Cross-Platform Mobile Application Development

Introduction to Cross-Platform Mobile Development

Week 1

Agenda

- Introduction to Cross-Platform development
- Advantages and Disadvantages of Cross-Platform design.
- Introduction to primary concepts of React Native
  - What is React Native?
  - Why use React Native?
  - Advantages and Disadvantages
  - How does React Native work?
  - Reviewing React Native Syntax

Course Learning Objectives:

- In this course, you will become familiarized with:
  - Purpose of cross-platform development.
  - Different methods to test and start a React Native mobile application.
  - Using React Native to develop cross-platform mobile applications.
  - Preparing the development environment for React Native application development.
  - Integrating Flux pattern into React Native mobile application projects using Redux.
  - Designing appropriate styles and layouts for mobile applications.
  - Exploring and Utilizing React Native Core and Native components.
  - Employing platform-specific code and components.
  - Creating and publishing mobile application package files for Android and iOS platforms.
  - Integrating a navigation system into the React Native applications using React Navigation library.
  - Demonstrating the use of activity indicators to show progress.
  - Creating simple and complex animations.
  - Use of modal screens to present content to users over the main view.
  - Using a gesture responder system to handle user touches and gestures.
  - Collecting and displaying user data in React Native applications.

What is Cross-Platform Development?

- Cross-platform application development refers to the development of applications that can run on multiple platforms and devices.
- Cross-platform design may require you to build your project separately for each platform. However, this does not require you to re-code everything.
- For cross-platform development, developers may use languages such as JavaScript, Java, ReactNative, Flutter, etc.

Advantages of Cross-Platform Development:

- You re-use the code you wrote to target multiple platforms and hence you do not need to rewrite anything for a separate platform.
- It reduces the development time, and you can deliver your app to the market quicker.
- It is often cheaper.
- You target a wider range of audiences.
- Easier code maintenance.
- Consistency of design across all platforms.

Disadvantages of Cross-Platform Development:

- Depending on the platform, it may result in performance issues or bigger file sizes. However, this is really scenario dependent and is not a blanket statement for all cross-platform approaches.

- Cross-Platform IDEs may take longer to release new updates and as a result, this may affect how quickly you update your applications to address a specific bug that is due to a problem with the IDE, compiler or the intermediatory language and cannot be addressed at the application level.

Examples of Cross-Platform Development:

- Cross platform development is done in both mobile app development and game development.
- For instance:
  - Unity Engine is used to develop cross-platform games for Android, Linux, Windows and iOS.
  - React-Native is used to develop applications for web, Android and iOS.

What is React?

- React is an open-source JavaScript library used in many popular online platforms such as Facebook (Meta).
- It was originally used to build user interfaces for web applications.
- A component-based concept to create UIs for web applications.

What is React Native?

- React-Native is a framework for building native mobile apps in JavaScript using the React JS library.
- React-Native code compiles to real native components.

Advantages of React Native

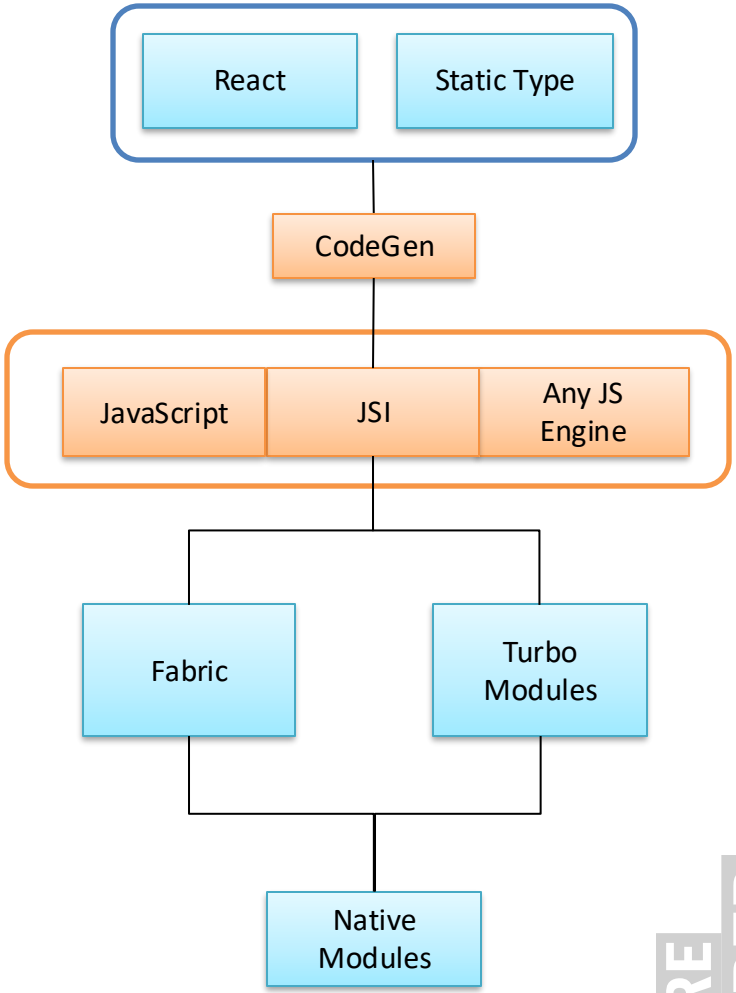
- If you are a web developer, it gives you a head start with React Native development as it uses markup language.
- It uses JavaScript, and as such, you do not need to learn any additional language!
- You can use a Hot Reload feature to reload the application on your phone instantly after you make any changes in the code during the development process.
- If you already know React, You are just a few steps away from knowing React Native! The concepts are very similar.
- Target multiple platforms, it saves your time and energy.

Disadvantages of React Native

- React does not work as closely as a Native development environment that is dedicated to a specific platform. A native approach may give you more flexibility in accessing specific OS functionality.
- You may still need to have a knowledge of native programming if you want to create native modules within your React Native application.
- At enterprise level applications, that have too many complex navigations and features, performance issues may arise. An example would be developing complex 3D games.

React Native Architecture

- JSI (**JavaScript Interface**) is an interface that allows JavaScript to hold a reference to a C++ object and vice-versa.
- **Fabric** is React Native's new rendering system, a conceptual evolution of the legacy render system. The core principles are to unify more render logic in C++, improve interoperability with host platforms, and to unlock new capabilities for React Native.
- **Turbo Modules** facilitate more efficient type-safe communication between JavaScript and native, without relying on the React Native bridge. It will also enable new extensions that weren't possible with the legacy Native Module system.



Benefits of New Architecture

The new architecture also provides benefits in code quality, performance, and extensibility:

- **Type safety:** code generation to ensure type safety across the JS and host platforms. The code generation uses JavaScript component declarations as source of truth to generate C++ structs to hold the props. Mismatch between JavaScript and host component props triggers a build error.
- **Shared C++ core**: the renderer is implemented in C++ and the core is shared among platforms. This increases consistency and makes it easier to adopt React Native on new platforms.
- **Better Host Platform Interoperability**: Synchronous and thread-safe layout calculation improves user experiences when embedding host components into React Native, which means easier integration with host platform frameworks that require synchronous APIs.

Benefits of New Architecture

The new architecture also provides benefits in code quality, performance, and extensibility:

- **Consistency**: The new render system is cross-platform, it is easier to keep consistency among different platforms.
- **Faster Startup**: Host components are lazily initialized by default.
- **Less serialization of data between JS and host platform**: React used to transfer data between JavaScript and *host platform* as serialized JSON. The new renderer improves the transfer of data by accessing JavaScript values directly using JavaScript Interfaces (JSI).

Concepts in common with React:

- Among various concepts present in React Native, the following are almost identical to that of React.js.
  - Components
  - Props and States
  - JSX (A Syntax Extension to allow writing HTML code in JavaScript)

- During this course you will learn stuff that is specific to React Native, such as native components.
- **It is easy to use props and states as the concept is the same for React and React Native.**

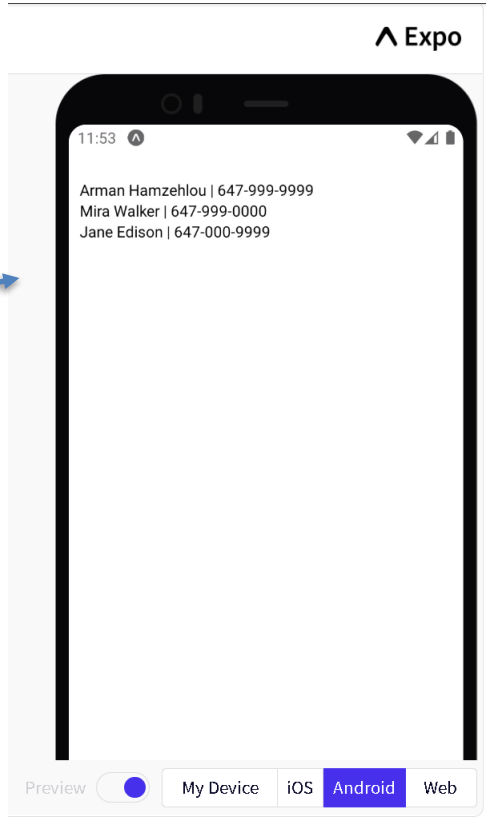
React Native Syntax

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

const User= (props) => {
 return (
  <View>
   <Text>{props.name} {props.lname} | {props.number}</Text>
  </View>
 );
}

const Users = () => {
 return (
  <View style={{top:50,left:10}}>
   <User name='Arman' lname="Hamzehlou" number="647-999-9999"/>
   <User name='Mira' lname="Walker"  number="647-999-0000"/>
   <User name='Jane' lname="Edison"  number="647-000-9999"/>
  </View>
 );
}

export default Users;
```



React Native Syntax

- In the syntax example you can see that very similar to React.js we start off by creating components.
- All components can receive props.
- You can see we are using “react-native” to import a few components. These components will be reviewed in upcoming chapters, which are very useful.
- You can see two function components.
- We have used styles to add padding to the components output so they are not too close to the sides of the phone screen.
- This code is being run in Expo environment, a development environment we will be exploring in the upcoming lesson.

References:

- https://reactnative.dev/docs/native-modules-android
- https://reactnative.dev/architecture/overview