Internet Technologies

From Web Applications to Mobile Applications

University of Cyprus
Department of Computer Science

Web Apps: Mobile-first Design is essential

- Nearly 84% of the global population owns a smartphone and often multiple types of mobile devices. That’s far more than the number of people with access to PCs and laptop.
- Web apps can be optimized for a good mobile (phone/tablet) experience
  - Responsive design: CSS media queries
  - Reduced loading time: light and compact coding files (minified CSS/JS files), compressed images
- Web apps written in HTML/CSS aren't mobile applications; they run in a browser ➔ BUT they can be packaged in a browser-like shell or wrapper, transformed into mobile apps and provided via app stores

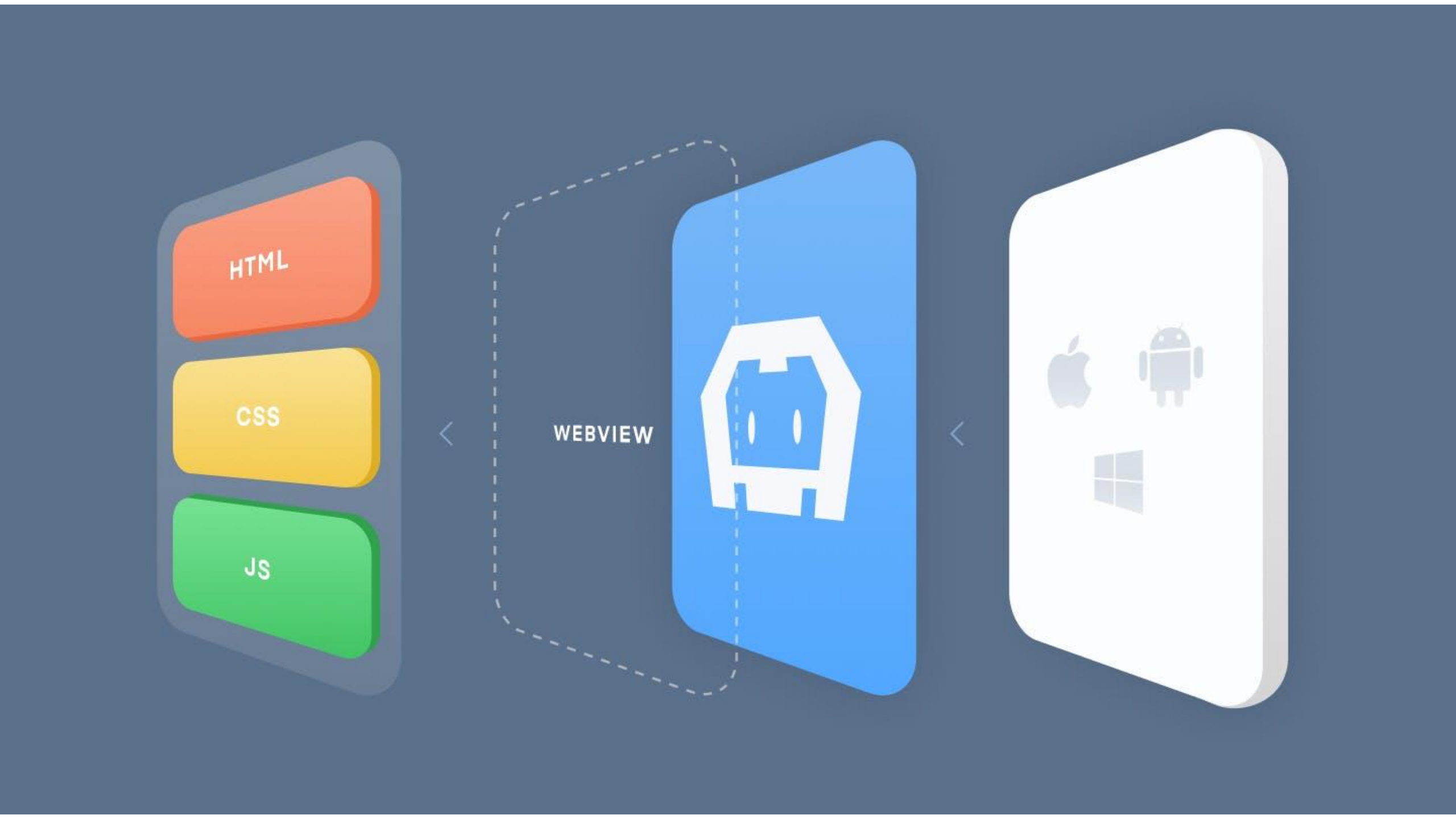
Mobile Applications

- **Native apps**: Software programs built for use on a particular mobile platform (Android, iOS), taking advantage of each platform's resources (e.g. GPS, Bluetooth, Camera). Installed via App stores.
  - Android native apps built using: Java, Kotlin
  - iOS native apps built using: Objective-C, Swift
- **Web apps**: Software developed using HTML, CSS, JavaScript, hosted on web servers, accessed via web browsers over a network. Not installed on mobile phones.
- **Hybrid apps**: Combination of native and web apps. Developed using web technologies (HTML, CSS, JavaScript) but packaged and installed like native apps. Hybrid apps have access to underlying platform APIs to use device resources. Installed via App stores.

Apache Cordova

APACHE CORDOVA™

- Open-source mobile development framework for developing hybrid cross-platform applications using HTML, CSS and JavaScript that can be packaged and installed like native apps
- Cordova takes a web application and renders it within a native WebView component
  - A WebView is an application component that is used to display web content
  - You can think of a WebView as a web browser without any of the standard user interface elements, such as a URL field or status bar

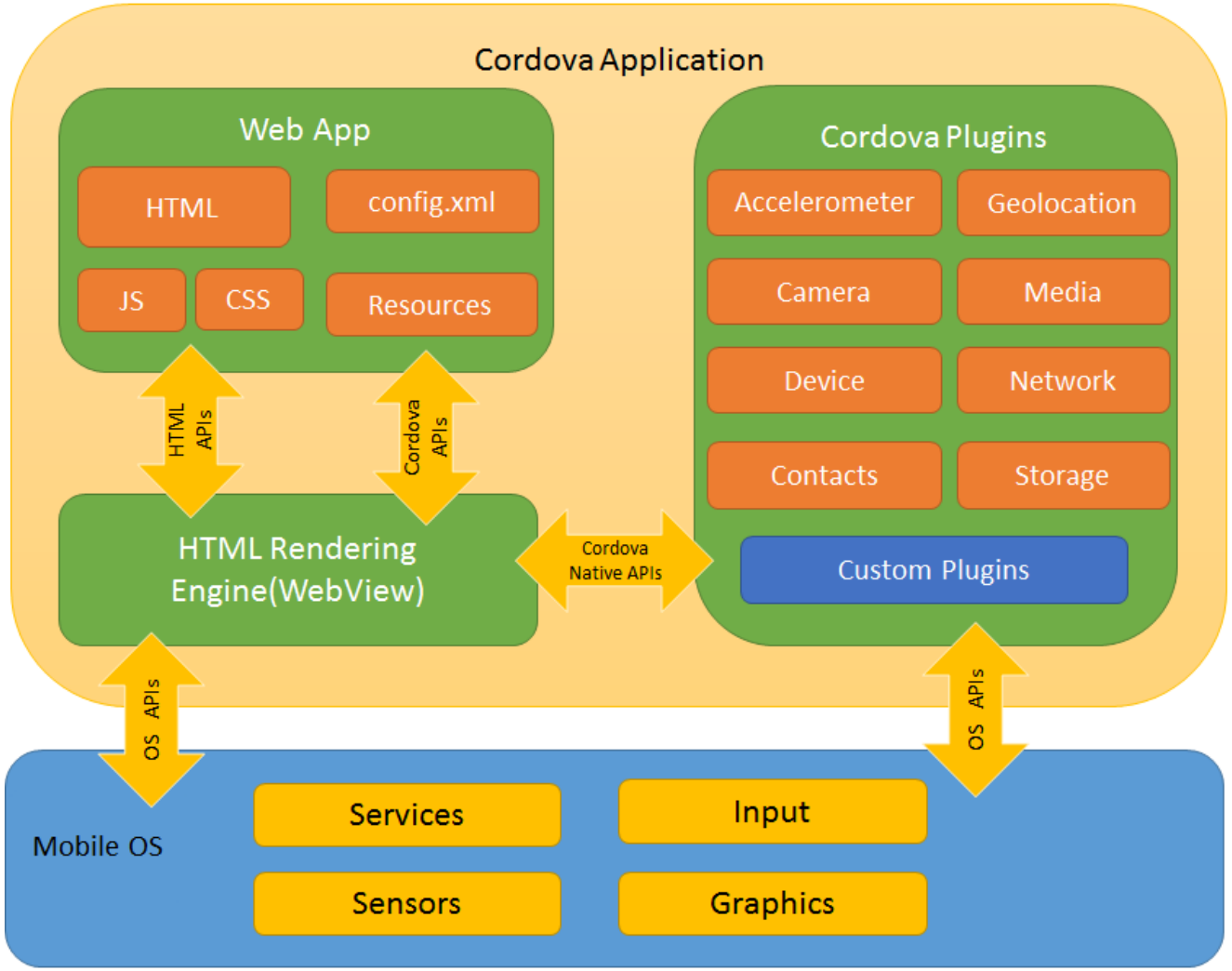

HTML
CSS
JS
WEBVIEW

Apache Cordova

APACHE CORDOVA™

- Typically, web-based applications are executed within a browser without direct access to various hardware and software features on the device
  - Example: a web app running within a browser does not have access to the contact database (names, phone numbers, emails), GPS module etc. of the mobile device
- Cordova provides (a) the basic framework to run a web app within a native application container (WebView) as well as (b) JavaScript APIs to allow access to a wide variety of device features, like the contacts database, etc.
- These capabilities are exposed through the use of a collection of plugins
  - Plugins provide a bridge between a web application running within a WebView and the device's native features

Cordova Architecture



Apache Cordova is used by:

- mobile developers eager to develop applications across more than one platform (Android, iOS), without having to re-implement it with each platform's language and tool set
- web developers that want to deploy existing web apps in various app store portals
- mobile developers interested in mixing native application components with a *WebView* that can access device-level APIs, or interested to develop plugin interfaces between native and WebView components

Development path

- Cordova provides you two basic workflows to create a mobile app:
  - Cross-platform (command-line interface - CLI) workflow (recommended)
    - Preferable when building an app to run on as many different mobile operating systems as possible, with little need for platform-specific development
  - Platform-centered workflow
    - Preferable when building an app for a single platform and need to be able to modify it at a lower level (modify the project within the SDK)

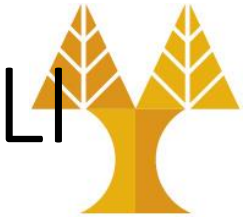
System Requirements*

- Install Java Development Kit (JDK) >= JDK11
  - set the JAVA_HOME environment variable to the location of your JDK installation
- Install Gradle (binary-only version)
  - add the path to the Gradle's binary directory to your path environment variable
- Install Android Studio & add SDK packages
  - set the ANDROID_SDK_ROOT environment variable to the location of the Android SDK installation

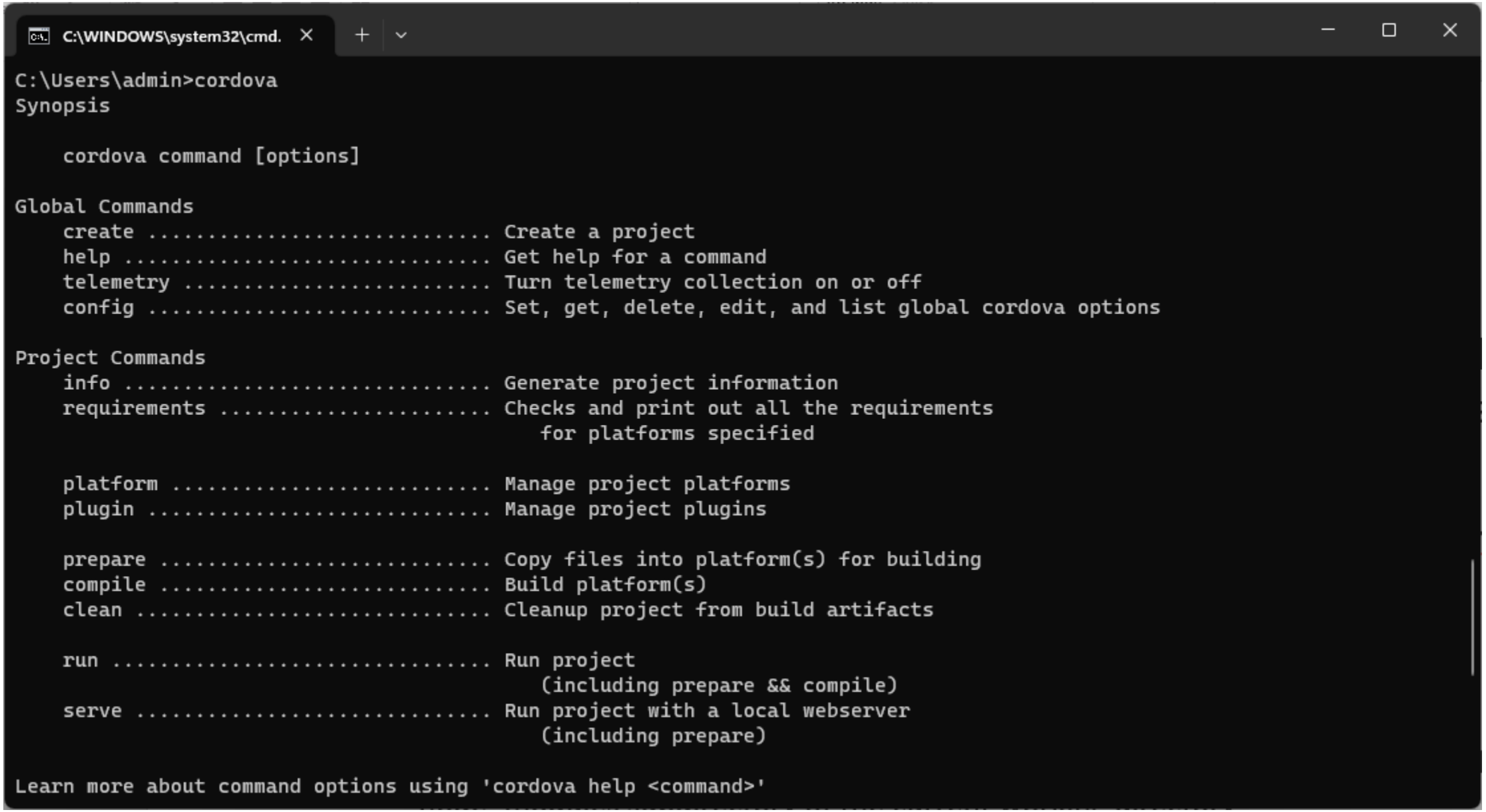
Please refer to: https://cordova.apache.org/docs/en/latest/guide/platforms/android/#system-requirements

(*) for Android application development on a Windows machine. For macOS and Linux see here

Cross-platform workflow using Cordova CLI



- Download and install Node.js (if not already installed). On installation you should be able to invoke node and `npm` on your command line (CMD on Windows, terminal on macOS and Linux).
- Install the cordova module using `npm` utility of Node.js. The cordova module will automatically be downloaded by the npm utility.
  - On macOS and Linux terminal: `sudo npm install -g cordova`
  - On Windows CMD: `npm install -g cordova`
    - The -g flag above tells npm to install cordova globally. Otherwise it will be installed in the node_modules subdirectory of the current working directory
- Following installation, you should be able to run cordova on the command line with no arguments and it should print help text.



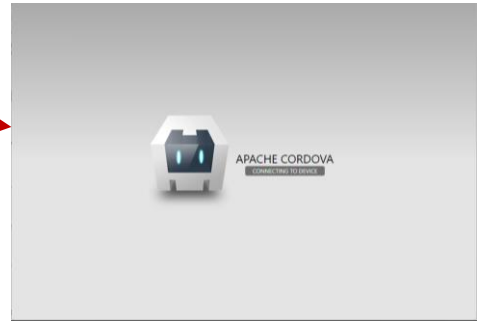
If you get the message “'cordova' is not recognized as an internal or external command, operable program or batch file.” see here.

Create Cordova App

- Go to the directory where you want to maintain your source code, and create a Cordova project:

```
cordova create myapp cy.ac.ucy.cs.epl425 MyAndroidApp
```

- This creates the required directory structure for your cordova app within `myapp/`. The name of the app (in mobile device) will be `MyAndroidApp`. By default, the cordova create script generates a skeletal web-based application whose root folder is `myapp/www/`
  - The web-based application folder structure:
    - myapp/www/index.html
    - myapp/www/css/index.css
    - myapp/www/js/index.js
    - myapp/www/img/logo.jpg



Default web app index.html file

Convert existing Web App to Mobile App: Convert your HW1 to Android App

- Remove everything from `myapp/www`
- Place all files (except .php) of your HW1 within `myapp/www`
  - Last 5 requests functionality can be served by the php file running on your departmental account
    - Edit your JavaScript file to use absolute (full) URL to your php file(s) such as https://www.cs.ucy.ac.cy/~<username>/epl425/<yourfilename>.php
- Optimizations:
  - keep content locally on mobile device: you can avoid links to CDN-based CSS/JS files for faster application loading time and less data exchange over the network
    - You can download Bootstrap CSS/JS files locally and replace links in the <head> section of the index.html

Add Platforms to your Cordova App

- All subsequent commands need to be run within the project's directory
  - `cd myapp`
- Add the platforms that you want to target your app. We will add the 'android' and 'browser'* platform and ensure they get saved to config.xml and package.json:
  - `cordova platform add android`
  - `cordova platform add browser`
- To check your current set of platforms:
  - `cordova platform ls`

(*) Adding Cordova's browser platform to a hybrid app allows us to run and debug apps using the regular web browser without deployment to a device or server. However, if the plugins we're using don't support the browser platform, then they won't be available at runtime and we would have to code around that in our app logic.

Running commands to add/remove platforms affects the contents of the project's platforms directory, where each specified platform appears as a subdirectory.
Note: When using the CLI to build your application, you should not edit any files in the /platforms/ directory. The files in this directory are routinely overwritten when preparing applications for building, or when plugins are re-installed.

Check prerequisites

- To **build** and **run** apps, you need to install SDKs for each platform you wish to target. Alternatively, if you are using browser for development you can use browser platform which does not require any platform SDKs.
- To check if you satisfy requirements for building the platform:
  - `cordova requirements`
- Note: make sure Gradle version is compatible with JDK version
  - In this lab we used Gradle 8.1 and JDK 15 to build / run the Cordova App

Build the App

- This step builds the app for a specified platform so we can run it on mobile device or emulator
- Run the following command to build the project for all platforms:
  - `cordova build`
- You can optionally limit the scope of each build to specific platforms - 'android' in this case:
  - `cordova build android`
    - This process creates the .apk file in myapp\platforms\android\app\build\outputs\apk\debug\app-debug.apk
    - Copy .apk file to your Android device, double click to install and test

Run the App on a real device

- If you want to use a real device for testing, connect your mobile device to your development machine via USB cable, enable USB debugging on your Android device(*) and execute the command:
  - `cordova run android --device`
    - If you get errors related to missing “Android build tools”, you will need to install them via Android Studio (see here).
    - If you get an error message like "No devices found" then make sure that you have developer mode and USB Debugging enabled properly on the device
  - This process, builds the app, transfers .apk to mobile device, installs it and launches it

(*)

- **Enable Developer options (if System → Developer Options does not appear within Settings) - FIRST TIME ONLY**
- **Enable USB debugging on your device**

Create application icon

- Icons help your users identify your app
- Create custom icons for your app using Image Asset Studio (Android Studio) and Xcode (iOS)
- Third-party services are also available, such as IconKitchen (Android, iOS), Icon Themer (iOS), etc.

- Example:
  - Create icon using IconKitchen, download .zip file, extract it, and place android folder next to myapp/www/ folder
  - Configure application to use the icon (see next slide)

Modify Mobile App Configuration

- config.xml (application version, application name, application icon, etc.)

```
<?xml version='1.0' encoding='utf-8'?>
<widget id="cy.ac.ucy.cs.epl425" version="1.0.0" android-versionCode="1" xmlns="http://www.w3.org/ns/widgets" xmlns:cdv="http://cordova.apache.org/ns/1.0">
    <name>Tourist Destination</name>
    <description>Apache Cordova for Tourist Destination</description>
    <author email="antoniou.pavlos@ucy.ac.cy" href="https://www.cs.ucy.ac.cy/~csp5pa1">
        Pavlos Antoniou
    </author>
    <content src="index.html" />
    <allow-intent href="http://*/*" />
    <allow-intent href="https://*/*" />

    <platform name="android">
     <resource-file src="android/res/mipmap-anydpi-v26/ic_launcher.xml" target="/app/src/main/android/res/mipmap-anydpi-v26/ic_launcher.xml" />
     <icon background="android/res/mipmap-mdpi/ic_launcher_background.png" density="mdpi" foreground=" android/ res/mipmap-mdpi/ic_launcher_foreground.png" />
     <icon background="android/res/mipmap-hdpi/ic_launcher_background.png" density="hdpi" foreground=" android/ res/mipmap-hdpi/ic_launcher_foreground.png" />
     <icon background="android/res/mipmap-xhdpi/ic_launcher_background.png" density="xhdpi" foreground=" android/ res/mipmap-xhdpi/ic_launcher_foreground.png" />
     <icon background="android/res/mipmap-xxhdpi/ic_launcher_background.png" density="xxhdpi" foreground=" android/ res/mipmap-xxhdpi/ic_launcher_foreground.png" />
     <icon background="android/res/mipmap-xxxhdpi/ic_launcher_background.png" density="xxxhdpi" foreground=" android/ res/mipmap-xxxhdpi/ic_launcher_foreground.png" />
    </platform>
</widget>
```

APPLICATION VERSION (needed by Google Playstore)

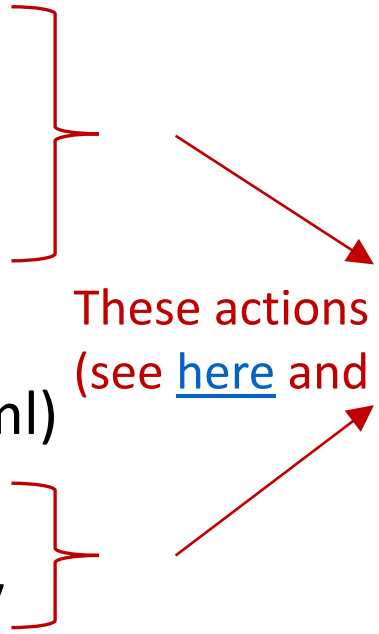
APPLICATION NAME

APPLICATION ICON

- Re-run application to see changes: `cordova run android --device`

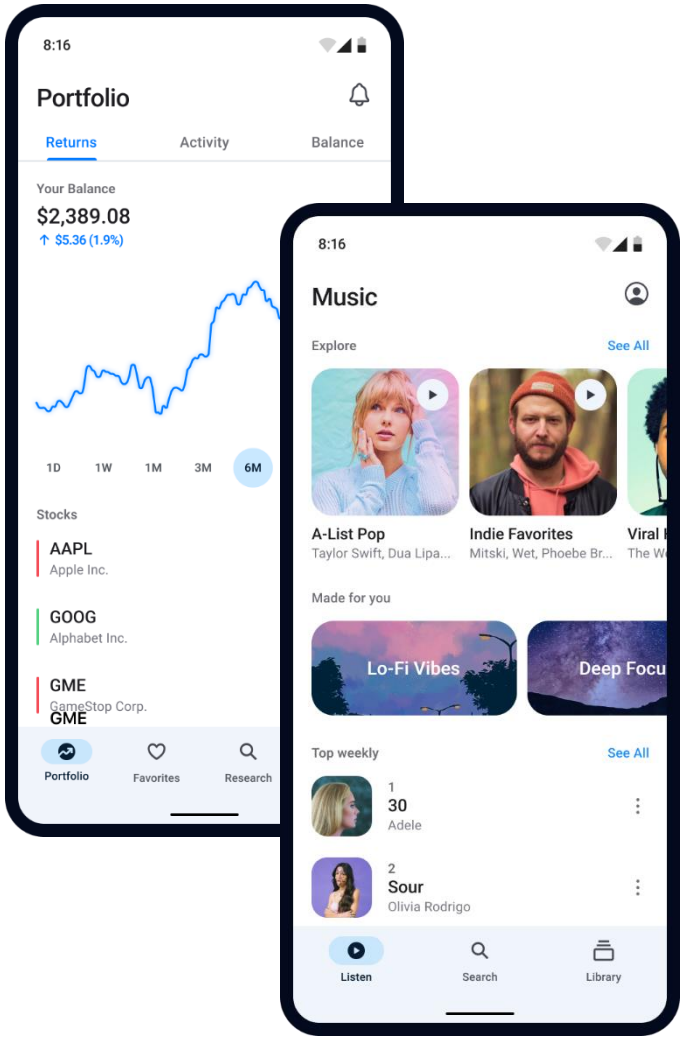
Publish Cordova App to App Store

- Publish your app for the first time in Google Playstore (see here for info)
  1. Set version number (in config.xml)
  2. Generate upload key and keystore
  3. Generate (build) apk in **release mode**
  4. Sign your app
- Update your App
  1. Increase version number (in config.xml)
  2. Generate (build) apk in **release mode**
  3. Sign your apk with the old upload key



These actions can be performed in Android Studio (see here and Appendix)

Ionic Framework

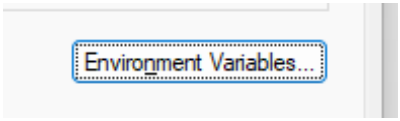


- Ionic Framework is an open-source mobile **UI toolkit** for building modern, high quality cross-platform mobile apps with popular front-end JavaScript frameworks (Angular, React, Vue) or without a JavaScript framework
  - No need to have a web app to convert to mobile app
- Provides mobile-based UI components such as menus, sliders, alerts, checkboxes, radio buttons, input elements, modals, cards, datetime pickers
  - No need to use bootstrap
- Installation: `npm i -g @ionic/cli`

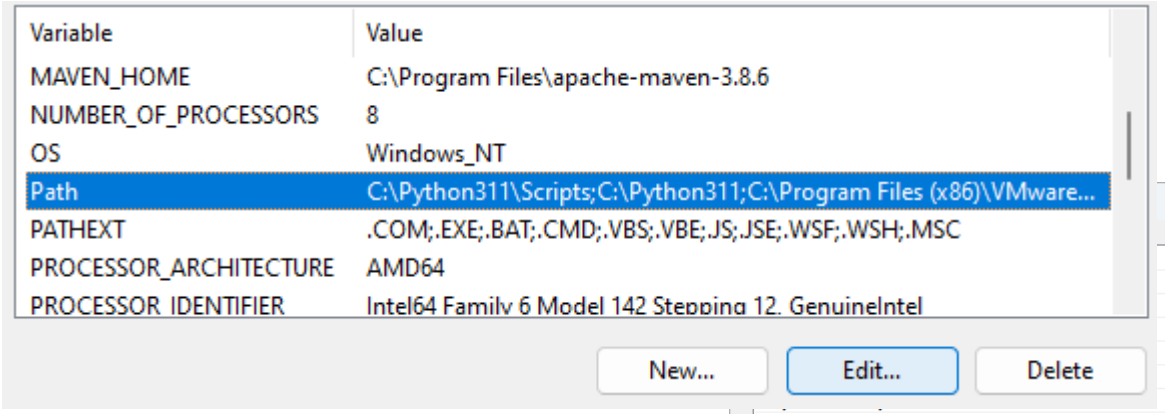
Appendix

Add NPM roaming data to your PATH

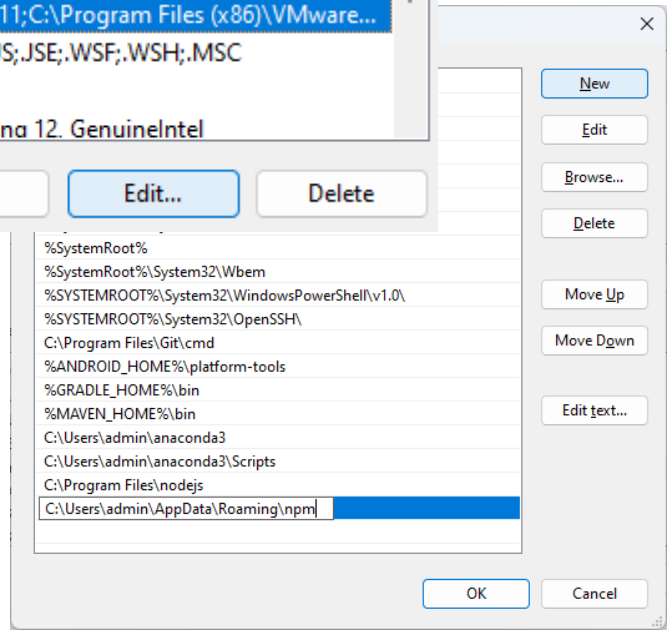
- My PC → Right Click → Properties → Advance System Settings → Environment Variables button



- Click on Path and then edit

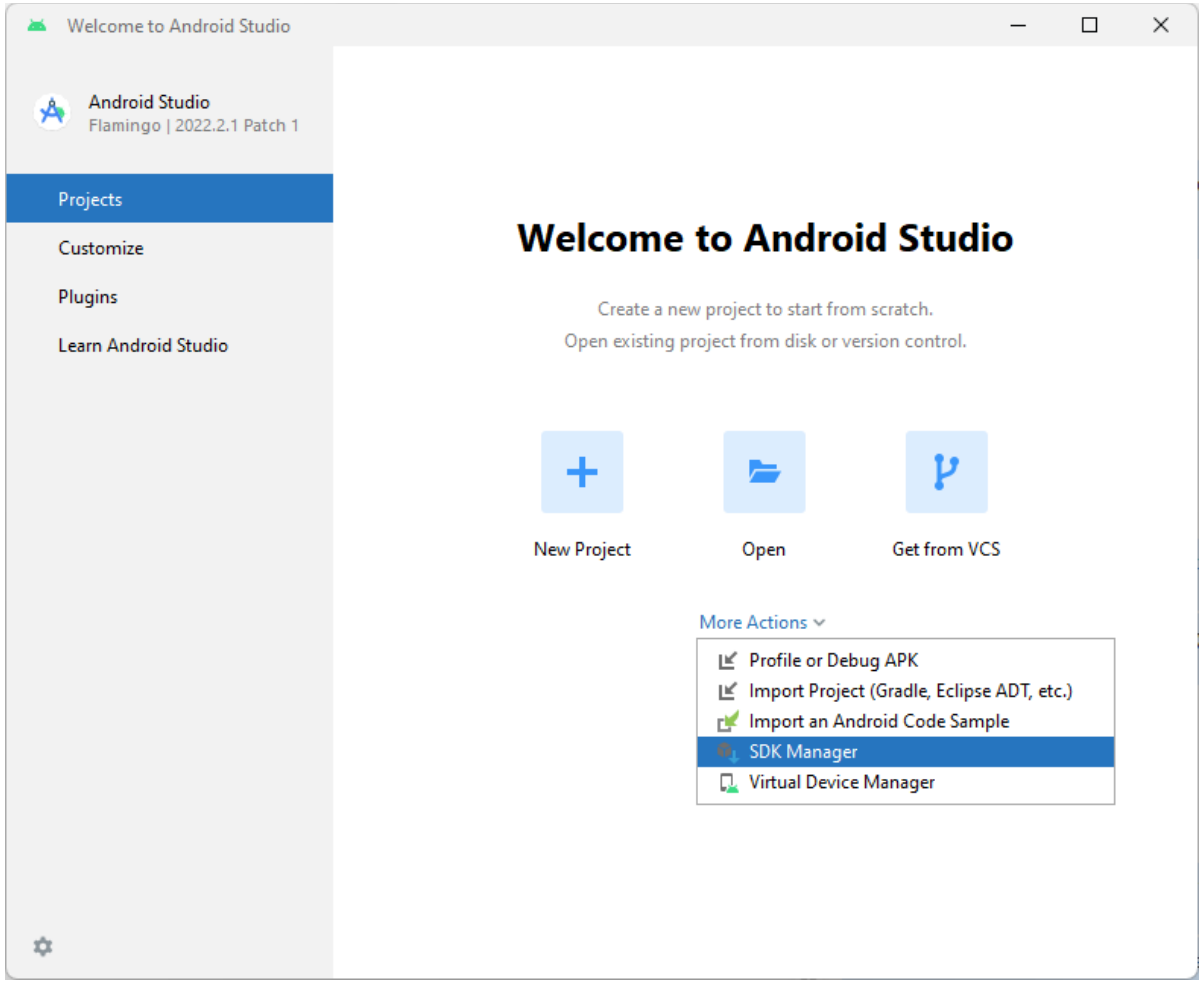


- Click New and enter c:\users\YourUserName\AppData\Roaming\npm\
  - (replace admin with the name of your user profile)



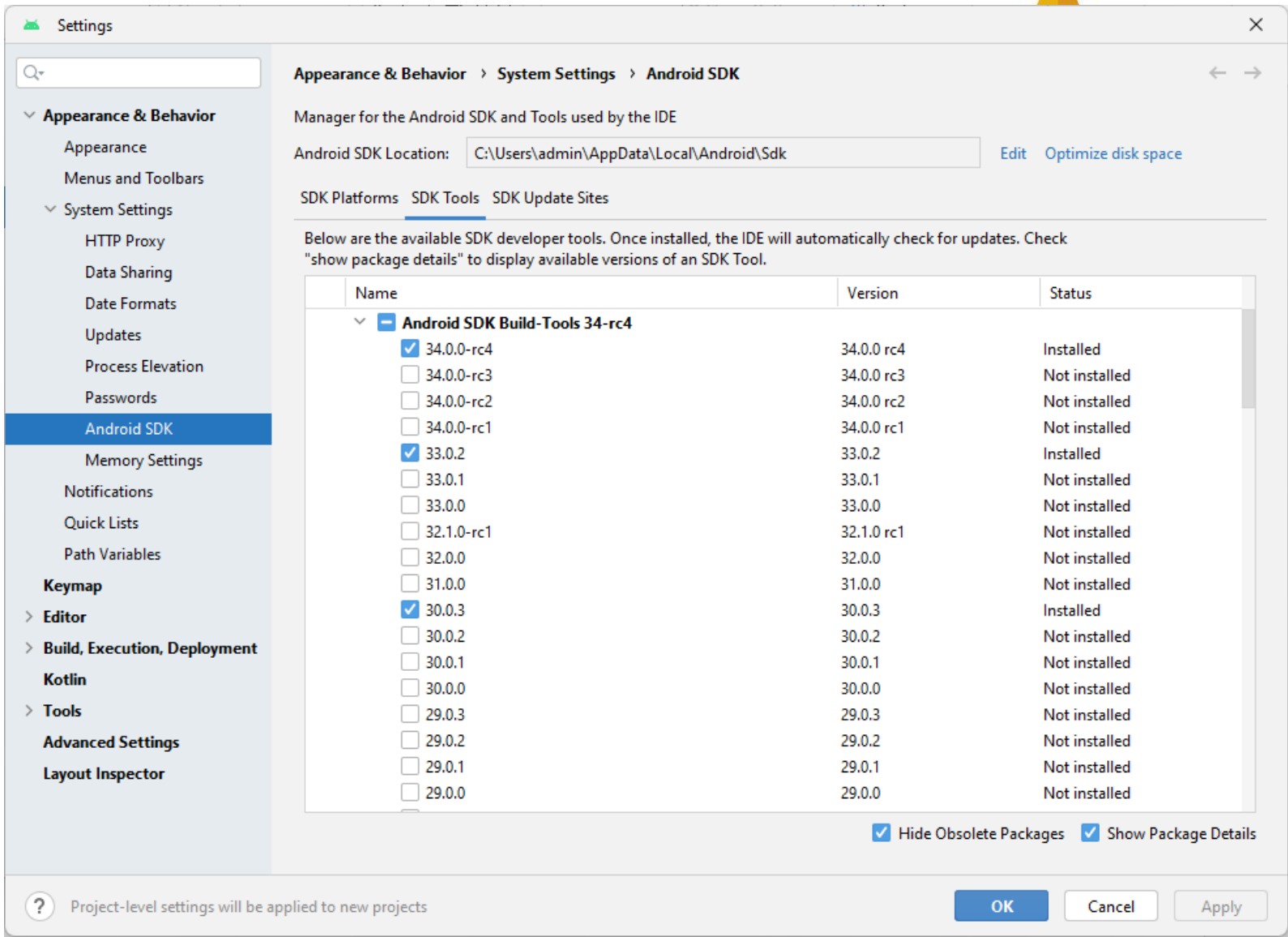
Install Android Build Tools

- Launch Android Studio
- From the home page, under More Actions, select SDK Manager



Install Android Build Tools

- Click on the “Show Package Details” on the bottom left side
- In the SDK Tools tab, open the dropdown menu under Android SDK Build, select the required version of the Android Tools, and then select Apply to start downloading

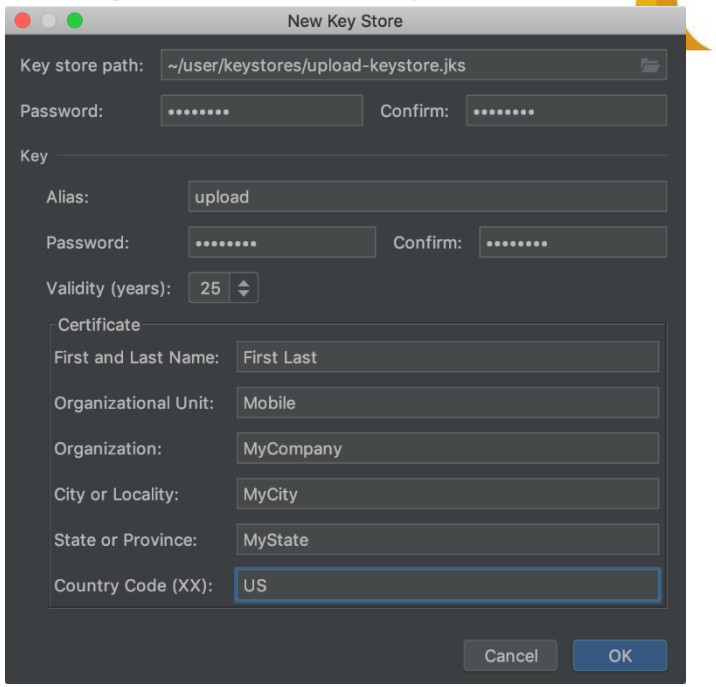


Import Cordova app to Android Studio

- Launch Android Studio
- Choose **File -> New -> Import Project**
- Select the Android platform directory in your project (<your-project>/platforms/android) and Press Ok
- After that the Gradle build will begins and let it to be finish
- Once it finishes importing, you should be able to build and run the app directly from Android Studio.

Sign your app for release to Google Play

- FIRST TIME: Generate an upload key and keystore → https://developer.android.com/studio/publish/app-signing#generate-key



- Sign your app with your upload key → https://developer.android.com/studio/publish/app-signing#sign_release