Cordova - Guide - Install & Setup - Windows 10

- Dr Nick Hayward

A brief overview of install and setup for Cordova v.6.x onwards with Windows 10.

Platform specific guides,

- Android install and setup - see `android-platform-guide` document
- iOS install and setup - see `ios-platform-guide` document

Contents

- Intro
- Chocolatey installs
- Cordova create

Intro

We may install Android on Windows 10 for Cordova development using either *Android Studio* or the *SDK command line tools*.

Each Android option requires installation and configuration of Node.js, Git, and Cordova CLI prior to a build and compile of applications.

Chocolatey installs

For Windows, we may consider installing these tools using *Chocolatey*,

- https://chocolatey.org/

We may use this tool to install the required Node.js and Git, plus the following

- openjdk
- android-sdk
- gradle
- ...

We may also use Chocolatey to manage dependencies, updates, and configure install paths.

Using Powershell with *admin* privileges, we may install packages as follows,

```
choco install nodejs
```

So, install and setup with Chocolatey is as follows,

- open Powershell as **admin**
- install Node.js, Git, and Android SDK (n.b. Android Studio may also be installed using Chocolatey)

```
choco install nodejs
choco install git
choco install android-sdk
```

- **n.b.** `android-sdk` will install the SDK command line tools
  - Android Studio may be installed instead but is not essential to build and compile Cordova apps.
- Chocolatey will check for existing JDK whilst installing Android
  - if not available, it will download and install JDK
  - it will also update path settings
- install a local copy of Gradle

```
choco install gradle
```

- switch Powershell to **non-admin** user
- then, install Cordova using NPM
- test Cordova by creating a default app, e.g.

```
cordova create test com.example.test Test
```

- use `sdkmanager` to install any required Android packages, e.g.
  - build-tools
  - platform-tools
  - platforms
  - tools
  - emulator
  - e.g. install `build-tools` and `platforms` as follows,

```
sdkmanager --install "build-tools;29.0.1"
sdkmanager install "platforms;android-28"
```

- test Android SDK install with `cordova build`
- then, setup Android emulator with `avd`
  - install `system-images` using sdkmanager
    - e.g. `sdkmanager --install "system-images;android-28;google_apis;x86_64"`
  - create default avd using `avdmanager`
    - e.g. `avdmanager create avd -n win-avd -k "system-images;android-28;google_apis;x86_64"`
  - test emulator launch
    - e.g. `emulator -avd win-avd -gpu host`
    - starts `win-avd` with default host hardware gpu acceleration

- test Cordova app with emulator
  - e.g. `cordova emulate android`
- test run Cordova app with connected device
  - e.g. `cordova run android`
  - *n.b.* `adb` adb should be configured for connection on device - i.e. device set to usb-debugging & developer settings turned on...
- test app in the cloud, e.g.
  - Firebase test lab (access from firebase console for Android and iOS testing)
  - Visual Studio App Center - https://docs.microsoft.com/en-us/appcenter/sdk/getting-started/cordova

Cordova create

We may start by creating an initial, basic Cordova app.

In the root of a project's directory, issue the following command to create a new, default Cordova project

```
cordova create basic_spa com.example.basic BasicSpa
```

The above Cordova command defines the following,

- local directory name = `basic_spa`
- namespace for Cordova app = `com.example.basic`
- name of app - `BasicSpa`

Each of these values may be customised to match project requirements.

We can then change directory to our newly created Cordova project,

```
cd basic_spa
```

References

- Android SDK command line tools - https://developer.android.com/studio/command-line
- Android Studio - https://developer.android.com/studio/
- Chocolatey - https://chocolatey.org/
- Cordova - https://cordova.apache.org/
- Firebase test lab -
- Git -
- Node.js -
- Visual Studio App Center -