Cordova - App Emulation - Test locations for Geolocation

A brief introduction to using test location data with Cordova emulation options.

Contents

Intro

Application emulators for Cordova apps may be considered relative to an installed native SDK, browser, serve, and hosted cloud options.

For the mobile device emulator included with the Android SDK, we may control and update settings and data using a standard command line, terminal, or console application.

Using the command line, we may test and debug an application using simulated *geo* data.

We may also test our app using a connected device, and the Chrome browser developer tools.

Android emulation

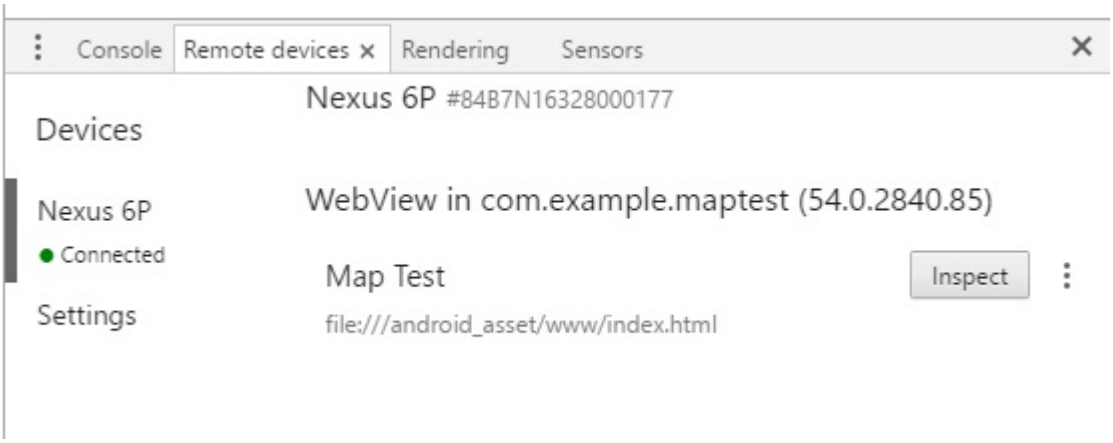
We can test Android based apps with many different types of emulators, including the browser and the Android SDK emulator. We can also run an application on a connected device, and simulate geolocation data for a compatible app.

Device and browser testing

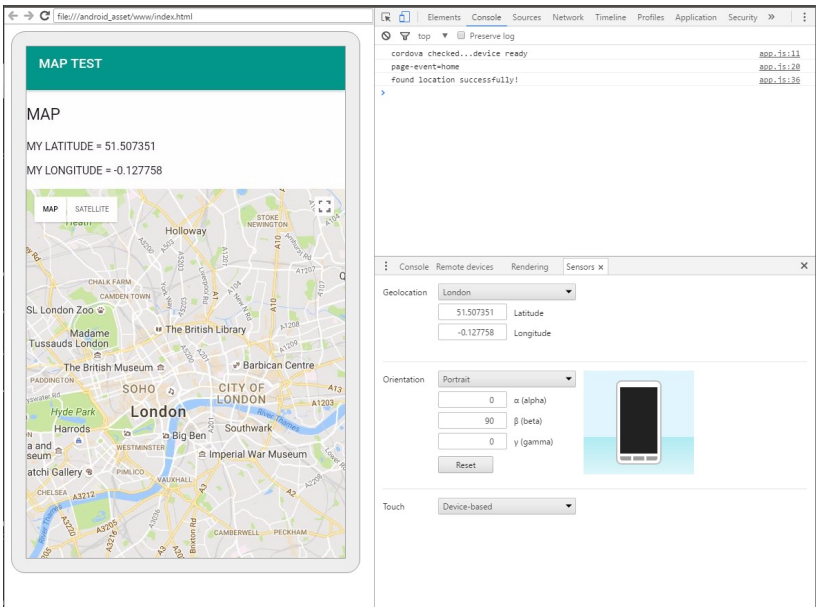
To test our geolocation based app with a connected device, issue the following command for the required Cordova app,

```
cordova run android
```

With the device still connected to our local machine, open the Chrome web browser, and select the 'Developer Tools' option. Then select 'More tools' and 'Remote Devices'. From the list of connected devices, select your current device and click on the 'Inspect' button. e.g.



After selecting the 'Inspect' option, we can then test the current app on the connected device. To test geolocation usage, select 'More Tools' and 'Sensors'. For the 'Geolocation' sensor, we can test various preset and custom locations, e.g.



After updating the latitude and longitude for the 'Geolocation' sensor, simply refresh the 'Developer Tools' window and the new location will be loaded on the connected device. We may use this pattern to test multiple locations for our app.

Android SDK emulator - console usage

To connect to the Android emulator, ensure that the emulator has been launched for the required app, e.g.

```
cordova emulate android
```

After the emulator has launched and loaded successfully, we may then connect using the following command,

```
telnet localhost port-number
```

Modify this command to include the port number for the current running emulator. This will be shown in the title of the emulator window, e.g. Android Emulator - Nexus_5X_API_24:5584

In this example, the port number for telnet connection will be 5584.

After initial connection, an auth token will be created. Details of storage location, and file name for this token, will be shown in the console output. Copy the token from the file, and then authenicate this telnet session with the following command,

```
auth YOUR_AUTH_TOKEN
```

After successfully authenticating, you'll then be able to perform various tasks with the connected emulator and app. Use the following command to check available options,

```
help
```

To use geolocation options, issue the available geo command plus required location data. e.g.

```
geo fix -0.127758 51.507351 4392
```

This command accepts longitude, latitude, and an optional altitude setting.

Further details on emulator usage with the command line can be found in the Android Studio User Guide.