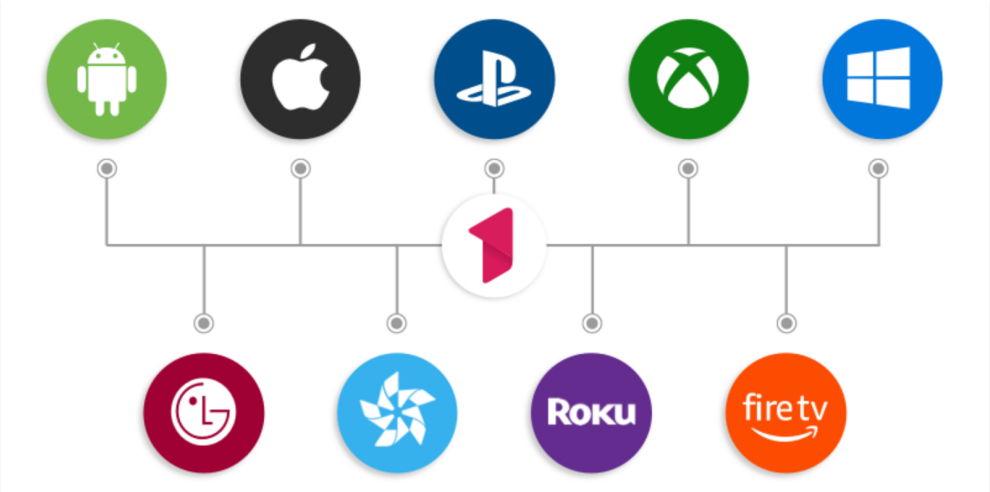
Test on Multiple Platforms

Let's get into one of the best parts about You.i Engine One: multi-platform support. You.i Engine One acts as a bridge for your code on any of the following platforms: Android (mobile and TV), Apple (iOS, tvOS), PlayStation, Xbox, LG, Tizen, Roku, and Amazon Fire TV.



Use the instructions below to run your app on an Android device or emulator and on Roku.

Try it on Android

Pro Tip: To load your application on an Android device, you'll need to install Android Studio. You'll also need Android NDK version **17c**, which you may need to retrieve from Android's NDK archives. It doesn't matter where you save NDK version 17c; just remember the path because you'll need to set it as an environment variable.

1. Configure the following environment variables.

Mac Linux Windows

```
# add to your ~/.bash_profile and reload
export ANDROID_HOME=~/Library/Android/sdk
export ANDROID_NDK_HOME=<path-to-your-downloaded-ndk-17c>
export JAVA_HOME=<path-to-jdk>  # you can use the Android Studio JDK: "/Applications/Android
export PATH=$PATH:$ANDROID_HOME:$ANDROID_HOME/tools:$ANDROID_HOME/platform-tools:$ANDROID_NDK
```

2. Use `youi-tv doctor` to validate your Android configuration. Make any changes needed to meet the requirements.

3. Generate and build the Android project.

```
cd MyApp
youi-tv build -p android
```

4. Connect your device to your development computer, or choose to use the Android Studio emulator.

Note: Since we have no need to edit any files right now, you don't need to open the project in Android Studio. To use the emulator (virtual device), start it from the Android Studio splash screen by clicking Configure > AVD Manager.

5. Run the following command to load your app onto your mobile device or simulator.

```
cd youi/build/android/project
```

```
adb devices # to make sure your device is connected & accessible
./gradlew installDebug
```

6. If not already running, start the Metro bundler (yarn server) from your development platform.

```
yarn start  # can be from any folder in your project
```

> **Note:** Because we're running the yarn server on your development computer, both it and your mobile device must be on the same WiFi network.

7. Start the application. Go to the device and launch your application, or use the command line below.

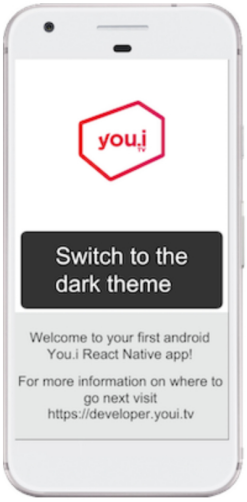
```
./gradlew startApplication
```

Your single-source app is now able to run on both your development platform and a mobile device.

It doesn't look quite as nice on the phone as it does on your development platform. Let's fix this.

Device Form Factor

As you no doubt noticed, the app that looked great as a landscape desktop app has some display issues on your mobile phone. It's even worse if you rotate your phone to landscape mode.



A clever way to approach this problem is to use the custom `FormFactor` RN module that comes with You.i Engine One. With this module, we can make our app behave differently for each type of form factor it supports.

> **Note:** See the docs for a list of our custom modules and custom components.

1. To fix the problems we saw above, we'll employ a stylesheet per form factor. Conveniently, `index.youi.js` already imports the `FormFactor` component. Take a look at the list of imports.

2. FormFactor returns one of: TV, Mobile, or Tablet. By default, You.i Engine One development platforms report themselves as "TV", so let's create a TV stylesheet. Tablet and Mobile can share a second (default) stylesheet.

   Add a new variable to your `YiReactApp` component near the top of `index.youi.js` that checks the current form factor and uses that to choose a stylesheet.

```
platformStyles = FormFactor.select({
  TV: stylesTV,
  default: stylesTouch
});
```

3. Add stylesheets for our new styles `stylesTv` and `stylesTouch` by pasting the following at the bottom of `index.youi.js`, just above our current `styles` stylesheet.

```
const stylesTV = StyleSheet.create({
  bodyText: {
    fontSize: 20
  },
  headlineText: {
    fontSize: 30,
    textAlign: "center",
    marginBottom: 15
  }
});

const stylesTouch = StyleSheet.create({
  bodyText: {
    fontSize: 7
  },
  headlineText: {
    fontSize: 10,
    textAlign: "center",
    marginBottom: 10
  }
});
```

4. Also, take a look at the initial `styles` stylesheet at the bottom of `index.youi.js`. Notice that the default app created when you run `youi-tv init` already uses `FormFactor` to determine a height and width for the image.

```
image: {
  height: FormFactor.isTV ? 225 : 75,
  width: FormFactor.isTV ? 225 : 75,
  resizeMode: "contain"
},
```

> **Note:** Images from URIs always need their size explicitly specified rather than relying on them growing to fill their parent view.

Together, the stylesheets above configure image size, font sizes, and margins that are smaller for touch devices.

5. Update your app to use these new styles by swapping `styles.<stylename>` with `this.platformStyle.<stylename>` in each of the three text components.

```
<View style={[styles.bodyContainer, themeStyles.body]} focusable={true} accessible={true}>
  <Text
    style={[this.platformStyles.headlineText, themeStyles.headlineText]}
    accessibilityLabel="Welcome to your first {PlatformName.name} You I React Native app"
  >
    Welcome to your first {PlatformName.name} You.i React Native app!
  </Text>
  <Text
     style={[this.platformStyles.bodyText, themeStyles.bodyText]}
  >
    For more information on where to go next visit
  </Text>
  <Text
    style={[this.platformStyles.bodyText, themeStyles.bodyText]}
    accessibilityLabel="https://developer dot you i dot tv"
  >
    https://developer.youi.tv
  </Text>
</View>
```

6. Because `bodyText` and `headlineText` are now being used from the specific form factor style sheets, you can remove these from the original `styles` stylesheet. This step is optional, but helps keep your code clean.

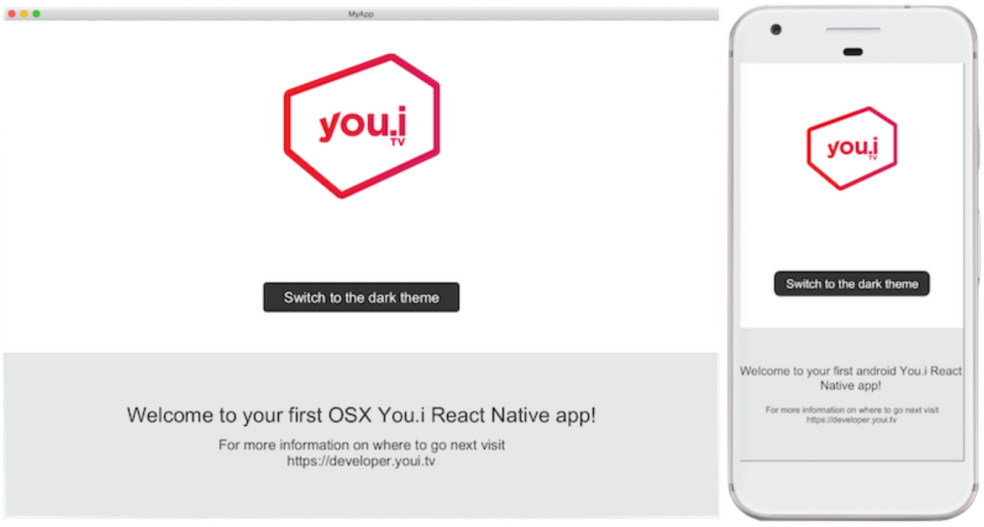
7. Run your app again to see an improvement in some aspects, but the button text is still too large.

   Replace the contents of `button.js` with this version that includes a stylesheet based on the form factor.

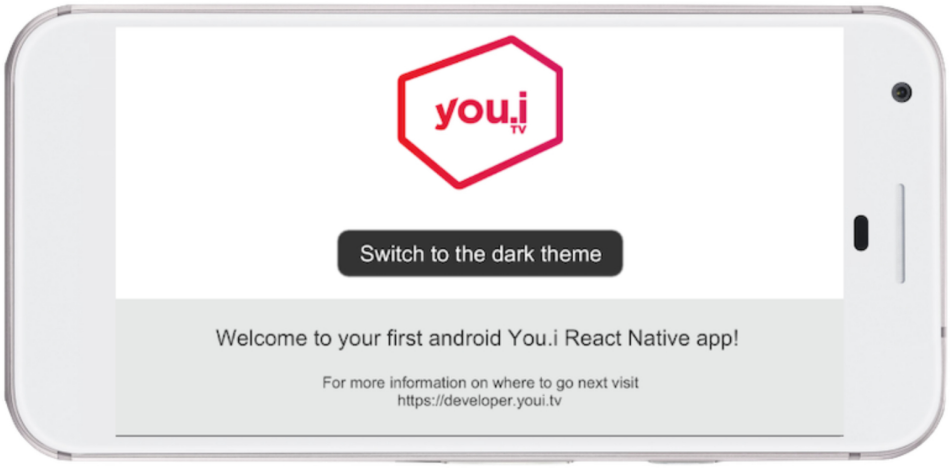
```
import React from "react";
import { StyleSheet, Text, TouchableOpacity, View } from "react-native";
import { FocusManager, FormFactor } from "@youi/react-native-youi";

export const Button = ({
  buttonStyle,
  defaultFocus,
  onPress,
  textStyle,
  title
}) => {
```

8. Run your app on both your development environment and on Android. It now looks great on both!



   Even if you rotate the screen to portrait.



Try it on Roku

Do you have a Roku device or Roku-capable TV? Testing your application on Roku is quick and easy with the You.i Engine One Roku Cloud Solution.

Our Cloud Solution uses a combination of a small client application that's distributed through the Roku store and a server-side application that resides in the cloud. While developing your app, you'll side-load the client application to your test device and use your local macOS or Linux machine instead of the cloud.

Note: Roku development isn't supported from Windows.

1. Connect your Roku device to the same network as your development computer.
2. Put your Roku into developer mode. Take note of your Roku device's IP address and developer password; you'll need these below.
3. Build the Roku server application from the application folder.

   Note: When you're moving between builds that run on your development computer (`osx` or `linux` targets) and server builds for Roku, call `youi-tv clean` before your first build to get a clean configuration.

   Mac Linux

   ```
   youi-tv clean osx
   youi-tv build -p osx -d YI_BUILD_CLOUD_SERVER=ON
   ```

4. Run the server application. First, start the Metro Bundler.

   ```
   yarn start
   ```

   Warning: A known issue may affect placement of visual elements in your development environment. For best results, look at the screen connected to your Roku device instead of your development environment, or run the app in headless mode by adding the `-n1` flag to your server command, as shown below. We're working hard to resolve this issue in a future release.

   Open a second terminal window. In that window, go to your app folder (`cd MyApp`) and start the server application.

   Mac Linux

   ```
   cd youi/build/osx/Debug
   ./MyApp -n1
   ```

   Unlike other platforms, the Roku server application doesn't launch a graphical component immediately. The graphical interface opens after the client connects.

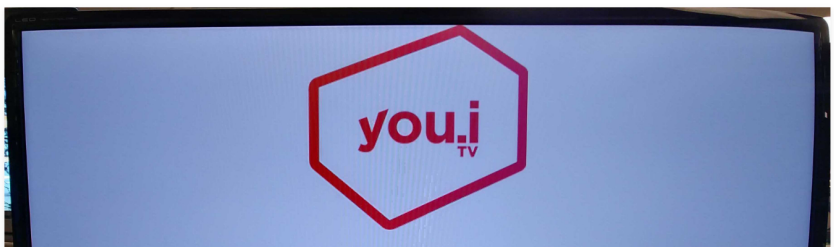
5. Build and side-load the client application on your Roku device.

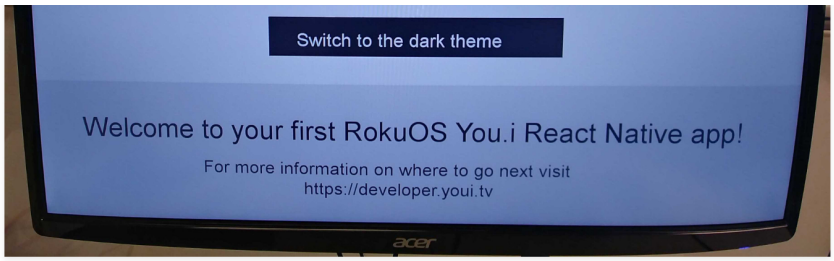
   Open a third terminal window to run the following commands.

   ```
   cd MyApp/client
   ./build.rb -u rokudev:<roku password> -l -r <roku ip address>
   ```

   Pro Tip: The `-l` option for the Roku client build script uses the default network IP address for your development computer. If you need to specify a different IP address (for example, if you have two network interfaces), use `-s <ip address>` to specify the address where the Roku device can find the Metro Bundler running.

6. After a few seconds, the server application appears on your development computer and is mirrored by the client on your Roku device. Notice the platform name is `RokuOS`!



Interact with the application with your Roku remote. Continue development activities like using `Ctrl-R` to reload or open the dev panel from the server app on your development computer.

Fantastic! You've completed our getting started tutorial. Take a look at the rest of our documentation for tips and information on how to develop the rest of your You.i RN application.

If you had any issues with getting this sample running, you can download a zip file here with the completed JavaScript and C++ files.

Updated: February 18, 2020