Mobile App Bootstrap: React Native

Provides a look at example code for how to embed Tableau vizzes inside of a React Native app that runs on iOS and Android.

Installation

Mac:

1. Download the code for the app from the github repository
2. Install Visual Studio Code
- VS Code is the main IDE we use for React Native development; it's where you'll do most of your editing.
3. Install Xcode for creating iOS builds
4. Install Android Studio for creating Android builds
- You might need these added to your `~/.bash_profile`: `export ANDROID_HOME=$HOME/Library/Android/sdk` `export PATH="$PATH:$ANDROID_HOME/platform-tools:$ANDROID_HOME/tools"`
5. Install homebrew
6. `brew install node`
- This uses homebrew to install node
7. `brew install watchman`
- watchman enables some fun auto-reload-on-file-change capabilities in React Native
8. `npm install -g react-native-cli`
- This installs the React Native Command Line Interface

Windows:

1. Download the code for the app from the github repository
2. Install Visual Studio Code
- VS Code is the main IDE we use for React Native development; it's where you'll do most of your editing.
3. Follow Step 1 and Step 2 in these excellent Windows installation instructions

Note that we've run into problems running watchman on Windows, so the auto-reload functionality won't be available. Also note that iOS building/

running is only possible on a Mac. We do almost all of our mobile development using Macs.

Download Modules

From the `Mobile_App_Bootstrap` directory, run `npm install`

- This installs the various modules needed into a `node_modules` directory.

Starting the sample app

iOS

- `react-native run-ios` runs the iOS build in the default simulator.
    - `react-native run-ios --simulator="MySimulatorName"` will run the iOS build using the simulator you specify.
        - More details on specifying simulators
        - List of available simulators: `xcrun simctl list`
        - Example to create your own named simulator for an iPad Air 2 running iOS 11.3: `xcrun simctl create MySimulatorName com.apple.CoreSimulator.SimDeviceType.iPad-Air-2 com.apple.CoreSimulator.SimRuntime.iOS-11-3`
- ⌘-d in the iOS simulator brings up the React Native debug menu.

Android

- Run an AVD (Android Virtual Device) emulator. This could be started via Android Studio (Tools -> AVD Manager) or the command line
- `react-native run-android` runs the Android build
- ⌘-m in the Android emulator brings up the React Native debug menu.

For running on a physical iOS/Android device, take a look at the docs for the subtleties involved.

Debugging

React Native has some great debugging documentation, it's well worth reading through. A common setup for us is to have the standalone React Developer Tools running and the simulator with Live Reload and Hot Reloading turned on.

Project Layout

The main app files you'll want to edit are in `src`. The app uses React Navigation to provide the bottom tab navigation.
The core of this project was created using `create-react-native-app`, as outlined in the React Native Getting Started docs. After the app was made, it was "ejected" and modified.

Customization Points

- `Home.js` shows the Home tab. A tap on one of the cards opens a CardDetails object, displaying the provided URL in a webview.
- `Viz1.js`, `Viz2.js`, `Viz3.js` show the three viz tabs. They open the hard-coded URLs provided. Note that WKWebView is used behind the scenes on iOS, while the default WebView is used on Android. You'll want to change these URLs to point to your own vizzes.
- `router.js` configures the bottom tabs using React Navigation.

URL parameters

The sample URLs use several query parameters which are especially helpful for mobile embedded vizzes:

- `:embed=y`: Requests the embedded version of the viz, without the server navigation UI.
- `:tooltip=n`: Removes tooltips. Used in this demo to keep the user on the viz.
- `:toolbar=n`: Removes the viz toolbar.
- `:showVizHome=no`: Used in this demo to not show Tableau Public home information. Unnecessary when connecting to servers not named Tableau Public.
- `:mobile=y`: Explicitly requests a touch-friendly UI, rather than relying on Tableau Server's User-Agent sniffing.
- `:showAppBanner=n`: Removes the "Open in Tableau Public" banner at the top of the viz.