NOVEMBER 25

React Native (and Expo)

Created By
Connor Wells

Referencing
React Native Tutorial
Expo Tutorial

Brief Introduction to React Native

React

- JavaScript
- UI Components
- Code + Markup
- Dynamic States

Native

- React Syntax
- Device-Specific UI Elements
- Handles Multi-Platform

Key Concepts

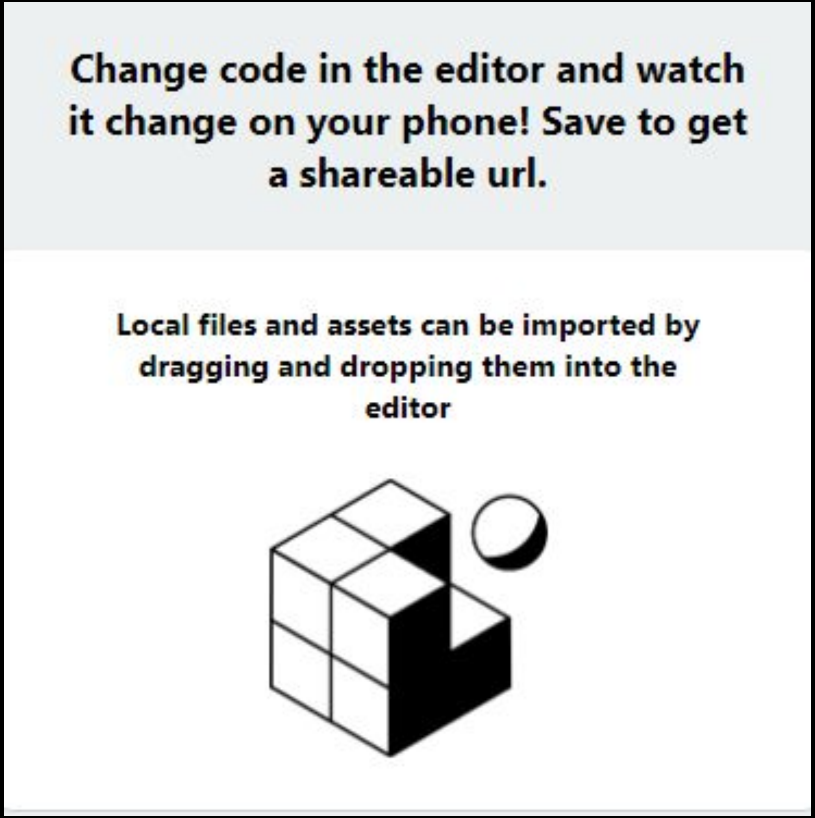
1. Components
   a. Anything seen on screen
2. Props
   a. On-creation variables
3. State
   a. Initialized and writeable

React Native – Practical

Navigate to:
https://snack.expo.dev

Sign up if you want to save any work.

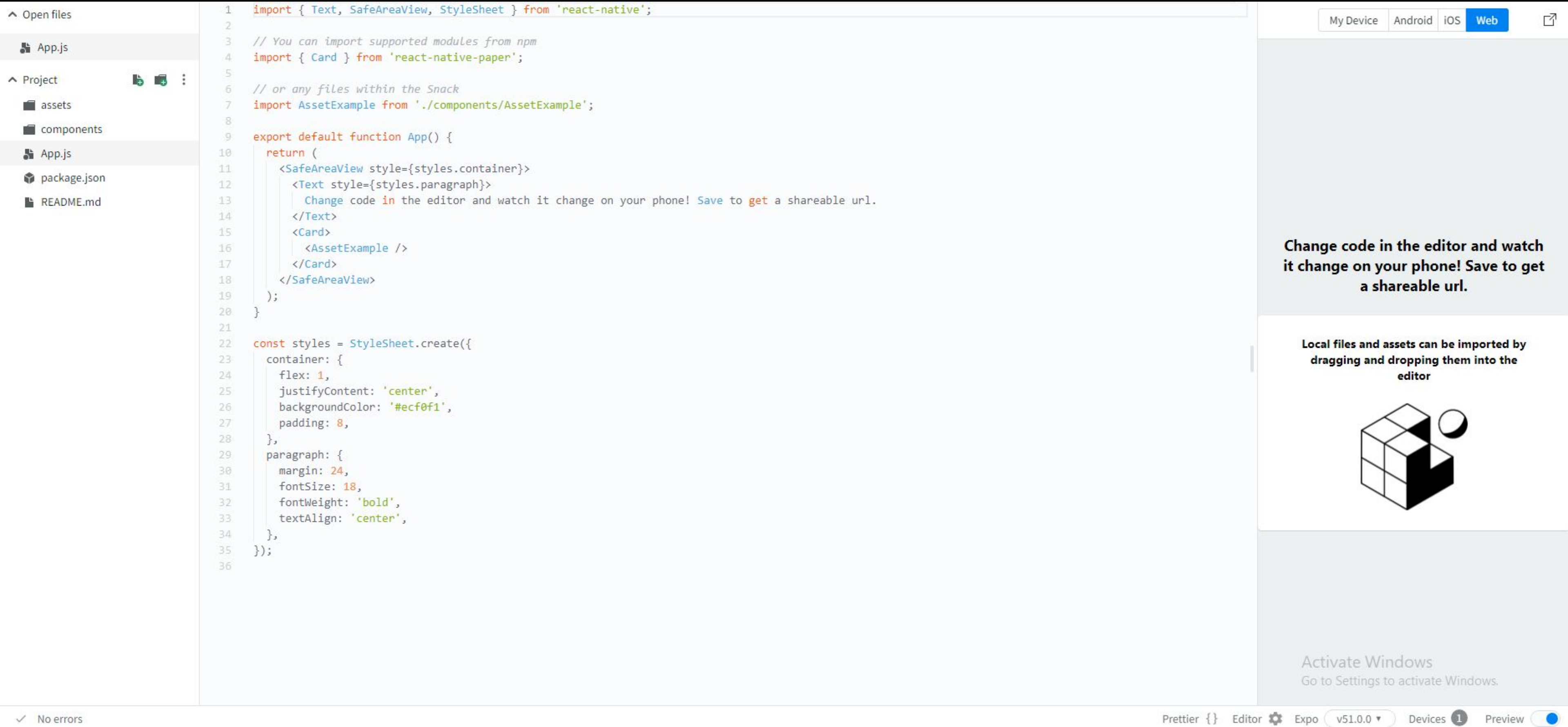
Expo will be useful later.



Change code in the editor and watch it change on your phone! Save to get a shareable url.

Local files and assets can be imported by dragging and dropping them into the editor

React Native – Expo Explanation (https://snack.expo.dev)



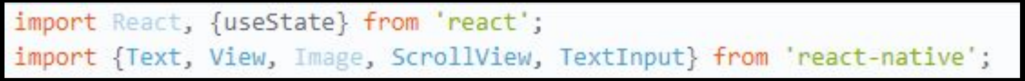
```
import { Text, SafeAreaView, StyleSheet } from 'react-native';

// You can import supported modules from npm
import { Card } from 'react-native-paper';

// or any files within the Snack
import AssetExample from './components/AssetExample';

export default function App() {
  return (
    <SafeAreaView style={styles.container}>
      <Text style={styles.paragraph}>
        Change code in the editor and watch it change on your phone! Save to get a shareable url.
      </Text>
      <Card>
        <AssetExample />
      </Card>
    </SafeAreaView>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    backgroundColor: '#ecf0f1',
    padding: 8,
  },
  paragraph: {
    margin: 24,
    fontSize: 18,
    fontWeight: 'bold',
    textAlign: 'center',
  },
});
```

Change code in the editor and watch it change on your phone! Save to get a shareable url.

Local files and assets can be imported by dragging and dropping them into the editor

React Native – Core Components & Imports

import {component} from 'react-native'



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

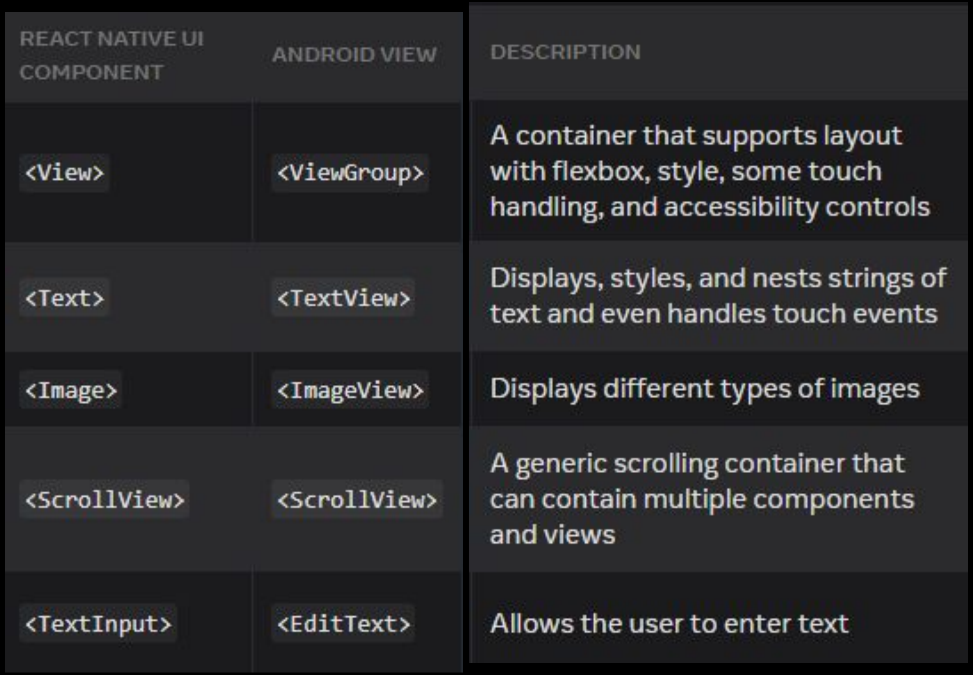
View – Layout grouping

Text – String display

Image – Image display

ScrollView – Scrolling container

TextInput – Text box for user input



REACT NATIVE UI COMPONENT	ANDROID VIEW	DESCRIPTION
`<View>`	`<ViewGroup>`	A container that supports layout with flexbox, style, some touch handling, and accessibility controls
`<Text>`	`<TextView>`	Displays, styles, and nests strings of text and even handles touch events
`<Image>`	`<ImageView>`	Displays different types of images
`<ScrollView>`	`<ScrollView>`	A generic scrolling container that can contain multiple components and views
`<TextInput>`	`<EditText>`	Allows the user to enter text

React Native – Function

```
const HelloWorldApp = () => {};
```

- Functions return a component.
- () are input parameters
- return goes inside {}

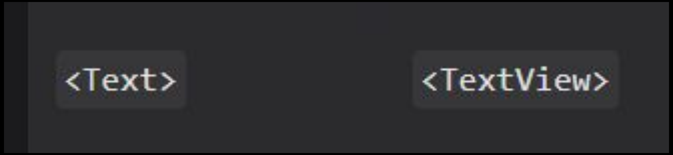
React Native – Function

```
const HelloWorldApp = () => {
  return <Text>Hello World</Text>;
};
export default HelloWorldApp;
```

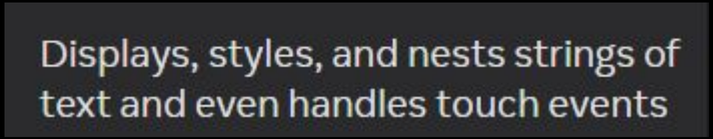
- Functions return a component.
- Returns “Hello World” text
- Export default “FunctionName” works for our purposes.

React Native – <Text>

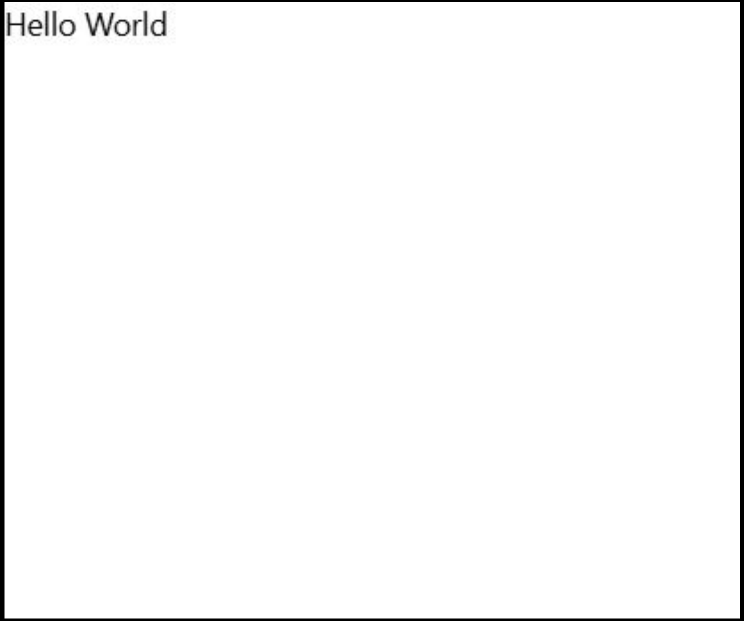
```
const HelloWorldApp = () => {
  return (
      <Text>Hello World</Text>
  );
};
export default HelloWorldApp;
```



<Text> <TextView>

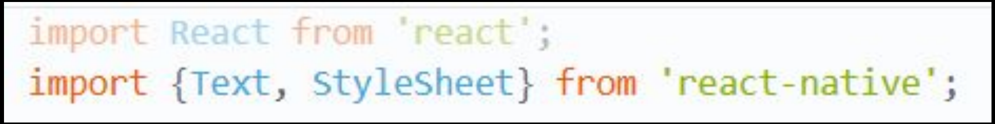


Displays, styles, and nests strings of text and even handles touch events



```
Hello World
```

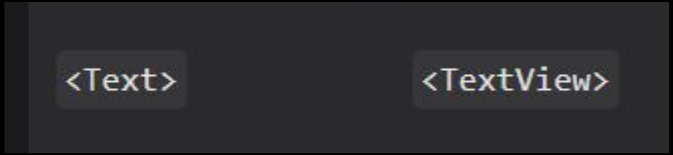
React Native – 〈StyleSheet〉

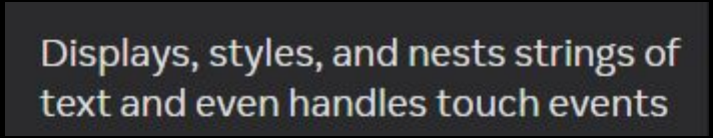

```
import React from 'react';
import {Text, StyleSheet} from 'react-native';
```

```
const page = StyleSheet.create({
  text: {
    color: '#000',
    fontSize: 14,
    fontWeight: 'bold',
  },
  header: {
    color: '#61dafb',
    fontSize: 30,
    marginTop: 36,
  },
});
```

- Like CSS, various styles
- Easy style definition per element
- style = {page.text}

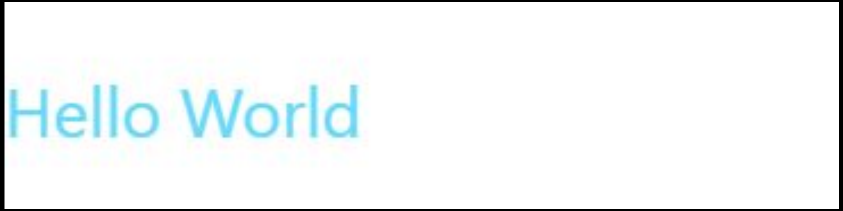
React Native – ⟨Text⟩ + ⟨StyleSheet⟩


<Text>
<TextView>

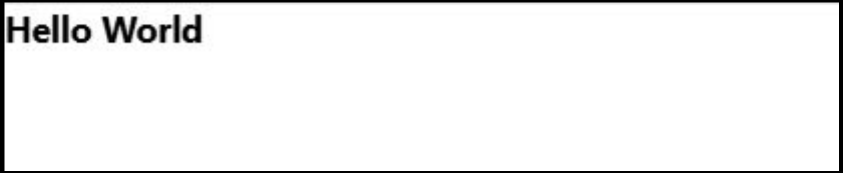

Displays, styles, and nests strings of text and even handles touch events

```
const HelloWorldApp = () => {
 return (
    <Text
     style = {page.______}>
     Hello World
    </Text>
 );
};
export default HelloWorldApp;
```

page.header



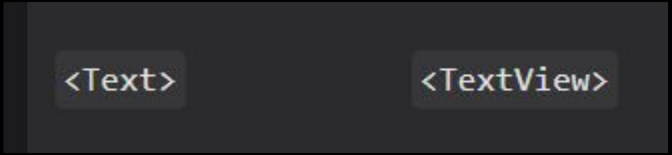
page.text

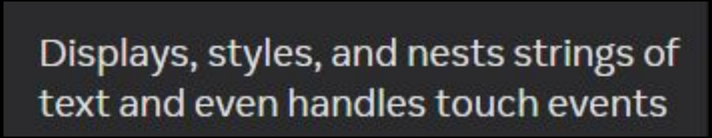


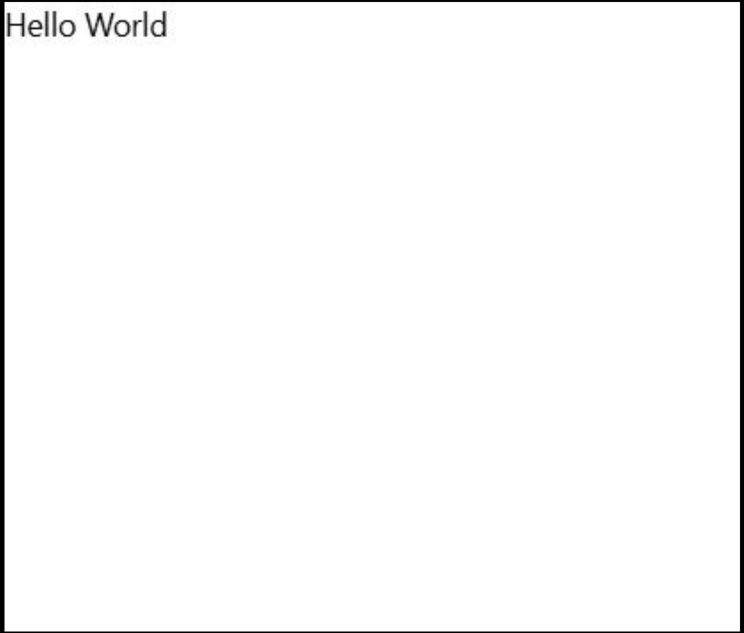
React Native – <Text> + <StyleSheet>

Want multiple <Text>s in a row? Easy!

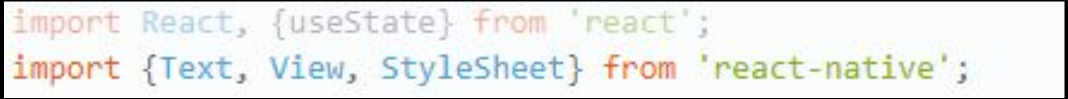
```
const HelloWorldApp = () => {
  return (
      <Text>Hello World</Text>,       WRONG
      <Text>Hello World</Text> ← only one shown
  );
};
export default HelloWorldApp;
```


<Text> <TextView>


Displays, styles, and nests strings of text and even handles touch events


Hello World

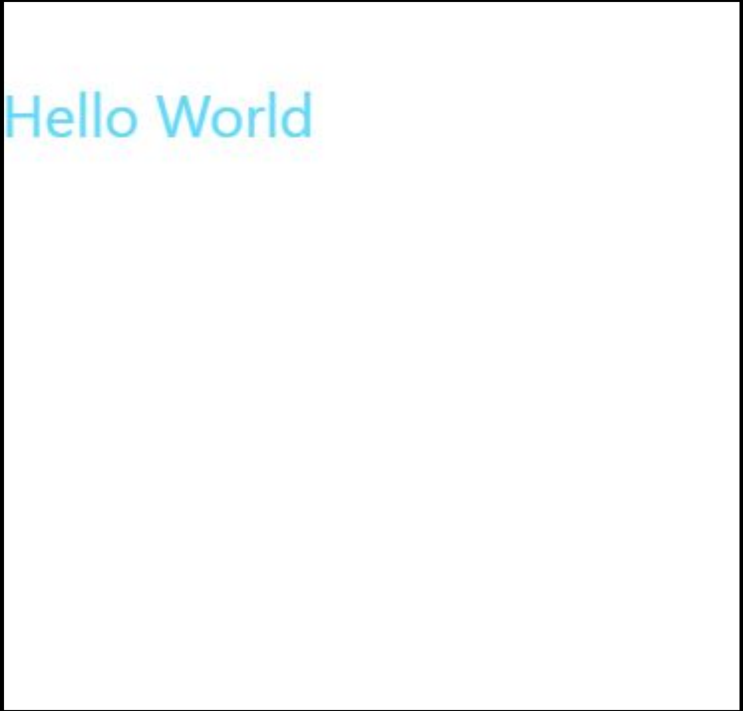
React Native – <View>



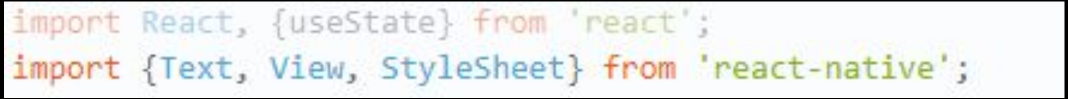
```
import React, {useState} from 'react';
import {Text, View, StyleSheet} from 'react-native';
```

Want multiple <Text>s? Use a <View>!

```
return (
  <View>
    <Text
      style = {page.header}>
      Hello World
    </Text>
  </View>
);
```



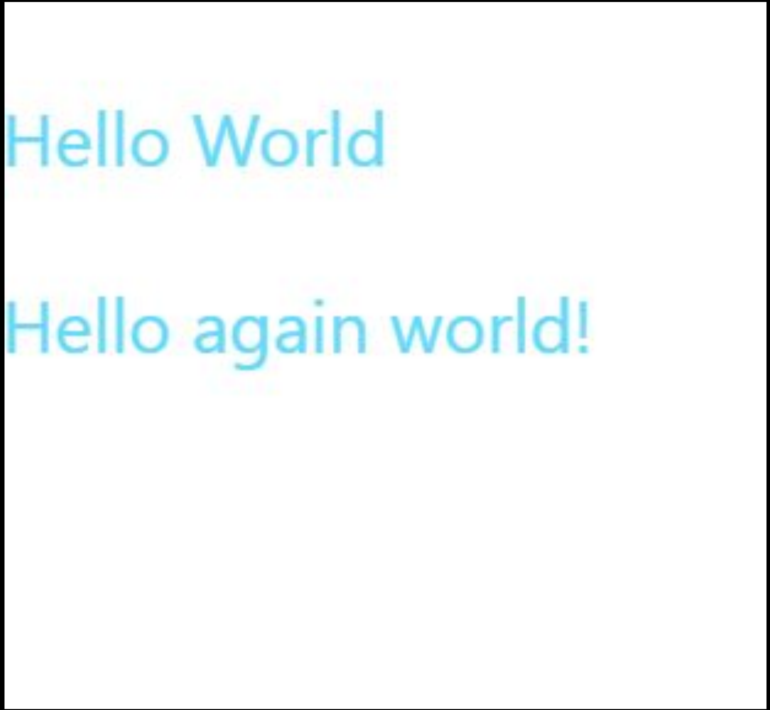
React Native – ⟨View⟩



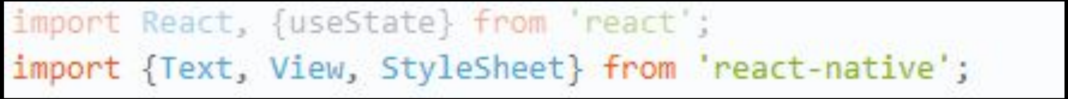
```
import React, {useState} from 'react';
import {Text, View, StyleSheet} from 'react-native';
```

Want multiple <Text>s? Use a <View>!

```
<View>
  <Text
    style = {page.header}>
    Hello World
  </Text>
  <Text
    style = {page.header}>
    Hello again world!
  </Text>
</View>
```



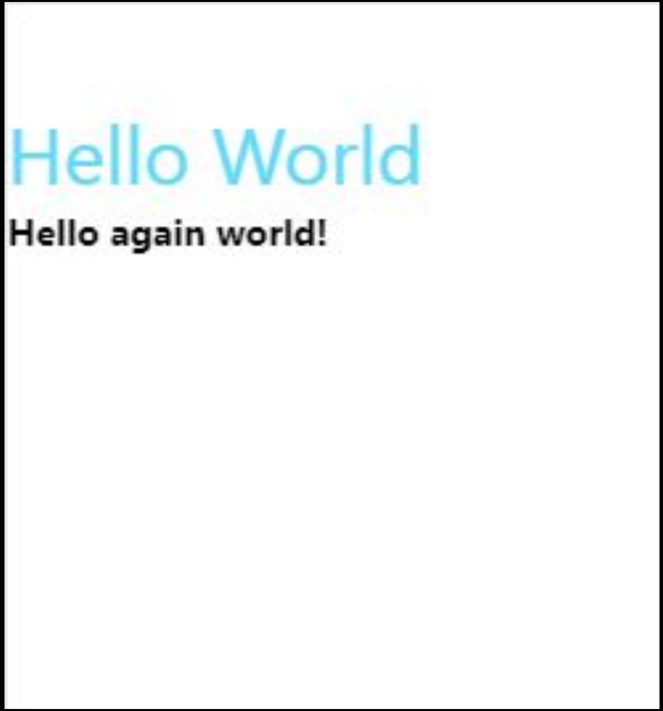
React Native – ⟨View⟩



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

Different <Text>s can have different styles.

```
<View>
  <Text
    style = {page.header}>
    Hello World
  </Text>
  <Text
    style = {page.text}>
    Hello again world!
  </Text>
</View>
```



Hello World

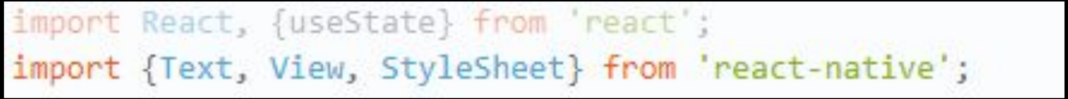
Hello again world!

React Native – 〈StyleSheet〉, revisited

```
const page = StyleSheet.create({
  …
  center: {
    alignItems: 'center',
    backgroundColor: '#eaeaea',
    flex: .7,
  },
});
```

- Keep text + header styles
- Add a style for a <View>
- style = {page.center}

React Native – 〈View〉



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

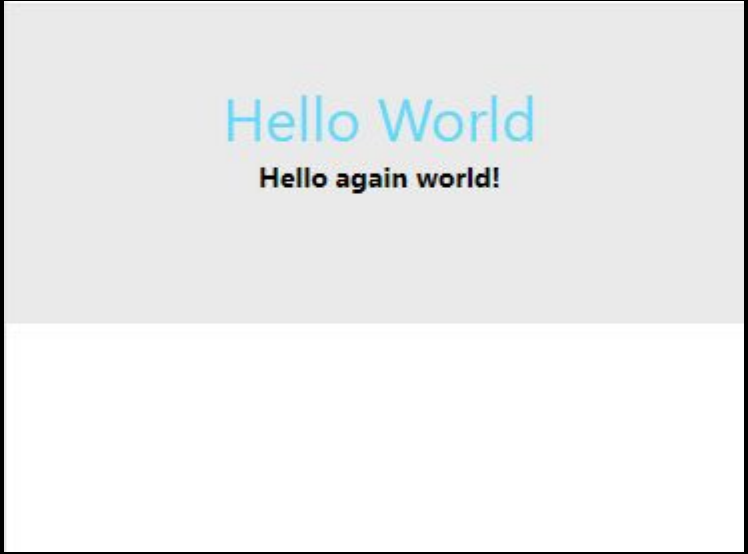
Keeping <Text> from previous example, add a style to the <View>.

<View

  style={page.center}>

  (Texts go here)

</View>



Hello World

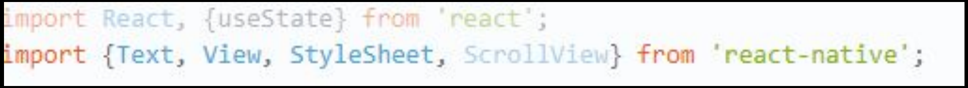
Hello again world!

React Native – 〈StyleSheet〉, revisited again

```
const page = StyleSheet.create({
 …
 center: {
   alignItems: 'center',
   backgroundColor: '#eaeaea',
   flex: 1,     // CHANGED TO 1
   paddingBottom: 20, // ADDED
 },
 scroll: {
   backgroundColor: 'white',
   flex: 1,
 },
```

- Add a <ScrollView> style
- Modify the <View> style
- style = {page.scroll}

React Native – <ScrollView>



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

What if we want multiple <View>s? <ScrollView>!

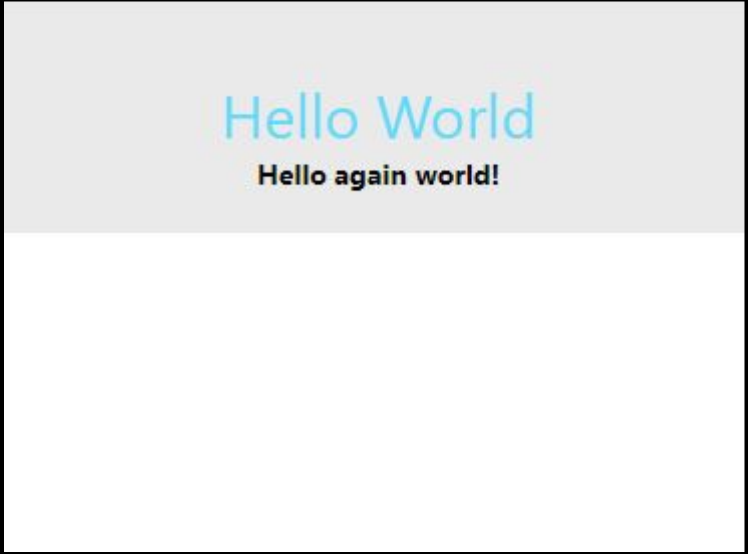
(Among others).

<ScrollView

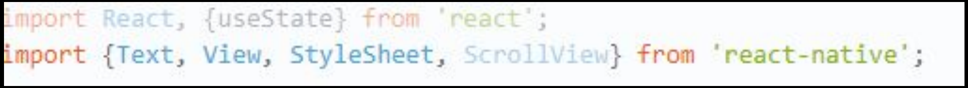
  style={page.scroll}>

  (Views go here)

</ScrollView>



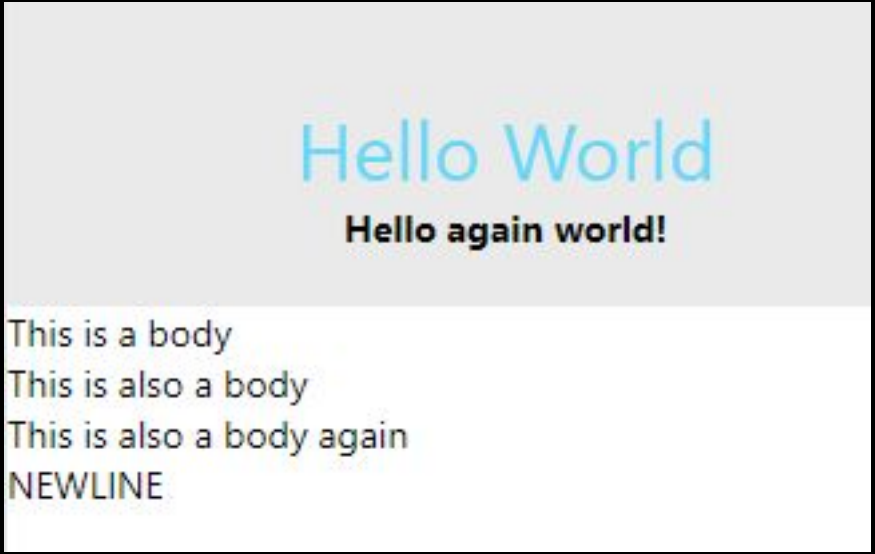
React Native – 〈ScrollView〉



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

Let's add some more content!

```
<ScrollView
  style={page.scroll}>
  (Views go here)
  <View>
      <Text>This is a body</Text>
      <Text>This is also a body</Text>
      <Text>This is also a body
      again{'\n'}NEWLINE</Text>
    </View>
</ScrollView>
```

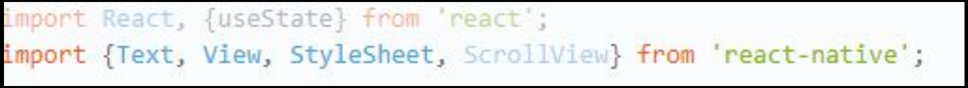


Hello World

Hello again world!

This is a body
This is also a body
This is also a body again
NEWLINE

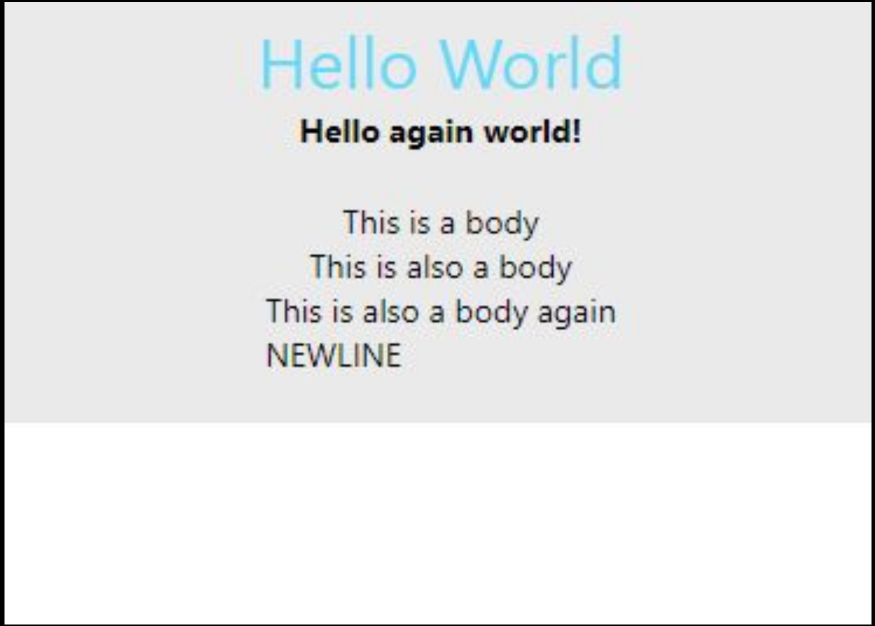
React Native – [〈ScrollView〉]



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

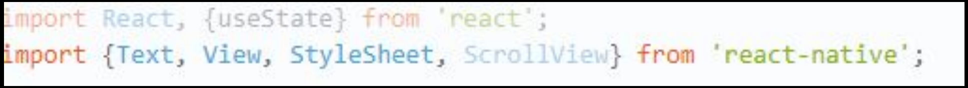
Let's add some more content!

```
<ScrollView
  style={page.scroll}> …
  <View style={page.center}>
        <Text>This is a body</Text>
        <Text>This is also a body</Text>
        <Text>This is also a body
        again{'\n'}NEWLINE</Text>
      </View>
</ScrollView>
```



```
Hello World
Hello again world!

This is a body
This is also a body
This is also a body again
NEWLINE
```

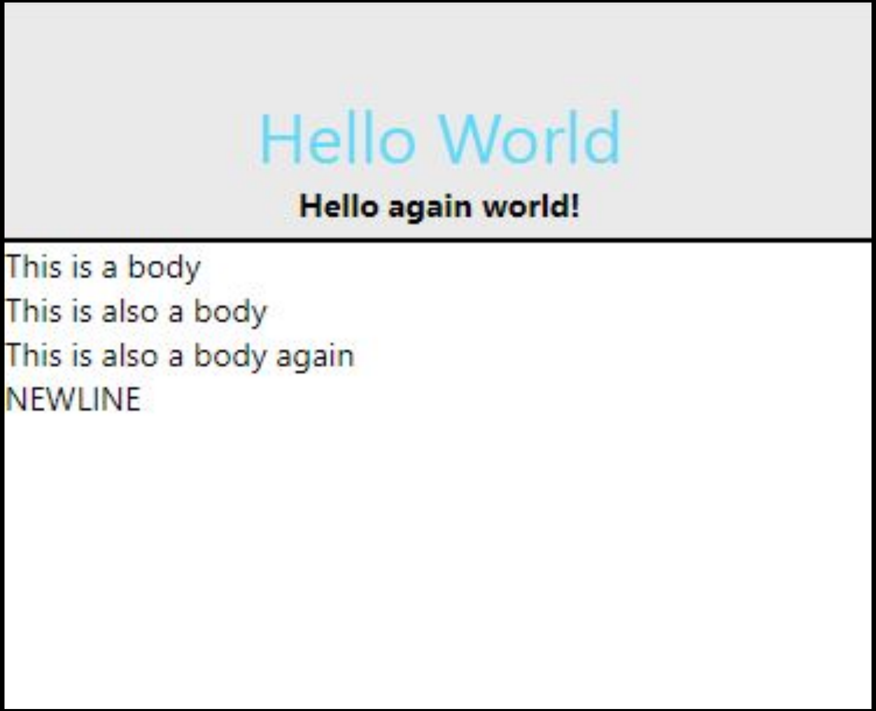
React Native – <ScrollView>



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

Cheesy header border? Also custom style in declaration.

```
<ScrollView
  style={page.scroll}> …
  <View style={{
        backgroundColor: 'black',
        height: 2,
      }}></View>
  …
</ScrollView>
```

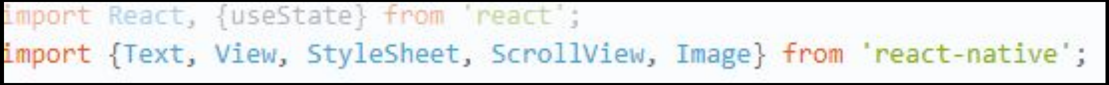


Hello World

Hello again world!

This is a body
This is also a body
This is also a body again
NEWLINE

React Native – ⟨Image⟩

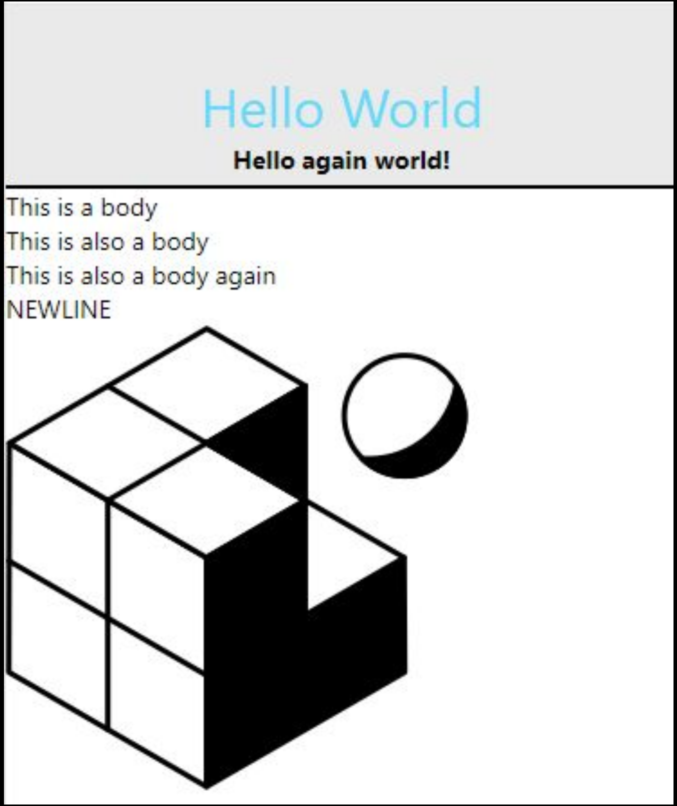


```
import React, {useState} from 'react';
import {Text, View, StyleSheet, ScrollView, Image} from 'react-native';
```

Let’s add an image to the mix.

```
<ScrollView
  style={page.scroll}>
  …
  <Image source={require('./assets/snack-icon.png')} />
</ScrollView>
```

NOTE (if drawable): `source={{uri: 'app_icon'}}`



Hello World

Hello again world!

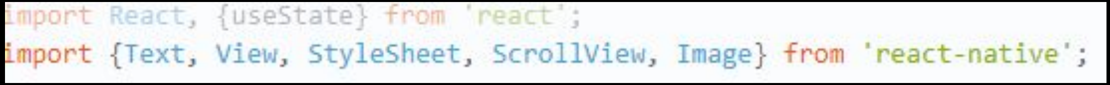
This is a body
This is also a body
This is also a body again
NEWLINE

React Native – 〈StyleSheet〉, revisited again again

```
const page = StyleSheet.create({
  …
  imageSize:
    width: 100,
    height: 100,
    opacity: .5,
  },
});
```

- Add a <Image> style
- Resizes the image, and transparency!
- style = {page.scroll}

React Native – ⟨Image⟩

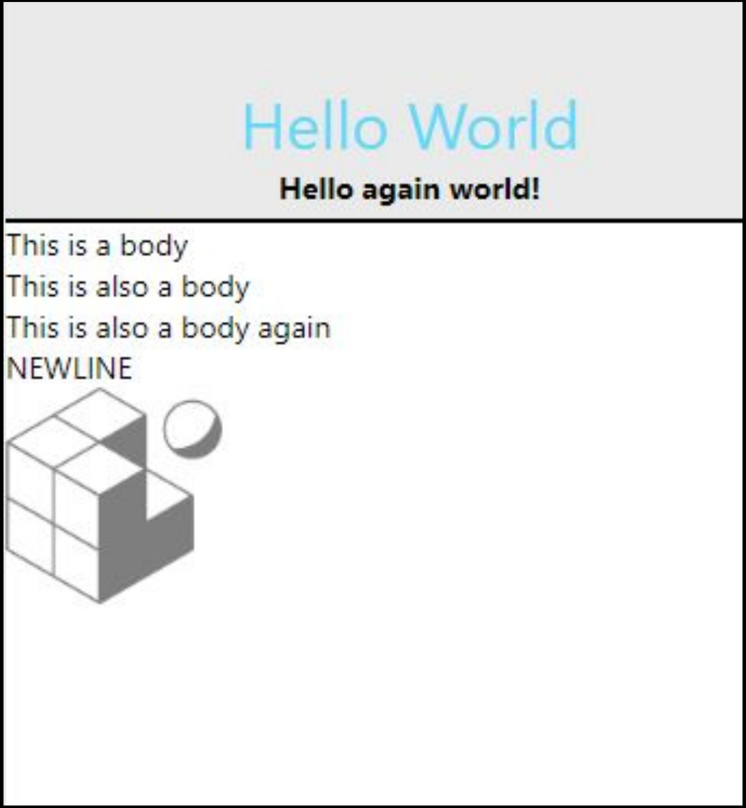


```
import React, {useState} from 'react';
import {Text, View, StyleSheet, ScrollView, Image} from 'react-native';
```

Styling the image.

```
<ScrollView
  style={page.scroll}>
  …
  <Image source={require('./assets/snack-icon.png')}
    style={page.imageSize}/>
</ScrollView>
```

NOTE (if drawable): `source={{uri: 'app_icon'}}`



Hello World

Hello again world!

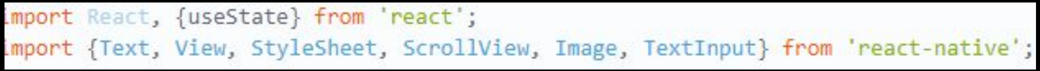
This is a body
This is also a body
This is also a body again
NEWLINE

React Native – States

```
const HelloWorldApp = () => {
  const [text, setText] = useState(“”);


FUNCTION:
{newText => setText(newText)}


Subtitle text (replace “Hello world again”):
<Text
  style = {page.text}>
  {text}
</Text>
```

- text is the state (variable)
- useState sets default value to “”
- setText is a setter for text
- Function takes a string into newText, sets setText to newText

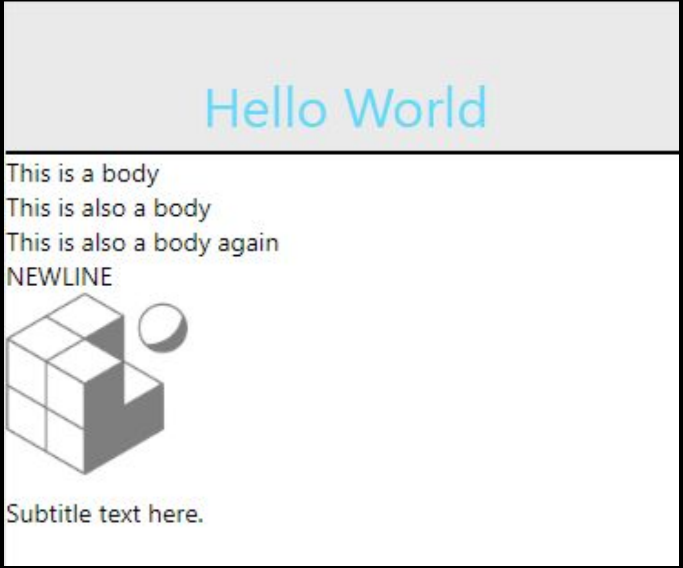
React Native – 〈TextInput〉



```
mport React, {useState} from 'react';
mport {Text, View, StyleSheet, ScrollView, Image, TextInput} from 'react-native';
```

Let's add some text input and use it!

```
<ScrollView
  …
  <TextInput
        style={{height: 40}}
        placeholder="Subtitle text here."
        onChangeText={newText => setText(newText)}
        defaultValue={text}
      />
</ScrollView>
```

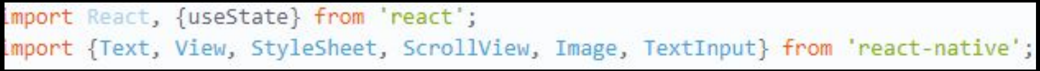


Hello World

This is a body
This is also a body
This is also a body again
NEWLINE

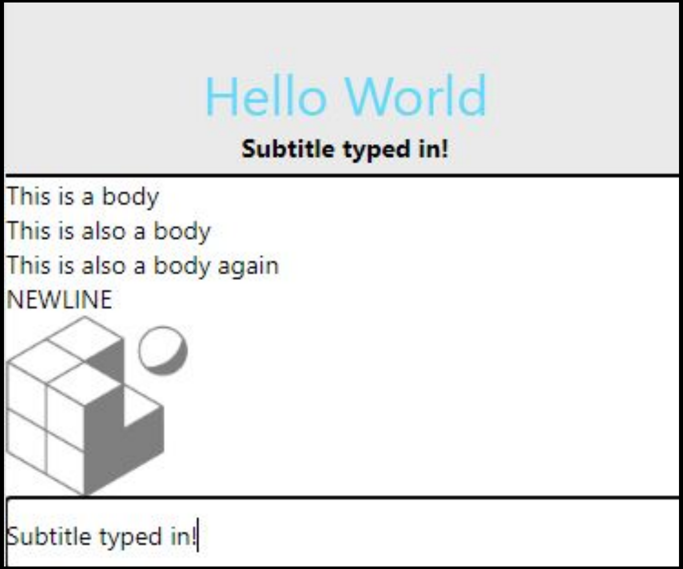
Subtitle text here.

React Native – <TextInput>



```
mport React, {useState} from 'react';
mport {Text, View, StyleSheet, ScrollView, Image, TextInput} from 'react-native';
```

It automatically updates the subtitle with the text you type in the <TextInput>. How cool!



Hello World

Subtitle typed in!

This is a body
This is also a body
This is also a body again
NEWLINE

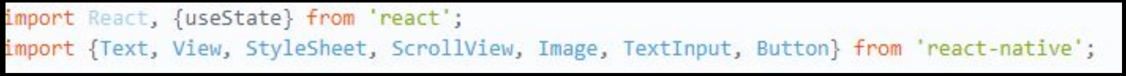
Subtitle typed in!

React Native – ⟨StyleSheet⟩, revisited one last time

```
const page = StyleSheet.create({
  …
  imageBig: {
    width: 300,
    height: 300,
    opacity: 1,
  },
});
```

- Add another <Image> style
- Bigger, not transparent
- style = {page.imageBig}

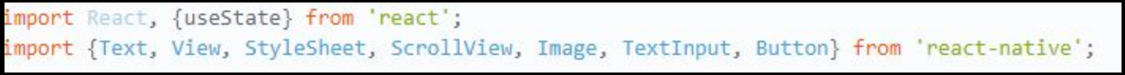
React Native – 〈Button〉



```
import React, {useState} from 'react';
import {Text, View, StyleSheet, ScrollView, Image, TextInput, Button} from 'react-native';
```

Basic button. If you want more customization, use a <Pressable>. We'll need some setup for this.

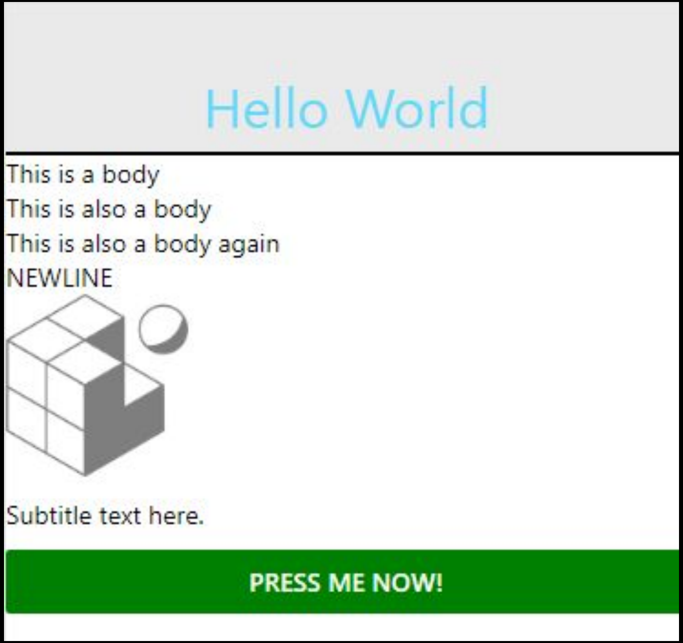
```
const HelloWorldApp = () => {
  …
  const [clickToggle, setClickToggle] = useState(false);
  …
  <Image source={require('./assets/snack-icon.png')}
        style=
{ clickToggle ? page.imageSize : page.imageBig }/>
TERNARY OPERATOR ↑
```


```
import React, {useState} from 'react';
import {Text, View, StyleSheet, ScrollView, Image, TextInput, Button} from 'react-native';
```

React Native – 〈Button〉

Basic button. If you want more customization, use a <Pressable>. We'll need some setup for this.

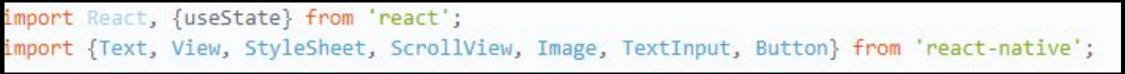
```
<ScrollView…
  <Button
      color='green'
      title='Press me now!'
      onPress={() => setClickToggle(!clickToggle)}
      />
```


Hello World

This is a body
This is also a body
This is also a body again
NEWLINE

Subtitle text here.

PRESS ME NOW!

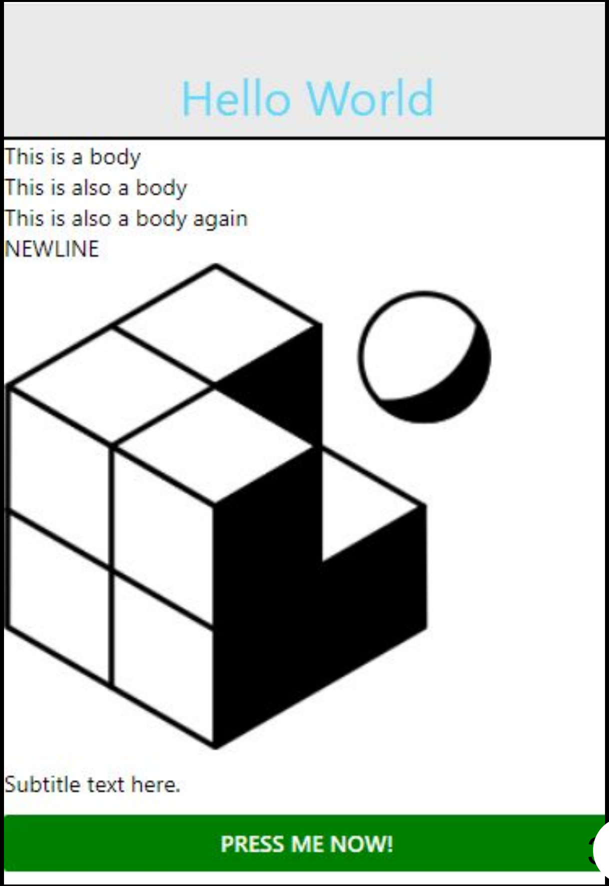


```
import React, {useState} from 'react';
import {Text, View, StyleSheet, ScrollView, Image, TextInput, Button} from 'react-native';
```

React Native – <Button>

Basic button. If you want more customization, use a <Pressable>. We'll need some setup for this.

```
<ScrollView…
  <Button
      color='green'
      title='Press me now!'
      onPress={() => setClickToggle(!clickToggle)}
      />
```



Hello World

This is a body
This is also a body
This is also a body again
NEWLINE

Subtitle text here.

React Native – The end of the basic demo!

Thank you!

Expo

And how to use it

Now, for Android Studio!
(+ Expo)
(In Windows)

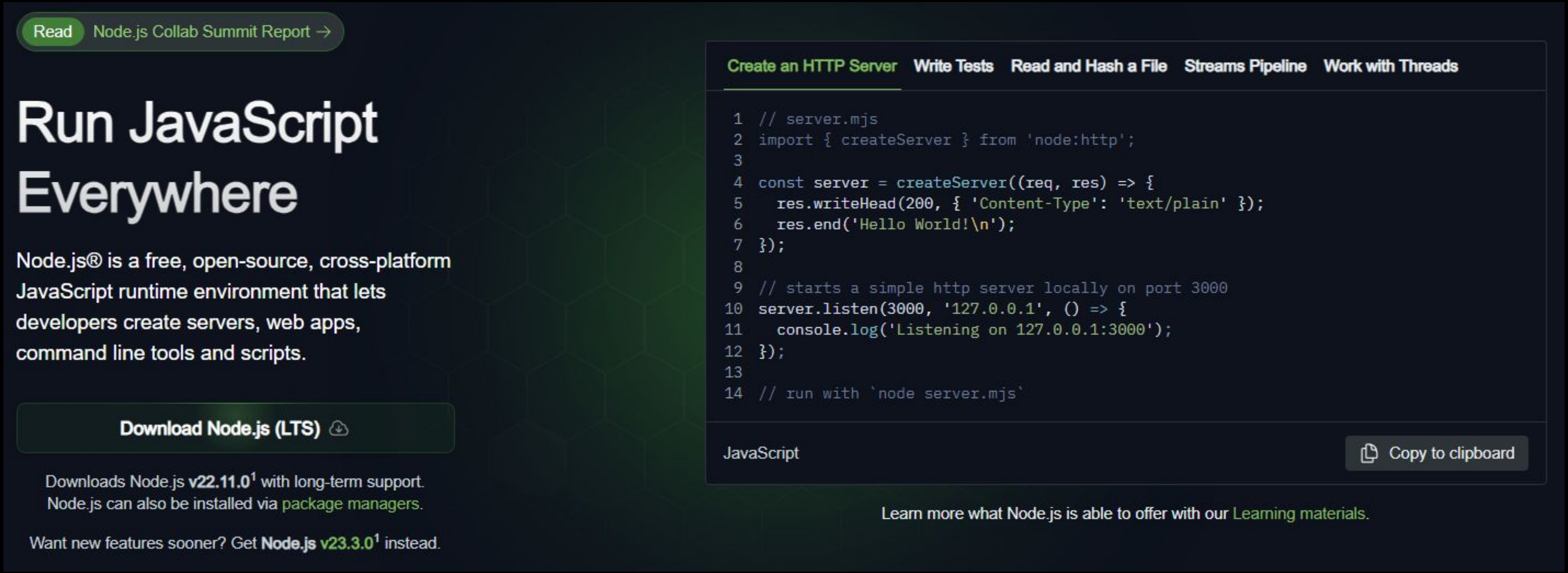
Download NodeJS (nodejs.org)
(Including Chocolatey)

Set up your environment
(See later slides for variables)

Create a project
“npx create-expo-app@latest”

Start your expo server
“npx expo start --android”

Node.js (Windows) – Download and run installer



Run JavaScript Everywhere

Node.js® is a free, open-source, cross-platform JavaScript runtime environment that lets developers create servers, web apps, command line tools and scripts.

Downloads Node.js **v22.11.0**[1] with long-term support.
Node.js can also be installed via package managers.

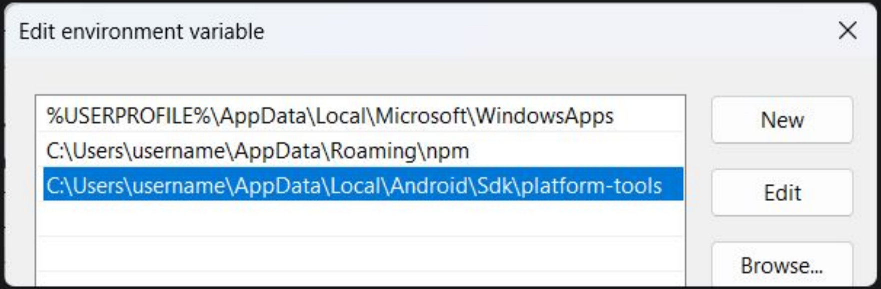
Want new features sooner? Get **Node.js v23.3.0**[1] instead.

```javascript
// server.mjs
import { createServer } from 'node:http';

const server = createServer((req, res) => {
  res.writeHead(200, { 'Content-Type': 'text/plain' });
  res.end('Hello World!\n');
});

// starts a simple http server locally on port 3000
server.listen(3000, '127.0.0.1', () => {
  console.log('Listening on 127.0.0.1:3000');
});

// run with `node server.mjs`
```

Learn more what Node.js is able to offer with our Learning materials.

Environment Variables (Windows)

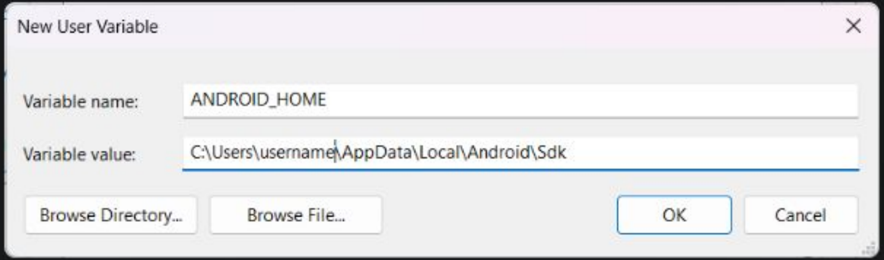
8 To add platform-tools to the Path, go to **Windows Control Panel** > **User Accounts** > **User Accounts** (again) > **Change my environment variables** > **Path** > **Edit** > **New** and add the path to the platform-tools to the list as shown below:



Edit environment variable

```
%USERPROFILE%\AppData\Local\Microsoft\WindowsApps
C:\Users\username\AppData\Roaming\npm
C:\Users\username\AppData\Local\Android\Sdk\platform-tools
```

6 After the tools installation is complete, configure the `ANDROID_HOME` environment variable. Go to **Windows Control Panel** > **User Accounts** > **User Accounts** (again) > **Change my environment variables** and click **New** to create a new `ANDROID_HOME` user variable. The value of this variable will point to the path to your Android SDK:



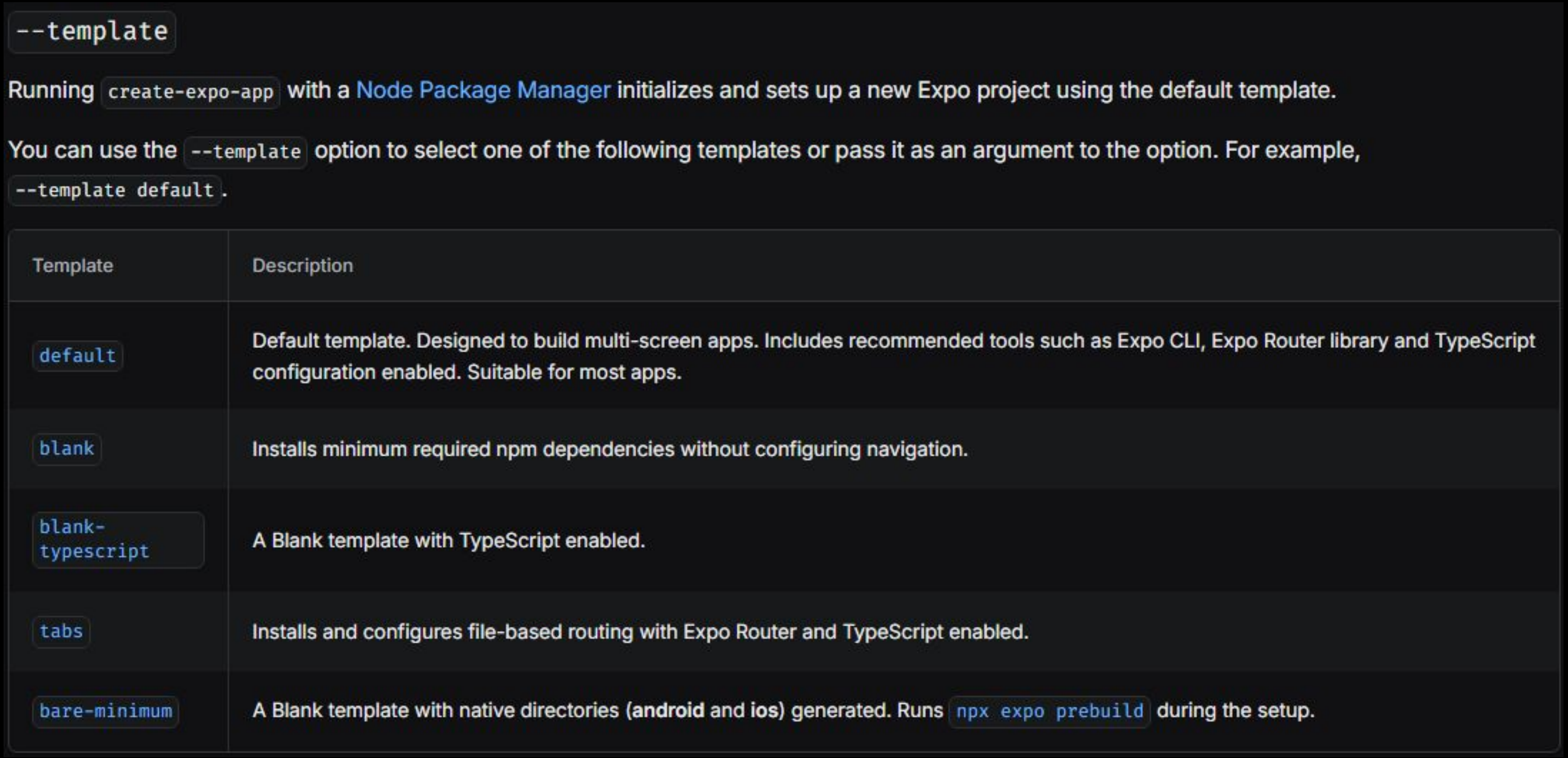
New User Variable

Variable name: ANDROID_HOME

Variable value: C:\Users\username\AppData\Local\Android\Sdk

Project Creation (Git Bash) – Project Creation Info

Command: npx create-expo-app --template default   OR npx create-expo-app@latest



`--template`

Running `create-expo-app` with a Node Package Manager initializes and sets up a new Expo project using the default template.

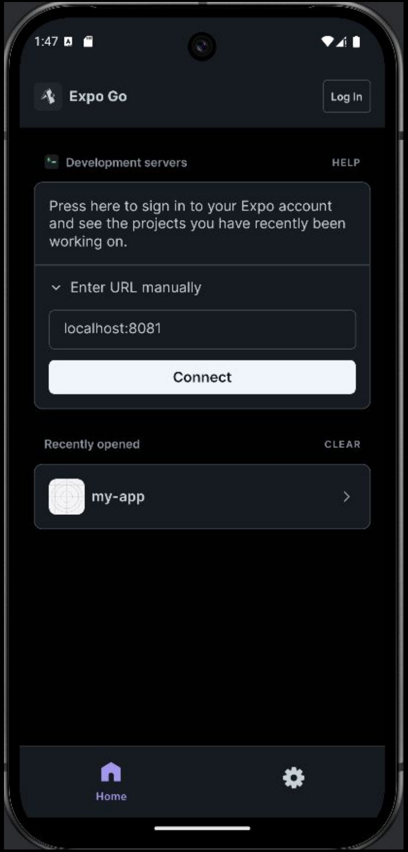
You can use the `--template` option to select one of the following templates or pass it as an argument to the option. For example, `--template default`.

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.

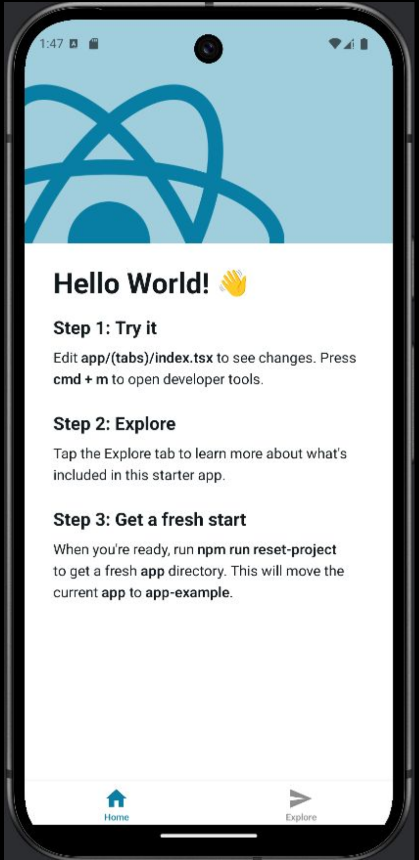
Start Development (Expo) – Dev Server Info

Command: npx expo start --android ← This is for an emulator (Android Studio)

- Starts an Expo server in CLI
- Installs/launches Expo Go app
- Connects automatically
  - Reconnect @ given IP on launch, defaults to: localhost:8081
- Launches app inside of Expo Go



Press here to sign in to your Expo account and see the projects you have recently been working on.



Hello World! 👋

Step 1: Try it

Edit app/(tabs)/index.tsx to see changes. Press cmd + m 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.

Explore (Expo + Android Studio) – Expo Docs

Explore the “latest” template.

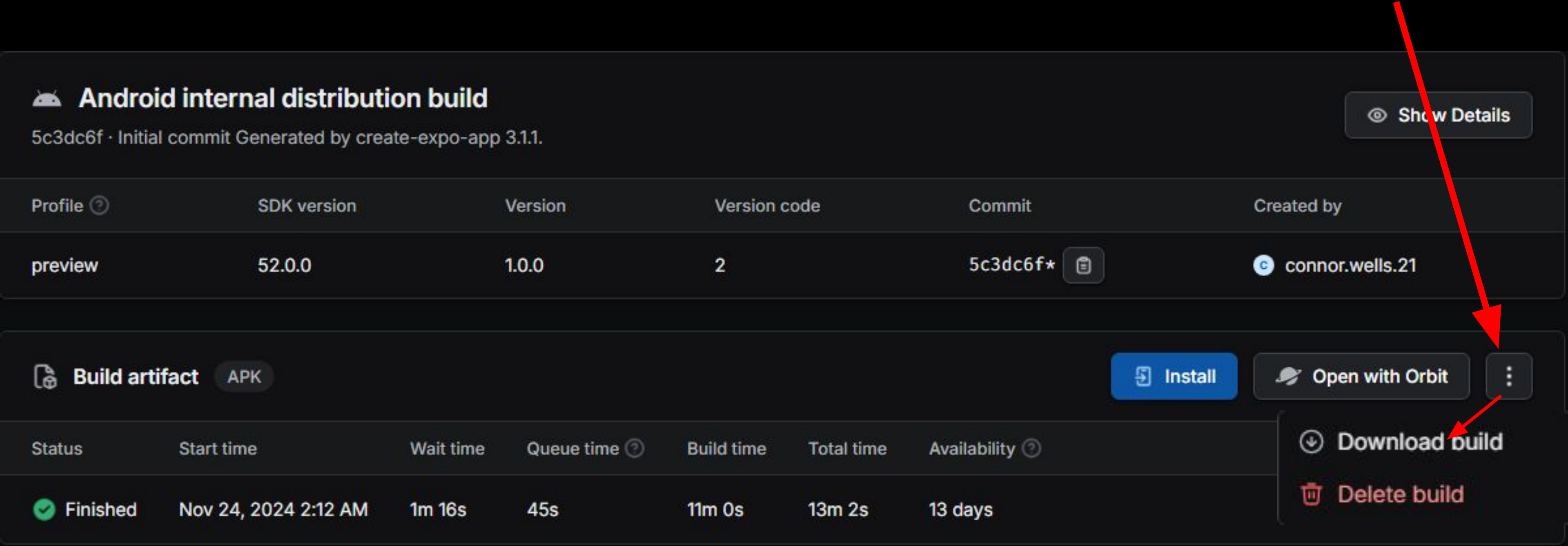
Once you’re done, run:

“npm run reset–project”

For a fresh, base project to build off of.

Building – USE POWERSHELL. GIT BASH DOESN'T WORK

1) npm install -g eas-cli (Installs/updates eas, which is the build tool)
2) (Sign in to Expo on your browser)
3) eas login -s (Uses SSO login through browser)
4) eas build:configure (Setup)
5) eas build -p android --profile preview (Build APK)
6) Once build is done, you can run the APK on your emulator, or download it from Expo.dev



Android internal distribution build

5c3dc6f · Initial commit Generated by create-expo-app 3.1.1.

Profile	SDK version	Version	Version code	Commit	Created by
preview	52.0.0	1.0.0	2	5c3dc6f*	connor.wells.21

Build artifact APK

Status	Start time	Wait time	Queue time	Build time	Total time	Availability
Finished	Nov 24, 2024 2:12 AM	1m 16s	45s	11m 0s	13m 2s	13 days

Future Readings/More Work

If you want to actually get some practice building a more complex app, expo has a fantastic tutorial, broken up into 9 chapters, available on their website:
https://docs.expo.dev/tutorial/introduction/

Also helpful might be:

A basic React Native tutorial: https://reactnative.dev/docs/tutorial

React Native Introduction/Documentation: https://reactnative.dev/docs/getting-started

Expo Documentation: https://docs.expo.dev

Sources

React Information: https://legacy.reactjs.org

React Native Information: https://reactnative.dev

React Native Introduction/Documentation: https://reactnative.dev/docs/getting-started

Expo Documentation: https://docs.expo.dev