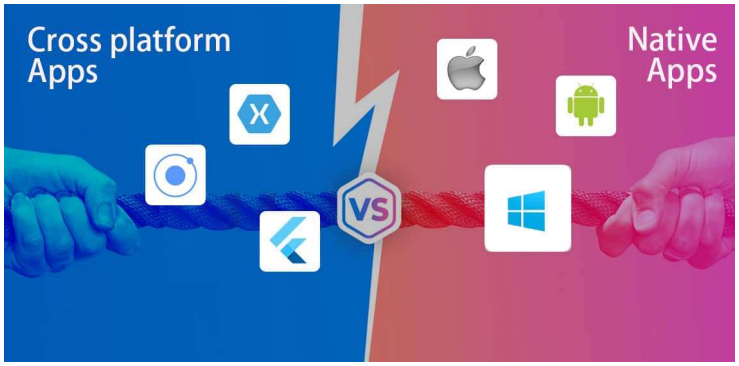
1

I. Introduction

- In the world of mobile app development, we have two primary approaches: multi-Platform (Cross-Platform) and Native Development



2

Native Mobile Programming

- Native development involves creating separate versions of an app for each platform—iOS and Android.
- Advantages: Maximum Performance, Full Access to Platform Features, Native Look and Feel, Platform-Specific Optimizations
- Disadvantages: Higher Development Cost , Longer Development Time, Platform-Specific Codebase, Double Maintenance Effort

3

Multi-Platform (Cross-Platform) Mobile Programming

- Multi-platform development, also known as cross-platform development, enables you to write code once and deploy it across multiple platforms.
- React native
- Flutter
- Xamarin
- Advantages: Cost and Time Efficiency, Shared Codebase, Easier Maintenance, Rapid Prototyping
- Disadvantages: Slightly Lower Performance, Limited Access to Platform-Specific Features, Potential Compatibility Issues, Framework Learning Curve

4

What is React Native?

- React Native is an open-source framework for building mobile apps.
- Developed by Facebook, it allows you to use React and JavaScript/Typescript to create native-quality mobile apps for iOS and Android.

5

Advantages of React Native

- Cross-Platform Development: Write code once, run it on both iOS and Android.
- Reusable Components: Components can be reused across platforms.
- Hot Reloading: Real-time code changes without app restart.
- Strong Community: Active community and third-party libraries.
- Performance: Near-native performance thanks to native modules.
- Faster Development: Streamlined development process.

6

Disadvantages of React Native

- Limited Access: Some native features may require native code.
- Debugging: Debugging can be complex compared to web development.
- Frequent Updates: Keeping up with platform updates can be challenging.
- Navigation: Complex navigation can be harder to implement.
- Third-Party Dependencies: Reliance on third-party libraries.

7

Use Cases

- Prototyping: Quickly develop and test app ideas.
- MVPs: Build Minimum Viable Products rapidly.
- Content-Driven Apps: News, blogs, social media.
- E-commerce: Shopping apps with dynamic content.
- Showcase examples like Facebook, Instagram, Airbnb, and Tesla using React Native for their apps.

8

II. Setting up the development environment

9

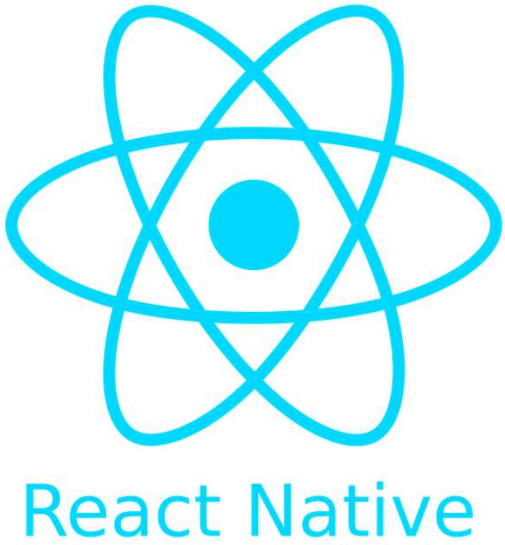
Expo CLI

- Expo CLI is a command-line tool provided by Expo(**https://expo.dev/),** a framework built around React Native. It simplifies the development process and provides additional features.
- Ease of Setup, Development Speed, Over-the-Air Updates
- Limitations: Expo CLI may limit access to some native modules and configurations

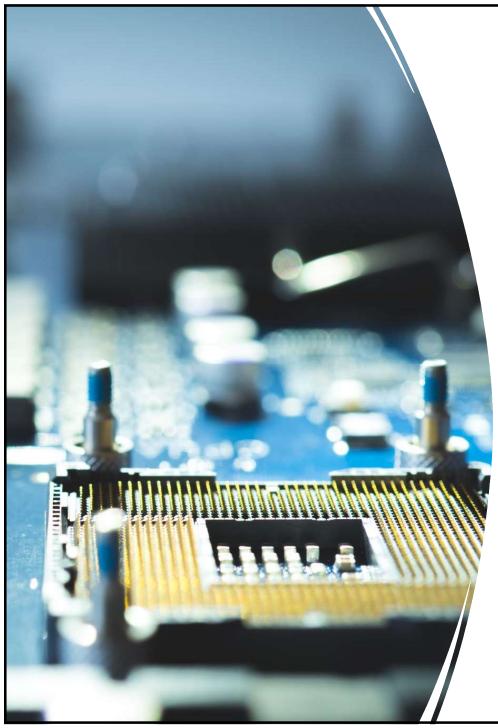
10

React Native CLI

- React Native CLI is the official command-line tool for React Native development. It provides core functionality for creating, managing, and building React Native apps.
- An open source tool
- Full Control
- Native Module Integration
- Access to Third-Party Libraries



11



Prerequisites

- Node.js (LTS version)
- npm (Node Package Manager)
- Yarn (optional but recommended)
- Xcode (for macOS users)
- Android Studio (for Android development)"

12

Install Chocolatey, Node.Js, NpN, Yarn, Expo CLI, React Native CLI

- Download and install from: https://chocolatey.org/install
- Open an Administrator Command Prompt (right click Command Prompt and select "Run as Administrator")
- Install Node.js.
- *choco install -y nodejs-lts microsoft-openjdk11*
- Install Yarn (Optional)
  - *choco install yarn*
- Install Expo CLI: *npm install -g expo-cli*
- Install the React Native CLI globally: *npm install -g react-native-cli*

13

Install Android Studio (All Platforms)

- For Android development, install Android Studio from https://developer.android.com/studio."

Select the "SDK Platforms" tab from within the SDK Manager, then check the box next to "Show Package Details" in the bottom right corner. Look for and expand the `Android 13 (Tiramisu)` entry, then make sure the following items are checked:

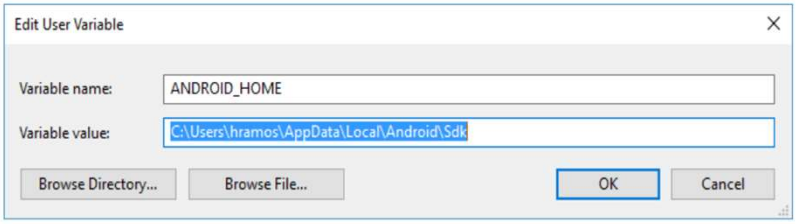
- `Android SDK Platform 33`
- `Intel x86 Atom_64 System Image` or `Google APIs Intel x86 Atom System Image`

14

Configure the ANDROID_HOME environment variable

The React Native tools require some environment variables to be set up in order to build apps with native code.

1. Open the **Windows Control Panel.**
2. Click on **User Accounts,** then click **User Accounts** again
3. Click on **Change my environment variables**
4. Click on **New...** to create a new `ANDROID_HOME` user variable that points to the path to your Android SDK:



15

III. Structure a program

```
mport React from 'react';
mport { Text, View } from 'react-nat

onst HelloWorld=()=>{
 return (
   <View style={{alignItems: 'center
   backgroundColor: '#F5FCFF', top:1
     <Text>Hello, World!</Text>
   </View>
 );
;

xport default HelloWorld;
```


16

Creating a Project with Expo CLI

npm Yarn

```
npx create-expo-app AwesomeProject

cd AwesomeProject
npx expo start
```

- Run the online application at https://snack.expo.dev/

17

Create a New React Native App with React native CLI

- Present the steps to create a new app:
- Let's create your first React Native app with the following command:
- *npx react-native init MyFirstApp*

18

Project Structure

- "Your project is structured like this:
  - **MyFirstApp/**
    - **android/**: Android-specific files
    - **ios/**: iOS-specific files
    - **node_modules/**: Third-party libraries
    - **App.js**: Main app component"

19

Open and Edit App.js/App.tsx file

```
demo2 > JS App.js > ...
1  import { StatusBar } from 'expo-status-bar';
2  import React from 'react';
3  import { StyleSheet, Text, View } from 'react-native';
4
5  export default function App() {
6    return (
7      <View style={styles.container}>
8        <Text>Open up App.js to start working on your app!</Text>
9        <StatusBar style="auto" />
10     </View>
11   );
12 }
13
14 const styles = StyleSheet.create({
15   container: {
16     flex: 1,
17     backgroundColor: '#fff',
18     alignItems: 'center',
19     justifyContent: 'center',
20   },
21 });
22
```

Allows import/export components (functions, classes, variables) from one file to another for use.

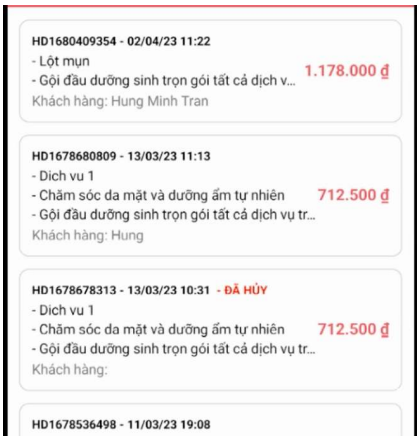
20

Run the App

- To run your app, use:

  npx react-native run-android (for Android)

  npx react-native run-ios (for iOS)"



Imports and Exports

- ES6 provides two ways to export multiple components from one file:
- **named export**, **default export**.
  - Named export: use export and {}

```
//functionsFile.js

//exporting a function
export function squareNumber(x) {
  return x * x;
}

//exporting a variable
export const pi = 3.14;

//Cách khác để export:

//exporting a function
function squareNumber(x) {
  return x * x;
}

//exporting a variable
const pi = 3.14;
export {squareNumber, pi} ;
```

Import with Named export

- Syntax: import { <Name 1> , <name 2>…} form "fileJS";

```
//main.js
import {squareNumber, pi} from "functionsFile";
const radius = 7;
console.log("Area of a circle is", pi * squareNumber(7));

//Cách khác để import

import * as mathFuncs from "functionsFile";
console.log("Area of circle is ", mathFuncs.pi * mathFuncs.squareNumber(7));
```

Use * to import all

Default export

- Systax: import, export default …

```
//functionsFile.js

export default function(x) {
  return x * x ;
}
```

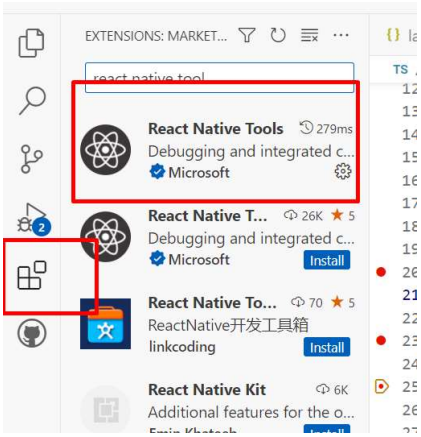
import vào 1 file khác

Not included in {}

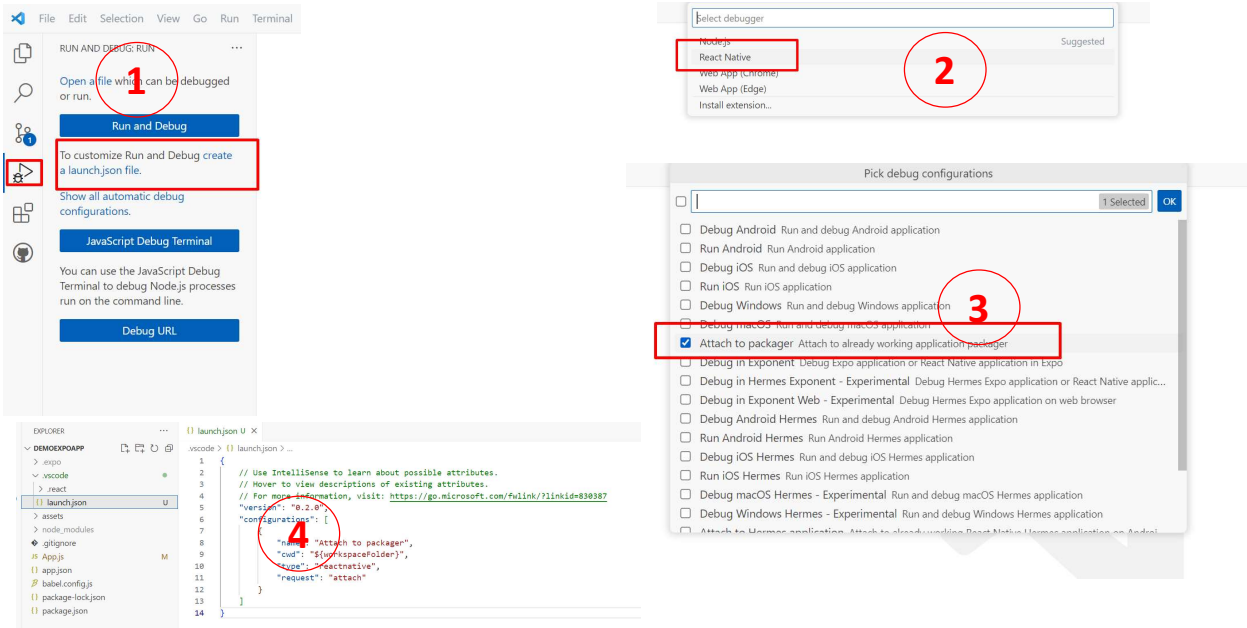
```
//main.js

import squareNumber from "functionsFile";
squareNumber(7);
```

IV. Debugging

- Setup extension ‘React native tool’ on Visual studio Code



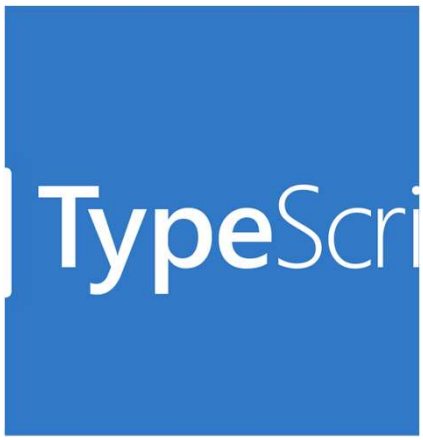
25



26

V. Typescript

- TypeScript is a statically typed superset of JavaScript that enhances code quality and developer productivity by adding strong typing to your code.



27

Why TypeScript in React Native?

- In React Native, TypeScript offers:
  - Better code quality
  - Enhanced developer tooling
  - Improved collaboration in teams
  - Reduced runtime errors
- TypeScript is used to:
  - Define component props and states
  - Annotate function parameters and return types
  - Type-check your code at compile-time

28

Basic Types

- number, string, boolean, array
- null, undefined, any, void“

- Example: let age: number = 30;
- let name: string = "John";
- let isCompleted: boolean = true;
- let numbers: number[] = [1, 2, 3, 4, 5];

29

React Native Component, function with TypeScript

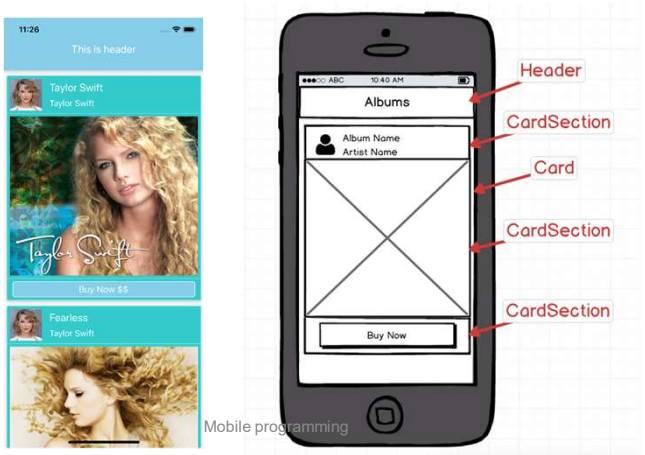
```tsx
import React, { Component } from 'react';
import { View, Text } from 'react-native';

interface Props {
  name: string;
}

class Greeting extends Component<Props> {
  render() {
    return (
      <View>
        <Text>Hello, {this.props.name}!</Text>
      </View>
    );
  }
}
```

```tsx
function greet(name: string): string {
  return `Hello, ${name}!`;
}
```


30

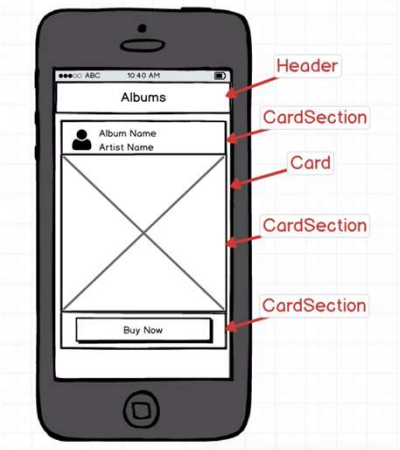
VI. Components

- Components are a fundamental concept of both React and React native. It is the breakdown of the application into small components that create their high reusability and scalability



Components

- In addition to the components we build and reuse, Reat native provides a lot of default components.
  - **Native Components**
  - **Custom Component**



Native Components

- React Native provides a set of core/native components that allow you to build mobile user interfaces using familiar concepts from web development.
- These components are part of the React Native framework and offer a bridge to native UI elements on both iOS and Android platforms
- Can use community-made Native Components: https://reactnative.directory/

V	IOS VIEW	WEB ANALOG	DESCRIPTION
	<UIView>	A non-scrollling	A container that suppo with flexbox, style, son handling, and accessib controls
	<UITextView>	<p>	Displays, styles, and ne strings of text and eve touch events
	<UIImageView>	<img>	Displays different type images
	<UIScrollView>		A generic scrolling con that can contain multip components and views
	<UITextField>	<input type="text">	Allows the user to ente

Native Components

- The following core component APIs Documents can be found: https://reactnative.dev/docs/components-and-apis

```
mport { View, Text, Image, ScrollView, TextInput } from 'react

onst App = () => {
 return (
   <ScrollView>
     <Text>Some text</Text>
     <View>
       <Text>Some more text</Text>
       <Image
         source={{
           uri: 'https://reactnative.dev/docs/assets/p_cat2.pn
         }}
         style={{ width: 200, height: 200 }}
       />
     </View>
     <TextInput
       style={{
         height: 40,
         borderColor: 'gray',
         borderWidth: 1
       }}
       defaultValue="You can type in me"
     />
   </ScrollView>
 );
```

Custom Component

- There are two main types of custom components in React Native:
  - Functional Components
  - Class Components"

Custom Component

- Functional Components:

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

const HelloWorld = () => {
  return (
    <View>
      <Text>Hello, World!</Text>
    </View>
  );
};

export default HelloWorld;
```

```
import React from 'react';
import HelloWorld from './Helloworld'
export default App =()=>{

  return (
    <HelloWorld/>
  )
}
```

Custom Component

- Class Components

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

class MyComponent extends Component {
  constructor(props) {
    super(props);
    this.state = {
      count: 0,
    };
  }

  handleIncrement = () => {
    this.setState((prevState) => ({
      count: prevState.count + 1,
    }));
  }

  render() {
    return (
      <View>
        <Text>Count: {this.state.count}</Text>
        <Button title="Increment" onPress={this.handleIncrement} />
      </View>
    );
  }
}
export default MyComponent;
```

```
import React from 'react';
import MyComponent from './MyComponent'
export default App =()=>{

  return (
    <MyComponent/>
  )
}
```


Props

- 1. Data in the React Component is managed by state and props.
- 2. The state can change while prop is immutable. This means that the state can update in the future while prop is impossible.
- 3. Props: allows changing the React Component through properties.

Multiple Props

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

const Cat = (props) => {
  return (
    <View>
      <Text>Hello, I am {props.name}!</Text>
    </View>
  );
}

const Cafe = () => {
  return (
    <View>
      <Cat name="Maru" />
      <Cat name="Jellylorum" />
      <Cat name="Spot" />
    </View>
  );
}

export default Cafe;
```

Nhúng thuộc tính name của Cat vào JSX qua props, mặt định tất cả component điều có props

Hello, I am Maru!
Hello, I am Jellylorum!
Hello, I am Spot!

39

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

const CatApp = () => {
  return (
    <View>
      <Image
        source={{uri: "https://reactnative.dev/docs/assets/p_cat1.png"}}
        style={{width: 200, height: 200}}
      />
      <Text>Hello, I am your cat!</Text>
    </View>
  );
}

export default CatApp;
```

Hello, I am your cat!

40

State

- Props are considered parameters passed to configure components to use and remain immutable.
- While State handles when data componets changes due to interaction from users
  - Use prop when rendering components.
  - Enable state to keep track of the component's data that wants to change over time.

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

const Counter = () => {
  const [count, setCount] = useState(0);

  const handleIncrement = () => {
    setCount(count + 1);
  };

  return (
    <View>
      <Text>Count {count}</Text>
      <Button title="Increment" onPress={handleIncrement} />
    </View>
  );
};
```


Publishing

- Step 1: On Windows keytool must be run from C:\Program Files\Java\jdkx.x.x_x\bin, as administrator. Open powershell.

Run keytool -genkeypair -v -storetype PKCS12 -keystore my-upload-key.keystore -alias my-key-alias -keyalg RSA -keysize 2048 -validity 10000

- Step 2: Setting up Gradle variables

Place the my-upload-key.keystore file under the android/app directory in your project folder

Edit the file ~/.gradle/gradle.properties or android/gradle.properties, and add the following (replace ***** with the correct keystore password, alias and key password)

MYAPP_UPLOAD_STORE_FILE=my-upload-key.keystore
MYAPP_UPLOAD_KEY_ALIAS=my-key-alias
*MYAPP_UPLOAD_STORE_PASSWORD=*****

*MYAPP_UPLOAD_KEY_PASSWORD=*****

Publishing

- Step 3: Edit the file android/app/build.gradle in your project folder, and add the signing config

```
android {
    ...
    defaultConfig { ... }
    signingConfigs {
        release {
            if (project.hasProperty('MYAPP_UPLOAD_STORE_FILE')) {
                storeFile file(MYAPP_UPLOAD_STORE_FILE)
                storePassword MYAPP_UPLOAD_STORE_PASSWORD
                keyAlias MYAPP_UPLOAD_KEY_ALIAS
                keyPassword MYAPP_UPLOAD_KEY_PASSWORD
            }
        }
    }
    buildTypes {
        release {
            ...
            signingConfig signingConfigs.release
        }
    }
}
```


Publishing

- Step 4.1 : Generating the release AAB

```
npx react-native build-android --mode=release
```

Optional: divided by CPU type

```
android {
    splits {
        abi {
            reset()
            enable true
            universalApk false
            include "armeabi-v7a", "arm64-v8a", "x86", "x86_64"
        }
    }
}
```

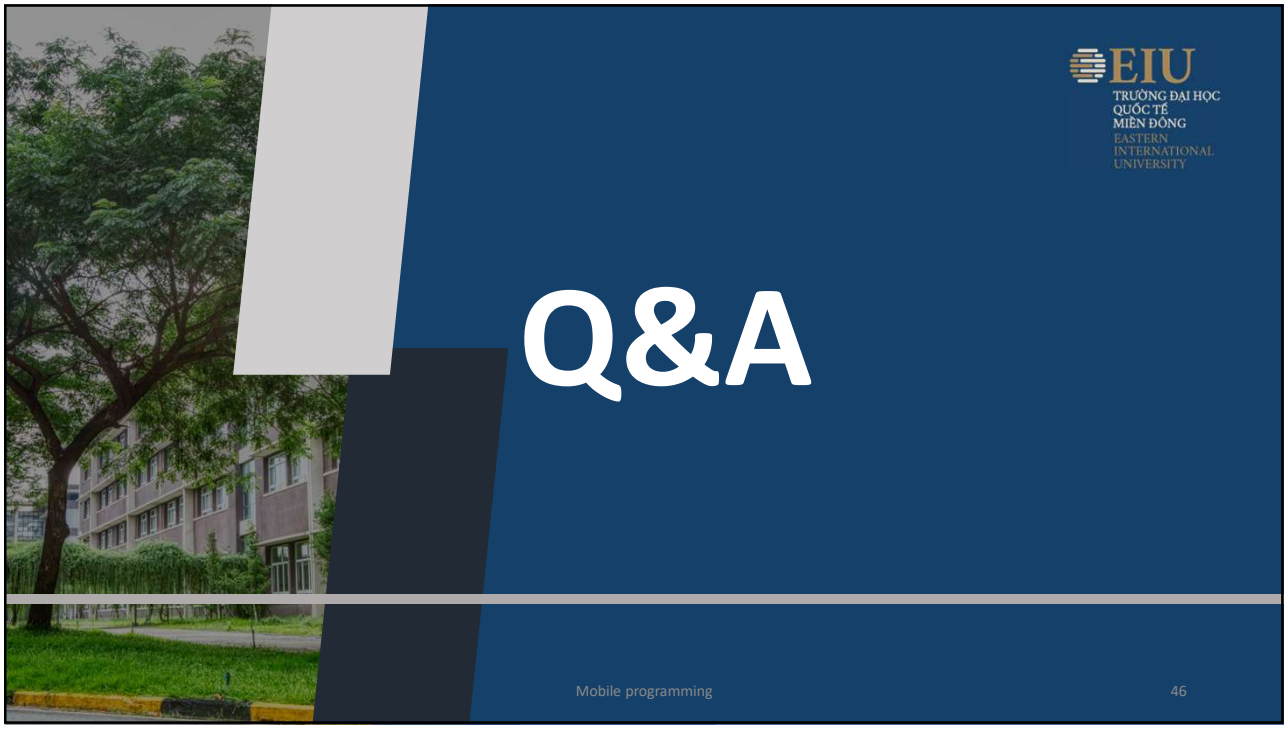
- Step 4.2: Build React native App to apk file

```
./gradlew assembleRelease
```

Exercises

1. Install the environment and create your first project with Expo CLI and React Native CLI
2. Rewrite the example code in the Props, state, component slide.

45



46