OBJECT COMPUTING

WEBINAR

React Native Overview

mark@objectcomputing.com

PREREQUISITES

- This talk assumes familiarity with JavaScript, React, and CSS

REACT NATIVE OVERVIEW ...

- Open-source framework for building mobile apps with React, JavaScript, and CSS
- Developed by Facebook
- Uses native components, not WebViews
- Can mostly focus on React Native components without learning details of iOS and Android APIs
- Can integrate native code when needed (if not using Expo)
  - Java and Kotlin for Android; Swift and Objective-C for iOS

… REACT NATIVE OVERVIEW

- Uses CSS Flexbox for layout
- Supports automatic reloading in simulators and real devices
- Can debug with Chrome devtools
- Supports over-the-air (OTA) app updates without a new app review
- Used by
  - Adidas, Bloomberg, Facebook, Instagram, Tesla, Pinterest, Salesforce, Skype, Uber, Walmart, Wix, & more

WHY USE REACT NATIVE?

- Want to develop native mobile apps for Android and iOS
- Want most/all of the code to work on both
- Don’t want to learn Java, Kotlin, or Swift
- Want to leverage expertise in JavaScript and CSS
- Maybe already know React
- Want automatic reloading
- Don’t have high performance requirements (not a video game)

COMPETITORS

all open source

apps built on WebViews lack native look-and-feel and are slower than native apps

Name	From	Native or WebView	Programming Language	Frameworks Supported	Notes
PhoneGap	Adobe	WebView	JavaScript, HTML, & CSS	custom	• contributed to Apache in 2011 and renamed Cordova • development continued outside of Apache • now a fork of Cordova • uses plugins for accessing device capabilities
Cordova	Apache	WebView	JavaScript, HTML, & CSS	custom	• uses plugins for accessing device capabilities
Ionic	Drifty	WebView	JavaScript, HTML, & CSS	Angular (working on React & Vue)	• also creates web apps and Electron-based desktop apps • has commercial support for CI/CD and themes
Xamarin	Microsoft	Native	C#	custom	• also creates desktop apps for Windows and macOS • has free and enterprise tiers
NativeScript	Progress (Telerik)	Native	JavaScript	Angular & Vue	
React Native	Facebook	Native	JavaScript & CSS	React	
Vue Native		Native	JavaScript & CSS	Vue	• builds on React Native and can use all its features and components • transpiles `.vue` files to React components
Flutter	Google	Native	Dart	custom	• as of March 2019 not ready for serious use • Dart is a hot mess

SAME IN REACT AND REACT NATIVE

- Language - JS or TS
- Components - how defined
- Props
- State management
- Lifecycle methods
- Sending HTTP requests
- Linting tools - ESLint
- Formatting tools - Prettier
- Unit testing - Jest?

DIFFERENT IN REACT AND REACT NATIVE

- Elements used
  `div` vs. **`View`**, **`span`** vs. **`Text`**, ...
- Platform-specific components -
  Android and iOS
- Styling approach -
  CSS vs. style objects
- Debugging approach -
  web browser vs. simulators & Expo
- Deployment -
  web server vs. app stores
- Access to device capabilities -
  ex. camera
- End-to-end testing - Cypress vs. ?
- Devtools -
  browser extensions vs. simulators

ARCHITECTURE

- Uses JavaScriptCore to execute JavaScript code at runtime
  - Safari JavaScript engine implemented in C/C++
  - bundled with Android apps
- JavaScript thread communicates with native threads using the “bridge”
  - acts as an asynchronous message broker
  - implemented in C/C++
- Android details
  - uses Android Runtime (ART), not JVM

RESOURCES

- React Native - https://facebook.github.io/react-native/
- React Native Express - http://www.reactnativeexpress.com/
- Awesome React Native - http://www.awesome-react-native.com/
- Expo - https://expo.io/
  - set of React Native tools, libraries and services
- npmjs.org
  - search for “react-native”; almost 16,000 packages as of 3/24/19

TWO WAYS TO START

- Expo CLI
  - includes Expo SDK which is a library of components and services
  - can "eject" to switch to just using React Native directly later
  - install "Expo Client" on devices to easily run apps on them
    - just need to scan QR code displayed in web browser
- React Native CLI

EXPO CLI

- Pros
  - can run app on devices over wifi — on macOS why must the firewall be turned off to use LAN mode?
  - version upgrades are easier
  - easier to deploy to app stores — handles keys, signing credentials, and certificates for you
  - over-the-air updates — skips review process after first release
- Cons
  - can't use native modules that require building native code — can use Expo libraries that wrap them
  - some device APIs aren't supported yet (ex. Bluetooth) — see status at https://expo.canny.io/feature-requests
  - resulting app sizes may be larger
  - lags behind latest React Native release by a few months

native code is written in Java, Kotlin, Objective-C, or Swift

REACT NATIVE CLI

- Pros
  - can use native modules
- Cons
  - linking libraries and using CocoaPods is brittle
  - upgrades are often a nightmare

EXPO (v32)

- Free tool that wraps React Native and adds many features
- https://expo.io/
- Runs on macOS, Windows, and Linux
- Supports Android 5+ and iOS 10+
  - don’t use if older versions must be supported
- Each version of Expo works with a specific version of React Native
- Simplifies updating to new versions of React Native

“Expo is kind of like Rails for React Native.
Lots of things are set up for you,
so it's quicker to get started
and on the right path.”

EXPO MODES

- "managed"
  - no need to install Android Studio or Xcode
  - after code changes, Expo rebuilds app, hosts in local server, and deploys to simulators/devices
  - some native APIs are not supported
  - **`expo upload`** deploys to stores
  - **`expo publish`** deploys an update to stores
- "bare"
  - can use all **native APIs** and include native code (Java, Kotlin, Swift)
  - you handle steps to build, upload, and publish using Android Studio and Xcode

devices must have "Expo Client" installed from app store

examples include Bluetooth, in-app purchases, and WebRTC

for details on how it works, see https://docs.expo.io/versions/v32.0.0/workflow/how-expo-works/

EXPO MANAGED MODE (does not apply to "bare" mode)

- Pros
  - testing on real devices using Expo Client
  - creates binaries without interacting with Android Studio or Xcode
    - in "managed" mode, not "bare" mode
  - supports over-the-air updates to apps in stores
- Cons
  - no support for native modules
    - must "eject" to use native code
  - no background code execution
  - results in larger apps
    - ~15MB for Android
    - ~20MB for iOS

but these are close to average size of mobile apps

EXPO ADDED COMPONENTS AND APIS

can use outside Expo, but need extra setup

- Components
  - **`Camera`**, **`MapView`**, **`Svg`**, **`Video`**, and more
- APIs
  - **`AppAuth`**, **`Audio`**, **`Calendar`**, **`Contacts`**, **`DeviceMotion`**, **`FaceDetector`**, **`Font`**, **`Haptics`**, **`ImagePicker`**, **`LocalAuthentication`**, **`Localization`**, **`Location`** (geolocation), **`MailComposer`**, **`MediaLibrary`**, **`Notifications`**, **`Payments`**, **`Permissions`**, **`Print`**, **`SecureStore`**, **`Sensors`**, **`SMS`**, **`Speech`**, **`SQLite`**, **`Updates`**, and more

see "API Reference" in left nav. at https://docs.expo.io/versions/latest/

Which of these support push notifications?

SETUP

- There are many steps required to get started
- Step 1: install Node.js from https://nodejs.org/

OPTION #1 - React Native CLI … skip ahead to option #2

- Browse
  https://facebook.github.io/react-native/docs/getting-started
- Click “React Native CLI Quickstart”
- Select development OS and target OS
- Follow instructions

```
npm install -g react-native-cli
react-native init MyProject
cd MyProject
react-native start
```

… OPTION #1 - React Native CLI …

- To run on Android simulator
  - start simulator as shown on slide 23
  - create file ***project-directory*/android/local.properties**
    - add following line

```
sdk.dir=/Users/USERNAME/Library/Android/sdk
```

  - **`react-native run-android`**
  - to reload app press "r" twice

- To run on iOS simulator
  - start simulator as shown on slide 24
  - open a new terminal window
  - **`react-native run-ios`**
  - takes about 5 minutes the first time!
  - to reload app press cmd-r

… OPTION #1 - React Native CLI

- To run on an iOS device
  - attach device to Mac with USB cable
  - in Finder locate ***project-directory/ios/project-name.xcodeproj*** file and double-click it to open in Xcode app
  - wait for processing of files to complete
  - select Product … Destination … device-name
- first time only on a device
  - select top folder in left nav.
  - click "General" tab
  - in the "Signing" section "Team" drop-down select your name
  - at top after project name select the real device or a specific device to simulate
- click play button at top or select Product … Run
- takes about a minute to start

couldn't get this to work the last time I tried with a new react-native-cli project

OPTION #2 - Expo CLI

our focus

OBJECT COMPUTING

- **`npm install -g expo-cli`**
- **`expo init`** ***`my-project`***
  - choose a template, "blank", "tabs", or "bare-minimum"
  - enter app name as it should display on devices
  - enter "Y" to install dependencies including react-native and much more
- **`cd`** ***`my-project`***
- **`npm start`** or **`yarn start`** or **`expo start`**
  - opens a browser tab shown on next slide
  - can start a simulator by pressing "a" for Android or "i" for iOS

Expo projects are tied to specific versions of React and React Native. As of 4/9/2019, cannot use hooks.

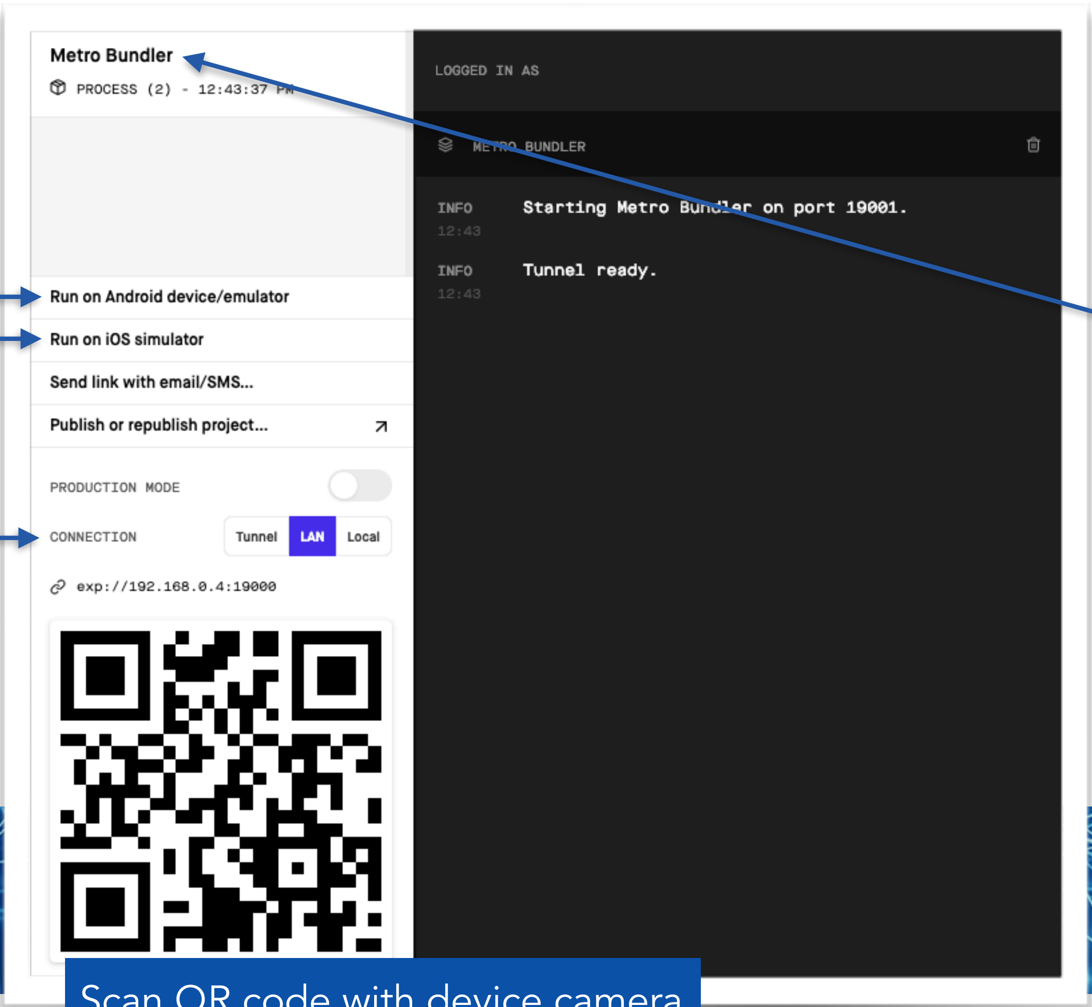
WARNING!
expo v32.0.0 requires restart of this and simulators if a JS syntax error is saved! Watch for percentage less than 100 in window where this is running. Close old browser tab and use new one.

Expo App Page

Click these to run on a simulator. Android simulator must already be running, but iOS simulator will be started.

When **CONNECTION** is set to **LAN**, computer and device must be on same wifi network. This works in macOS if firewall is turned off (System Preferences ... Security & Privacy ... Firewall).

When **CONNECTION** is set to **Tunnel** a cloud-based proxy is used. This is a little slower, but works.


Metro Bundler
PROCESS (2) - 12:43:37 PM
Run on Android device/emulator
Run on iOS simulator
Send link with email/SMS...
Publish or republish project...
PRODUCTION MODE
CONNECTION
Tunnel
LAN
Local
LOGGED IN AS
METRO BUNDLER

```
exp://192.168.0.4:19000
```

```
INFO   Starting Metro Bundler on port 19001.
12:43

INFO   Tunnel ready.
12:43
```

"Metro Bundler" is a JavaScript bundler for React Native that supports watch and live reload in Android and iOS simulators.

Scan QR code with device camera to run app in "Expo Client".

OS-SPECIFIC REQUIREMENTS

- Developing for Android in Windows requires Python 2
  - can’t use Python 3?
- Does “Xcode for Windows” allow developing iOS apps in Windows?

SIMULATORS

- Can use Android and iOS simulators to test apps
- iOS simulator requires macOS
  - true?
  - Does "Xcode for Windows" support the iOS simulator?

INSTALL ANDROID SIMULATOR

- Browse https://developer.android.com/studio
- Press “DOWNLOAD ANDROID STUDIO”
  - based on Intellij
- Double-click downloaded installer and follow instructions

RUN ANDROID SIMULATOR

- Launch Android Studio app and follow one-time setup instructions
- Select Tools … AVD Manager stands for "Android Virtual Device"
- Click "+ Create Virtual Device…" to add a new device simulator
- Click "Download" after an existing device simulator
- Click green triangle "play" button after a device simulator
    - takes a couple of minutes the first time
- Can quit Android Studio

INSTALL AND RUN IOS SIMULATOR

- Install Xcode
  - browse https://developer.apple.com/xcode/ and click Download button
- To run simulator
  - launch Xcode app
  - select Xcode … Open Developer Tool … Simulator
  - select device by selecting Hardware … Device … *os* … *device-name*
  - can quit Xcode

Alternative to launching Xcode:

```
npm install -g ios-sim
ios-sim showdevicetypes
ios-sim start --devicetypeid "full-device-type"
```

Handy npm script:

```
"ios-sim": "ios-sim start --devicetypeid 'iPhone-XR, 12.1'",
```

RUN APP ON SIMULATOR

- Start one or both simulators
- In Expo web app, press
  "Run on Android device/emulator" or "Run on iOS simulator"
  - takes a couple of minutes the first time
  - installs Expo Client in simulator

INSTALL EXPO CLIENT ON DEVICES

- Install "Expo Client" app on mobile devices
- Launch "Expo Client"
- Press "Sign up for Expo"
- Enter requested info. and press "Sign Up"
    - will receive an authentication email

RUN APP ON ANDROID DEVICE USING EXPO CLIENT

- TRY THIS AND DOCUMENT!
- SHOULD BE SIMILAR TO NEXT SLIDE

RUN APP ON iOS DEVICE USING EXPO CLIENT

- Open Camera app on device
- Point at QR code in Expo web app
- Press "Open in Expo"
  - will launch Expo Client app if not already running

When "CONNECTION" is "LAN", computer and phone must be on same Wifi network.
May work better when "CONNECTION" is "Tunnel".

PROJECT STRUCTURE

- `index.js` - app entry point
- `App.js` - top component
- `package.json` - describes dependencies and scripts
- `node_modules` directory - holds libraries installed by npm or yarn
- `android` directory - holds Android-specific code
- `ios` directory - holds iOS-specific code
- `.flowconfig` - configures the Flow type checker
- `.gitignore` - lists directories and files that should not be saved in the Git repository
- `.watchmanconfig` - configures the Facebook Watchman file watcher that supports hot reloading

```
import {AppRegistry} from 'react-native';
import App from './App';
import {name as appName} from './app.json';

AppRegistry.registerComponent(appName, () => App);
```

often don't need to edit these files

INITIAL FILES TO EDIT

- `package.json` - dependencies and scripts
- `app.json` - app configuration
- `app.js` - top-most component

REACT NATIVE DOES NOT USE HTML

- `div -> View` main building block component
- `span -> Text`
- `button -> Button`
- `img -> Image`
- and many more differences