Getting Started with React Native and Expo

Table of Contents

1. Introduction 3
2. Overview 3
3. Installing the Development Environment 3
    3.1 Computer Installation 3
        3.1.1 Install Node.js 3
    3.2 Phone Installation (iOS or Android) 4
4. Creating a New React Native App with Expo 4
    4.1 Initializing the Project 4
    4.2 Viewing Your App on Your Phone 4
5. Understanding the Project Structure 6
    5.1 Create metro.config.js File 8
6. Network Configuration 9
    6.1. Set up IP Address 9
    6.2. Ports 10
7. Modifying Your App 10
    7.1 Open and Edit Your index.tsx File 10
8. Troubleshooting 11
9. Additional Resources 11
10. Glossary 12
11. Additional Terms 12

1. Introduction

React Native is an open-source framework for building mobile applications using JavaScript, allowing developers to create apps for both Android and iOS with a single codebase. This approach greatly reduces development time and effort, making it an efficient choice for cross-platform app development.

Expo, on the other hand, is a platform and set of tools that simplify React Native development, especially for beginners. It offers a streamlined development experience by providing pre-configured workflows, a built-in library of components, and easy access to native device features. With Expo, you can avoid the complexities of native code, making it easier to get started and focus on building your app.

This guide is designed for people who are new to React Native and want a hands-on introduction. It will walk you through installing the necessary development environment, setting up Expo, and creating your first React Native app. By the end of this guide, you'll have a foundational understanding of how React Native works and how to use Expo to develop and run your applications efficiently.

2. Overview

What is React Native? React Native Combines React, a JavaScript library for building user interfaces, with the capabilities of Android or iOS platforms. It allows you to create mobile apps using JavaScript without needing to learn native languages like Kotlin, Java, Swift, or Objective-C.

Who is this guide for? It is intended for those with very basic knowledge of JavaScript and React, although prior experience is helpful but not required.

3. Installing the Development Environment

3.1 Computer Installation

You need to install Node.js programs on your computer. You can use either PowerShell or the Windows Command Prompt for this process. It's recommended to run the installation as an administrator to ensure you have the necessary permissions to modify system files.

3.1.1 Install Node.js

1. Visit the [Node.js download page](https://nodejs.org/en/download/).

2. Download the latest LTS version suitable for your operating system (Windows, macOS, or Linux).

3. Open the downloaded file and follow the setup wizard:

   - Click “Next” to proceed.

   - Accept the license agreement and click “Next.”

   - Choose your preferred installation location, then click “Next.”

   - Ensure the "Add to PATH" option is selected (important).

   - Click “Install” to begin, and then “Finish” when the installation completes.

4. Verify the installation by running the following commands in your terminal/command prompt:

node –v and **npm -v**

3.2 Phone Installation (iOS or Android)

Download the **Expo Go** app from the App Store (iOS) or Google Play Store (Android).

Ensure your computer and phone are connected to the **same Wi-Fi network** for seamless communication.

4. Creating a New React Native App with Expo

4.1 Initializing the Project

1. Open your terminal/command prompt.

2. Navigate to your preferred folder:

cd path/to/your/folder

Create a new project by running:

npx create-expo-app my-app

- This command will fetch and run the latest "create-expo-app" tool without needing a global installation.

4. Navigate into your project folder:

cd my-app

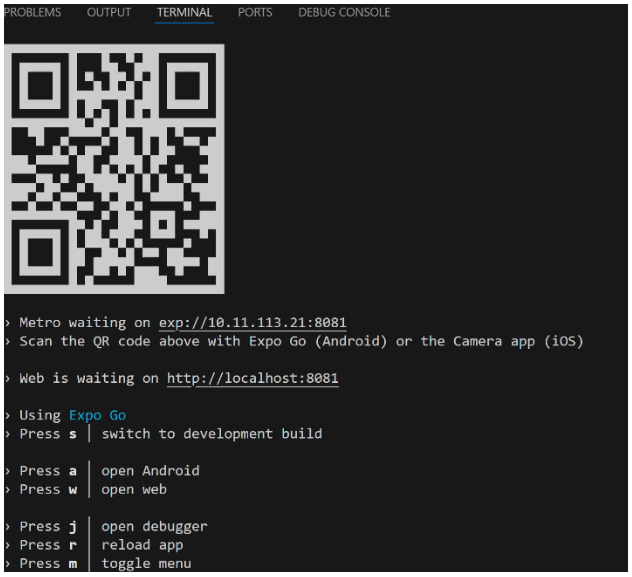
5. Start your project:

npx expo start

4.2 Viewing Your App on Your Phone

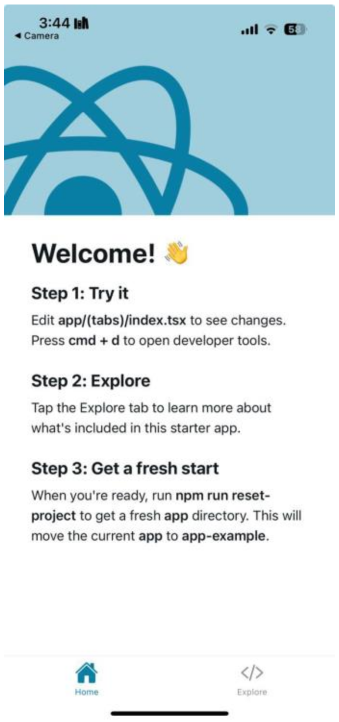
1. Open the Expo Go app on your phone.

2. Scan the QR code displayed in your terminal or web browser.



```
› Metro waiting on exp://10.11.113.21:8081
› Scan the QR code above with Expo Go (Android) or the Camera app (iOS)

› Web is waiting on http://localhost:8081

› Using Expo Go
› Press s │ switch to development build

› Press a │ open Android
› Press w │ open web

› Press j │ open debugger
› Press r │ reload app
› Press m │ toggle menu
```

3. You should see your app running on your phone.



Welcome! 👋

Step 1: Try it

Edit **app/(tabs)/index.tsx** to see changes. Press **cmd + d** to open developer tools.

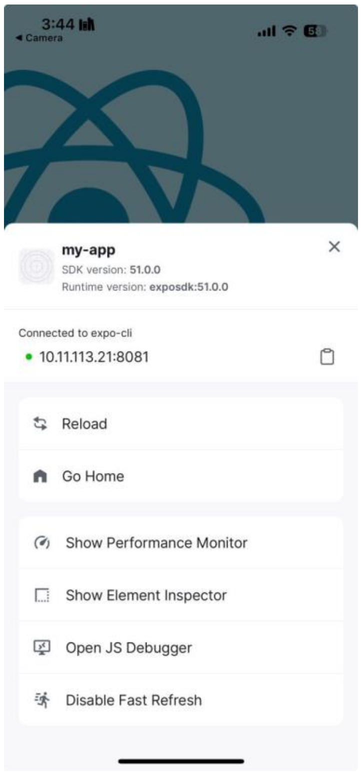
Step 2: Explore

Tap the Explore tab to learn more about what's included in this starter app.

Step 3: Get a fresh start

When you're ready, run **npm run reset-project** to get a fresh **app** directory. This will move the current **app** to **app-example**.

You can access the Dev Menu to test and debug your project efficiently. To open the Dev Menu, simply shake your device if you're running the app on a physical device. The Dev Menu provides options such as reloading the app, opening debugger, and inspecting elements, making it easier to troubleshoot and enhance your project during development.



my-app

SDK version: **51.0.0**

Runtime version: **exposdk:51.0.0**

Connected to expo-cli

● 10.11.113.21:8081

5. Understanding the Project Structure

When you create the project using “npx create-expo-app”, it initializes a TypeScript project by default, meaning most of the files will have the. tsx extension. You may need to keep most of the folders and files, as they are important for development, configuration, and functionality. However, a few files and folders may be unnecessary and can be safely deleted if not in use. Below is the complete structure where most components serve key purposes:

1. **.expo:** Stores Expo's internal data. Hidden by default and should only be modified if troubleshooting.

2. **app/:** This folder contains the core structure of your project and is critical for the app's functionality. Inside, you'll find “index.tsx”, which serves as the main page of the app. Additionally, there is a “tab/” folder, which is part of the sample project created by default. While it provides examples for organizing tabs, it's not necessary to keep all the tabs. You can remove them and just retain the “index.tsx” file as the primary page for your app.

3. **assets/:** Stores static resources like fonts and images, Necessary if your app uses such resources.

4. **components/:** stores component test files, keep to maintain organized UI components.

5. **hooks/:** Contains custom React hooks, keep if using hooks for state management.

6. **node_modules/:** Contains all project dependencies. Do not delete as it's essential for your app to run.

7. **scripts/:** Stores automated task scripts. Delete if empty or unused.

8. **app.json:** Expo project configuration, including app metadata and settings, Essential for the project.

9. **babel.config.js:** Configures Babel for transpiling JSX and modern JavaScript/TypeScript. Necessary for correct compilation.

10. **expo-env.d.ts:** TypeScript declaration for Expo environments. Keep if using TypeScript.

11. **package-lock.json:** Locks npm dependency versions for consistency across environments. Keep for stability.

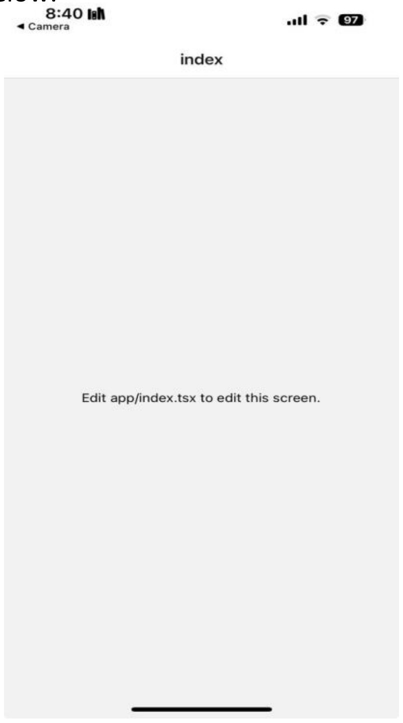
12. **package.json:** Core project file that lists dependencies and scripts. Essential for the app's configuration.

13. **tsconfig.json:** TypeScript configuration file for compiling and type-checking. Necessary if using TypeScript.

In essence, in a simple React Native project, the essential files and structure include: the **app/** folder, which contains components or pages, with **index.tsx** acting as the main page of your app. The **app.json** file provides the configuration for Expo, while **node_modules/** holds the installed dependencies. The **metro.config.js** file handles Metro bundler configurations, and **package.json** manages the project's dependencies and scripts. You can choose to work with either TypeScript (.tsx) or JavaScript (.js), depending on your preference, as both are equally valid.

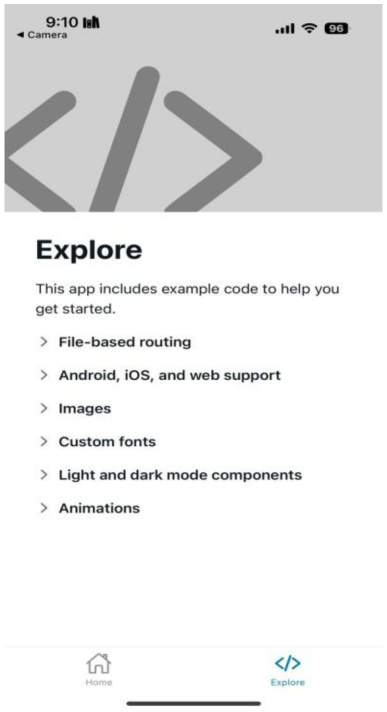
Important Note: Please explore both tabs in the sample app.

- In the first tab (Home), you'll find helpful information on how to begin modifying the app. Specifically, Step 3 provides instructions for resetting the project. By running the command "**npm run reset-project**", you can replace the current sample files with a fresh version of the app and all the necessary files. Once the command has been executed, you'll notice the app has been updated as shown below.



Edit app/index.tsx to edit this screen.

- For additional details on how to get started working with the app, refer to the other sections (Explore).



Explore

This app includes example code to help you get started.

5.1 Create metro.config.js File

The metro.config.js file is essential for configuring the Metro bundler, which React Native uses to compile and bundle your app's assets. This file allows you to customize how the bundler handles your project's modules, paths, and more. It's particularly useful for ensuring compatibility with specific node modules and setting up custom configurations like working with WebSockets, encrypted communications, or fixing IP addresses.

To create this file, navigate to the root of your project and add metro.config.js. This file will reside in the main directory to ensure Metro applies your custom settings when running the project.

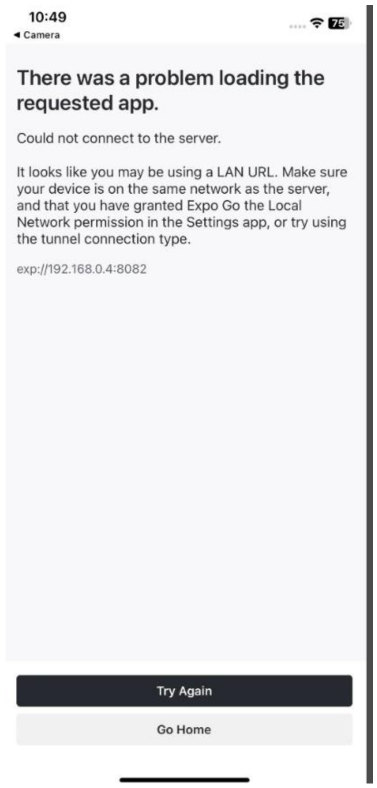
For example, you can add this code to the metro.config.js file to ensure compatibility with certain node modules like url, buffer, stream and process which are not natively supported in React Native. This configuration allows you to use these modules seamlessly within the project by resolving their browser-compatible versions.

```
const { getDefaultConfig } = require('expo/metro-config');

const defaultConfig = getDefaultConfig(__dirname);

defaultConfig.resolver.extraNodeModules = {
  ...defaultConfig.resolver.extraNodeModules,
  url: require.resolve('react-native-url-polyfill'),
  buffer: require.resolve('buffer'),
  stream: require.resolve('stream-browserify'),
  process: require.resolve('process'),
};

module.exports = defaultConfig;
```

Tip: If the Expo Go app is unable to connect to your project, particularly on a public network, it might be necessary to modify your network settings. For step-by-step guidance, check the "Network Configuration" section. Otherwise, feel free to skip Section 6 and proceed to Section 7.

6. Network Configuration

1. Make sure your device is on the same network as the server, and
2. that you have granted Expo Go the Local Network permission in the Settings app, or
3. try using the tunnel connection type.
   "**npx expo start --tunnel**"


There was a problem loading the requested app.

Could not connect to the server.

It looks like you may be using a LAN URL. Make sure your device is on the same network as the server, and that you have granted Expo Go the Local Network permission in the Settings app, or try using the tunnel connection type.

exp://192.168.0.4:8082

6.1. Set up IP Address

If you want to set a fixed IP address for your React Native project, follow these steps:

1. Create or edit the "metro.config.js" file in your project root and add the following configuration:

```
// metro.config.js
const { getDefaultConfig } = require("expo/metro-config");
const defaultConfig = getDefaultConfig(__dirname);
module.exports = {
  ...defaultConfig,
  server: {
    port: 19000, // Ensure this matches the desired Metro Bundler port
    enhanceMiddleware: (middleware) => {
      return (req, res, next) => {
        if (req.url.startsWith('/node_modules/')) {
          req.url = req.url.replace('/node_modules/', '/');
        }
        return middleware(req, res, next);
      };
    },
  },
};
// Set REACT_NATIVE_PACKAGER_HOSTNAME as a backup
process.env.REACT_NATIVE_PACKAGER_HOSTNAME = '10.11.113.21'; // Ensure this matches your
current LAN IP
```

2. Set the "REACT_NATIVE_PACKAGER_HOSTNAME" environment variable by running the following command in your terminal:

"**export REACT_NATIVE_PACKAGER_HOSTNAME=<your-computer-ip>**"

Replace "<your-computer-ip>" with your actual IP address, for example, "192.168.1.10".

6.2. Ports

The Metro Bundler starts working by default with "localhost" on port 8081. If it doesn't work, you can switch to using your computer's IP address with port 19000, which is commonly used by the Expo development server for better connectivity with your device.

Port Used By Purpose

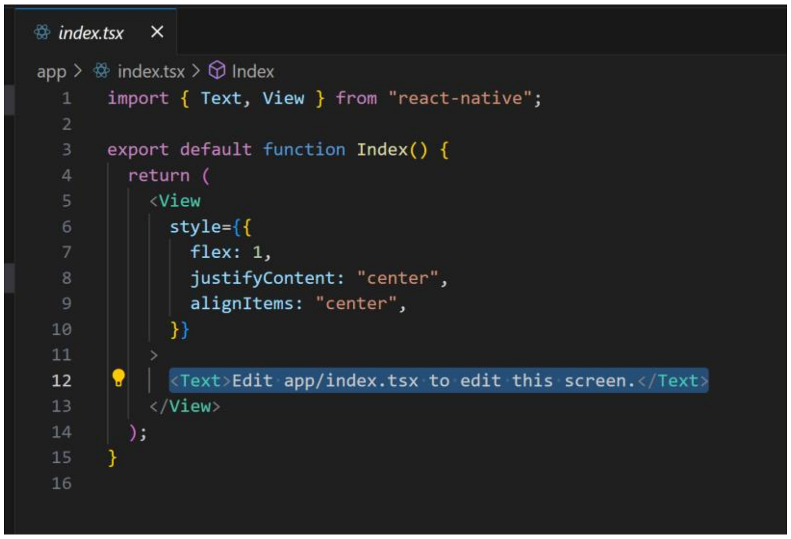
8081	React Native Metro Bundler	Serves the JavaScript bundle and handles live reloading for React Native projects.
19000	Expo Dev Server	Main server for serving JavaScript code to the Expo Go app.
19001	Expo WebSocket Server	Enables real-time communication, debugging, and logs for Expo projects.
19002	Expo Developer Tools UI	Hosts the web-based Expo Developer Tools interface.

7. Modifying Your App

7.1 Open and Edit Your index.tsx File

1. Open "index.tsx" in a text editor (e.g., Visual Studio Code, Notepad).

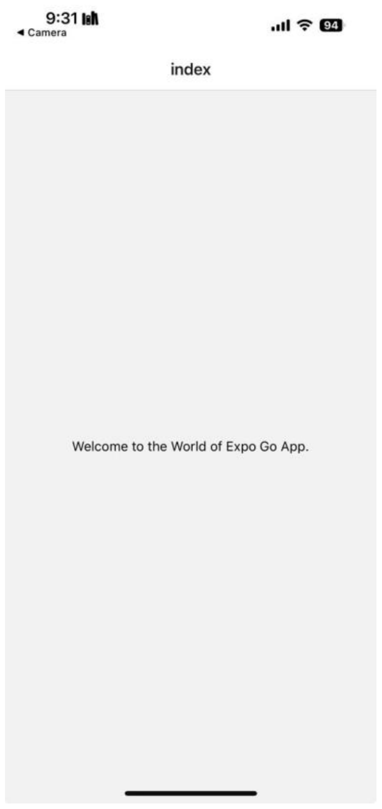
2. Find the line with "<Text>" containing "Edit app/index.tsx to edit this screen.".



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

export default function Index() {
  return (
    <View
      style={{
        flex: 1,
        justifyContent: "center",
        alignItems: "center",
      }}
    >
      <Text>Edit app/index.tsx to edit this screen.</Text>
    </View>
  );
}

```

3. Change this text to " Welcome to the World of Expo Go App." and save the file.

You should see the changes reflected in your app automatically.



Welcome to the World of Expo Go App.

8. Troubleshooting

If you encounter issues during setup, refer to the React Native [Setup Troubleshooting Guide] (https://reactnative.dev/docs/troubleshooting).

Common issues include:

Expo CLI not recognized: Ensure you installed it with “npm install -g expo-cli” and restart your terminal.

9. Additional Resources

[React Native Documentation] (https://reactnative.dev/docs/getting-started)

[JavaScript Tutorials by Mozilla] (https://developer.mozilla.org/en-US/docs/Web/JavaScript)

[React JavaScript Library] (https://reactjs.org/)

[Expo Documentation] (https://docs.expo.dev/)

Your glossary is a great start! Here are a few additional key terms that might be useful, especially considering the context of React Native, Expo, and your project:

10. Glossary

Node.js: JavaScript runtime environment that allows JavaScript to be run on the server side.

npx: A tool that runs Node.js packages without installing them globally, making it convenient for one-time use.

JSX: JavaScript XML syntax used in React to describe what the UI should look like. It combines JavaScript and HTML-like syntax.

State: Data that controls a component's behavior in React. When the state changes, the component re-renders.

11. Additional Terms

Props: Short for "properties," props are read-only inputs passed to components in React, allowing for dynamic content.

Component: A building block of React applications, representing parts of the user interface.

Expo: A framework and platform for universal React applications that helps streamline the development process, especially for mobile apps.

React Native: A framework that enables building mobile apps using JavaScript and React.

Metro Bundler: A development server that compiles and bundles assets in React Native projects.

npm: Node Package Manager, used to manage and install JavaScript libraries and dependencies in Node.js projects.

Redux: A state management library commonly used in React applications to manage application-level state.

TypeScript: A typed superset of JavaScript that adds static types, improving code readability and reducing bugs.