Building Your First Application

Objectives

Setting up your environment

- Running React Native Application
- Modifying Your App
- Exploring the Sample code

Build an App

- Building an Android App with React Native
- React Native – Default Application
- Build Your First App with React Native
- Creating an App

Setting up your environment

Expo Go

- If you are new to mobile development, the easiest way to get started is with Expo Go.
- Expo is a set of tools and services built around React Native.
- You will only need a recent version of Node.js and a phone or emulator.
- If you'd like to try out React Native directly in your web browser before installing any tools, you can try out Snack

Environment setup

React Native project with Expo

- NodeJS
- Device (Expo go) => LdPayer, Nox, BlueStack

IDE

- Visual Code

IDE Visual Code

- Extention
  - React-native tool
  - ES7 + React
  - Prettier code formatter
  - color picker
  - auto rename tag
  - Auto close tag
  - Auto import
  - Auto complete tag

Extention

- bracket pair color
- Npm intelligent
- intelligent css
- intelligent code
- npm intelligent
- path intellisense
- Material icon
- Javascript ES6
- JavaScript & typeScript

Expo Go (contd.)

Run the following command to create a new React Native project called

>>npx create-expo-app@latest

>>npx create-expo-app@latest --template

You can use the `--template` option to select one of the following templates or pass it as an argument to the option. For example, `--template default`.

Template	Description
`default`	Default template. Designed to build multi-screen apps. Includes recommended tools such as Expo CLI, Expo Router library and TypeScript configuration enabled. Suitable for most apps.
`blank`	Installs minimum required npm dependencies without configuring navigation.
`blank-typescript`	A Blank template with TypeScript enabled.
`tabs`	Installs and configures file-based routing with Expo Router and TypeScript enabled.
`bare-minimum`	A Blank template with native directories (**android** and **ios**) generated. Runs `npx expo prebuild` during the setup.

Expo Go (contd.)

- **Running your React Native application**
  - Install the Expo Go app on your iOS or Android phone and connect to the same wireless network as your computer.
  - On Android, use the Expo Go app to scan the QR code from your terminal to open your project.
  - On iOS, use the built-in QR code scanner of the default iOS Camera app.
- **Modifying your app**
  - Open **App.js** in your text editor of choice and edit some lines.
  - The application should reload automatically once you save your changes.

DEMO

React Native CLI

- If you are already familiar with mobile development, you may want to use React Native CLI.
- For example, if you are integrating React Native into an existing application, or if you ran "prebuild" from Expo to generate your project's native code, you'll need this section.
- **Development OS**
  - Windows + Android
  - MacOS + iOS and MacOS + Android

Windows + Android

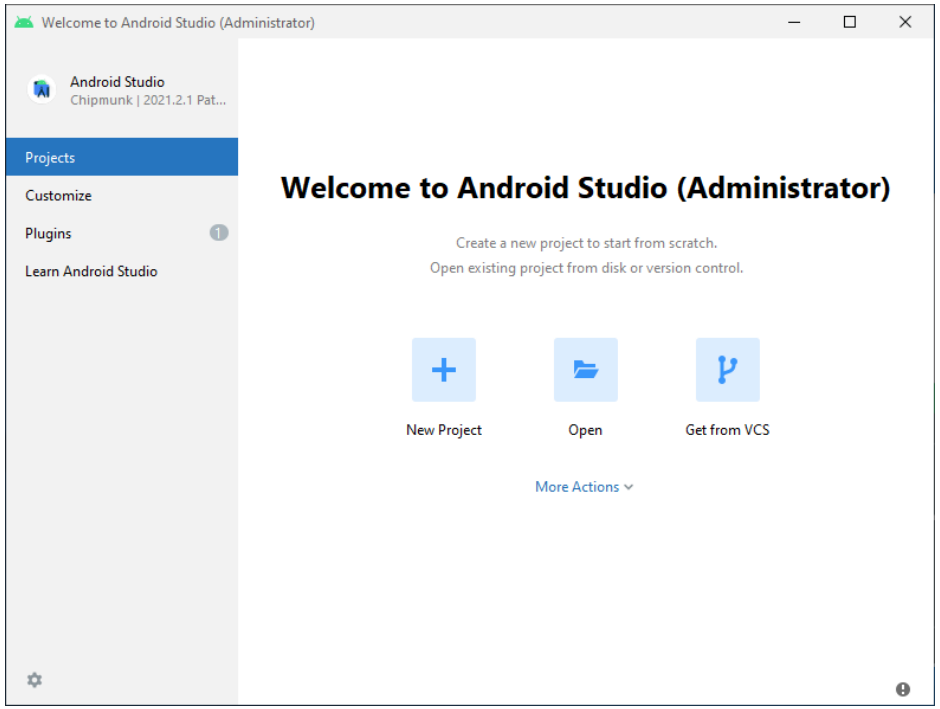
- **Installing dependencies**
  - We will need **Node**, the **React Native** command line interface, a **JDK**, and **Android Studio**.
- **Node (v18.16.0)**
  - Download link: https://nodejs.org/en/download
- **OpenJDK (v17)**
  - Download link: https://learn.microsoft.com/en-us/java/openjdk/download

Windows + Android (contd.)

- **Install Android Studio**
  - Download link: https://developer.android.com/studio/index.html
  - While on Android Studio installation wizard, make sure the boxes next to all of the following items are checked:
    - Android SDK
    - Android SDK Platform
    - Android Virtual Device

Windows + Android (contd.)

- **Install the Android SDK**
  - To do that, open Android Studio, click on "More Actions" button and select "SDK Manager".

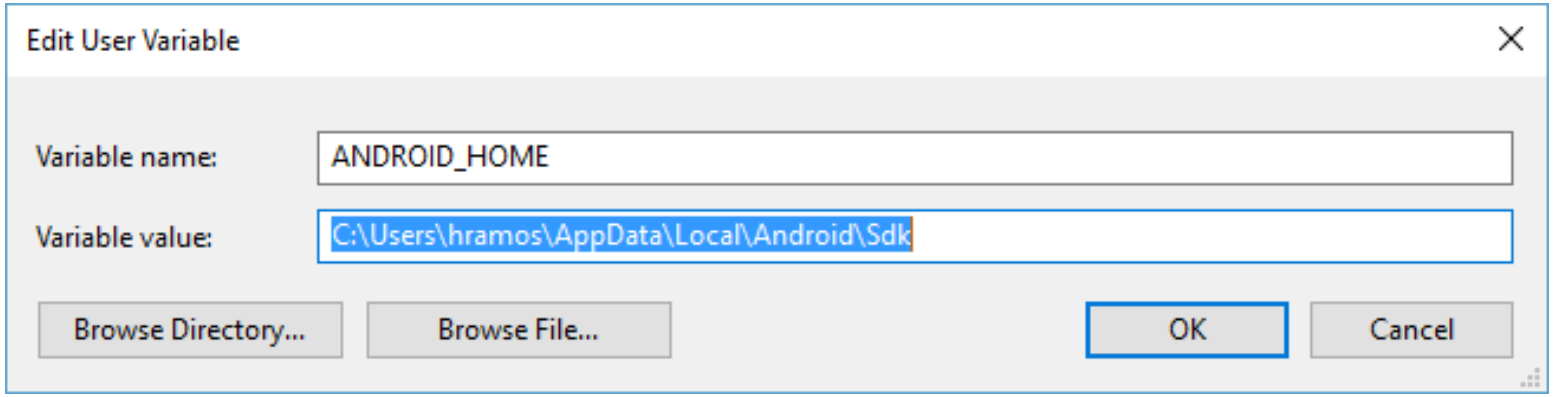


Windows + Android (contd.)

- **Install the Android SDK (contd.)**
  - 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
  - Next, select the "**SDK Tools**" tab and check the box next to "**Show Package Details**" here as well. Look for and expand the Android SDK Build-Tools entry, then make sure that 33.0.0 is selected.
  - Finally, click "**Apply**" to download and install the Android SDK and related build tools.

Windows + Android (contd.)

- **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:



Windows + Android (contd.)

- **Add platform-tools to Path**

    1. Open the **Windows Control Panel**.
    2. Click on **User Accounts**, then click **User Accounts** again.
    3. Click on **Change my environment variables.**
    4. Select the **Path** variable.
    5. Click **Edit**.
    6. Click **New** and add the path to platform-tools to the list.

The default location for this folder is:

```
%LOCALAPPDATA%\Android\Sdk\platform-tools
```

Windows + Android (contd.)

- **Creating a new application**
  - If you previously installed a global react-native-cli package, please remove it as it may cause unexpected issues:

```
npm uninstall -g react-native-cli @react-native-community/cli
```

  - Let's create a new React Native project called "**AwesomeProject**":

```
npx react-native@latest init AwesomeProject
```

Windows + Android (contd.)

- **Preparing the Android device**
  - Using a virtual device
    - If you use Android Studio to open **./AwesomeProject/android**, you can see the list of available **Android Virtual Devices** (AVDs) by opening the "**AVD Manager**" from within Android Studio
    - If you have recently installed **Android Studio**, you will likely need to create a new AVD. Select "**Create Virtual Device...**", then pick any Phone from the list and click "Next", then select the **Tiramisu API Level 33 image**.
    - Click "Next" then "**Finish**" to create your AVD. At this point you should be able to click on the green triangle button next to your AVD to launch it, then proceed to the next step.

Windows + Android (contd.)

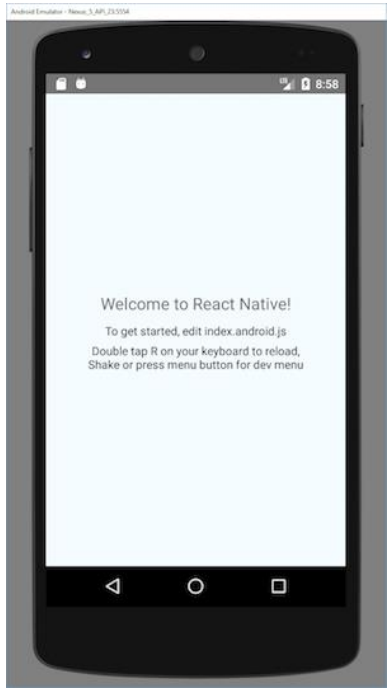
- **Running your React Native application**
  - Step 1: Start Metro

```
npx react-native start
```

  - Step 2: Start your application

```
npx react-native run-android
```

If everything is set up correctly, you should see your new app running in your Android emulator shortly.



What are props?

Example HTML tag attribute

- Tag <img/> HTML
  -
  -
  -
  -

What are props?

- Props (Properties) are attributes passed data from a parent component to a child component.
- Props are immutable data (cannot be changed) and can only be read, not modified.
- Props allow we to create reusable components by passing different values into those components.

Props definition

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

interface GreetingProps {
    name: string;
    address?: string;
}

export default function Greeting(props: GreetingProps) {
    return (
        <View>
            <Text>Hello {props.name}</Text>
            <Text>Address: {props.address}</Text>
        </View>
    );
}
```

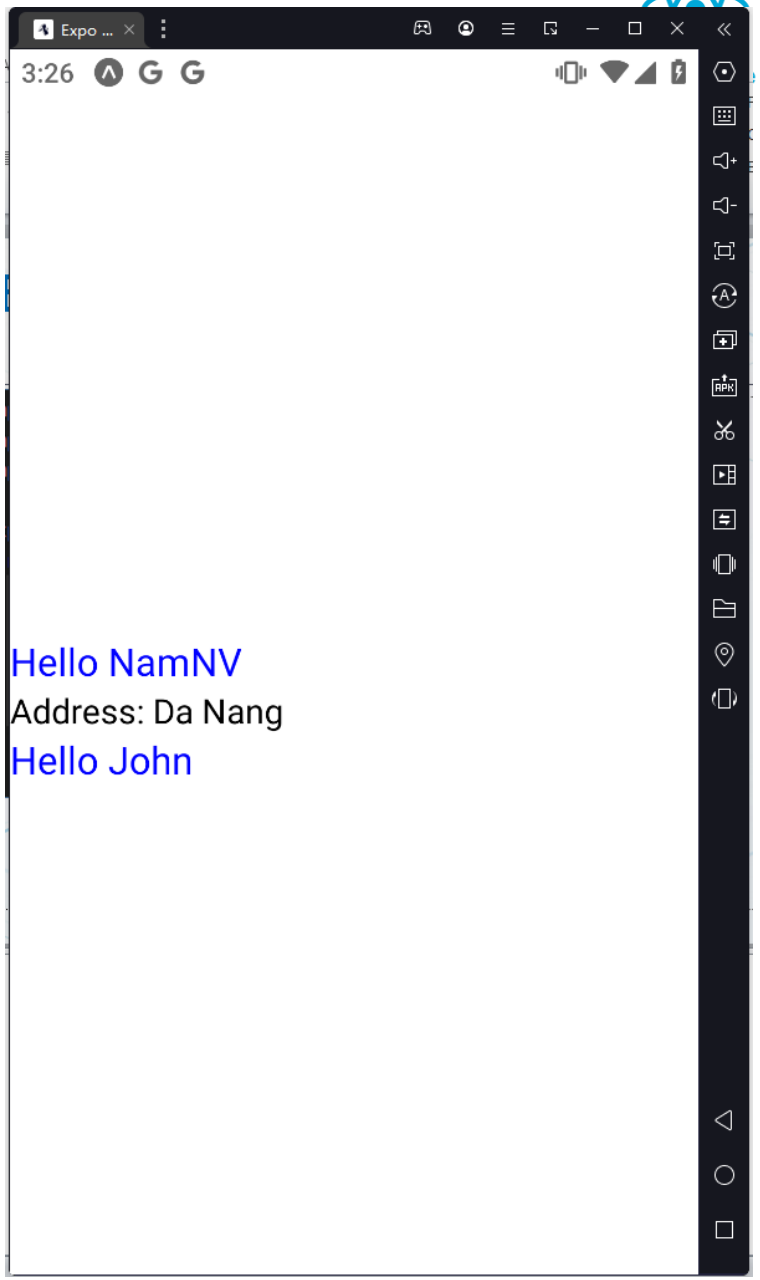
- In this example, data was a string variable. But **props** can be anything such as integers, objects, arrays, and even React components.

Props definition (contd.)

```
import { StyleSheet, View,} from 'react-native';
import React from 'react';
import Greeting from './components/Greeting';

export default function App() {
  return (
      <View style={styles.container}>
        <Greeting name="NamNV" address="Da Nang" />
        <Greeting name="John" />
      </View>

  );
}
```



When are Props used?

- You need to use Props when you want to pass data from a parent component to a child component.
- Note
  - The **Greeting** component can be used multiple times in different places, with different values passed into the props.name.
  - **Remember**, you should not modify the value of the **props.name** inside the **Greeting** component. We only read and display the value of props.name on the screen.

DEMO

What is state?

What is state?

- **State** (mutable) is the internal data of a component that can change throughout the component's lifecycle.
- **State** is initialized in the constructor function and can be updated using the setState method.
- When the state changes, the component will re-render

What is state?

- Define state
  - import {useState} from 'react'

```
const [name, setName] = useState('')
const [count, setCount] = useState(0)
const [show, setShow] = useState(false)
const [products, setProducts] = useState([])
```

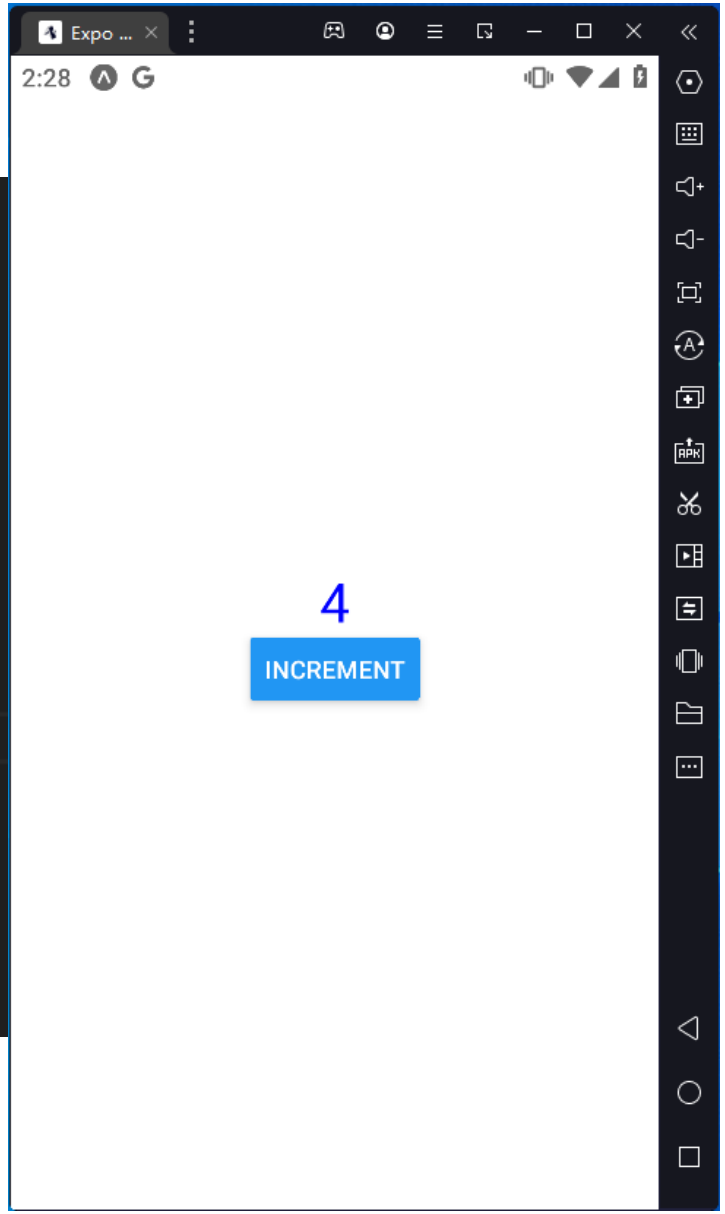
JAVA

```
private String name;
private int age;

public String getName() {
}
public void setName(String name) {
}
public int getAge() {
}
public void setAge(int age) {
}
```

Example

```
1  import { useState } from "react";
2  import { Button, Text, View } from "react-native";
3
4  export default function Counter() {
5      const [count, setCount] = useState(0);
6      const increment = () => {
7          setCount(count + 1);
8      }
9      return(
10         <View>
11             <Text>{count}</Text>
12             <Button title="Increment" onPress={increment} />
13         </View>
14     )
15 }
16
```



Note

- InitState is used to initialize the initial value
- setState() update the state with a new value.
- You should using the setState method to update the state.
- You can use setState with a callback.

DEMO

Summary

- In this session, we learned the following:
  - Step-by-Step to setting up your environment
    - Using Expo Go
    - Using React Native CLI
  - Understand to creating a new application
    - Project initialization
    - Passing and change state data between components
    - Run the application on a device emulator