iOS and you build it, you're going to learn all these technologies, all these languages, all these frameworks. You're going to get it, start it and create it and ship it.
  - Later on, if you want to build something in virtual reality, you'll be able to use those same skills that you learn to bring an experience in virtual reality. This is something to be very excited about.
- What technical skills and soft skills are used in cross-platform developer?
  - Technology, Framework, Languages, Tools.
  - Communication with other developers, product managers, ...
  - Good practice in school to work with teammates.
- What is your favorite part of being a cross-platform dev?
  - Ship products to people.
  - Get some feedback, help people out.
- What do you value most about your work as a cross-platform dev?
  - Using a cross platform framework to build an application that is deployed, ship to somebody, and then makes a difference in their life.

1.5. Dialog: Bridging the Gap: Your React Foundation for Mobile Apps

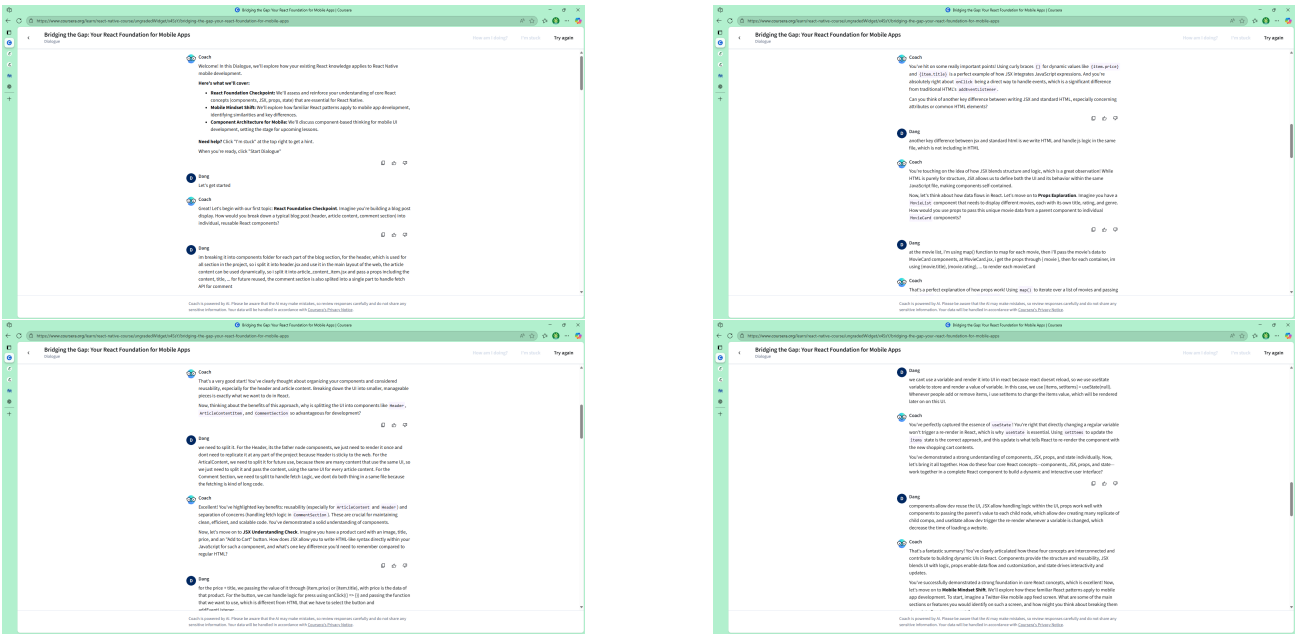

Figure 1.1: My dialog about React Foundation with coach

1.6. Reading: Course Syllabus

Week 1: Introduction to React Native

- Overview of React Native and its role in mobile development.
- Setting up the development environment and core components.

Week 2: Lists and Text Input

- Techniques for rendering large lists and managing user input.
- Creating a login screen using TextInput.

Week 3: Interactivity and Visuals

- Adding clickable areas with Pressable and displaying images.
- Utilizing advanced hooks for better responsiveness.

Week 4: Navigation

- Setting up React Navigation for screen transitions.
- Configuring header bars and navigation systems.

Week 5: Final Assessment

- Building a newsletter sign-up page and peer review.
- Reflecting on the learning experience and next steps.

1.7. Reading: How to be successful in this course

- Set daily goals for studying.
- Create a dedicated study space.
- Schedule time to study on your calendar.
- Keep yourself accountable.
- Actively take notes.
- Join the discussion.
- Do one thing at a time.
- Take breaks.

1.8. Video: React Native with Expo

1.8.1. Required materials

- Install Node.
- Install emulator. I'm using android studio.
- Install IDE. I'm using VSCode.

1.8.2. Create a sample React Native App

- Command:

```
npx create-expo-app [app_name]
```

- My sample React Native App:

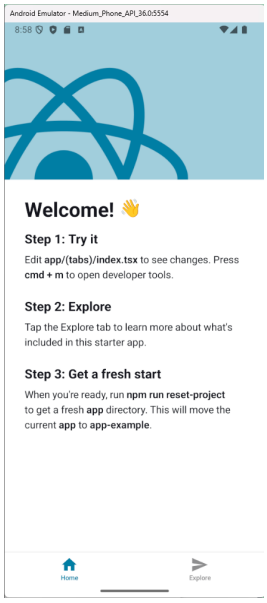

Figure 1.2: Sample React Native

- **Practive Question:** Before you install the Expo package, which preliminary steps should you take? → Install an emulator, Node, IDE

CHAPTER 2

Introduction to React Native

2.1. Video: How is React Native used in the real world?

Financial institution

- Adopted React Native to build their consumer mobile app for both iOS and Android.
- Instead of developing a separate iOS version and an Android version in parallel with no way to share code, choosing React Native instead to share code and reduce the delays and repetition.

Automatic Updates

- Automatically refreshes codes that accelerated the release of new product features.
- This resulted in avoiding delays in getting approval from App Stores for a new version of the app.

Multinational e-commerce platform provider.

- Developer who have used React switch to React Native quickly for their mobile apps since it uses React as well.
- The existing code could have potential reuse and the learning curve for moving from react to React Native is relatively low.
- The shared code between iOS and Android is 95 percent shared.

Financial Advisor

- Reducing the time to market.

- Their development velocity was very high and they were able to ship the first version of the app on both iOS and Android within five weeks of development. So they need to collaborate between the web and mobile team.

- **Practive Question:** What are the biggest benefits of using React Native are? Select all that apply. → Sharing of ideas and code between web and mobile or React and React Native, Sharing of code base across platforms, The speed of development.

2.2. Video: React Native Code

2.2.1. Sample code:

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

export default function WelcomeApp(){
  return (
    <View
      style={{
        flex: 1,
        justifyContent: 'center',
        alignItems: 'center'
      }}
    >
      <Text>Welcome to React Native</Text>
    </View>
  )
}
```

- **Practive Question:** Which of the following statements are true about React Native Code? → React Native code is made up of components and follows component design, React Native code is written in JavaScript and React.

2.3. Reading: Benefits of React Native

- Uses Javascript.
- Uses React.
- Builds Cross-Platform Native Apps.
- Cost Effective.
- Developer Experience.
  - Fast Refresh
  - Easy Debugging.
  - Over-the-Air Updates

2.4. Video: What is Expo

2.4.1. What is Expo?

- Expo is an open-source platform that is used for making cross platform native apps using React Native.
- Expo adds a layer of abstraction on top of React Native apps.
- It provides tools built for React Native, which gets you set up within a few minutes without much hassle.
- With Expo, however, you will never touch any native IOS or native android code like Swift, Java, or Kotlin because Expo automatically makes native code compatible with React Native code, and it is not available to the developers who are using it.

2.4.2. Over the Air (OTA)

- Push updates to your app anytime Over The Air not need app store approvals to push updates, reducing time waiting on approval from apps stores to publish updates to their apps.

2.4.3. API for Expo

- Camera.
- File systems.
- Location services.
- Push notification.

2.4.4. Some caution about Expo

- Expo does not have all the IOS and Android APIs.
- Expo has support from many device APIs but not all of them.
- The size of your app is not lean.
- Expo can be ejected from your app later on.
- **Practive Question:** Expo is an open-source platform that adds a layer of abstraction on top of React Native apps. Which of the following are benefits of this? Check all that apply.
  - You will only need a recent version of Node.js and an emulator to start building mobile apps in React Native using Expo.
  - With Expo, you will be able to start right away with minimal setup and build a complete cross-platform app in React Native.
  - Expo has a feature to do Over The Air (OTA) updates.

2.5. Reading: Building React Native apps with Expo

- Build with npm:

```
npx create-expo-app FirstProject
cd FirstProject
npm start
```

- Instead of using `npm start`, we can also use `npx expo start`.

- Build with yarn:

```
yarn create expo-app FirstProject
cd FirstProject
yarn start
```

- Instead of using `yarn start`, we can also use `yarn expo start`.

2.6. Knowledge check: Introduction to React Native

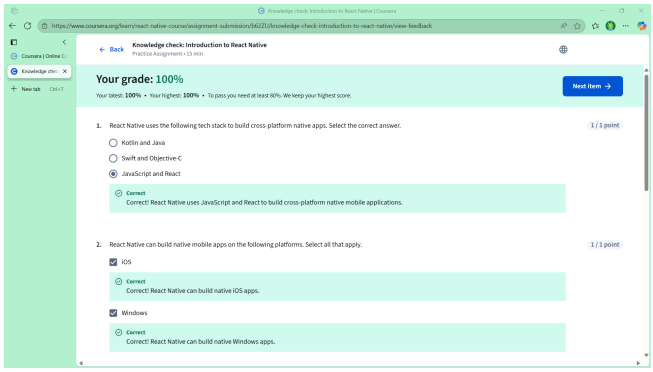

Figure 2.1: Knowledge check: Introduction to React Native

CHAPTER 3

React Native Components

3.1. Video: What are React Native Components

3.1.1. Basic Concept

- In React, you use a basic building block to build the UI, namely a component. Components let you split the UI into independent reusable pieces.
- When you put all of these independent pieces of components together, you build a complete app.
- The same concept is true for React Native since it is built using React.
- Some reusable React's components:
  - Headers
  - Footers
  - Menu bars
  - Images

3.1.2. Categorization

- Core components
  - View, Text, Image, TextInput, ScrollView, ...
  - They are ready to use.
  - They translate into native iOS and native Android components. This means they can adapt to work with your device's native functionality without the need for specialized code. You do not need to concern yourself about how that happens.

- Community components
  - React Navigation, React Native Screen, React Native Maps, React Native Videos, ...
  - The original package by itself is quite lean.
- Your native components
  - Your native components and you own them.
  - Written in native code. Built by developers who are experienced in native mobile development.
  - You can make them available as open source for the community.
- **Practive Question:** Which component type is built into the react-native package and comes ready-to-use? → Core components

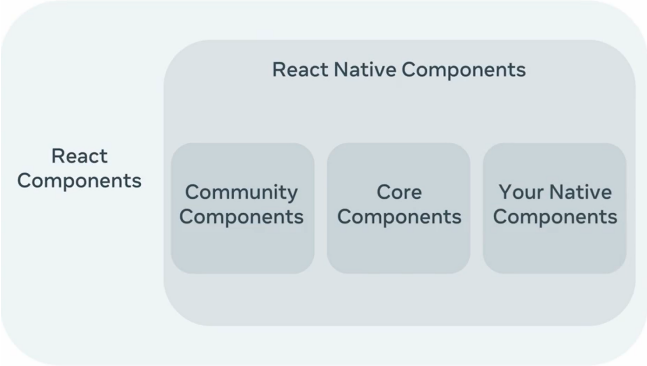

Figure 3.1: React Native Components

3.2. Video: Building a components

- Create a components folder. Add a `LittleLemonHeader.js` to this folder.
- Implement this header:

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


export default function LittleLemonHeader(){
```

```
        return (
          <View>
            <Text>Littel Lemon Restaurant</Text>
          </View>
        )
      }
```

- Use it in the `main.js`.
- **Practive Question:** Custom Components in React are reusable throughout the app. True or false? → True

3.3. Reading: Exploring building a components

This part is the same way to build a component as the above section.

3.4. Reading: Exercise: Your first React Native components

- This part is about creating a footer components after the header work.

Figure 3.2: Exercise: Your first React Native components

3.5. Self review: Your first React Native components

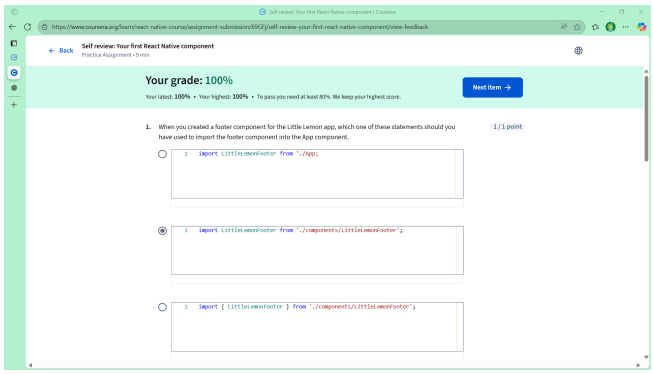

Figure 3.3: Self review: Your first React Native components

3.6. Knowledge checking: React Native Components

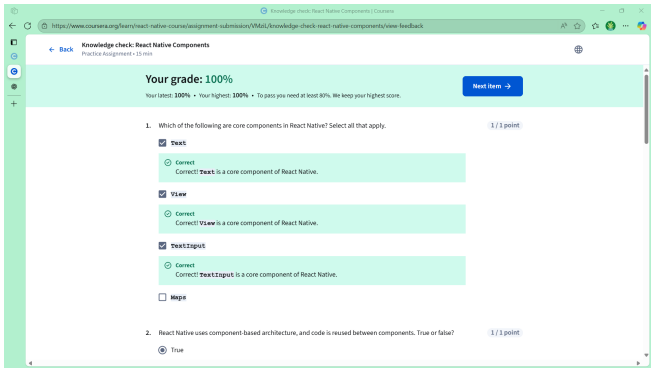

Figure 3.4: Knowledge checking: React Native Components

CHAPTER 4

Views, Text and Scrollable components

4.1. Video: What are View and Text components?

4.1.1. View

- View is the basic building block of the user interface.
- It is a small rectangular element on the screen, houses all elements within it.
- The view component is like a non-scrolling div tag in the web world.

4.1.2. Text

- A basic core component from React Native that is used for displaying text.
- Support styling, nesting and touch handling.
- **Practive Question:** The `View` component can be nested inside other views and can have zero or many children within it. True or false? → True

4.2. Video: Using the View and Text Components

- Using flex: 0.5 → the child component take 50% of parent's space.
- The text must be rendered within a text components.
- Use nested text components, the child text components can be inherited the style from parent's components, we can also override the style.
- Using `numberOfline=1` to avoid wrap.

- **Practive Question:** You have created a View component for your app and would like it to take 50% of the space on the screen. Which of the following styling parameter settings would you apply to achieve this? → flex: 0.5

4.3. Reading: Exercise: Build a React Native Screen

Figure 4.1: Exercise: Build a React Native Screen

4.4. Self Review: Build a React Native screen

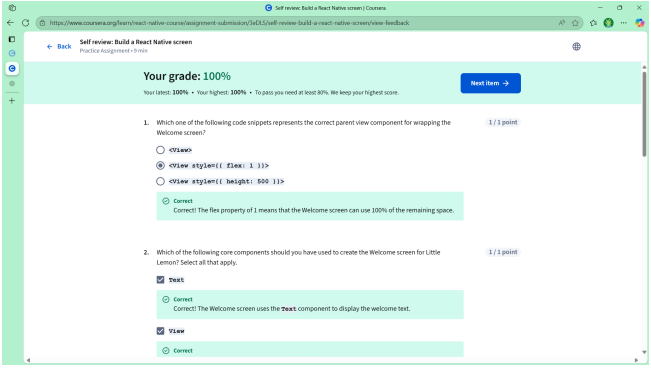

Figure 4.2: Self Review: Build a React Native Screen

4.5. Video: What is the ScrollView component?

4.5.1. Problem

- The menu items do not fit within the small mobile screen.
- Users should be able to scroll up and down the menu.

4.5.2. ScrollView

- The ScrollView component has only one rule. It must be bounded by a height in order to work.
- The parent node what wrap the ScrollView have to set a fix height.
- Use this core component, we have both scroll view for iOS and Android.
- **Practive Question:** Which of the following are true statements about the ScrollView component? Check all that apply. → ScrollView is a core React Native component and comes as a part of the React Native Package, It must be bounded by a height in order to work.

4.6. Video: Using the ScrollView component

Figure 4.3: Scroll View practice

4.7. Reading: Exercise: Build a scrollable component

Figure 4.4: Exercise: Build a scrollable component

4.8. Practice Assignment: Self review: Build a scrollable component

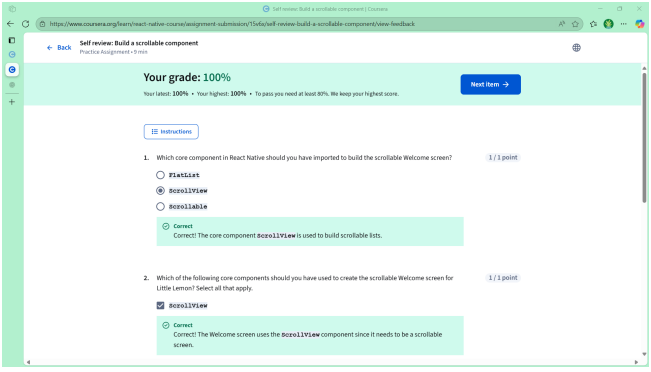

Figure 4.5: Self review: Build a scrollable component

4.9. Practive Assignment: Knowledge check: Views, Text and Scrollable components

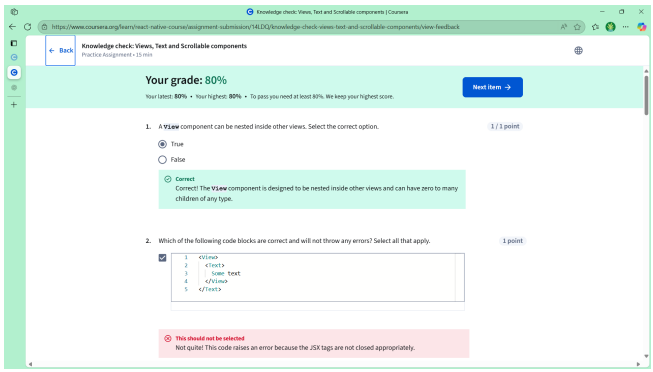

Figure 4.6: Knowledge check: Views, Text and Scrollable components

CHAPTER 5

Styling components

5.1. Video: StyleSheet API

- Extract styles from the component's render, keeping all the styles together.
- Inline styling is when you keep all the styles within the components render. As your application grows, this method of inline styling can get unmanageable very quickly and the code is harder to read.
- React Native provides a style sheet which is an API that is very similear to CSS style sheets.
- StyleSheet API allow moving styles away from the components render.
- Using:

```
<View style={styles.container}>
  <Text style={styles.title}>Hello World</Text>
</View>

const styles = StyleSheet.create({
  container: {
    ...
  },
  title: {
    ...
  }
})
```

- We can create a `global style` folder to reuse all the style for our components.

- **Practive Question:** The `StyleSheetAPI` helps abstract the component styles from the component's render and results inreadable and meaningful code. True or false? → True

5.2. Video: Practical Styling

- Convert previous work of MenuItems using StyleSheet:

```
<View style={menuStyle.container}>
  <ScrollView
    style={menuStyle.innerContainer}>
    <Text style={menuStyle.headerText}>
      View Menu
    </Text>
    <Text style={menuStyle.itemText}>
      {menuItemsToDisplay[0]}
    </Text>
  </ScrollView>
</View>

const menuStyle = StyleSheet.create({
  container: {
    flex: 0.75
  },
  innerContainer: {
    paddingHorizontal: 40,
    paddingVertical: 40,
    backgroundColor: 'black',
  },
  headerText: {
    color: 'white',
    fontSize: 40,
    flexWrap: 'wrap'
  },
  itemText: {
```

```
    color: '#F4CE14',
    fontSize: 36
  }
})
```

- Convert Header style:

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

export default function LittleLemonHeader() {
  return (
    <View style={headerStyle.container}>
      <Text
        style={headerStyle.headerText}>
        Little Lemon
      </Text>
    </View>
  );
}

const headerStyle = StyleSheet.create({
  container: {
    backgroundColor: '#F4CE14'
  },
  headerText: {
    padding: 40,
    fontSize: 30,
    color: 'black',
    textAlign: 'center'
  }
})
```

- Convert App:

```
import { StyleSheet, View } from 'react-native';
import LittleLemonHeader from
   './components/LittleLemonHeader';
import MenuItems from './components/MenuItems';
export default function App() {
  return (
    <>
      <View
        style={style.container}
      >
        <LittleLemonHeader />
        <MenuItems />
      </View>
    </>
  );
}

const style = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#495E57'
  }
})
```

5.3. Reading: Styling using StyleSheet

This section is to style all the components that I have done in the previous part.

5.4. Reading: Exercise: Style a component

- This part follow the same above process.
- Header convert:

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

export default function LittleLemonHeader() {
  return (
    <View style={headerStyle.container}>
      <Text
        style={headerStyle.headerTitle}
      >
        Little Lemon
      </Text>
    </View>
  );
}

const headerStyle = StyleSheet.create({
  container: {
    backgroundColor: '#F4CE14'
  },
  headerTitle: {
    padding: 40,
    fontSize: 30,
    color: 'black',
    textAlign: 'center'
  }
})
```

- Footer convert:

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

export default function LittleLemonFooter() {
  return (
```

```
        <View
          style={footerStyle.container}
        >
          <Text
            style={footerStyle.footerTitle}
          >
            All rights reserved by Little Lemon, 2022{' '}
          </Text>
        </View>
      );
    }

    const footerStyle = StyleSheet.create({
      container: {
        backgroundColor: '#F4CE14',
        marginBottom: 10,
      },
      footerTitle: {
        fontSize: 18,
        color: 'black',
        textAlign: 'center',
      }
    })
```

- Main app convert:

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

    import LittleLemonHeader from
       './components/LittleLemonHeader';
    import LittleLemonFooter from
       './components/LittleLemonFooter';
```

```
import WelcomeScreen from './WelcomeScreen';

export default function App() {
  return (
    <>
      <View
        style={style.HeaderContainer}
      >
        <LittleLemonHeader />
        <WelcomeScreen />
      </View>
      <View
        style={style.FooterContainer}
      >
        <LittleLemonFooter />
      </View>
    </>
  );
}

const style = StyleSheet.create({
  HeaderContainer: {
    flex: 1,
    backgroundColor: '#495E57',
  },
  FooterContainer: {
    backgroundColor: '#495E57'
  }
})
```

5.5. Self review: Style a component

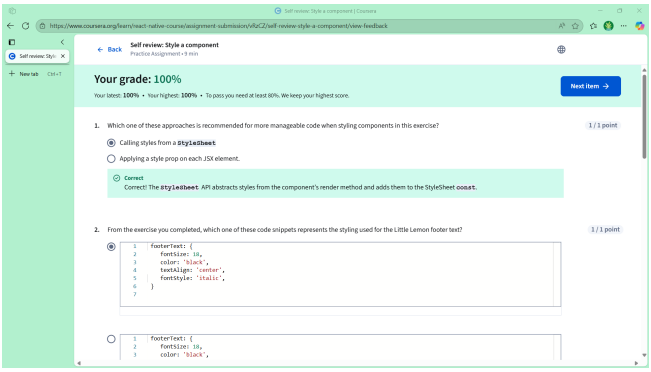

Figure 5.1: Self review: Style a component

5.6. Video: Module summary: Introduction to React Native

Introduction to React Native

- What is React Native
- Motivation behind using React Native
- Roles

Introduction to React Native

- How to be successful
- Setup React Native Project

Why React Native

- Describe the basics of React Native
- Benefits of React Native
- React Native in the real world

React Native and Expo

- What is Expo
- How to build an app with Expo code

Components

- Categorize React Native
- Use components to build apps
- Give an overview about React Native's components

Views and Text

- Demonstate the use of View and Text
- Describe basic features of View and Text

ScrollView

- Demonstate how to build a scrollable view
- Describe where you will use a scrollable view

(React Native Styling)

- Inline styling
- Benefits of using stylesheets
- Practical Styling

5.7. Module quiz: Introduction to React Native

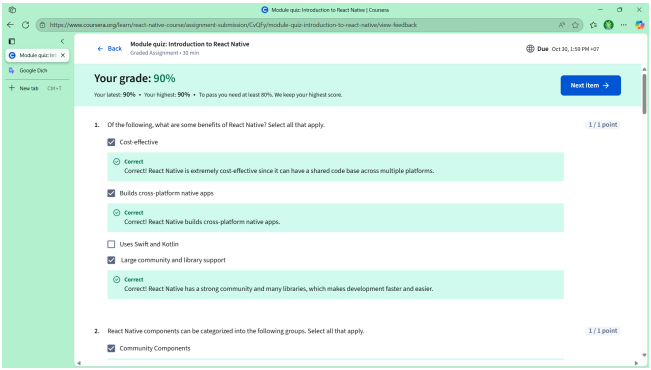

Figure 5.2: Module quiz: Introduction to React Native