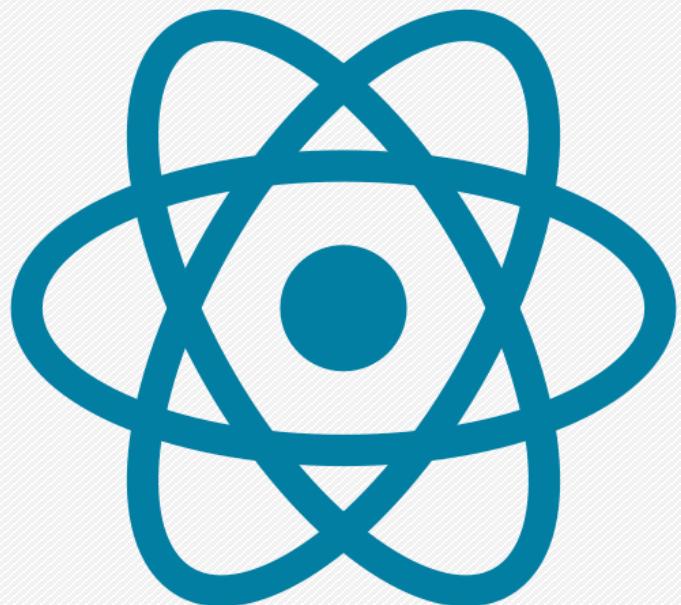


REACT NATIVE

FROM AN ANDROID DEV



Michael Fazio



Speaker notes

- ▶ I hope you've had a great week at THAT!

SPECIAL THANKS TO ALL OUR AWESOME CAMP SPONSORS!

Unspecified



DATASTAX



Temporal



PROPELAUTH



PIECES
FOR DEVELOPERS



redgate



MESCIUS
FORMERLY GRAPECITY, INC.

AWS Amplify

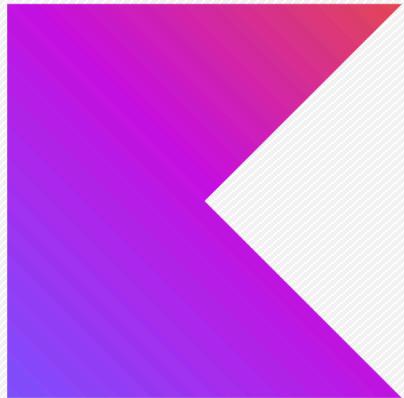
Speaker notes

- ▶ Thank you to all the sponsors for making THAT possible
- ▶ Food's expensive here. Like, *really* expensive.



Speaker notes

- ▶ It's me!
- ▶ Hi, I'm Michael.



Kotlin

Speaker notes

- ▶ I'm a fan of Kotlin
- ▶ It's from JetBrains

Speaker notes

- ▶ I'm also a fan of Android

Kotlin and Android Development

featuring Jetpack

Build Better, Safer Android Apps



Speaker notes

- ▶ I wrote a book
- ▶ "Kotlin and Android Development featuring Jetpack"
- ▶ Pragmatic Bookshelf, from the guys that brought you "The Pragmatic Programmer"



Speaker notes

- ▶ We're talking about React Native today
- ▶ I've been working with it lately



ReactNative



Xamarin



Android



Speaker notes

- ▶ Cross-platform tools are getting more and more popular
 - ▶ Flutter, React Native, Maui (Xamarin)
- ▶ It makes sense
 - ▶ Two platforms, one code base and language (in theory)
- ▶ Write once, deploy everywhere
- ▶ Again, I'm doing React Native now, and I've done Xamarin in the past



Speaker notes

- ▶ Old tools were "Write once, suck everywhere"
- ▶ They weren't very good
 - ▶ Cordova, PhoneGap, etc



Speaker notes

- ▶ They're pretty good now!
- ▶ Specifically React Native, Flutter, Xamarin/Maui



Speaker notes

- ▶ We're talking about React Native today
- ▶ I've been working with it lately

- Items
- Start w/
Hi, I'm Michael al I like Kotlin
I also like Android
I wrote this (Book)
But now I'm doing this (RN)
- Quick slides
- People use cross platform
 - And I think they're pretty cool, actually
 - Orange Cassidy thinks us
 - Overall
- Items to Note
- Intro
 - RN - It's React, but not everything works w/ RN
 - Pros - Hot reload
 - Animations
 - Cons - Not same as CSS animations
 - Styling in general
 - We're using v3 NativeScript
 - Rendering is slower than web
 - React Navigation
 - Stack nav
 - Edit Native in Android Studio/XCode
 - Monorepos
 - Probably avoid
 - Why would we want to start
 - Starting
- Items of Note, cont.
- React Programming paradigm
 - Not native integration
 - Use Expo for other pieces
 - Launched in 2015 (not new)
 - Still not at v1.0
 - RN apps
 - Facebook, FB Messenger, Meta Quest, Office
 - PS App, Discord
 - Walmart, Pinterest
 - Expo app vs. Snack player vs. stand alone
 - Does show platform-specific explanations
 - Native components vs. Core components
 - Platform intent
- Items to Note, cont.
- Moto devices kinda suck
 - Hermes vs. JSI
 - JavaScript Core
- Don't try to cover everything

- Don't try to teach React
 - Styling is similar to React
 - Touchables
 - No default styling
 - Can build your own
 - Use a library instead?
- Core Components
- ScrollView
 - Button
 - Modal
 - FlatList
 - Like
 - Section
 - View, - 3rd party libraries can change every fast ref
 - Upgrade guide
- Metro
- Redux + Redux Toolkit
 - Legacy vs. New Architecture
 - Native Modules → Turbo
 - In progress → ... since still

Speaker notes

- ▶ There's a lot to cover
- ▶ Point out the "Don't try to cover everything" note
- ▶ I'll try to avoid jamming too much in here
- ▶ Note that these are my personal thoughts and not reflective of anyone/anything else
 - ▶ That includes my employer
- ▶ We can dig into things after the talk if you want



Speaker notes

- ▶ From Meta
- ▶ One of the few good things (I also include Oculus in here)
 - ▶ Yes, they bought Oculus, so that only kind of counts



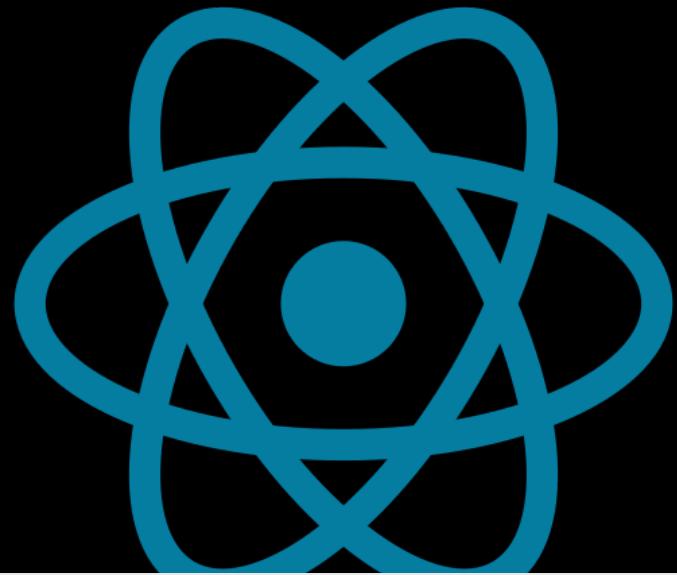
Speaker notes

- ▶ Not new, started in 2015
- ▶ A simpler time



Speaker notes

- ▶ Tons of apps are RN
 - ▶ Meta apps
 - ▶ Facebook/FB Messenger/Meta Quest
 - ▶ Amazon Shopping/Alexa/Kindle
 - ▶ Microsoft Office
 - ▶ Sorry, Teams as well. Don't hold it against RN.
 - ▶ PS App/Discord
 - ▶ Walmart/Pinterest
 - ▶ Tesla



Speaker notes

- ▶ React programming paradigm
- ▶ Mainly for UI
 - ▶ Need something else for most native integrations
- ▶ I'm not going to teach React at all here
 - ▶ Many people do that better than I



Speaker notes

- ▶ It's React so you can use stuff like Redux and Redux Toolkit easily



Speaker notes

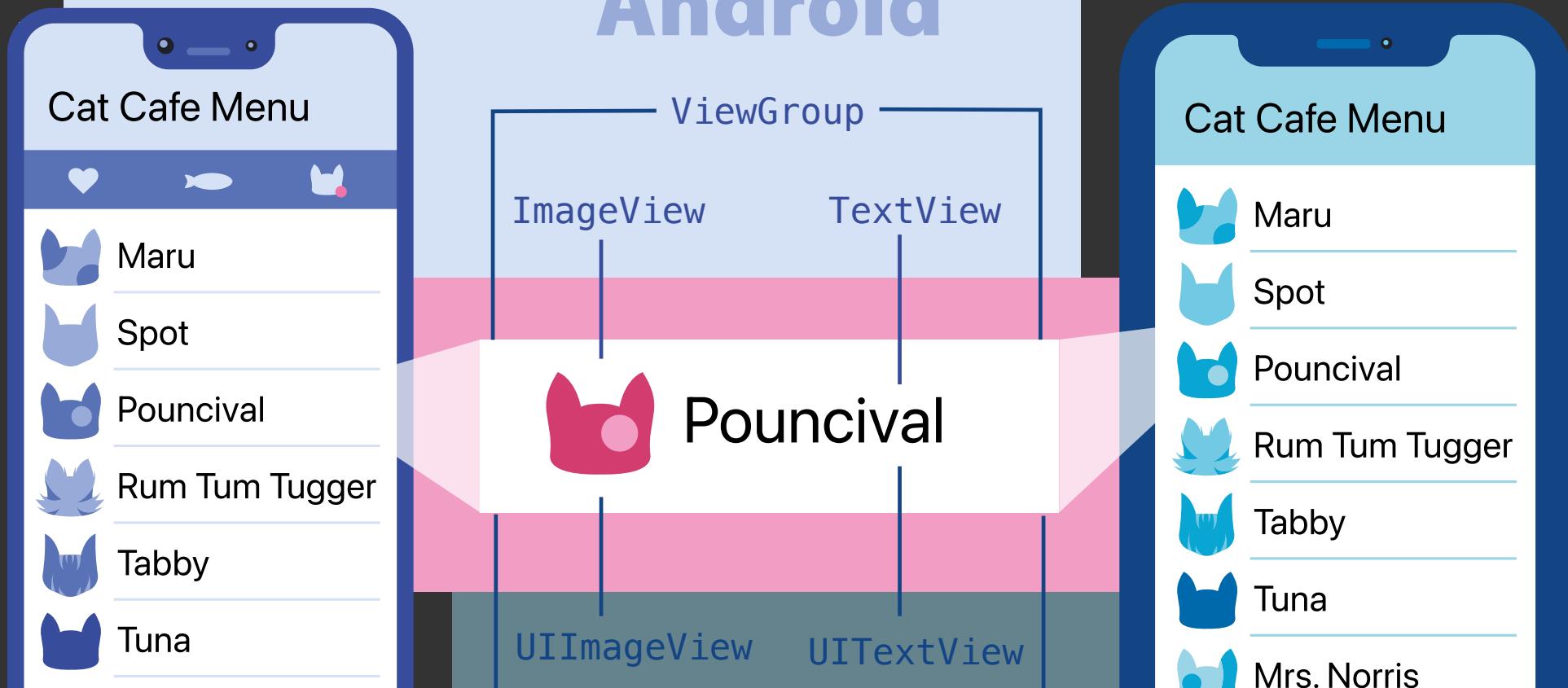
- ▶ You use JavaScript to write your code



Speaker notes

- ▶ Or even better, you can use TypeScript
- ▶ Enabled by default in new projects
- ▶ First time I didn't feel like I was fighting against TS
- ▶ JS or TS, you'll use your normal editor for dev
 - ▶ VS Code, WebStorm, etc.

Android



Speaker notes

- ▶ React Native components that are using the native (Android/iOS) views
- ▶ An image plus text becomes:
 - ▶ Android: ImageView and TextView inside a ViewGroup
 - ▶ iOS: UIImageView and UITextView inside a UIView

REACT NATIVE → ANDROID → IOS

- ▶ <View> → <ViewGroup> → <UIView>
- ▶ <Text> → <TextView> → <UITextView>
- ▶ <Image> → <ImageView> → <UIImageView>
- ▶ <ScrollView> → <ScrollView> → <UIScrollView>
- ▶ <TextInput> → <EditText> → <UITextField>

Android

Speaker notes

- ▶ View, Text, Image, ScrollView, TextInput
 - ▶ All have Android/iOS views under the hood
- ▶ FlatList/SectionList
 - ▶ Like RecyclerView
 - ▶ SectionList makes groupings easier
- ▶ Button
 - ▶ Not much here
 - ▶ “A basic button component that should render nicely on any platform. Supports a minimal level of customization.”

```
function Section(  
  {children, title}: SectionProps  
): React.JSX.Element {  
  const isDarkMode = useColorScheme() === 'dark';  
  const titleColor =  
    isDarkMode ? Colors.white : Colors.black;  
  return (  
    <View style={styles.sectionContainer}>  
      <Text  
        style={[  
          styles.sectionTitle,  
          { color: titleColor },  
        ]}>  
        {title}  
      </Text>  
    </View>  
  );  
}
```

Speaker notes

- ▶ One block of code, both platforms
- ▶ It's standard React in most ways
- ▶ Create reusable components to have in your app

The screenshot shows a Mac desktop with Xcode open. On the left, there's a "Running Devices - android" window showing an Android emulator with a "Welcome to React Native" screen. On the right, there's a "iPhone 15 Pro - iOS 17.5" window showing an iPhone simulator with the same screen. The main Xcode interface shows the "App.tsx" file with code for a React Native application. The code includes components like SafeAreaView, StatusBar, ScrollView, and View, along with styling logic for background colors based on isDarkMode. A yellow dot marker is placed on line 81 of the code.

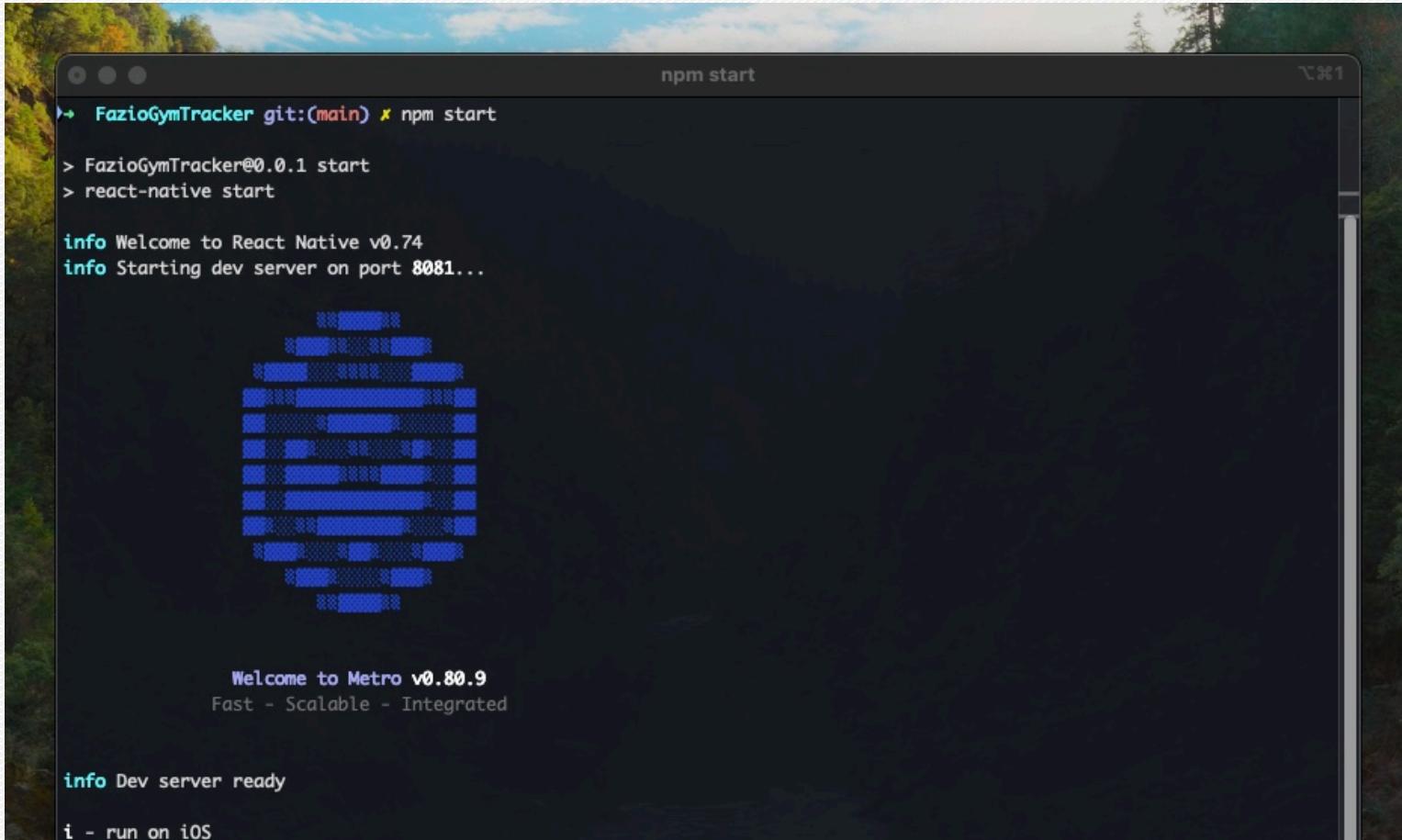
```
function App(): React.JSX.Element { Show usages
  ...
  <SafeAreaView style={backgroundStyle}>
    <StatusBar
      barStyle={isDarkMode ? 'light-content' : 'dark-content'}
      backgroundColor={backgroundStyle.backgroundColor}
    />
    <ScrollView
      contentInsetAdjustmentBehavior="automatic"
      style={backgroundStyle}>
      <Header />
      <View
        style={{
          backgroundColor: isDarkMode ? Colors.black : Colors.white,
        }}>
        <Section title="Step One">
          Edit <Text style={styles.highlight}>App.tsx</Text> to change this
          screen and then come back to see your edits.
        </Section>
        <Section title="See Your Changes">
          <ReloadInstructions />
        </Section>
        <Section title="Debug">
          <DebugInstructions />
        </Section>
        <Section title="Learn More">
          Read the docs to discover what to do next:
        </Section>
        <LearnMoreLinks />
      </View>
    </ScrollView>
  </SafeAreaView>
}

App()
```

Speaker notes

- ▶ One block of code, both platforms

METRO



```
npm start

→ FazioGymTracker git:(main) ✘ npm start

> FazioGymTracker@0.0.1 start
> react-native start

info Welcome to React Native v0.74
info Starting dev server on port 8081...



Welcome to Metro v0.80.9
Fast - Scalable - Integrated

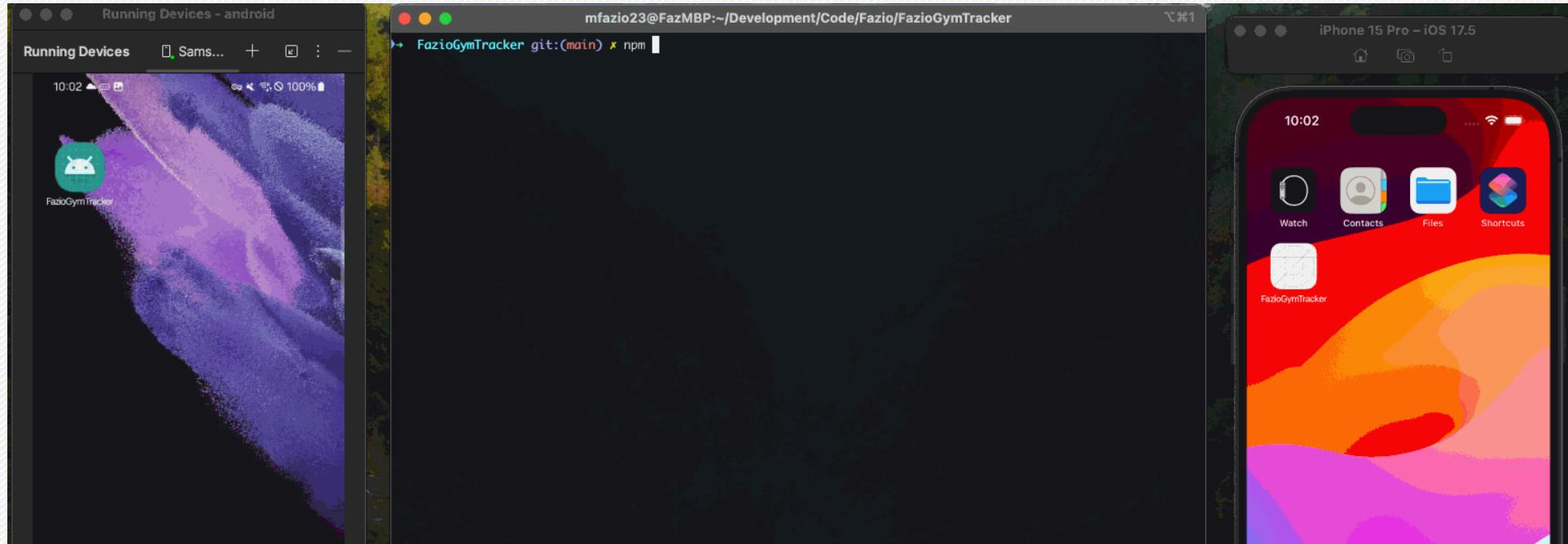
info Dev server ready

i - run on iOS
```

Speaker notes

- ▶ Metro
- ▶ Think of it as webpack for React Native
- ▶ You can start the app from Metro

METRO



Speaker notes

- ▶ Run `npm start` to fire up Metro
- ▶ Hit "a" in the window to start the Android app
 - ▶ Builds the app
 - ▶ Deploys to the device
 - ▶ Bundles the JS code
- ▶ Hit "i" in the window to start the iOS app
- ▶ Note that logs are shown in the Metro window

DEV MENU

The screenshot shows a development interface with three main sections:

- Running Devices - android**: Shows a connected Samsung Galaxy S20 device with 100% battery. The screen displays the "Welcome to React Native" logo.
- npm start**: The terminal output for the command "npm start". It shows the build process, including tasks like "app:checkDebugDuplicateClasses", "app:mergeExtDexDebug", and "app:installDebug". The output ends with "BUILD SUCCESSFUL in 7s" and a log message indicating the app is running on an iPhone 15 Pro simulator.
- iPhone 15 Pro - iOS 17.5**: Shows the "Welcome to React Native" screen running on the simulator.

Step One

Edit `App.tsx` to change this screen and then come back to see your edits.

See Your Changes

Double tap **R** on your keyboard to reload your app's code.

Debug

Press **Cmd or Ctrl + M** or **Shake** your device to open the Dev Menu.

Learn More

Read the docs to discover what to do next:

FazioGymTracker

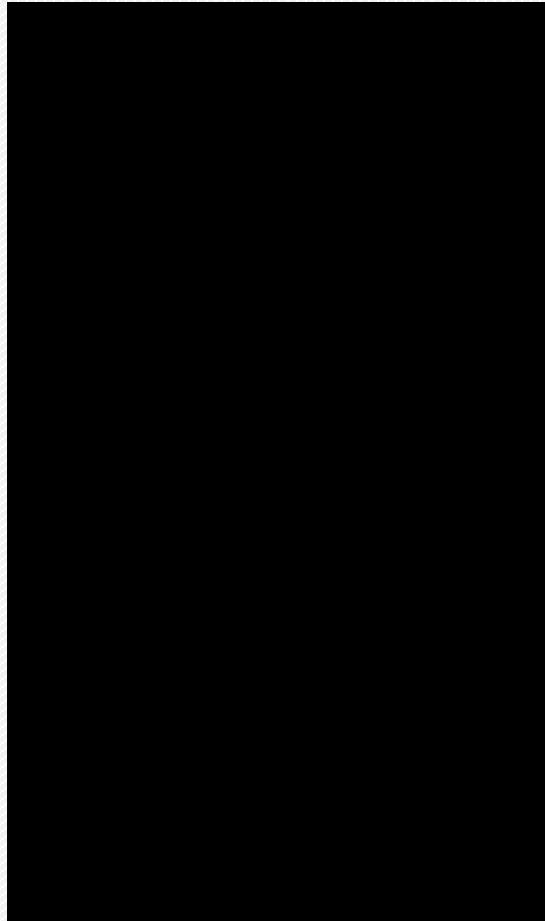
Explains a Hello World

The Basics Explains a Hello World for React Native.

Speaker notes

- ▶ Dev menu
- ▶ Hit "D" in the terminal
- ▶ Shake the device

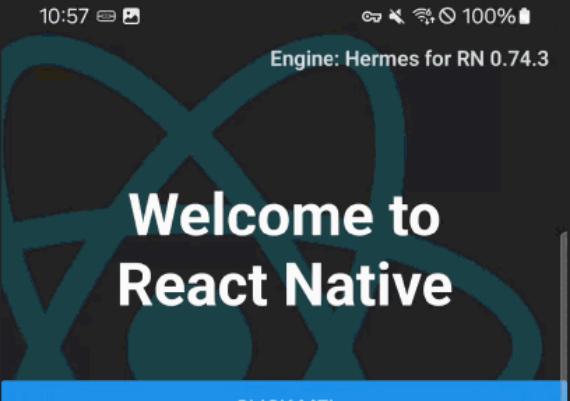
ELEMENT INSPECTOR



Speaker notes

- ▶ Dev menu
- ▶ Hit "D" in the terminal
- ▶ Shake the device

CHROME DEBUGGER



10:57 100%
Engine: Hermes for RN 0.74.3

Welcome to React Native

CLICK ME!

Step One
Edit `App.tsx` to change this screen and then come back to see your edits.

See Your Changes
Double tap **R** on your keyboard to reload your app's code.

Chrome DevTools interface:

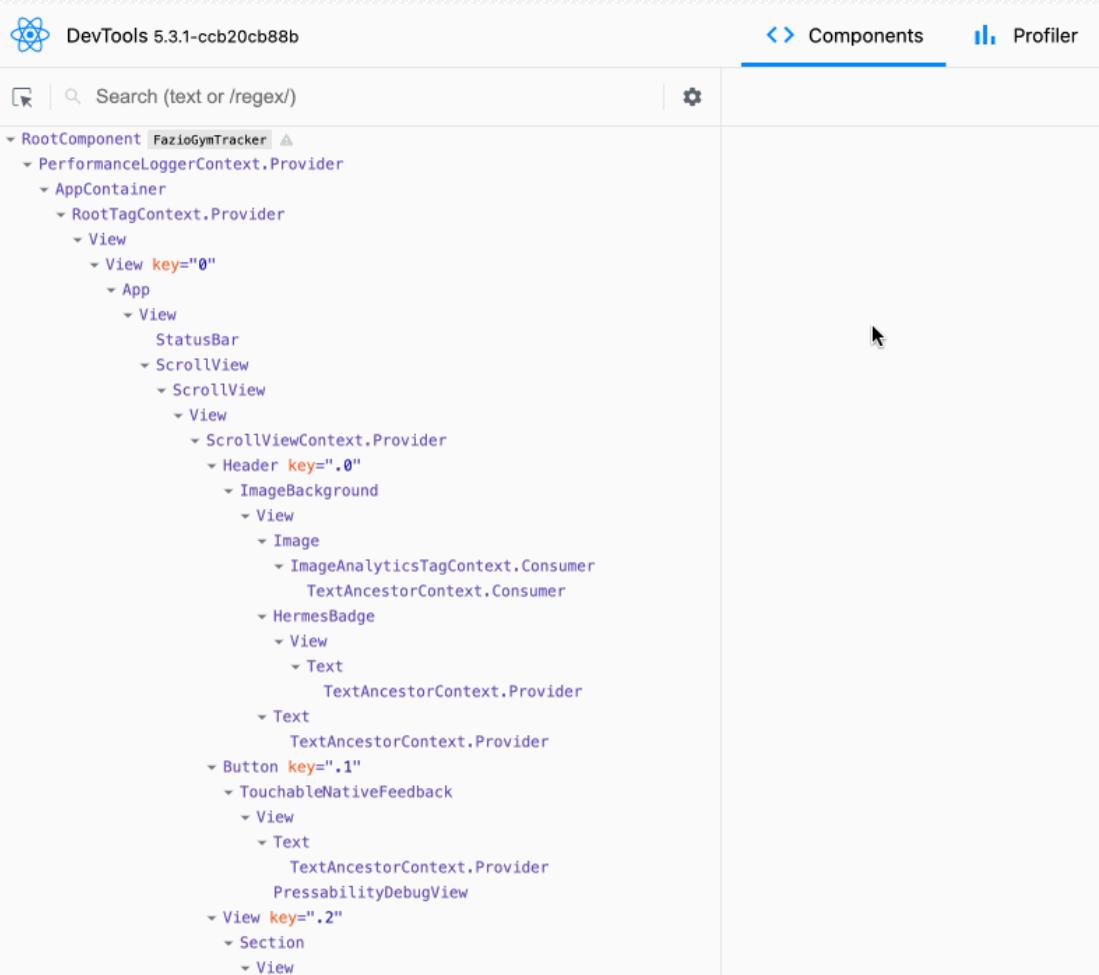
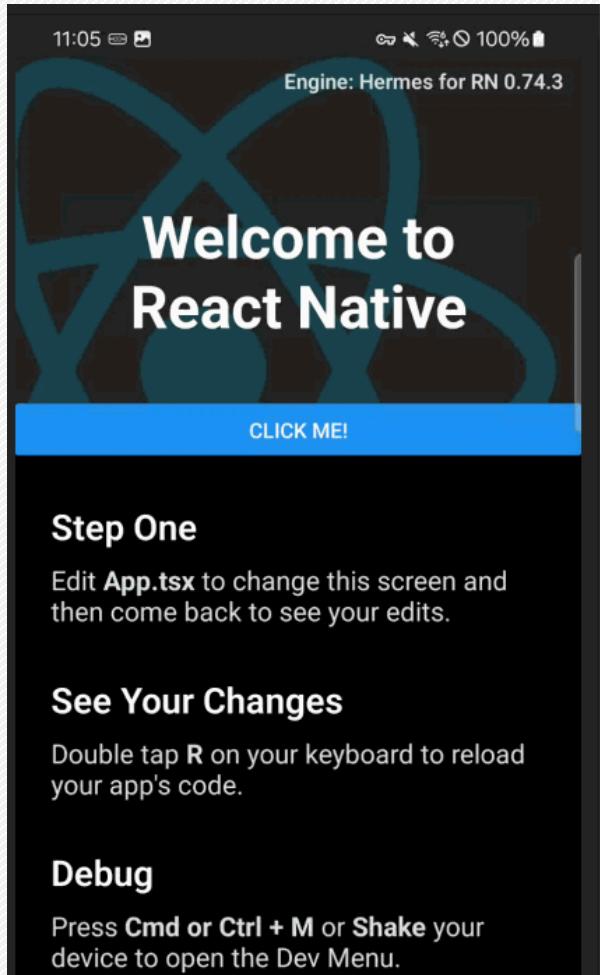
- Console tab is active.
- Sources tab is selected.
- Performance and Memory tabs are visible.
- Call Stack tab is visible.
- Scope tab shows "Not paused".
- Breakpoints tab is visible.
- Watch tab is visible.
- Pause on uncaught exceptions is checked.
- Pause on caught exceptions is unchecked.
- Scope is set to "Not paused".
- Call Stack is set to "Not paused".

```
function App(): React.JSX.Element {  
  const [lastClickTime, setLastClickTime] = React.useState<Date>(null);  
  const isDarkMode = useColorScheme() === 'dark';  
  
  const buttonClicked = () => {  
    const clickTime = new Date();  
    const last = lastClickTime ?? 'N/A';  
    console.log(`The button was clicked at ${new Date()}! Last tap was at ${last}`);  
    setLastClickTime(clickTime);  
  };  
  
  return (  
    <SafeAreaView style={backgroundStyle}>  
      <StatusBar  
        barStyle={isDarkMode ? 'light-content' : 'dark-content'}  
        backgroundColor={backgroundStyle.backgroundColor}  
      />  
      <ScrollView  
        contentInsetAdjustmentBehavior="automatic"  
        style={backgroundStyle}>  
        <Header />  
        <Button title="Click me!" onPress={buttonClicked}></Button>  
        <View  
          style={{  
            backgroundColor: isDarkMode ? Colors.black : Colors.white,  
          }}>  
          <Section title="Step One">  
            Edit <Text style={styles.highlight}>App.tsx</Text> to change this  
            screen and then come back to see your edits.  
          </Section>  
        </View>  
      </ScrollView>  
    </SafeAreaView>  
  );  
}
```

Speaker notes

- ▶ `chrome://inspect`
- ▶ Open the "Hermes React Native" link in there to bring up the Chrome dev tools
- ▶ Able to see the code, set breakpoints, view console logs
- ▶ React DevTools

REACT DEVTOOLS



Speaker notes

- ▶ React DevTools
 - ▶ Gives you React-specific inspection info



Speaker notes

- ▶ If you need to integrate at a lower level than what RN offers:
 - ▶ There's likely a library for whatever you need
 - ▶ But if not, you can create a native module
- ▶ Note that all native modules are initialized at app start, so be careful how many you have
- ▶ Do not have Coroutine support out of the box

NATIVE MODULES

```
class TimberModule(  
    reactContext: ReactApplicationContext  
) : ReactContextBaseJavaModule(reactContext) {  
  
    override fun getName() = "TimberModule"  
  
    @ReactMethod  
    fun log(level: String, message: String, tag: String = "") {  
        val logLevel = getLogLevelFromString(level)  
  
        Timber.tag(tag).log(logLevel, message)  
    }  
  
    private fun getLogLevelFromString(level: String): Int {...}  
}
```

Speaker notes

- ▶ You can do `console.log(...)` on the RN side, but if you want to integrate directly with Timber, here's how it could work
- ▶ Console logs from RN *will* show up in Logcat

NATIVE MODULES

```
class PrefsModule(reactContext: ReactApplicationContext) :  
    ReactContextBaseJavaModule(reactContext) {  
    private val sharedPrefs = ... // Get prefs from reactContext  
  
    @ReactMethod  
    fun setValue(id: String, value: String) {  
        with(sharedPrefs.edit()) {  
            putString(id, value)  
            apply()  
        }  
    }  
}
```

Speaker notes

- ▶ If you wanted to directly reach SharedPreferences, you could do something like this.
- ▶ Again, probably something that's already solved for you, but this is an example.
- ▶ All native module functions will be async
 - ▶ There's a flag to make them synchronous, don't use it
 - ▶ Either call something that can be run synchronously, or use a Promise
- ▶ promise.resolve comes back as a Promises that can be awaited in the JS side.
- ▶ I excluded the getName() function for space

NATIVE MODULES

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

const {TimberModule} = NativeModules;

export enum LogLevel {
  DEBUG = 'Debug',
  VERBOSE = 'Verbose',
  INFO = 'Info',
  ERROR = 'Error',
  WARN = 'Warn',
}

interface TimberInterface {
  log(level: LogLevel, message: string, tag?: string): void;
}

export default TimberModule as TimberInterface;
```

Speaker notes

- ▶ Reference the native modules by name in the JS/TS code
- ▶ Recommend creating a wrapper to make other calls cleaner
- ▶ No type safety between Native and JS layers (if you're using Typescript)

NATIVE MODULES

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

const {PrefsModule} = NativeModules;

interface PrefsInterface {
  getValue(id: string): Promise<string>;
  setValue(id: string, value: string): void;
}

export default PrefsModule as PrefsInterface;
```

```
import {PrefsModule} from './modules/PrefsModule';

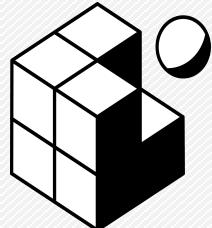
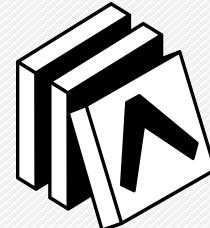
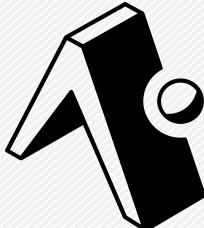
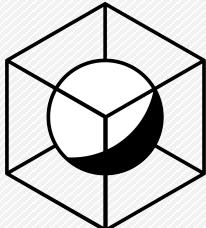
const currentValue = await PrefsModule.getValue('valueId');

PrefsModule.setValue('valueId', 'newValue');
```

Speaker notes



Expo



Speaker notes

- ▶ Framework on top of React Native
 - ▶ expo-cli - Command line tool to create and manage projects
 - ▶ expo-snack - Online editor to test out code
- ▶ Has a ton of libraries you can quickly use for your app
 - ▶ expo-camera, expo-location, expo-notifications, etc.
- ▶ Can use Expo libraries in a vanilla React Native app if you want

▶ Can I use React Native without a framework?

Start a new React Native project with Expo

Platform support



Android



iOS



TV



Web

Expo is a production-grade React Native Framework. Expo provides developer tooling that makes developing apps easier, such as hot reloading, native code generation, based routing, a standard library of native modules, and much more.

Expo's Framework is free and open source, with an active community on [GitHub](#) and [Discord](#). The Expo team works in close collaboration with the React Native team at Meta to bring the latest React Native features to the Expo SDK.

The team at Expo also provides Expo Application Services (EAS), an optional set of services that complements Expo, the Framework, and tools like [React Native CLI](#) and [React Native Doctor](#) at each step of the development process.

To create a new Expo project, run the following in your terminal:

```
npx create-expo-app@latest
```

Once you've created your app, check out the rest of Expo's getting started guide to start developing your app.

Speaker notes

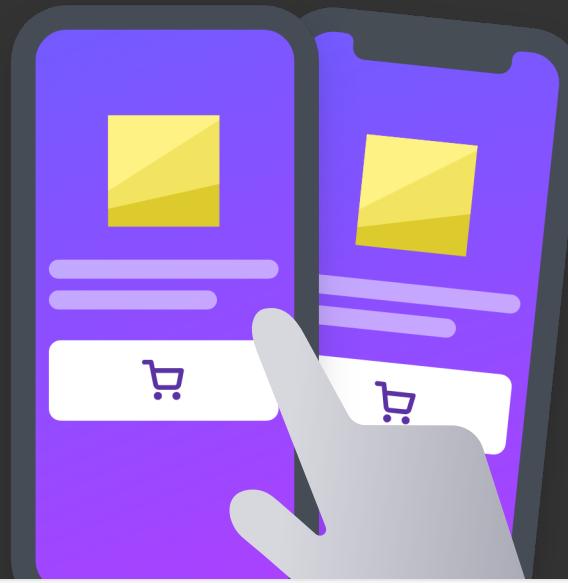
- ▶ The RN site in their intro says to use a framework and call out Expo

EXPO GO

```
$ npm start

Developer tools running on
http://localhost:19002
Starting bundler

> Press a | open Android emulator
> Press i | open iOS simulator
> Press w | open web
```



Speaker notes

- ▶ Quick way to test things out
- ▶ Don't use it long term
- ▶ No custom native modules
- ▶ Mentioned because Expo apps usually start here
- ▶ Easier to get up and running with iOS than deploying an app normally

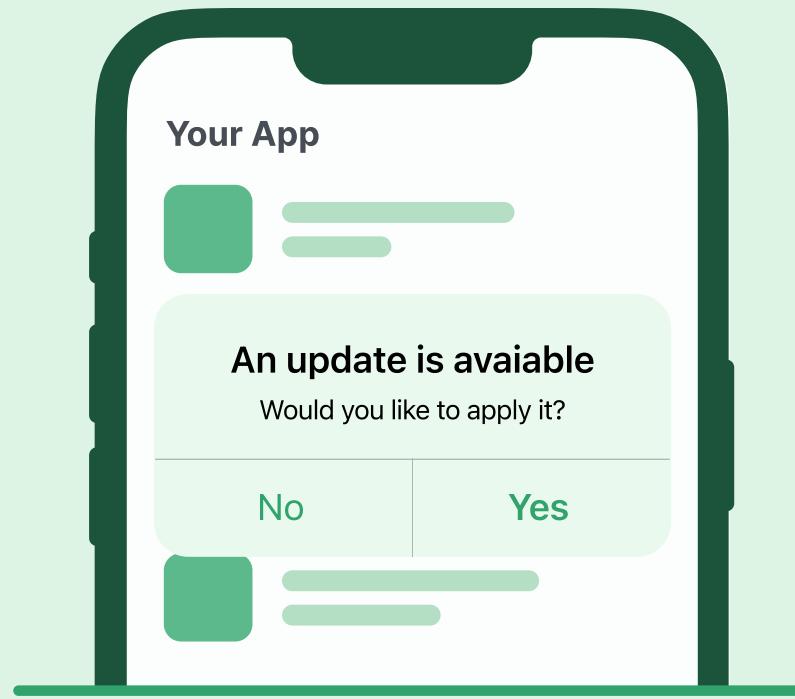
EXPO APPLICATION SERVICES



Speaker notes

- ▶ Expo Application Services (EAS)
- ▶ Haven't used EAS yet, but it's on my radar
- ▶ Cloud builds
 - ▶ Native should work as well
- ▶ Quick Play Store submissions

EAS UPDATE



Speaker notes

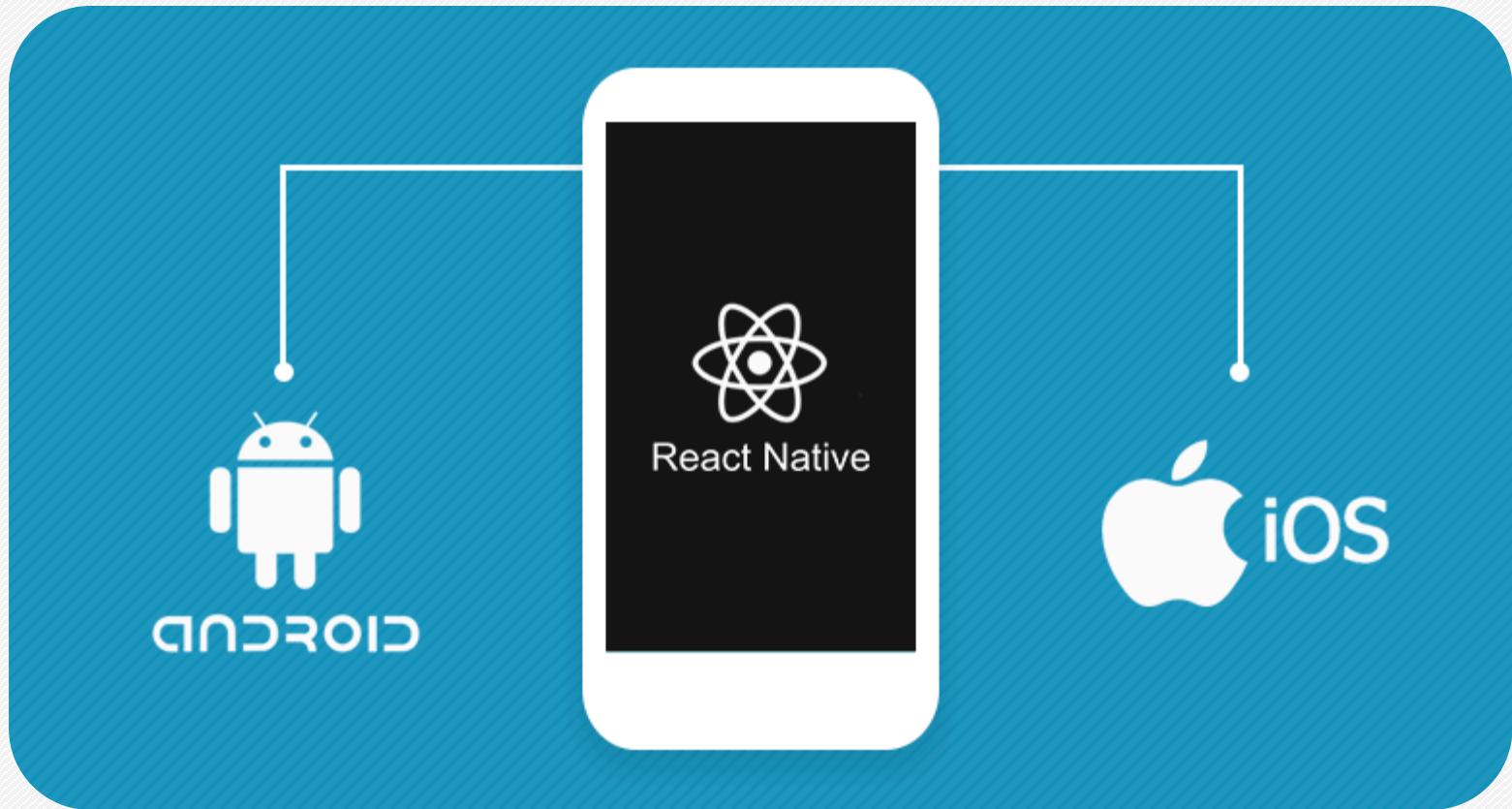
- ▶ EAS Update
 - ▶ Push out updates without a Play Store release
 - ▶ Able to hotfix without needing people to manually update
 - ▶ Works with Expo and pure RN apps
 - ▶ Only can do smaller fixes



Speaker notes

- ▶ A number of nice to great things about React Native

CROSS-PLATFORM



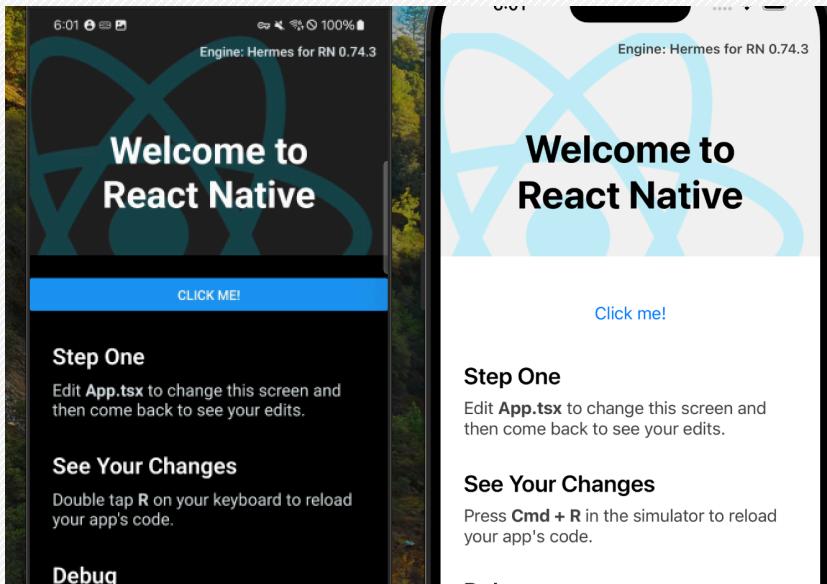
Speaker notes

- ▶ It *does* work pretty well with sharing code between platforms
 - ▶ Not always, but overall it's good, especially if you use libraries

CROSS-PLATFORM

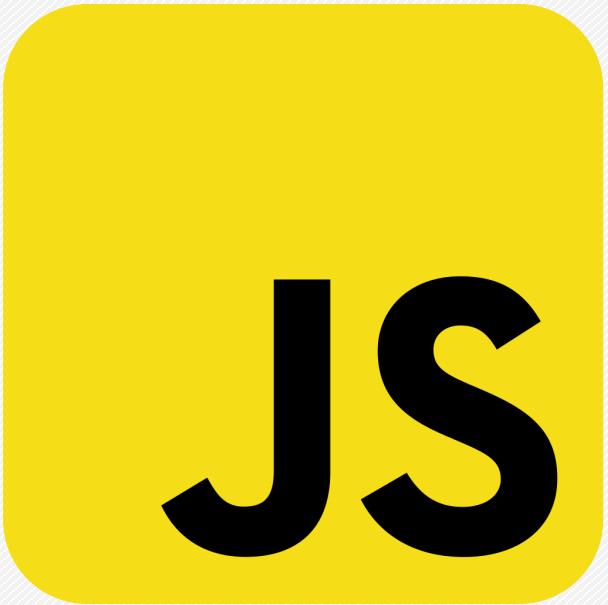
```
const topMargin = (Platform.OS === 'android') ? 24 : 40;
```

```
const topMargin = Platform.select({  
  ios: 40,  
  android: 24,  
});
```



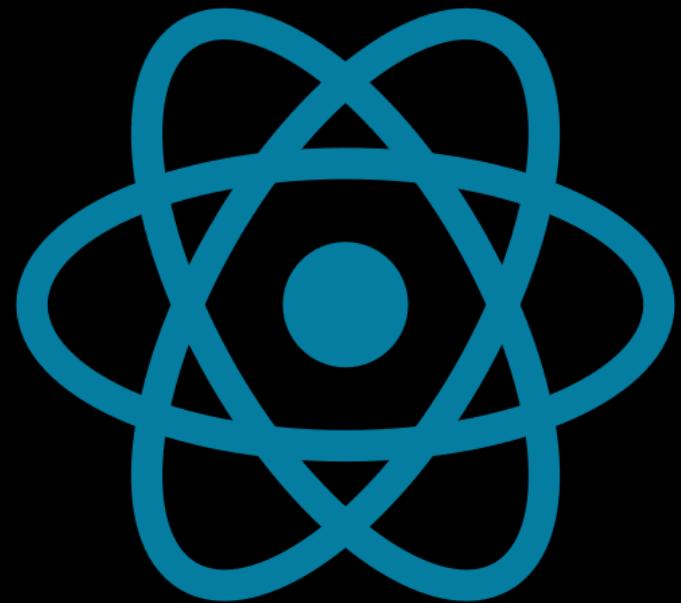
Speaker notes

- ▶ It *does* work pretty well with sharing code between platforms
- ▶ Not always, but overall it's good, especially if you use libraries



Speaker notes

- ▶ You use JavaScript to write your code
- ▶ It's well known
- ▶ Nothing weird like Dart or Objective-C
- ▶ TypeScript is extra handy
- ▶ Tons of docs, libraries, tooling is established
- ▶ Only need one language for most of your code



Speaker notes

- ▶ React's really nice to use
- ▶ Well known
- ▶ Tons of resources

FAST REFRESH

The screenshot shows a React Native development interface. On the left, there's a sidebar titled "Running Devices - android" showing a "Welcome to React Native" screen. Below it are sections for "Step One", "See Your Changes", "Debug", and "Learn More". In the center, a code editor titled "App.tsx" displays the following code:

```
function App(): React.JSX.Element {  Show usages
  const isDarkMode :boolean = useColorScheme() === 'dark';

  const backgroundStyle :{backgroundColor:any} = {
    backgroundColor: isDarkMode ? Colors.darker : Colors.lighter,
  };

  return (
    <SafeAreaView style={backgroundStyle}>
      <StatusBar
        barStyle={isDarkMode ? 'light-content' : 'dark-content'}
        backgroundColor={backgroundStyle.backgroundColor}
      />
      <ScrollView
        contentInsetAdjustmentBehavior="automatic"
        style={backgroundStyle}>
        <Header />
        <View
          style={{
            backgroundColor: isDarkMode ? Colors.black : Colors.white,
          }}>
          <Section title="Step One">
            Edit <Text style={styles.highlight}>App.tsx</Text> to change this
            screen and then come back to see your edits.
          </Section>
          <Section title="See Your Changes">
            <ReloadInstructions />
          </Section>
          <Section title="Debug">
            <DevMenu />
          </Section>
        </View>
      </ScrollView>
    </SafeAreaView>
  );
}
```

On the right, a simulator window titled "iPhone 15 Pro - iOS 17.5" shows the same "Welcome to React Native" screen. Below the interface, there are sections for "Step One", "See Your Changes", "Debug", and "Learn More" on both the left and right sides.

Speaker notes

- ▶ Quick feedback on changes in your app
 - ▶ Change a view, see it on your device pretty much right away
- ▶ Edits to a component used by multiple other components will cause the other components to be refreshed as well
- ▶ Worst-case scenario, the whole app refreshes for you

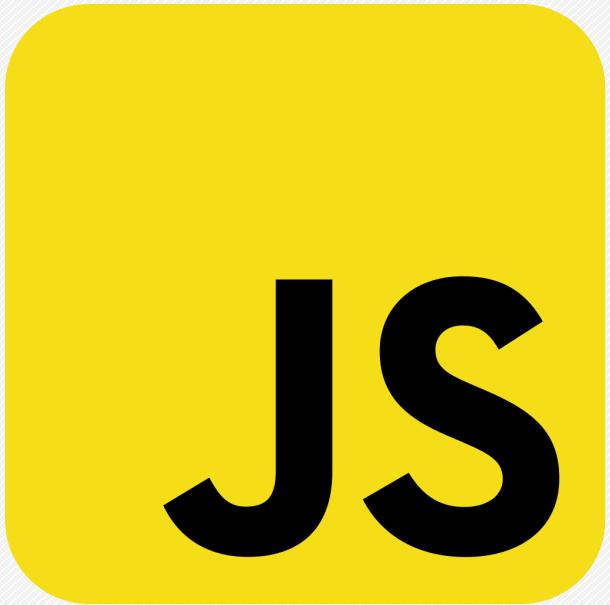
FIFA WORLD CUP™ - SEMIFINAL

BRA 0-4 GER 25:17



Speaker notes

- ▶ There are also some things that aren't so great
- ▶ 2014 World Cup in Brazil
- ▶ Germany/Brazil ended 7-1
- ▶ Semifinal match



Speaker notes

- ▶ You use JavaScript to write your code
- ▶ You're in the node space
 - ▶ Massive `node_modules` directory

JAVASCRIPT ...WAT

```
[ ] + [ ] // ""

[ ] + {} // "[object Object]"

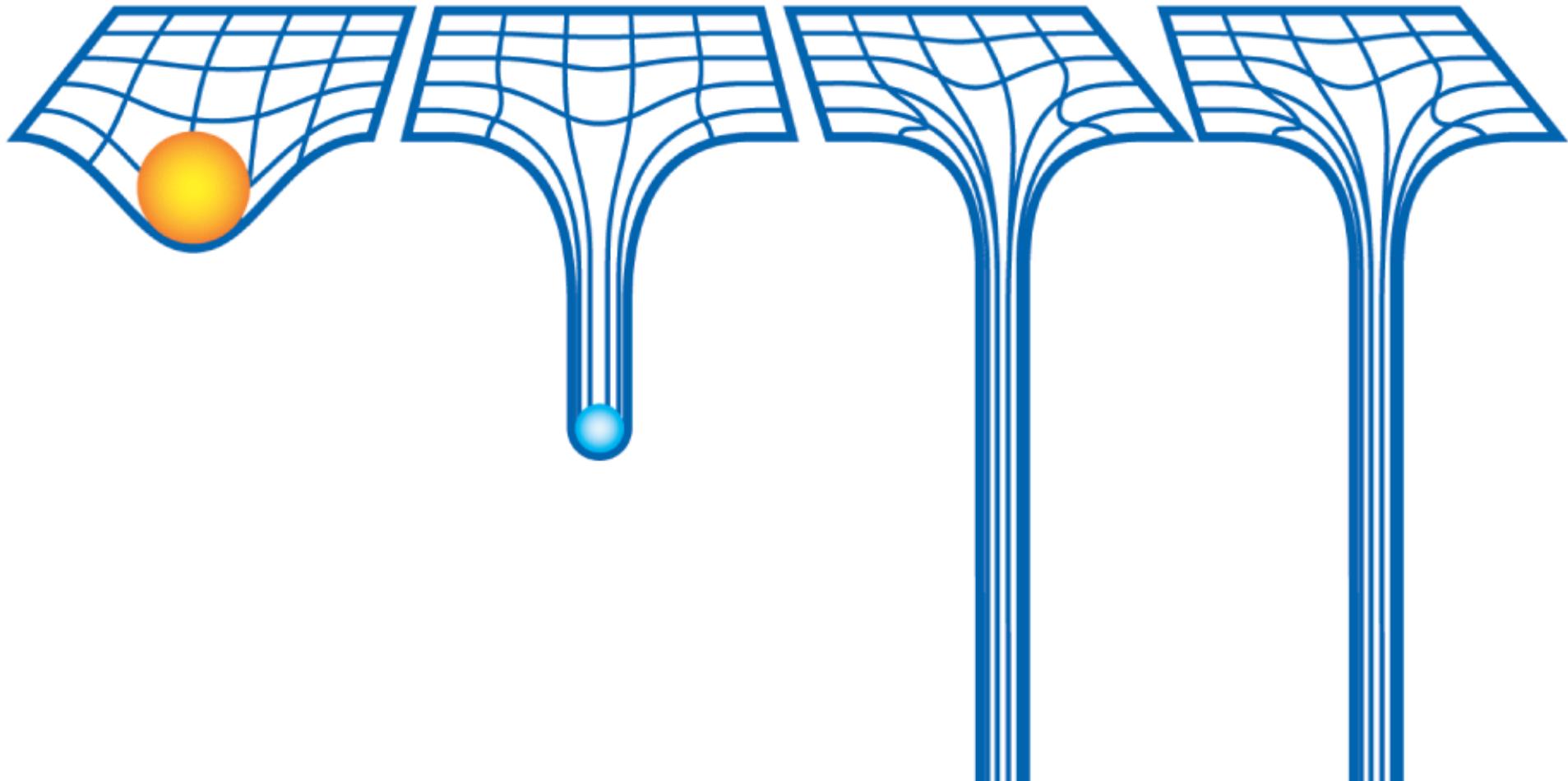
{} - [ ] // 0
```



Speaker notes

- ▶ <https://www.destroyallsoftware.com/talks/wat>
- ▶ A lightning talk by Gary Bernhardt from CodeMash 2012

Sun Neutron star Black hole node_modules



Speaker notes

- ▶ You're in the node space
- ▶ Massive node_modules directory

0.74.3

Latest

Compare ▾



Titozzz released this 3 weeks ago · 1833 commits to main since this release

<> v0.74.3

-o- 8e5a936

Added

- Add the ReactMarkerConstants.CONTENT_APPEARED support on Android in bridgeless mode. ([3c4d7618f0](#) by [@Kudo](#))

Changed

- Feat: update CLI to 13.6.9 ([d1e2a35061](#) by [@szymonrybczak](#))

iOS specific

- Support `customizeRootView` from `RCTRootViewFactory` ([3c4d761](#) by [@Kudo](#))

Fixed

- Codegen computes output path relative to project root instead of current working directory. ([d3e0430dea](#) by [@dmytrorykun](#))

Android specific

Speaker notes

- ▶ It's currently version 0.74.x
 - ▶ It's not even 1.0 yet!
 - ▶ Breaking changes every new major release
 - ▶ They *do* have an upgrade guide
 - ▶ 3rd party libraries don't always cooperate



React Native Upgrade Helper

Star 3,528



What's your app name?

FazioGymTracker

What's your app package?

dev.mfazio.gymtracker

What's your current react-native version?

0.72.1

To which version would you like to upgrade?

0.74.3

Show me how to upgrade!

Useful content for upgrading

Release 0.73

React Native 0.73 includes an updated process for the iOS privacy manifest, now required by Apple

1. [Learn how to update your app's Apple privacy settings](#)
2. [React Native 0.73 changelog](#)

Release 0.74

React Native 0.74 includes Yoga 3.0, Bridgeless by default under the New Architecture, batched onLayout updates, Yarn 3, removal of previously deprecated PropTypes, and some breaking changes, including updates to PushNotificationIOS. The Android Minimum SDK is now 23 (Android 6.0).

1. [Official blog post about the major changes on React Native 0.74](#)
2. [React Native 0.74 changelog](#)

You can use the following command to kick off the upgrade: `npx @rnx-kit/align-deps --requirements react-native@[major.minor]`.

`align-deps` is an OSS tool from Microsoft that automates dependency management. It knows which packages* versions are compatible with your specific version of RN, and it uses that knowledge to align dependencies, keeping your app healthy and up-to-date**. [Find out more here.](#)

* Not all packages are supported out-of-the-box.

** You still need to do the other changes below and verify the changelogs of the libraries that got upgraded.

Check out [Upgrade Support](#) if you are experiencing issues related to React Native during the upgrading process.

Speaker notes

- ▶ <https://react-native-community.github.io/upgrade-helper/>

✓ package.json MODIFIED

```

@@ -11,27 +11,27 @@
11  },
12  "dependencies": {
13    "react": "18.2.0",
14    "react-native": "0.72.1"
15  },
16  "devDependencies": {
17    "@babel/core": "^7.20.0",
18    "@babel/preset-env": "^7.20.0",
19    "@babel/runtime": "^7.20.0",
20    "@react-native/eslint-config": "0.72.2",
21    "@react-native/metro-config": "0.72.7",
22    "@tsconfig/react-native": "3.0.0",
23    "@types/metro-config": "0.76.3",
24    "@types/react": "^18.0.24",
25    "@types/react-test-renderer": "18.0.0",
26    "babel-jest": "29.2.1",
27    "eslint": "8.19.0",
28    "jest": "29.2.1",
29    "metro-react-native-babel-preset": "0.76.5",
30    "prettier": "2.4.1",
31    "react-test-renderer": "18.2.0",
32    "typescript": "4.8.4"
33  },
34  "engines": {
35    "node": ">=16"
36  }
37 }

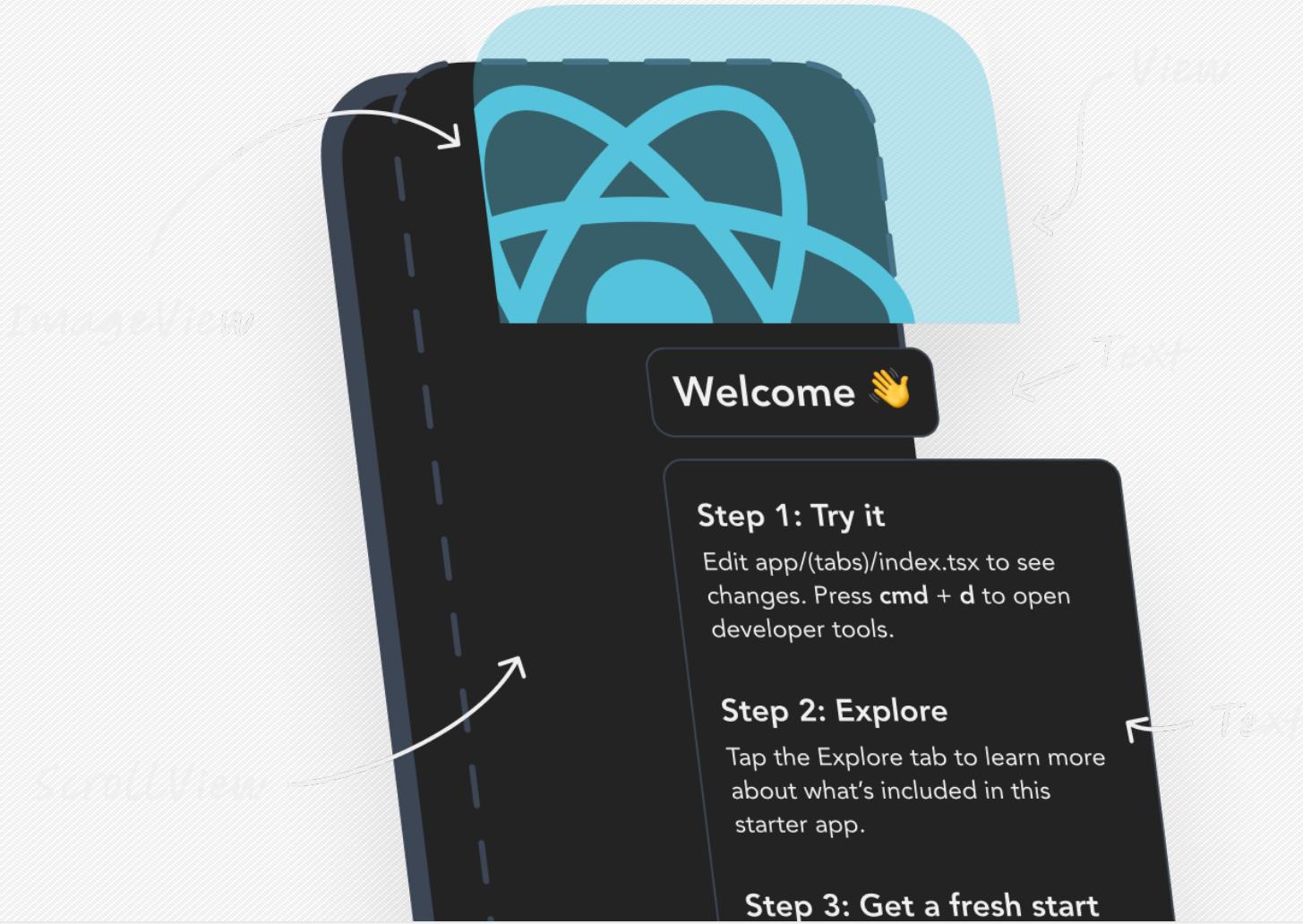
11  },
12  "dependencies": {
13    "react": "18.2.0",
14    "react-native": "0.74.3"
15  },
16  "devDependencies": {
17    "@babel/core": "^7.20.0",
18    "@babel/preset-env": "^7.20.0",
19    "@babel/runtime": "^7.20.0",
20    "@react-native/babel-preset": "0.74.85",
21    "@react-native/eslint-config": "0.74.85",
22    "@react-native/metro-config": "0.74.85",
23    "@react-native/typescript-config": "0.74.85",
24    "@types/react": "^18.2.6",
25    "@types/react-test-renderer": "18.0.0",
26    "babel-jest": "29.6.3",
27    "eslint": "8.19.0",
28    "jest": "29.6.3",
29    "prettier": "2.8.8",
30    "react-test-renderer": "18.2.0",
31    "typescript": "5.0.4"
32  },
33  "engines": {
34    "node": ">=18"
35  },
36  "packageManager": "yarn@3.6.4"
37 }

```

In React Native 0.74, for projects bootstrapped with React Native Community CLI, we've added first-class support for modern Yarn versions. For new projects Yarn 3.6.4 is the default package manager, and for existing projects, you can upgrade to Yarn 3.6.4 by running `yarn set version berry` in the project root. Read more [here](#).

Speaker notes

- ▶ <https://react-native-community.github.io/upgrade-helper/>



Speaker notes

- ▶ Extra layers between you and the device
 - ▶ RN -> Native code -> Device
 - ▶ Fragmentation effects get even worse
- ▶ Yes, we're using native components, but we're still a layer above them

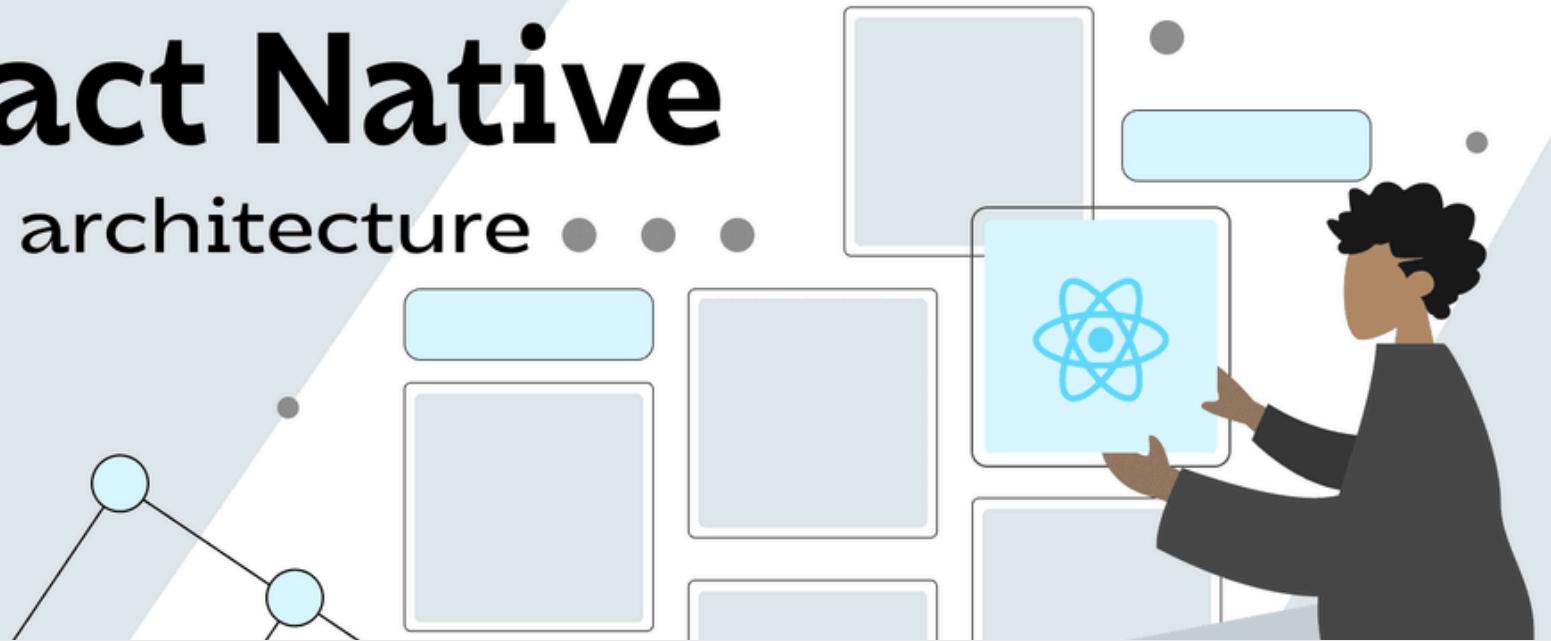


Speaker notes

- ▶ We've gone over some "cons", but here are some gotchas I've faced that could be useful
- ▶ These are some "heads up"s that I thought would be helpful

React Native

New architecture



Speaker notes

- ▶ New projects will likely use the current architecture
- ▶ There's a "New Architecture" that's available as well
- ▶ In progress since 2018
- ▶ Planned to be the default by end of 2024.
- ▶ "For most production apps, we do not recommend enabling the New Architecture today."
- ▶ Faster JS/Native interfacing
- ▶ Better layout and rendering



Speaker notes

- ▶ By default, this is the case, so don't try to send it to someone
- ▶ You *can* create a build with the JS files bundled, but it takes extra work

〈COCOAPODS〉

Speaker notes

- ▶ Cocoapods is a dependency manager for iOS projects
- ▶ Usually don't have to mess with it too much directly
 - ▶ Update your package.json and RN should autolink
- ▶ But if you delete your Podfile.lock, you could end up with slightly different versions of dependencies
- ▶ Heads up in case you didn't change anything (or think you did)

```
84 <TextInput  
85   style={{backgroundColor: 'purple'}}  
86   keyboardType={keyboardType}  
87   placeholder={'Issue'}  
88 />  
89  
90 <TextInput  
91   style={{backgroundColor: 'aqua'}}  
92   keyboardType={keyboardType}  
93   placeholder={'No issue here?'}  
94 />  
95  
96 <TextInput  
97   style={{backgroundColor: 'purple'}}  
98   keyboardType={keyboardType}  
99   placeholder={'First name will show up here'}  
100 />  
101  
102 <TextInput  
---
```

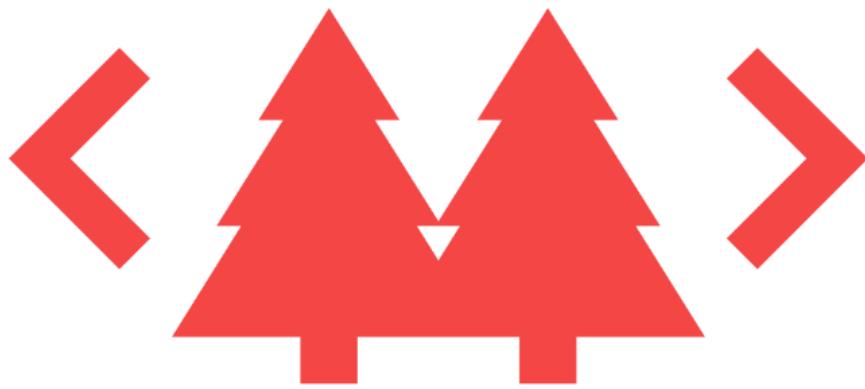
Speaker notes

- ▶ Here's a fun bug I found recently
- ▶ Some devices (this one's a Samsung S22+) will show suggestions for text input fields
- ▶ But the suggestions don't make sense, and the code is the same for all four fields.



Speaker notes

- ▶ What questions do you have?
- ▶ Available to chat otherwise, too



THANK YOU!

Speaker notes

- ▶ Thank you!