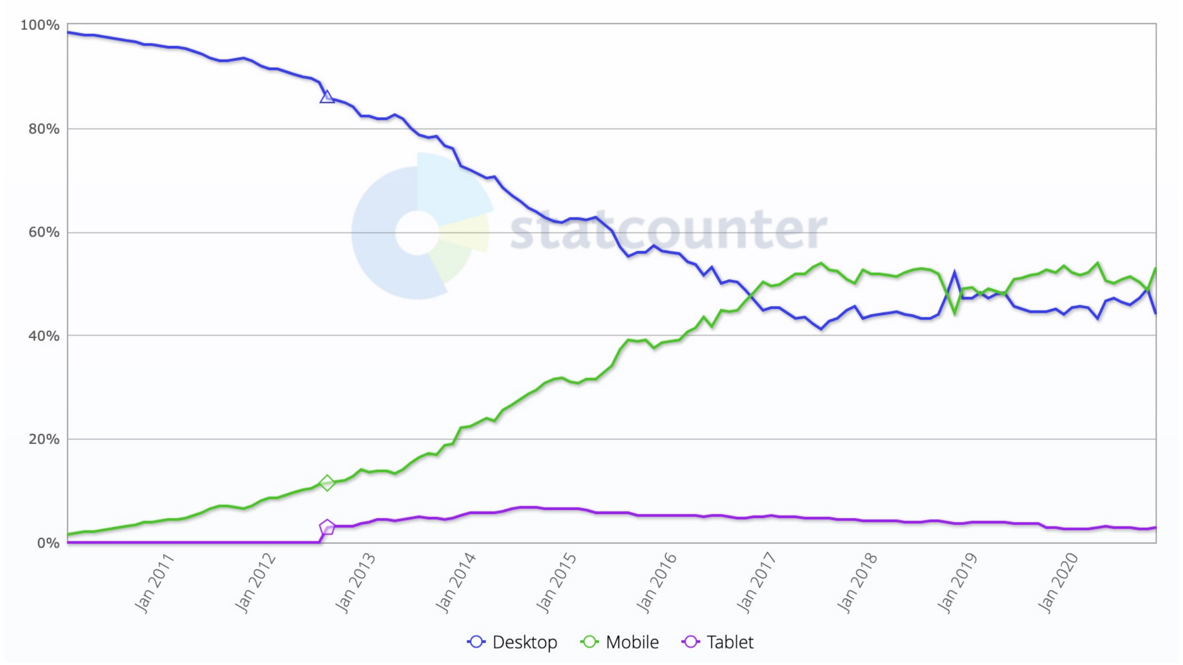
Tecnologie e Applicazioni Web

Mobile Apps

Filippo Bergamasco (filippo.bergamasco@unive.it)
http://www.dais.unive.it/~bergamasco/
DAIS – Ca' Foscari University of Venice
Academic Year: 2021/2022

Mobile web

Considering the massive diffusion of mobile devices, it is nowadays crucial the market share of the users accessing the web through a smartphone.



Mobile web

Usually we discuss about “mobile web”, but there is no actual difference to the “traditional” web we access from our computers.

Every smartphone now has a sophisticated browser as good as its desktop counterpart. (Full HTML, CSS, JavaScript support)

Mobile web

The main difference of a mobile web app relies in the user interaction:

- Touch-based interface, usually without physical keyboard
- Displays with different aspect ratio, completely different to a typical desktop screen (portrait vs. landscape)

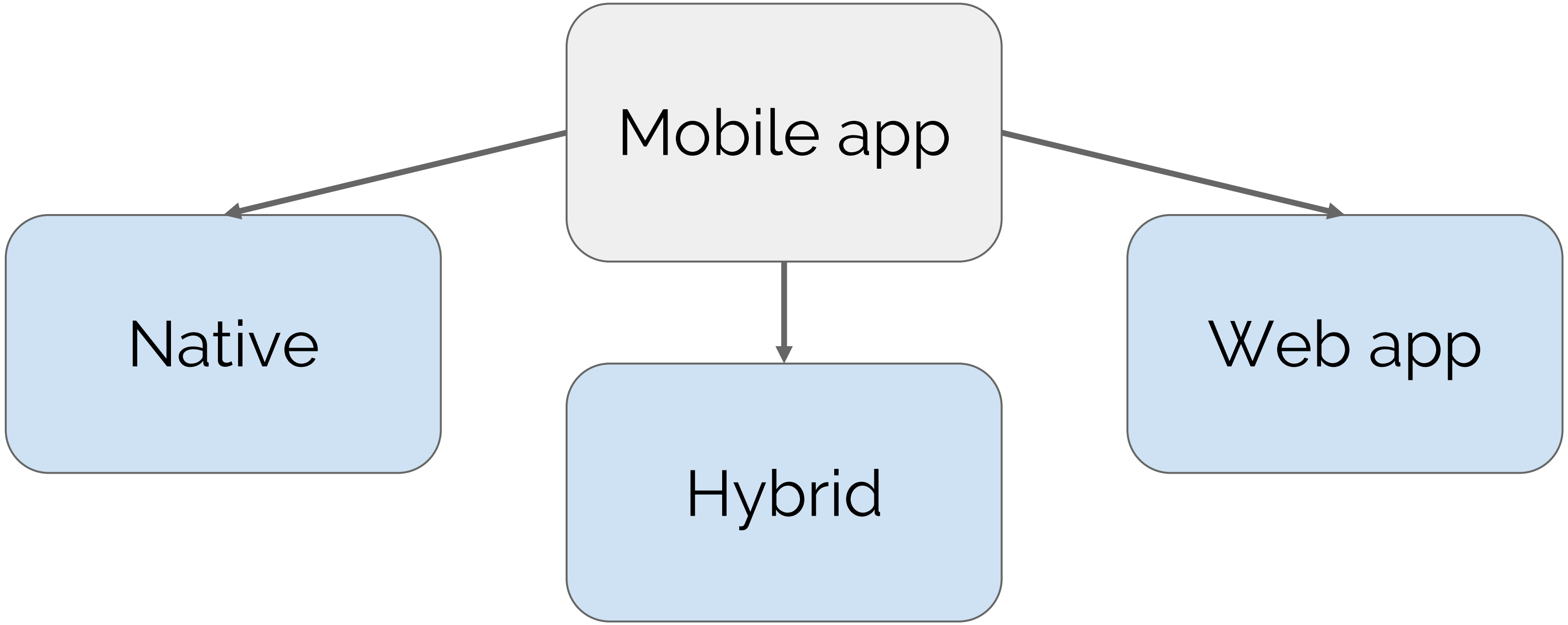
Mobile web

The main difference of a mobile web app relies in the user interaction:

- Reduced resolution and physical size
- Paradigms typical of the mobile ecosystem (navigation buttons, toolbars, pull-to-refresh, etc)

Problem: Great variability of mobile devices!

App development: what to choose?



Native Apps

A native app is written directly using the SDK of a certain platform, usually with the “official” language in which the API are provided.

Ex: Java for Android SDK, Objective-C or Swift for iOS SDK, etc.

Native Apps

Advantages:

- Fast and usually smaller
- Complete access to all the APIs (and functions). Access to device hardware (accelerometers, camera, GPS, etc.)
- Look-and-feel coherent with built-in apps
- Installation trough the official store

Native Apps

Drawbacks:

- Multi-platform apps require a complete porting, usually forcing a full rewrite of the entire code
- Complex development
- More expensive, especially for apps designed just to present (almost) static information (ex: a restaurant menu)

Mobile Web Apps

A WebApp Mobile is a web application (HTML5) that is visualized and executed in the system browser of a user device

Ex: Safari on iOS / Chrome on Android

Mobile Web Apps

Advantages:

- Use the same technologies of a desktop web application
- No installation needed. The developer may just provide the application URL
- Directly multi-platform. Differences among the devices are handled by the browser (development platform is essentially the web browser itself)

WebApp mobile

Drawbacks:

- Different look-and-feel compared to native apps
- Impossible to use API not standardized in HTML5 (ex. No access to system phonebook)
- Limited functionalities due to browser's security restrictions
- In general slower and with higher memory usage

Hybrid WebApp

An Hybrid WebApp is a web application (HTML5) encapsulated inside a native container providing a full-screen browser interface.

Ex: Applications developed with Apache Cordova or Ionic framework

Hybrid WebApp

Advantages:

- Same feeling of using a native app. Installation is performed via app store
- Developed with standard web technologies:
  - Porting of a desktop app highly simplified
  - Easy to develop a multi-platform application
- The native app incapsulating the code provides interface to access native APIs

Hybrid WebApp

Drawbacks:

- Not all the available APIs are exported by the native container
- Sometimes slower and with higher memory usage (the native container must still provide a full-featured web browser)
- Requires some customization for each platform

Hybrid app development

Apache Cordova is an open-source framework open-source to develop hybrid mobile apps using web technologies (HTML5)

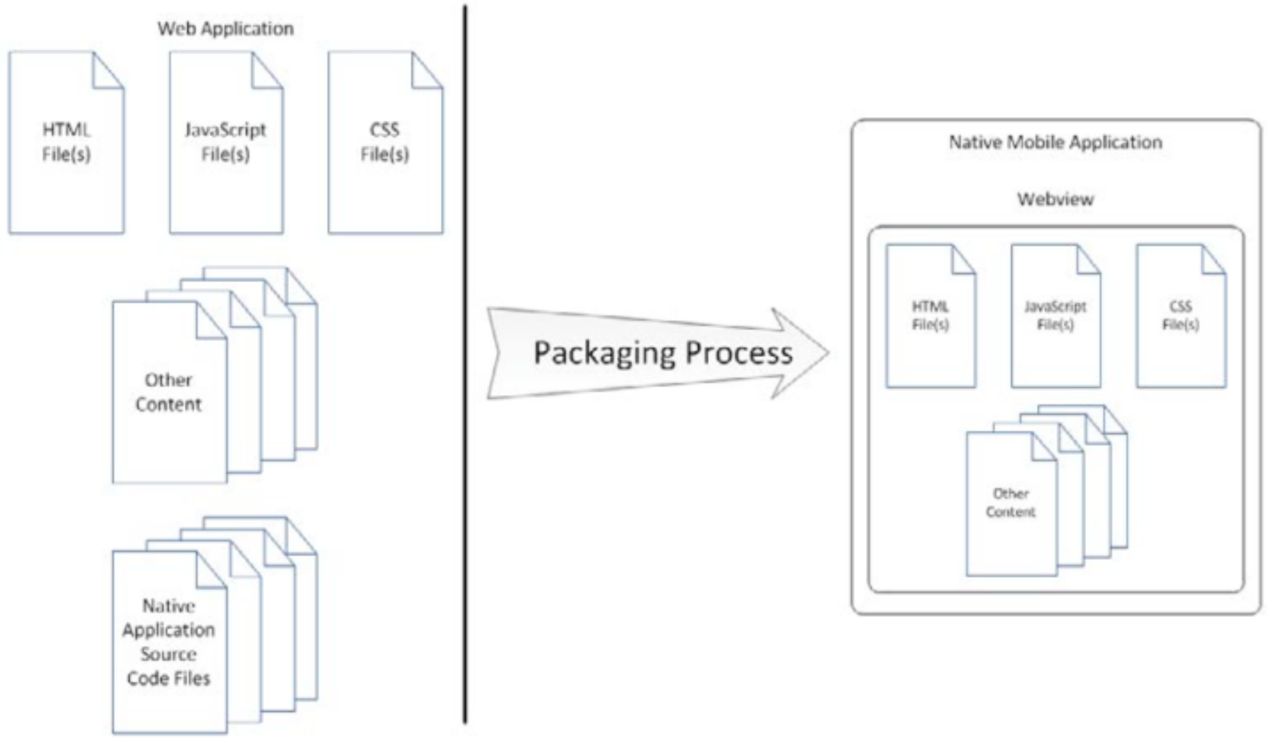
https://cordova.apache.org/

Apache Cordova

Comprises:

1. The native container source code for each supported platform. The container displays a full-screen browser to the user running our HTML5 application
2. APIs to allow the web application to natively access system functionalities
3. A set of tools to manage the build process, plugins, emulators, etc.

Apache Cordova



Cordova CLI

Cordova provides a command line tool (written in JavaScript) distributed via npm.

- `npm install -g cordova`

The –g flag installs the cordova package «globally», so it can be used as any other system tool.

Creating a new application

```
$ cordova create <project_path> <id>
<name>
```

For example:

```
$ cordova create ./myfirstapp it.unive.MyFirstApp
myfirstapp
```

Project structure

```
.
├── config.xml
├── hooks
│   └── README.md
├── platforms
├── plugins
└── www
    ├── css
    │   └── index.css
    ├── img
    │   └── logo.png
    ├── index.html
    └── js
        └── index.js

7 directories, 6 files
```

Global configuration file to control the application behaviour.

Contains information common to all platforms and platform-specific settings

Project structure

```
.
├── config.xml
├── hooks
│   └── README.md
├── platforms
├── plugins
└── www
    ├── css
    │   └── index.css
    ├── img
    │   └── logo.png
    ├── index.html
    └── js
        └── index.js

7 directories, 6 files
```

Contains the native application code to be built.

Project structure

```
.
├── config.xml
├── hooks
│   └── README.md
├── platforms
├── plugins
└── www
    ├── css
    │   └── index.css
    ├── img
    │   └── logo.png
    ├── index.html
    └── js
        └── index.js

7 directories, 6 files
```

Contains Cordova plugins that can be installed on-demand

Project structure

```
.
├── config.xml
├── hooks
│   └── README.md
├── platforms
├── plugins
└── www
    ├── css
    │   └── index.css
    ├── img
    │   └── logo.png
    ├── index.html
    └── js
        └── index.js

7 directories, 6 files
```

Contains the HTML+CSS+JavaScript files of our web application

Building

The project structure is independent to a particular mobile platform, that must be installed separately:

```
$ cordova add platform android
$ cordova build android
```

Android Platform

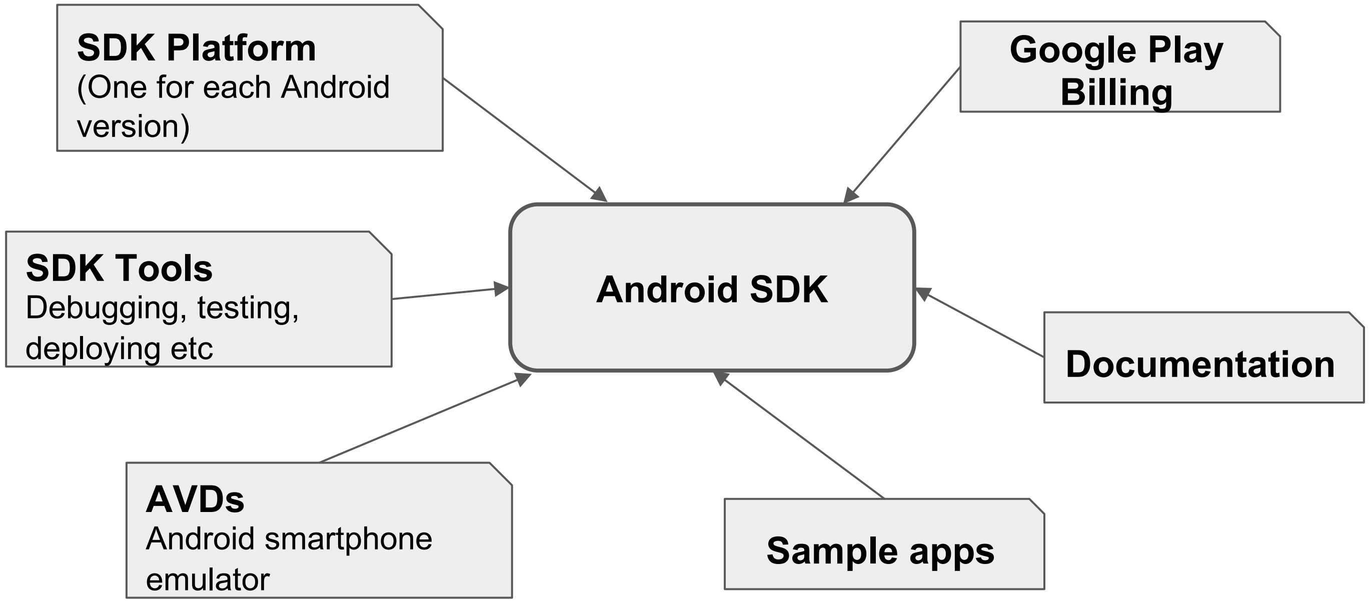
To develop applications for the Android platform it is necessary to download the Android SDK bundled with the Android Studio IDE

https://developer.android.com/studio

Build process is managed by Cordova, so Android Studio IDE will be sued just to manage the various SDKs and AVDs

Android SDK

Android SDK comprise all the tools needed to develop Android applications


SDK Platform
(One for each Android version)
Google Play Billing
SDK Tools
Debugging, testing, deploying etc
Android SDK
Documentation
AVDs
Android smartphone emulator
Sample apps

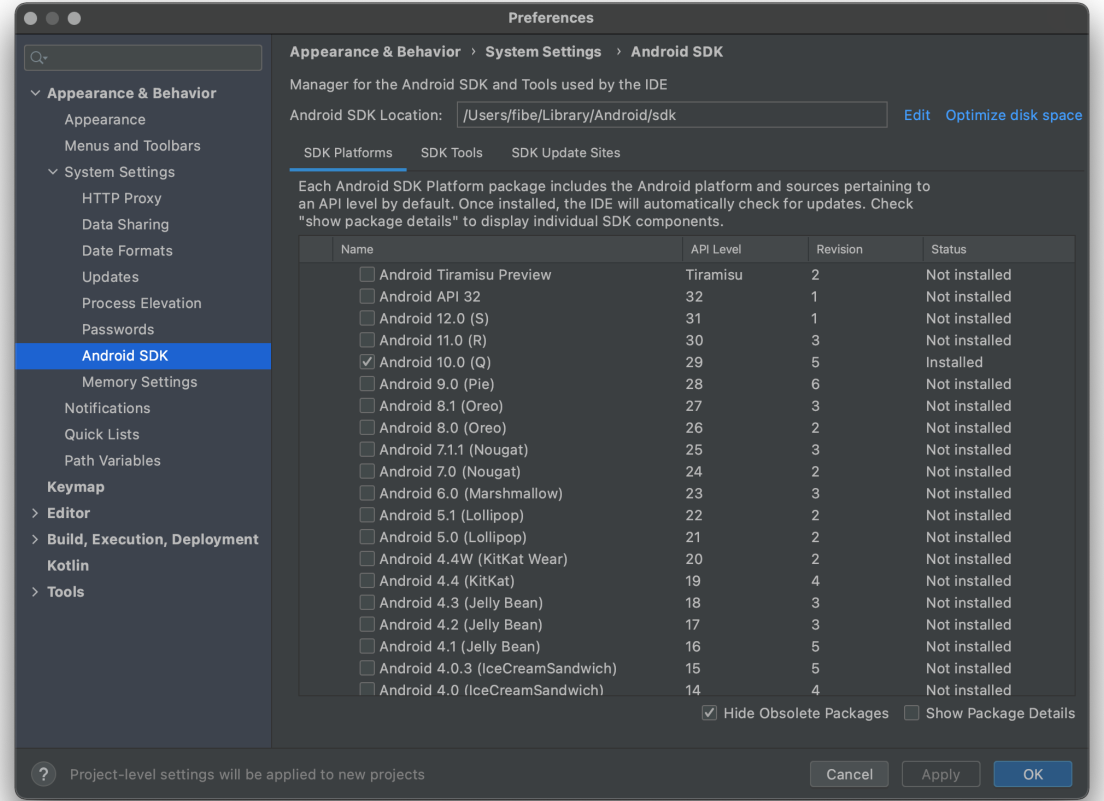
Android SDK Manager

Through the Android Studio's SDK manager it is possible to install:

- One or more SDK Platform (library) for each Android version
- SDK e Platform tools to manage the build and sign process
- Additional components like the Intel x86 Emulator Accelerator (HAXM), Google USB drivers for debugging, Google Play Services, etc.

It is also possible to execute a tool named AVD Manager to create new virtual machines.

SDK Platforms



Appearance & Behavior › System Settings › Android SDK

Manager for the Android SDK and Tools used by the IDE

Android SDK Location: /Users/fibe/Library/Android/sdk   Edit   Optimize disk space

SDK Platforms | SDK Tools | SDK Update Sites

Each Android SDK Platform package includes the Android platform and sources pertaining to an API level by default. Once installed, the IDE will automatically check for updates. Check "show package details" to display individual SDK components.

	Name	API Level	Revision	Status
☐	Android Tiramisu Preview	Tiramisu	2	Not installed
☐	Android API 32	32	1	Not installed
☐	Android 12.0 (S)	31	1	Not installed
☐	Android 11.0 (R)	30	3	Not installed
✓	Android 10.0 (Q)	29	5	Installed
☐	Android 9.0 (Pie)	28	6	Not installed
☐	Android 8.1 (Oreo)	27	3	Not installed
☐	Android 8.0 (Oreo)	26	2	Not installed
☐	Android 7.1.1 (Nougat)	25	3	Not installed
☐	Android 7.0 (Nougat)	24	2	Not installed
☐	Android 6.0 (Marshmallow)	23	3	Not installed
☐	Android 5.1 (Lollipop)	22	2	Not installed
☐	Android 5.0 (Lollipop)	21	2	Not installed
☐	Android 4.4W (KitKat Wear)	20	2	Not installed
☐	Android 4.4 (KitKat)	19	4	Not installed
☐	Android 4.3 (Jelly Bean)	18	3	Not installed
☐	Android 4.2 (Jelly Bean)	17	3	Not installed
☐	Android 4.1 (Jelly Bean)	16	5	Not installed
☐	Android 4.0.3 (IceCreamSandwich)	15	5	Not installed
☐	Android 4.0 (IceCreamSandwich)	14	4	Not installed

✓ Hide Obsolete Packages   ☐ Show Package Details

Project-level settings will be applied to new projects

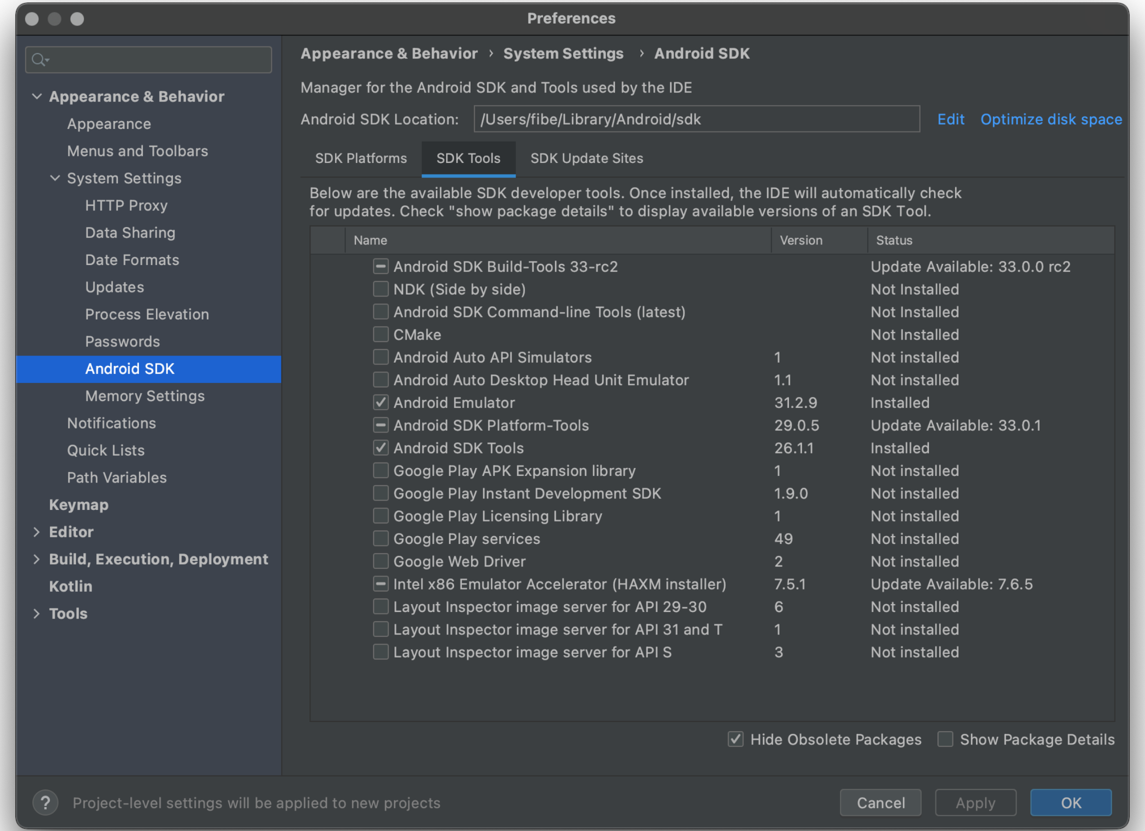
Versioni Android

Each Android version is characterized by:

- A codename
- An alphanumeric version number
- An **API level**: numerical id specifying the API version that can be used by the apps

Each version number corresponds to a certain API level. The codename can span multiple versions (in case of minor revisions)

SDK Tools



Preferences

- Appearance & Behavior
  - Appearance
  - Menus and Toolbars
  - System Settings
    - HTTP Proxy
    - Data Sharing
    - Date Formats
    - Updates
    - Process Elevation
    - Passwords
    - **Android SDK**
    - Memory Settings
  - Notifications
  - Quick Lists
  - Path Variables
- Keymap
- Editor
- Build, Execution, Deployment
- Kotlin
- Tools

Appearance & Behavior › System Settings › Android SDK

Manager for the Android SDK and Tools used by the IDE

Android SDK Location: /Users/fibe/Library/Android/sdk   Edit   Optimize disk space

SDK Platforms | **SDK Tools** | SDK Update Sites

Below are the available SDK developer tools. Once installed, the IDE will automatically check for updates. Check "show package details" to display available versions of an SDK Tool.

	Name	Version	Status
−	Android SDK Build-Tools 33-rc2		Update Available: 33.0.0 rc2
☐	NDK (Side by side)		Not Installed
☐	Android SDK Command-line Tools (latest)		Not Installed
☐	CMake		Not Installed
☐	Android Auto API Simulators	1	Not installed
☐	Android Auto Desktop Head Unit Emulator	1.1	Not installed
✓	Android Emulator	31.2.9	Installed
−	Android SDK Platform-Tools	29.0.5	Update Available: 33.0.1
✓	Android SDK Tools	26.1.1	Installed
☐	Google Play APK Expansion library	1	Not installed
☐	Google Play Instant Development SDK	1.9.0	Not installed
☐	Google Play Licensing Library	1	Not installed
☐	Google Play services	49	Not installed
☐	Google Web Driver	2	Not installed
−	Intel x86 Emulator Accelerator (HAXM installer)	7.5.1	Update Available: 7.6.5
☐	Layout Inspector image server for API 29-30	6	Not installed
☐	Layout Inspector image server for API 31 and T	1	Not installed
☐	Layout Inspector image server for API S	3	Not installed

[✓] Hide Obsolete Packages   [ ] Show Package Details

Project-level settings will be applied to new projects

Android Virtual Device (AVD)

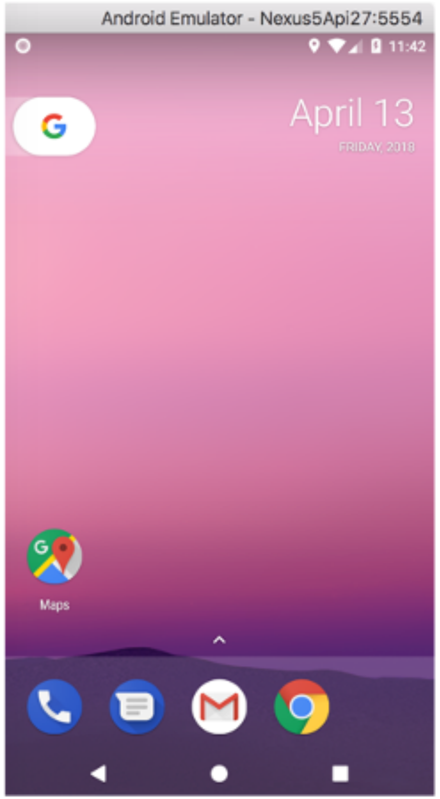
It is possible to execute the Android application under development in two ways::

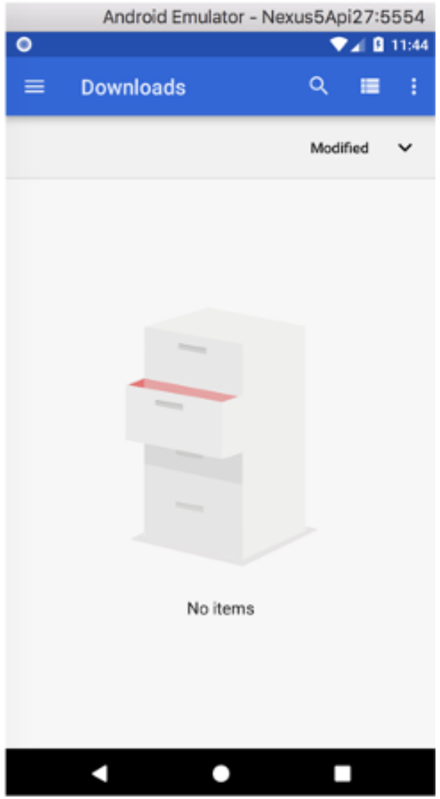
- On a physical device, by using the USB debugging drivers to manage the communication
- An Android Virtual Device (AVD) that is executed on the Android emulator provided with the SDK

Note: for good performance, it is highly suggested to install a “system image” for the Intel CPU and to enable the **Intel HAXM** accelerator

Android Virtual Device (AVD)

Once installed and configured, an AVD can be used as a standard Android phone

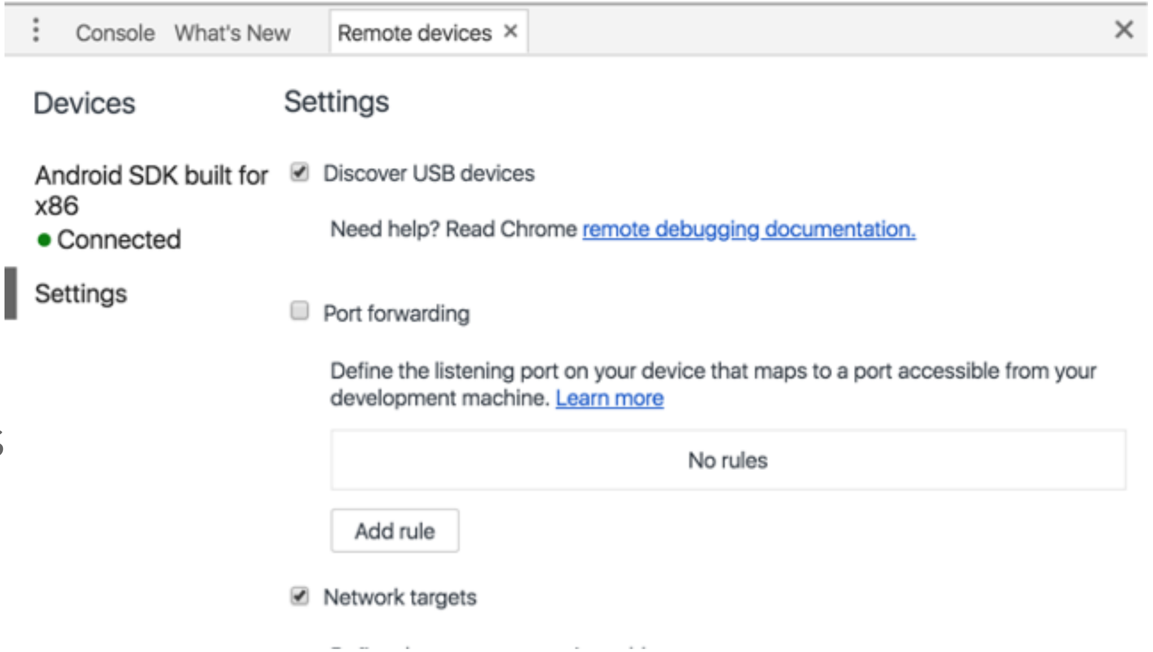




AVD debugging

If the Chrome browser (system or webview) is running in a device, it is possible to remotely debug the JavaScript interpreter by opening the URL chrome://inspect/#devices

from the host machine



Android SDK built for x86
● Connected

Settings

☑ Discover USB devices

Need help? Read Chrome remote debugging documentation.

☐ Port forwarding

Define the listening port on your device that maps to a port accessible from your development machine. Learn more

No rules

Add rule

☑ Network targets

Android platform

Once the Android SDK is installed with at least one platform, it is sufficient to:

1. Export the `ANDROID_HOME` environment variable according to the Android SDK installation directory
2. Add `ANDROID_HOME/platform-tools` to the current path

Then, execute the command:

```
$ cordova platform add android
```

to generate the directory “platform/android” containing all the source files used to build the native container for Android

Build

To compile the application and building the APK ready to be installed:

```
$ cordova build android
```

or:

```
$ cordova build android --release
```

To install the apk on an AVD:

```
$ adb install <package.apk>
```

Plugins

Interface to native APIs is implemented through cordova plugins plugins that can be installed on-demand:

https://cordova.apache.org/plugins/

Ufficial plugins:
Battery, camera, console, contacts, device, device motion, device orientation, dialog, file, geolocation, globalization, vibration, statusbar, etc..

How to use the plugins:

To install a plugin:

```
$ cordova plugin add <nome_plugin>
```

Some plugins add preference options to the cordova config.txt
Ex: cordova-plugin-statusbar

```
<preference name="StatusBarBackgroundColor"
value="#000000" />
```

Some plugins add methods and functions to the JavaScript window object (see the plugin documentation for details)

Angular to mobile

Suppose to have an existing Angular web frontend that needs to be ported to a Hybrid application

General procedure:

1. Modify the app source code to include cordova library (for event handling, etc.)
2. Package the application in the `www/` directory
3. cordova build

Step 1

Load the cordova.js library inside the existing application. Add the following line to **index.html:**

```
<script type="text/javascript" src="cordova.js"></script>
```

Note: The library is automatically «injected» when the application starts on the mobile device

Step 2

Modify the routing base address. Since application is locally served by the cordova container, we should modify the `<base>` URL in this way:

```
<base href="./">
```

Step 3

Let Angular bootstrap when the deviceready event is received. Modify **main.ts** in the following way:

```
import { enableProdMode } from '@angular/core';
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';

import { AppModule } from './app/app.module';
import { environment } from './environments/environment';

if (environment.production) {
 enableProdMode();
}
// Device bootstrap
document.addEventListener('deviceready', () => {
      platformBrowserDynamic().bootstrapModule(AppModule).catch(err =>
console.log(err));
}, false);
```

Step 3

Note: The application won’t run anymore on the browser because the deviceready event never occurs.

```
// Device bootstrap
document.addEventListener('deviceready', () => {
      platformBrowserDynamic().bootstrapModule(AppModule).catch(err =>
console.log(err));
}, false);
```

Step 4

Android applications running on the emulator are usually connected to the host network using a bridge.

The host IP address is `10.0.2.2`, so the sources might be modified accordingly:

```
~~public url = 'http://localhost:8080';~~
public url = 'http://10.0.2.2:8080';
```

Step 5

Modify `.Angular.json` to modify the property outDir from `dist/` to `<app-root>/www/`

This way, by executing the command

```
$ ng build
```

our application will be packed inside `<app-root>/www`

Step 6

Build everything and run:

```
$ cordova platform add android

$ ng build
$ cordova build
$ cordova run
```