Week 12 - React Native Development Guide

Introduction

Welcome to Week 12 of our React Native course! This week, we will cover the basics of setting up a React Native development environment, running a simple application, and understanding fundamental concepts through practical examples.

Setting Up the Development Environment

To get started with React Native development, students need to install the following software:

1. Install Visual Studio Code (VS Code)

VS Code is a lightweight yet powerful code editor, ideal for developing React Native applications.

- Download and install VS Code from https://code.visualstudio.com/.

2. Install Node.js

Node.js is required to run JavaScript-based applications.

- Download and install the latest stable version from https://nodejs.org/.
1. Verify the installation by running the following command in the terminal:
   node -v
-

3. Install Expo CLI

Expo provides an easy way to develop and test React Native applications without the need for Android/iOS emulators.

2. Install Expo CLI globally by running:
   npm install -g expo-cli
-
3. Verify the installation:
   expo --version
-

4. Create and Run a New React Native App

Once the setup is complete, create and run your first React Native application.

4. npx create-expo-app MyFirstApp
5. cd MyFirstApp
6. npm start

- Scan the QR code using the **Expo Go** app on your mobile device to run the application.

Understanding the Week 12 Code

This week, we will explore the following fundamental concepts through code examples:

- **Basic React Native Components**
- **Dynamic Text Rendering**
- **Passing Props**
- **Handling User Input**

1. PetInfo.js

Concepts Covered:

- Using `Text` and `View` components
- Storing and displaying a variable

Code Breakdown:

- A functional component `MyApp` is created.
- A constant variable `pet` is defined and assigned the value "Dog".
- The `View` component serves as a container to structure the UI.
- Two `Text` components display messages, with the second one dynamically incorporating the value of `pet`.
- Styles are applied using `StyleSheet` to enhance layout and readability.

2. FullNameWithPet.js

Concepts Covered:

- Creating reusable functions
- Using template literals for string concatenation
- Passing props with default values

Code Breakdown:

- The `getFullName` function takes three parameters (`fname`, `mName`, `lName`) and returns a formatted full name.
- The functional component `MyApp` uses `getFullName()` to display a full name dynamically.
- The `pet` variable is passed as a prop with a default value of "Dog".
- The `View` component structures the layout, and `Text` components are used to display information.
- The background color and spacing are enhanced using `StyleSheet`.

3. MyStudentProfile.js

Concepts Covered:

- Creating and using reusable components
- Passing props to child components
- Rendering multiple instances of a component dynamically

Code Breakdown:

- The `MyApp` component accepts `studentName` as a prop and displays a personalized greeting.
- The `MultiComp` component renders multiple instances of `MyApp`, each with a different student name.
- The `View` component acts as a container for structured alignment.
- The `title` style makes the heading prominent, and `card` style enhances the look of individual student entries.

4. TextInputExample.js

Concepts Covered:

- Handling user input dynamically using `useState`
- Using `TextInput` for interaction
- Updating UI in real-time based on user input

Code Breakdown:

- The `useState` hook is used to store and update user input dynamically.
- The `TextInput` component allows users to enter text, which updates the state using `onChangeText`.
- The entered text is displayed dynamically below the input field.

- The `View` component structures the layout, and `StyleSheet` is used for styling.
- The `input` style enhances usability by adding borders, padding, and width constraints.

Key Learning Outcomes

By the end of this week, students will:

1. Successfully set up a React Native development environment.
2. Understand the structure of a basic React Native application.
3. Learn how to use fundamental components such as `View`, `Text`, and `TextInput`.
4. Pass props and render dynamic content.
5. Use state to handle user input.

How This Helps

These concepts are the foundation of mobile app development. Understanding them will help students:

- Build their own interactive applications.
- Develop dynamic and responsive user interfaces.
- Prepare for more advanced topics like navigation, API integration, and state management.

Next Steps

For next weekS, we will dive deeper into **handling user interactions, event handling, and navigation between screens** in a React Native app.

Happy coding! 🚀