REACT NATIVE

2022 Learn once, write anywhere

Contents

- What is React Native ?
- React Native Environment setup window & mac
- React Native cli Vs Expo cli
- Understanding Project Structure
- React Native Lifecycle Methods.
- React Hook & State Management
- Understanding third party packages & installing
- React Navigation
- React Native Components & props
- Custom hook
- Local storage (Encryption methods)
- React Context vs Redux store
- React Context
- Redux store
- Data encryption and decryption
- RestApi fetching data
- Local push notifications (using Firebase)
- App Security
- Understanding code push
- Publishing to google play store

What is React Native ?

React Native is an open-source UI software framework created by Meta Platforms, Inc. It is used to develop applications for Android, Android TV, iOS, macOS, tvOS, Web, Windows and UWP by enabling developers to use the React framework along with native platform capabilities.

Android

Android TV

IOS

tvOs

React Native Environment setup window

JDK

- ✓ https://openjdk.org/projects/jdk/11/

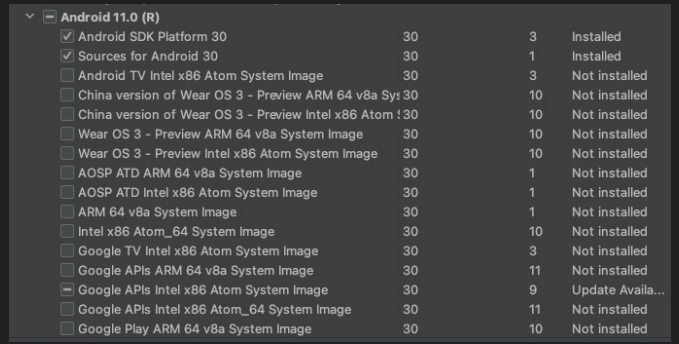
Node

- ✓ https://nodejs.org/en/download/

Android Studio

- ✓ https://developer.android.com/studio/index.html

- `Android SDK Platform 30`
- `Intel x86 Atom_64 System Image` or `Google APIs Intel x86 Atom System Image`

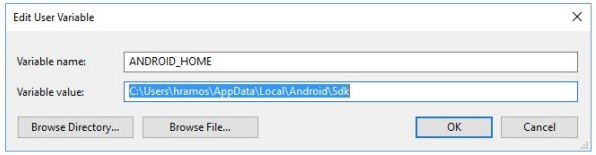


Android 11.0 (R)			
Android SDK Platform 30	30	3	Installed
Sources for Android 30	30	1	Installed
Android TV Intel x86 Atom System Image	30	3	Not installed
China version of Wear OS 3 - Preview ARM 64 v8a Sys	30	10	Not installed
China version of Wear OS 3 - Preview Intel x86 Atom S	30	10	Not installed
Wear OS 3 - Preview ARM 64 v8a System Image	30	10	Not installed
Wear OS 3 - Preview Intel x86 Atom System Image	30	10	Not installed
AOSP ATD ARM 64 v8a System Image	30	1	Not installed
AOSP ATD Intel x86 Atom System Image	30	1	Not installed
ARM 64 v8a System Image	30	1	Not installed
Intel x86 Atom_64 System Image	30	10	Not installed
Google TV Intel x86 Atom System Image	30	3	Not installed
Google APIs ARM 64 v8a System Image	30	11	Not installed
Google APIs Intel x86 Atom System Image	30	9	Update Availa...
Google APIs Intel x86 Atom_64 System Image	30	11	Not installed
Google Play ARM 64 v8a System Image	30	10	Not installed

Configure the ANDROID_HOME environment variable

Window

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:



MacOs

- $HOME/.bash_profile or $HOME/.bashrc
- ~/.zprofile or ~/.zshrc

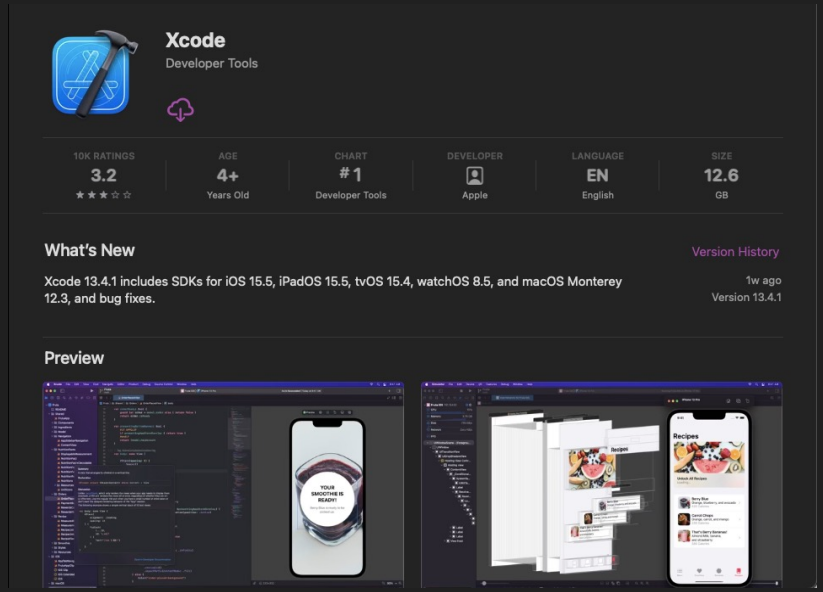
```
export ANDROID_SDK_ROOT=$HOME/Library/Android/sdk
export PATH=$PATH:$ANDROID_SDK_ROOT/emulator
export PATH=$PATH:$ANDROID_SDK_ROOT/platform-tools
```

React Native Environment setup macOs

- brew install node
- brew install watchman

Download XCode

- Download xcode
- sudo gem install cocoapods



Create React Native Project

- npx i –g react-native
- npx react-native init SampleProject
- cd SampleProject

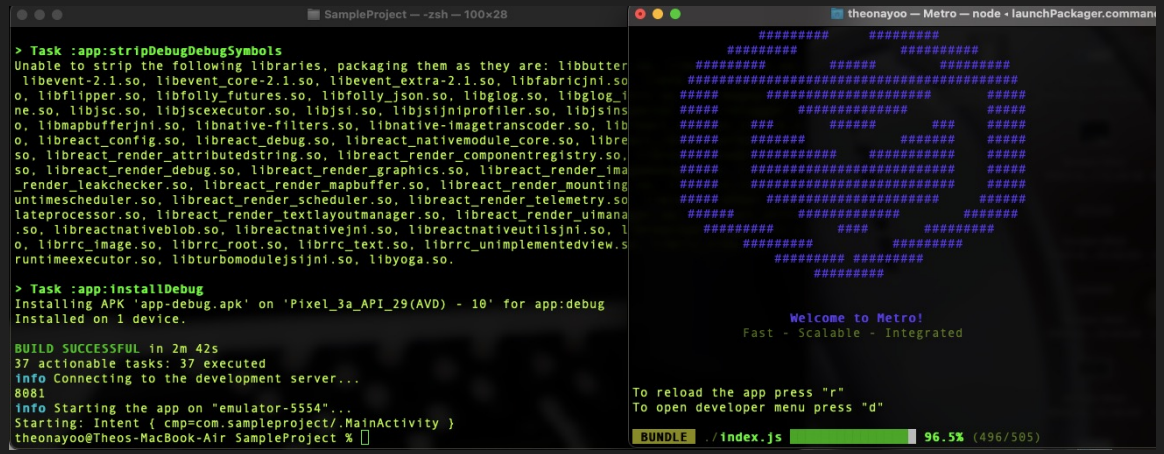
```
theonayoo@Theos-MacBook-Air React-Native % npx react-native init SampleProject

Welcome to React Native!
Learn once, write anywhere

✔ Downloading template
✔ Copying template
✔ Processing template
⠸ Installing dependencies
```

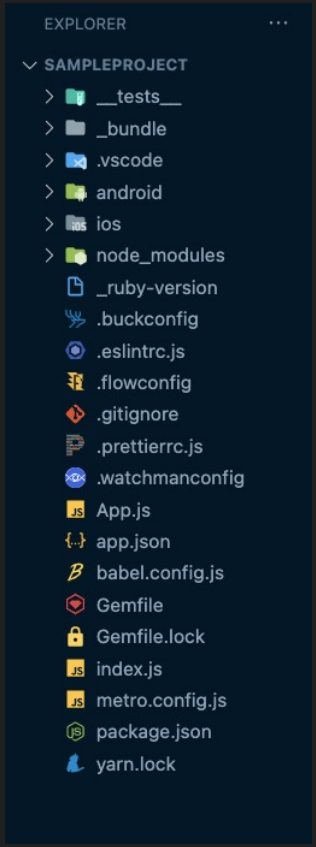
Run React Native Project

- npx react-native run-android (for android)
- npx react-native run-ios (for IOS)

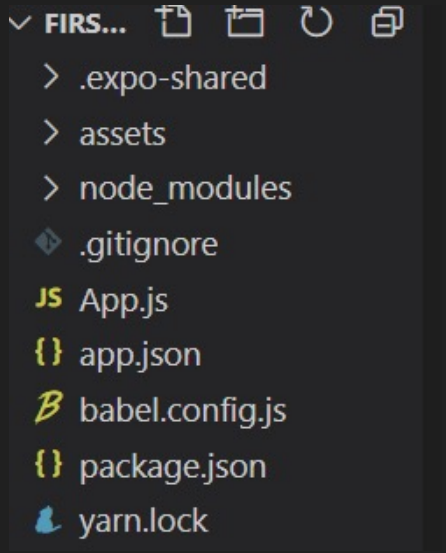


React Native cli Vs Expo cli

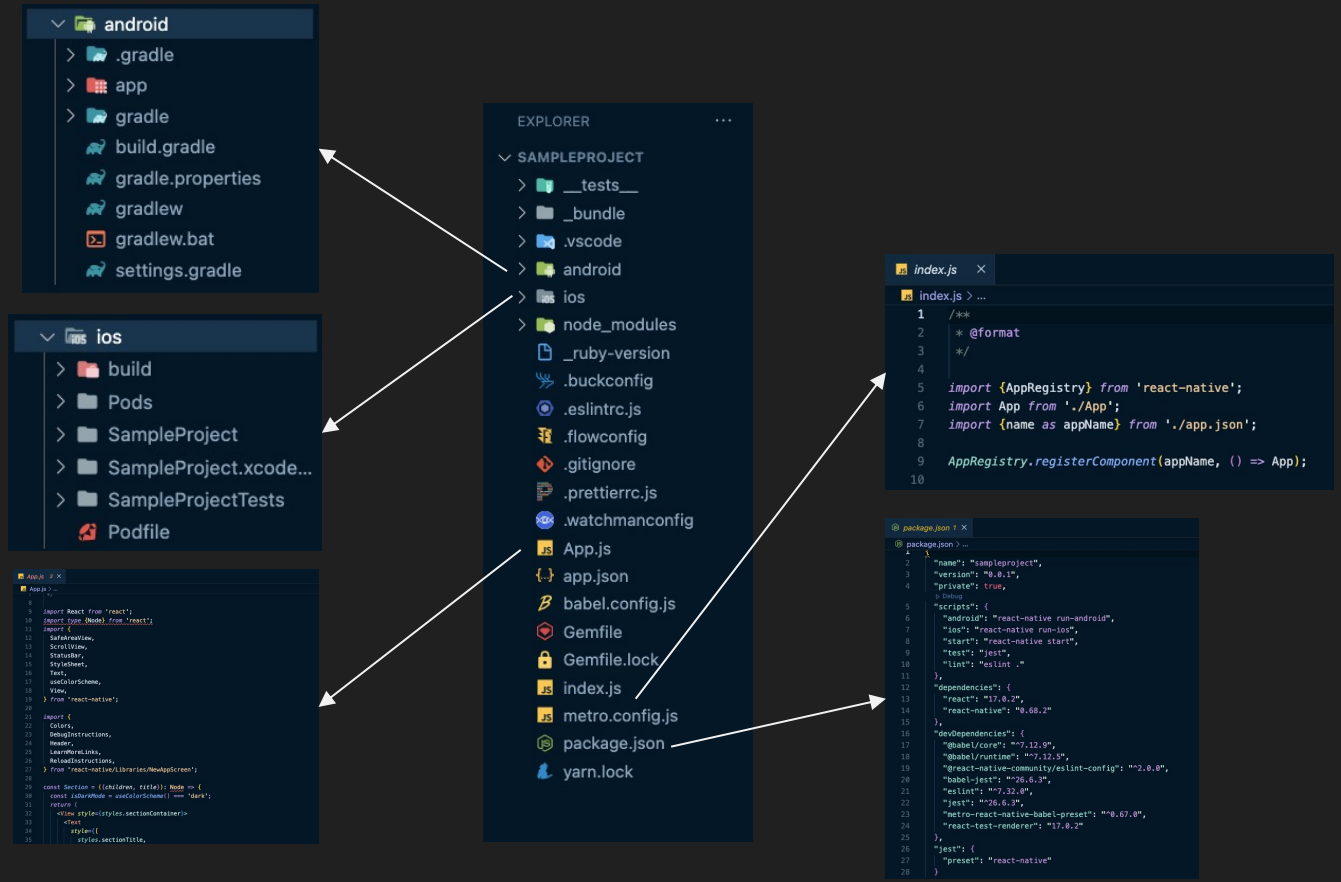
React Native Cli



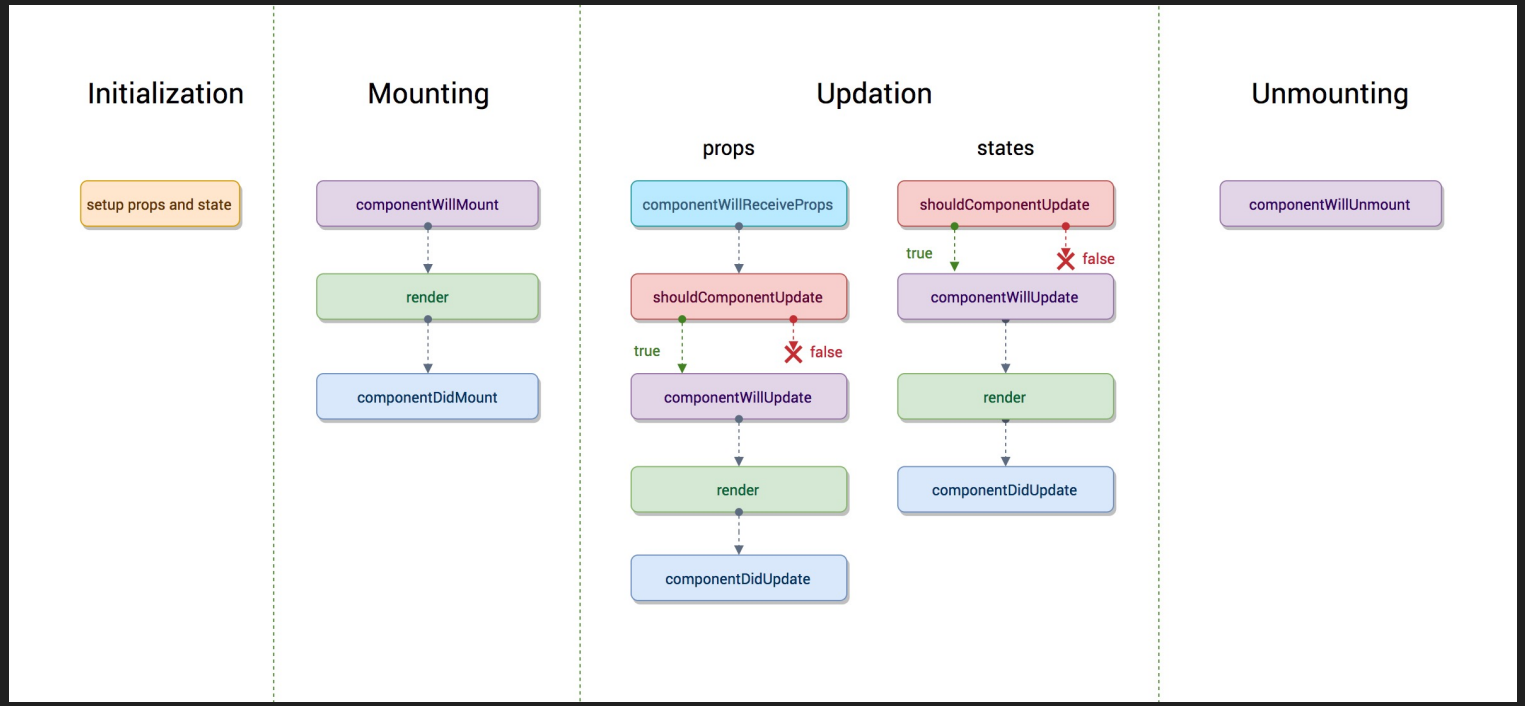
Expo Cli



Understanding Project Structure



React Native Lifecycle Methods



React Hook

- What is React Hook?

Hooks are a new addition in React 16.8. They let you use state and other React features without writing a class.

- Rules of hooks

  - Only call hooks at the top level
  - Only call hooks from React function

React Hook & State Management

- useState
- useEffect
- useReducer
- useContext
- useMemo
- useRef
- useCallback

```
import React,{useState} from 'react';
```

```
import React,{useEffect} from 'react';
```

```
import React,{useReducer} from 'react';
```

```
import React,{useContext} from 'react';
```

```
import React,{useMemo} from 'react';
```

```
import React,{useRef} from 'react';
```

```
import React,{useCallback} from 'react';
```

```
import React,{useState,useEffect,useContext,useReducer,useCallback,useRef,useMemo} from 'react';
```

useState Hook

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

const App = () => {
  const [title, setTitle] = useState('');

  return (
    <View>
      <Text>show title {title}</Text>
    </View>
  );
};

export default App;
```

useEffect Hook

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

const App = () => {
  const [title, setTitle] = useState('');

  useEffect(() => {
    // do something
  }, []);

  return (
    <View>
      <Text>show title {title}</Text>
    </View>
  );
};

export default App;
```

useContext Hook

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

const AuthContext = React.createContext();

const App = () => {
  const [title, setTitle] = useState('');

  const context = {
    title,

    getTitle: value => {
      setTitle(value);
    },
  };

  return (
    <AuthContext.Provider value={context}>
      {/* screen here */}
    </AuthContext.Provider>
  );
};

export default App;
```

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

const AuthContext = React.createContext();

const App = () => {
  const {title} = useContext(AuthContext);

  return (
    <View>
      <Text>{title}</Text>
    </View>
  );
};

export default App;
```

useReducer Hook

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

const initialState = {count: 0};
// Reducer function
const reducer = (state = initialState, action) => {
  switch (action.type) {
    case 'increment':
      return {count: state.count + 1};
    case 'decrement':
      return {count: state.count - 1};
    default:
      return {count: state.count};
  }
};
```

```
const Counter = () => {
  const [state, dispatch] = useReducer(reducer, initialState);

  return (
    <View>
      {/* count title */}

      <Text>Count : {state.count}</Text>

      {/* increment action */}

      <TouchableOpacity onPress={() => dispatch({type: 'increment'})}>
        <Text>Increment</Text>
      </TouchableOpacity>


      {/* Decrement action */}

      <TouchableOpacity onPress={() => dispatch({type: 'decrement'})}>
        <Text>Decrement</Text>
      </TouchableOpacity>

    </View>
  );
};
export default Counter;
```

Understanding third party packages & installing

react-native-qrcode-scanner TS

1.5.5 • Public • Published 4 months ago

Readme | Explore BETA | 3 Dependencies | 23 Dependents | 51 Versions

react-native-qrcode-scanner

npm package 1.5.5 | backers 0 | sponsors 0

A QR code scanner component for React Native built on top of react-native-camera by Lochlan Wansbrough

Please note, this will also function as a generic barcode scanner by the virtue of the above module supporting barcode scanning, however, this module was initially built as a QR code scanner.

This module was originally written because the author couldn't find a module that could be simply plugged into a project without requiring too much setup.

Looking for active contributors. See Contribution guide for more details.

***Please note**: Most of the support on debugging new issues, especially with Android, relies on the open source community. The project is on the look out for active contributors who want to maintain this library more rigorously.*

Getting started

Install

```
> npm i react-native-qrcode-scanner
```

Repository

github.com/moaazsidat/react-native-qr...

Homepage

github.com/moaazsidat/react-native-qr...

♥Fund this package

Weekly Downloads

26,445

Version	License
1.5.5	MIT
Unpacked Size	Total Files
37.1 kB	10

React Navigation

- npm install @react-navigation/native
- npm install react-native-screens react-native-safe-area-context
- npm install @react-navigation/native-stack

```
import * as React from 'react';
import { View, Text } from 'react-native';
import { NavigationContainer } from '@react-navigation/native';
import { createNativeStackNavigator } from '@react-navigation/native-stack';

function HomeScreen() {
  return (
    <View style={{ flex: 1, alignItems: 'center', justifyContent: 'center' }}>
      <Text>Home Screen</Text>
    </View>
  );
}

const Stack = createNativeStackNavigator();

function App() {
  return (
    <NavigationContainer>
      <Stack.Navigator>
        <Stack.Screen name="Home" component={HomeScreen} />
      </Stack.Navigator>
    </NavigationContainer>
  );
}
export default App;
```

Tabs Navigation

- npm install @react-navigation/native
- npm install react-native-screens react-native-safe-area-context
- npm install @react-navigation/native-stack

```
import * as React from 'react';
import { View, Text } from 'react-native';
import { NavigationContainer } from '@react-navigation/native';
import { createNativeStackNavigator } from '@react-navigation/native-stack';

function HomeScreen() {
  return (
    <View style={{ flex: 1, alignItems: 'center', justifyContent: 'center' }}>
      <Text>Home Screen</Text>
    </View>
  );
}

const Stack = createNativeStackNavigator();

function App() {
  return (
    <NavigationContainer>
      <Stack.Navigator>
        <Stack.Screen name="Home" component={HomeScreen} />
      </Stack.Navigator>
    </NavigationContainer>
  );
}
export default App;
```

React Native Components & props

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

// Components
import LoginHeader from './login';

const App = () => {
  const [title, setTitle] = useState('Login');

  return (
    <View style={{flex: 1, alignItems: 'center'}}>
      <LoginHeader title={title} goLogin={() => console.log('login....')} />
    </View>
  );
};

export default App;
```

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

const login = props => {
  return (
    <View>
      <Text>{props.title}</Text>

      <TouchableOpacity onPress={props.goLogin}>
        <Text>Login</Text>
      </TouchableOpacity>
    </View>
  );
};

export default login;
```

Components & Styling

- Inline style

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

const App = () => {
  return (
    <View style={{flex: 1, alignItems: 'center', justifyContent: 'center'}}>
      <Text>App</Text>
    </View>
  );
};

export default App;
```

External Styling

- External style

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

const App = () => {
  return (
    <View style={styles.container}>
      <Text>App</Text>
    </View>
  );
};

const styles = StyleSheet.create({
  container: {
    flex: 1,
    alignItems: 'center',
    justifyContent: 'center',
  },
});

export default App;
```

External Styling

- External style

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

const App = () => {
  return (
    <View style={styles.container}>
      <Text>App</Text>
    </View>
  );
};

const styles = StyleSheet.create({
  container: {
    flex: 1,
    alignItems: 'center',
    justifyContent: 'center',
  },
});

export default App;
```

External Styling

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

// Style
import styles from './Style';

const App = () => {
  return (
    <View style={styles.container}>
      <Text>App</Text>
    </View>
  );
};

export default App;
```

```
import {StyleSheet} from 'react-native';

const styles = StyleSheet.create({
  container: {
    flex: 1,
    alignItems: 'center',
    justifyContent: 'center',
  },
});

export default styles;
```

External Styling

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

// Style
import styles from './Style';

const App = () => {
  return (
    <View style={[styles.container, {backgroundColor: 'green'}]}>
      <Text>App</Text>
    </View>
  );
};

export default App;
```

Import Fonts

Config

- Create react-native.config.js

```
module.exports = {
    project: {
      ios: {},
      android: {},
    },
    assets: ['./assets/fonts/'],
};
```

Local storage (Using AsyncStorage)

Installing

- npm install @react-native-async-storage/async-storage

Importing

- import AsyncStorage from '@react-native-async-storage/async-storage';

Local storage

Store

```
const storeData = async value => {
  try {
    await AsyncStorage.setItem('@storage_Key', value);
  } catch (e) {
    console.log('error', e);
  }
};
```

Get data

```
const getData = async () => {
  try {
    const store = await AsyncStorage.getItem('@storage_Key');
    setData(store);
  } catch (e) {
    // error reading value
    console.log('error', e);
  }
```

Local storage

Store

```
const storeData = async value => {
  try {
    await AsyncStorage.setItem('@storage_Key', value);
  } catch (e) {
    console.log('error', e);
  }
};
```

Get data

```
const getData = async () => {
  try {
    const store = await AsyncStorage.getItem('@storage_Key');
    setData(store);
  } catch (e) {
    // error reading value
    console.log('error', e);
  }
```

Local storage (using mmkv or secure key store)

Installing

- npm install react-native-mmkv

Importing

- import { MMKV } from 'react-native-mmkv

Local storage (using mmkv)

Store

```
import {MMKV} from 'react-native-mmkv';

export const storage = new MMKV();

export const appStorage = {
  setItem: (key, value) => {
    try {
      storage.set(key, value);
    } catch (e) {
      console.log("Can't save data : ", e);
    }
  },

}

export default appStorage;
```

Get data

```
import {MMKV} from 'react-native-mmkv';

export const storage = new MMKV();

export const appStorage = {
  getItem: key => {
    if (storage.contains(key)) {
      return storage.getString(key);
    } else {
      console.log("Can't get data : ", key);
      return undefined;
    }
  },
};

export default appStorage;
```

Local storage (using mmkv)

Delete store

```
import {MMKV} from 'react-native-mmkv';

export const storage = new MMKV();

export const appStorage = {
  deleteItem: key => storage.delete(key),
};

export default appStorage;
```

Custom Hook

- Create React Context
- Create custom hook

```
export default {
    login: 'login',
}
```

```
import React, {useContext} from 'react';
import {AuthContext} from '../context/Context';
import en from '../component/helper/local/en';
import mm from '../component/helper/local/mm';

export const useLocal = () => {
  const {lang} = useContext(AuthContext);
  if (lang === 'en') {
    return en;
  } else {
    return mm;
  }
};
```

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

import {useLocal} from '../../../hook';

const login = () => {
  const local = useLocal();

  return (
    <View>
      <Text>{local.login}</Text>
    </View>
  );
};

export default login;
```

Npm package installing knowledge

Installing with package

- npm install <package>

Installing with package & version

- npm install <package>@<version>

Example

- npm install cowsay@1.2.0

yarn package installing knowledge

Installing with package

- yarn add <package>

Installing with package & version

- yarn add <package>@<version>

Example

- yarn add cowsay@1.2.0

Data Encryption & Decryption (CryptoJS)

Installing with version

- npm i crypto-js@3.1

Encryption

```
import React, {useState} from 'react';
import {View, Text} from 'react-native';
import CryptoJS from 'crypto-js';

let KEY = '1234567890';

const Login = () => {
  const [data, setData] = useState('My messages');

  // Encrypt
  var ciphertext = CryptoJS.AES.encrypt(data, KEY).toString();

  console.log('encryption ::', ciphertext);

  return (
    <View>
      <Text>Login</Text>
    </View>
  );
};
export default Login;
```

Decryption

```
import React, {useState} from 'react';
import {View, Text} from 'react-native';
import CryptoJS from 'crypto-js';

let KEY = '1234567890';

const Login = () => {
  const [data, setData] = useState('U2FsdGVkX19ndwlovTIWKyD3Vuk9PJIdtC8lZ+Sjx2c=');

  // Decrypt
  var bytes  = CryptoJS.AES.decrypt(data, KEY);
  var originalText = bytes.toString(CryptoJS.enc.Utf8);

  console.log('original messages ::',originalText)

  return (
    <View>
      <Text>Login</Text>
    </View>
  );
};
export default Login;
```

Get data from .env

Installing react-native-config

- npm i react-native-config

Configuration

---

android/settings.gradle

- include ':react-native-config'
- project(':react-native-config').projectDir = new File(rootProject.projectDir, '../node_modules/react-native-config/android')

android/app/build.gradle

```
dependencies {
    implementation project(':react-native-config')
}
```

RestApi Fetching data

Method

- GET
- POST
- PUT
- PUSH
- DELETE

Fetching data

Get Method

```
const fetchData = async () => {
  const response = await fetch(`https://jsonplaceholder.typicode.com/posts`, {
    method: 'GET',
    headers: {
      Accept: 'application/json',
      'Content-Type': 'application/json',
    },
  });

  let data = await response.json();
  console.log('data ::', data);
};
```

Post Method

```
const fetchData = async () => {
  const response = await fetch(`https://jsonplaceholder.typicode.com/posts`, {
    method: 'POST',
    headers: {
      Accept: 'application/json',
      'Content-Type': 'application/json',
    },
    body: JSON.stringify(data),
  });

  let data = await response.json();
  console.log('data ::', data);
};
```

Fetching data

Multi Upload

```
const fetchData = async () => {
  const uploadData = new FormData();
  let resData = {
    user: 'kyaw kyaw',
    email: 'test@gmail.com',
  };
  uploadData.append('data', resData);
  uploadData.append('images[]', {
    type: photo.type,
    uri: photo.uri,
    name: photo.fileName,
    width: photo.width,
    height: photo.height,
  });
  const response = await fetch(`https://jsonplaceholder.typicode.com/posts`, {
    method: 'POST',
    headers: {
      Accept: 'application/json',
      'Content-Type': 'multipart/form-data',
    },
    body: uploadData,
  });

  let data = await response.json();
  console.log('data ::', data);
};
```

Redux store

Action

```
import {AUTH, LOGOUT} from '../type';

export const authenticate = data => {
  return {
    type: AUTH,
    auth: data,
  };
};

export const logout = () => {
  return {
    type: LOGOUT,
  };
};
```

Reducer

```
import {AUTH, LOGOUT} from '../type';

const initialState = {
  data: null,
};

export default (state = initialState, action) => {
  switch (action.type) {
    case AUTH:
      return {
        data: action.auth,
      };
    case LOGOUT:
      return initialState;
    default:
      return state;
  }
};
```

Redux store

Redux

```
import {applyMiddleware, createStore,  combineReducers} from 'redux';
import ReduxThunk from 'redux-thunk';

// Reducer
import AuthReducer from './reducer/auth';

const rootReducer = combineReducers({
  auth: AuthReducer,
});

const store = createStore(rootReducer, applyMiddleware(ReduxThunk));

export default store;
```

Redux store

Set & Get data from Reducer

```
import React, {useState} from 'react';
import {useDispatch, useSelector} from 'react-redux';
import {View, Text, TouchableOpacity, TextInput} from 'react-native';

// From reducer
import * as actions from '../store/action/auth';

const Login = () => {
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');

  const dispatch = useDispatch();
  const authUser = useSelector(state => state.auth.data);

  const goLogin = () => {
    let data = {
      email: email,
      password: password,
    };
    dispatch(actions.authenticate(data));
  };
```

React Bottom Tabs

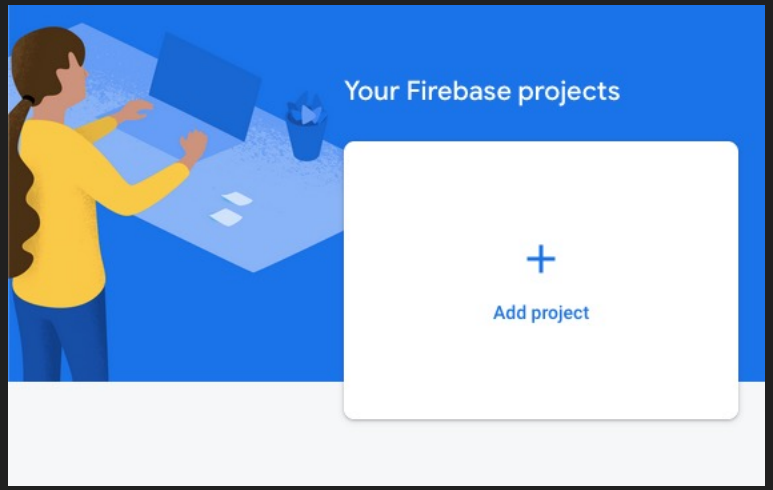
- npm install @react-navigation/bottom-tabs
- *import* { createBottomTabNavigator } *from* '@react-navigation/bottom-tabs';

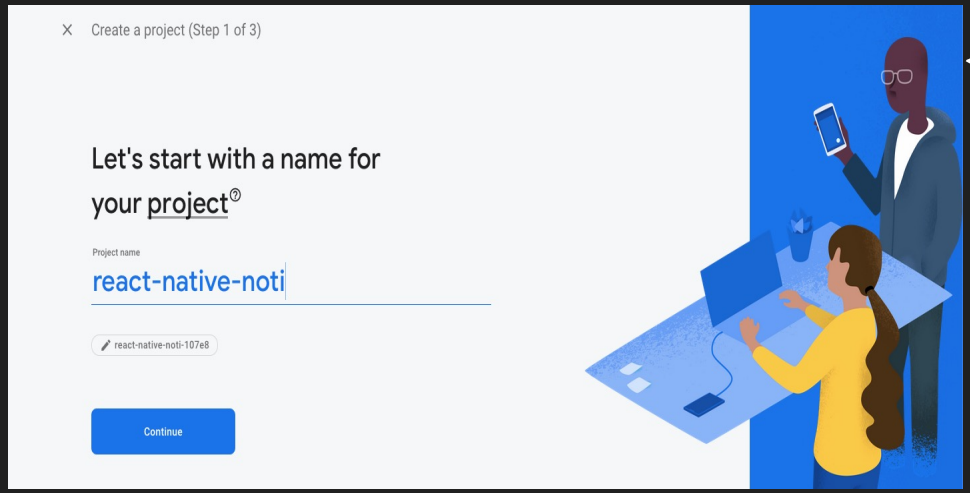
```
const Tab = createBottomTabNavigator();

export default function App() {
  return (
    <NavigationContainer>
      <Tab.Navigator>
        <Tab.Screen name="Home" component={HomeScreen} />
        <Tab.Screen name="Settings" component={SettingsScreen} />
      </Tab.Navigator>
    </NavigationContainer>
  );
}
```

Local push notifications (using Firebase)

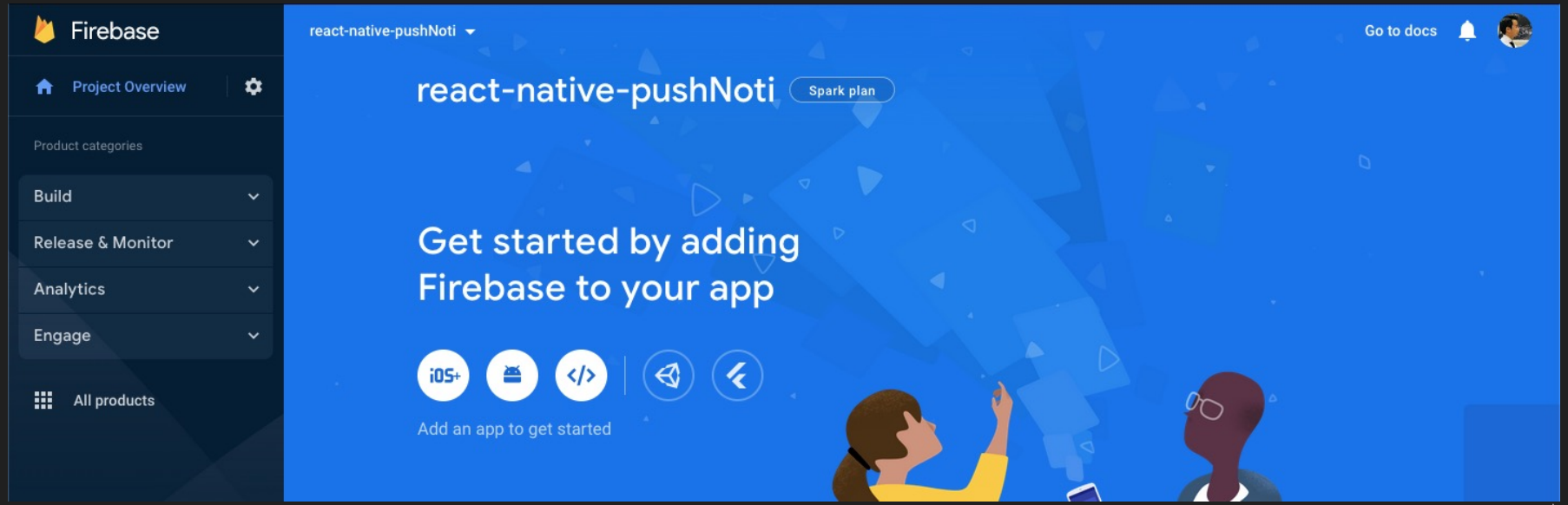
- https://console.firebase.google.com
- Create Project





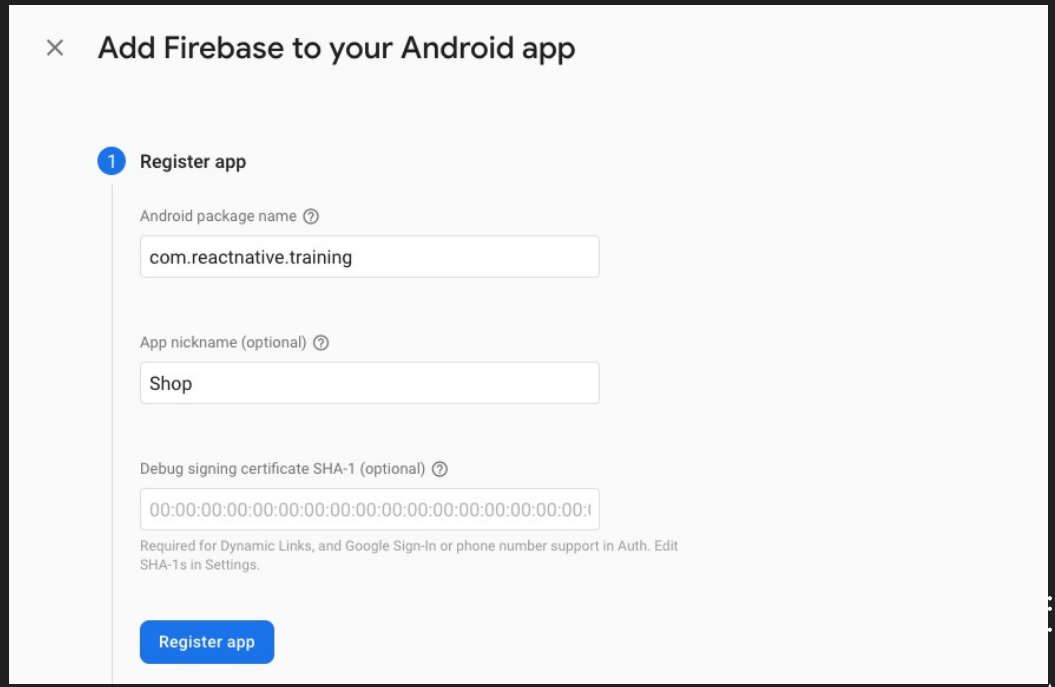
Local push notifications (using Firebase)

- Setup android / ios
- Click Android icon



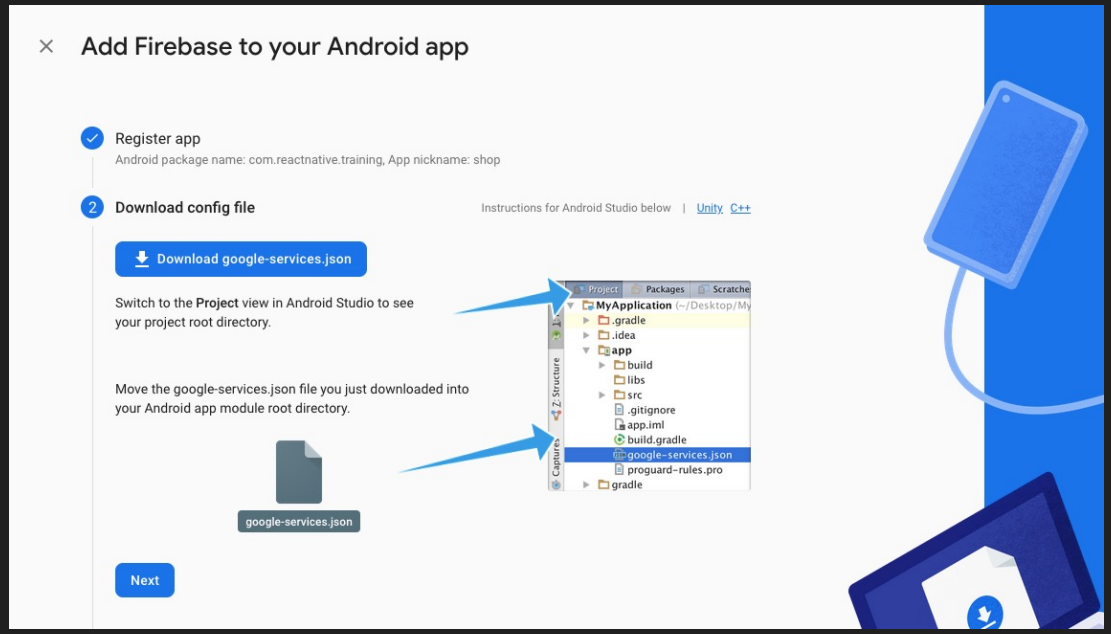
Local push notifications (using Firebase)

- Add project package
- Define app name
- Click Register app



Local push notifications (using Firebase)

- Download google-services.json file



Local push notifications (using Firebase)

- Add firebase sdk

Java Kotlin

App-level build.gradle (<project>/<app-module>/build.gradle):

```
apply plugin: 'com.android.application'
// Add this line
apply plugin: 'com.google.gms.google-services'

dependencies {
  // Import the Firebase BoM
  implementation platform('com.google.firebase:firebase-bom:30.2.0')

  // Add the dependency for the Firebase SDK for Google Analytics
  // When using the BoM, don't specify versions in Firebase dependencies
  implementation 'com.google.firebase:firebase-analytics'

  // Add the dependencies for any other desired Firebase products
  // https://firebase.google.com/docs/android/setup#available-libraries
}
```

By using the Firebase Android BoM, your app will always use compatible Firebase library versions. Learn more

Finally, press "Sync now" in the bar that appears in the IDE:

Gradle files have changed sin Sync now

Previous Next

Local push notifications (using Firebase)

- Install firebase package

```
# Using npm
npm install --save @react-native-firebase/app

# Using Yarn
yarn add @react-native-firebase/app
```

Local push notifications (using Firebase)

- Add firebase messaging code

```
import messaging from '@react-native-firebase/messaging';

// Local Storage
import {appStorage} from '../utils';

export const requestUserPermission = async () => {
  const authStatus = await messaging().requestPermission();
  const enabled =
    authStatus === messaging.AuthorizationStatus.AUTHORIZED ||
    authStatus === messaging.AuthorizationStatus.PROVISIONAL;

  if (enabled) {
    console.log('Authorization status:', authStatus);
    getToken();
  }
};
```

Redux (Add Products)

- Action

```
export const addProducts = data => {
  return {
    type: ADD_PRODUCTS,
    products: data,
  };
};
```

- Reducer

```
case ADD_PRODUCTS:
  return {...state, products: action.products};
```

Redux (Create)

- Action

```
export const createProducts = data => {
  return {
    type: CREATE_PRODUCTS,
    data: data,
  };
};
```

- Reducer

```
case CREATE_PRODUCTS:
  let createProduct;
  if (state.products.length > 0) {
    createProduct = {
      id: state.products.length + 1,
      quantity: 1,
      name: action.data.name,
      currency: action.data.currency,
      price: action.data.price,
      image: action.data.image,
    };
  } else {
    createProduct = action.data;
  }

  return {
    ...state,
    products: [...state.products, createProduct],
  };
```

Redux (Update)

- Action

```
export const updateProducts = data => {
  return {
    type: UPDATE_PRODUCTS,
    updateData: data,
  };
};
```

- Reducer

```
case UPDATE_PRODUCT:
  let updateData = action.updateData;
  let productIndex = state.products.findIndex(
    item => item.id === updateData.id,
  );

  const updateCurrentProducts = [...state.products];
  updateCurrentProducts[productIndex] = updateData;

  return {
    ...state,
    products: updateCurrentProducts,
  };
```

Redux (Add to Cart)

- Action

```
export const orderProducts = data => {
  return {
    type: ORDERS,
    orders: data,
  };
};
```

- Reducer

```
case ORDERS:
  let proId = action.orders.id;
  let proPrice = action.orders.price;
  let updateOrder = state.orders;

  if (state.orders[proId]) {
    updateOrder = {
      id: state.orders[proId].id,
      quantity: state.orders[proId].quantity + 1,
      name: state.orders[proId].name,
      currency: state.orders[proId].currency,
      price: state.orders[proId].price + proPrice,
      image: state.orders[proId].image,
    };
  } else {
    updateOrder = action.orders;
  }

  return {
    ...state,
    orders: {...state.orders, [proId]: updateOrder},
  };
```

Redux (remove from cart)

- Action

```
export const removeOrderFromList = data => {
  return {
    type: REMOVE_ORDER,
    removeItem: data,
  };
};
```

- Reducer

```
case REMOVE_ORDER:
  let updateOrdersItems;
  let removeOrderItem = state.orders[action.removeItem.id];
  let quantityCu = removeOrderItem.quantity;
  let currentAmt = removeOrderItem.price / quantityCu;
  if (quantityCu > 1) {
    const updateOrders = {
      id: removeOrderItem.id,
      quantity: removeOrderItem.quantity - 1,
      name: removeOrderItem.name,
      currency: removeOrderItem.currency,
      price: removeOrderItem.price - currentAmt,
      image: removeOrderItem.image,
    };
    updateOrdersItems = {
      ...state.orders,
      [action.removeItem.id]: updateOrders,
    };
  } else {
    updateOrdersItems = {...state.orders};
    delete updateOrdersItems[action.removeItem.id];
  }
  return {
    ...state,
    orders: updateOrdersItems,
  };
```

Redux (delete product)

- Action

```
export const deleteProduct = id => {
  return {
    type: DELETE_PRODUCT,
    id: id,
  };
};
```

- Reducer

```
case DELETE_PRODUCT:
  return {
    ...state,
    products: state.products.filter(state => state.id !== action.id),
  };
```

Code Push (Appcenter)

- npm install -g appcenter
- appcenter login (access browsing)
- appcenter apps create -d react-native-training -o Android -p React-Native
- appcenter apps create -d react-native-training -o iOS -p React-Native
- appcenter codepush release-react -a accountName/react-native-training -d Production\
- npm install react-native-code-push
- npm install -g appcenter-cli

Code push setup

---

android/settings.gradle

```
include ':app',':react-native-code-push'
project(':react-native-code-push').projectDir = new File(rootProject.projectDir, '../nod
```

android/app/build.gradle

```
apply from: "../../node_modules/react-native-code-push/android/codepush.gradle"
```

Code Push (Appcenter)

MainApplication.java

```
...
// 1. Import the plugin class.
import com.microsoft.codepush.react.CodePush;

public class MainApplication extends Application implements ReactApplication {

    private final ReactNativeHost mReactNativeHost = new ReactNativeHost(this) {
        ...

        // 2. Override the getJSBundleFile method in order to let
        // the CodePush runtime determine where to get the JS
        // bundle location from on each app start
        @Override
        protected String getJSBundleFile() {
            return CodePush.getJSBundleFile();
        }
    };
}
```

Code Push (Appcenter)

strings.xml

```xml
<resources>
    <string name="app_name">Training</string>
    <string moduleConfig="true" name="CodePushDeploymentKey">47uRBLgNydocOPgOyLjkE_9M6drglvOtcyHrK</string>
</resources>
```

Code Push (Appcenter in dashboard)

```
useFocusEffect(
  useCallback(() => {
    // code push
    var updateDialogOptions = {
      title: 'New version avaliable',
      optionalUpdateMessage: 'Upgrade your application now',
      optionalIgnoreButtonLabel: Platform.OS === 'android' ? '' : '',
      optionalInstallButtonLabel: 'Upgrade',
    };
    codePush.sync(
      {
        checkFrequency: codePush.CheckFrequency.ON_APP_RESUME,
        updateDialog: updateDialogOptions,
        installMode: codePush.InstallMode.IMMEDIATE,
      },
      status => {
        switch (status) {
          case codePush.SyncStatus.DOWNLOADING_PACKAGE:
            console.log('downloading....');
            break;
          case codePush.SyncStatus.INSTALLING_UPDATE:
            console.log('installing');
            break;
          case codePush.SyncStatus.UPDATE_INSTALLED:
            console.log('upgrade....');
            break;
          default:
            break;
        }
      },
      ({receivedBytes, totalBytes}) => {
        console.log(receivedBytes / totalBytes);
      },
```

Splash Screen & Icons generate

Generate icons & splash screen image

- https://appicon.co/ (generate icons & image)

android/settings.gradle

```
include ':react-native-splash-screen'
project(':react-native-splash-screen').projectDir = new File(rootProject.p
```

Publishing to Google Play Store

Generating an upload key

- keytool -genkeypair -v -storetype PKCS12 -keystore my-upload-key.keystore -alias my-key-alias -keyalg RSA -keysize 2048 -validity 10000
- Place the my-upload-key.keystore file under the android/app directory in your project folder.

android/gradle.properties

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

Publishing to Google Play Store

android/app/build.gradle

```
...
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
        }
    }
}
...
```

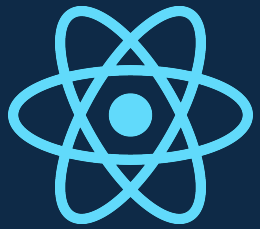
Publishing to Google Play Store

Generate for playstore

- ./gradlew bundleRelease

Generate for .apk

- ./gradlew assembleRelease



The End