Document de Travail pour le Développement Mobile Multiplateformes

Table des matières

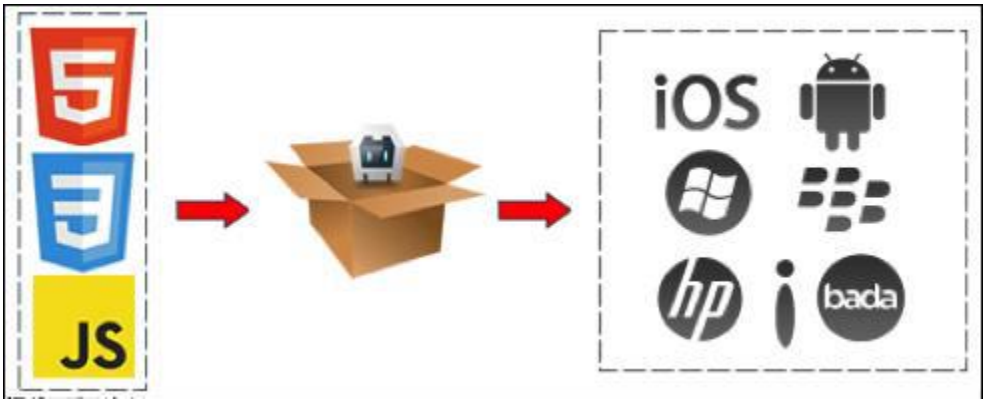
1. Développement mobile multiplateformes

«

- Bien que les deux géants que sont Google et Apple représentent une large majorité du marché des smartphones, tablettes, smart-tv et autres appareils connectés, ce dernier n'en est pas moins très fragmenté avec un certain nombre d'autres acteurs dont évidemment Windows Phone mais également BlackBerry OS, Firefox OS, Sailfish OS, Tizen…

- La tendance actuelle au "tout natif" n'est donc pas sans poser de sérieuses difficultés : Développer par exemple une même application native sur les 3 systèmes les plus répandus nécessite des connaissances spécifiques de 3 environnements et langages différents avec aucune portabilité possible si ce n'est une éventuelle architecture. Une telle optique peut rapidement revenir à doubler voire tripler les ressources nécessaires à un projet et maintenance comprise, représente un investissement considérable tout au long de la durée de vie de l'application.

- Observant le gain de notoriété des applications natives, et par conséquent, l'accroissement de la problématique du multiplateforme, des entreprises se sont rapidement rendues compte de l'importance de trouver des solutions permettant d'uniformiser le développement natif. Des noms tels que PhoneGap, Appcelerator et Xamarin sont ainsi devenus aujourd'hui des incontournables du développement mobile multiplateforme grâce au succès de leurs solutions. »[1]



1. Quelles sont les usages courants ?

- Jeux vidéos
- Applications Grand publique
- Applications Professionnelles

[1] Extrait de « Choix de développement mobile multiplateforme, application native ou hybride ? » Andy CHRISTEN

2. Forme du code exécutable sur un appareil mobile

- Natif
    a. Android (Java), IOS(Objectif-C), Windows(C++), …

- Web
    a. HTML, CSS, JavaScript

3. Avantages du code natif

- Intégration fine
- Interface et ergonomie
- Haute performance
- Outillage
- Markets

4. Inconvénients du code natif

- Plateforme spécifique
- Temps et argent
- Problème de ressources

5. Avantages des applications web sur appareil mobile

- Plusieurs plateformes
- Bonne interaction
- Performance
- IHM professionnelle
- Gain de temps / argent

6. Inconvénients des applications web sur appareil mobile

- Pas de hautes performances
- Pas d'intégration
- Ergonomie

7. Synthèse : natif versus Web

Natif

- IHM
- Hautes performances
- Outillages
- Markets

WebMobile

- Cross-platforms
- Gain de temps / argents
- Ressources
- Nombreuses solutions

8. WebMobile Hybride

1. *Qu'est-ce que c'est ?*

- Consiste généralement en des outils basés sur un langage ou une combinaison de langages génériques et populaires, qui vont servir d'interface avec les fonctionnalités de l'appareil mobile.

- Le développeur n'ayant plus à se soucier de connaître les outils spécifiques à chaque plateforme en développement plusieurs applications,

    - Il suffit de maîtriser ceux offerts par la solution choisie qui lui compilera une application native spécifique à chaque plateforme à partir de son unique projet de départ.

- Exemple : PhoneGap (Apache Cordova) est l'un des pionniers du développement mobile multiplateforme, proposant le populaire trio de langage web HTML5/CSS3/ JavaScript comme source pour ses applications.

2. *Avantages*

Permet de :

- Embarquer son application Web Mobile au sein d'une application native
- Bénéficier d'un pont entre le système et notre application Web Mobile
- Accéder aux markets

9. Modes de production (génération) de codes exécutables

- Ecrire en natif et générer en natif : 1 → 1
  a. API Java Android → Android
  b. Objectif-C → IOS
  c. C#, VB → WindowsPhone
  d. Etc.

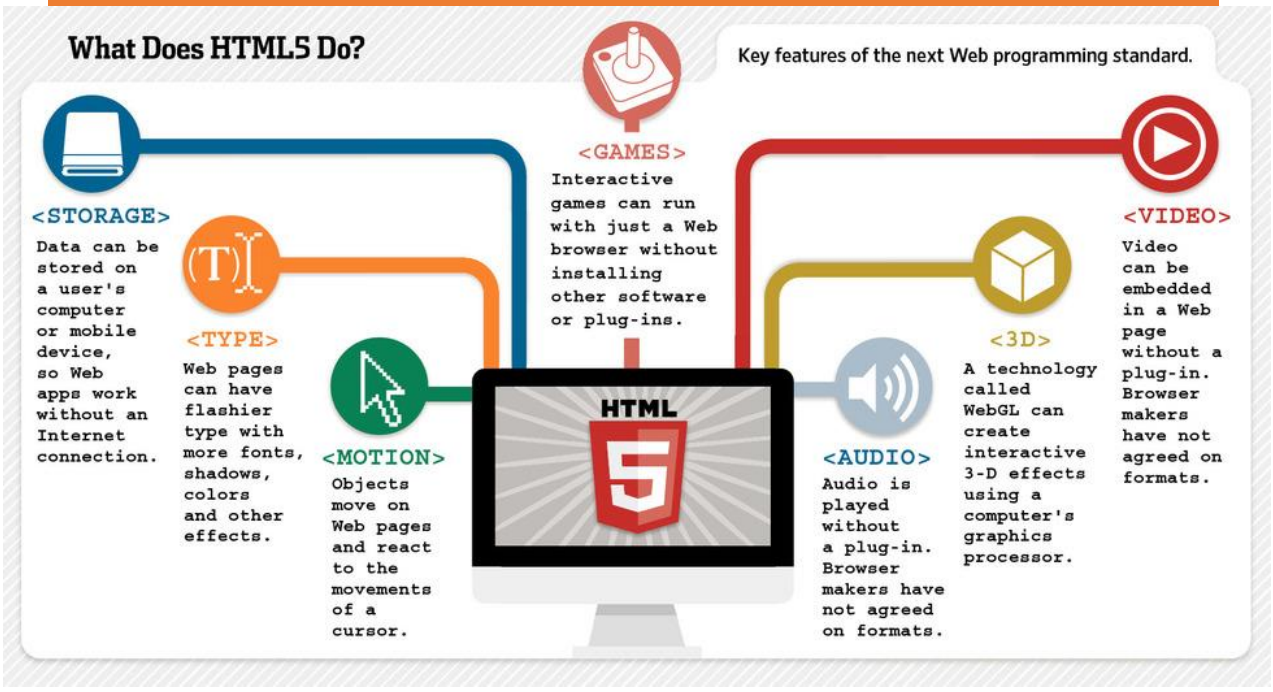
- Ecrire en HTML, CSS, JAVASCRIPT et générer une application web : 1→N
  a. Applications Web sur navigateur

- Ecrire en HTML, CSS, JavaScript + Code natif et générer le webHybrid : 1→ N
  a. Exemple Cordova

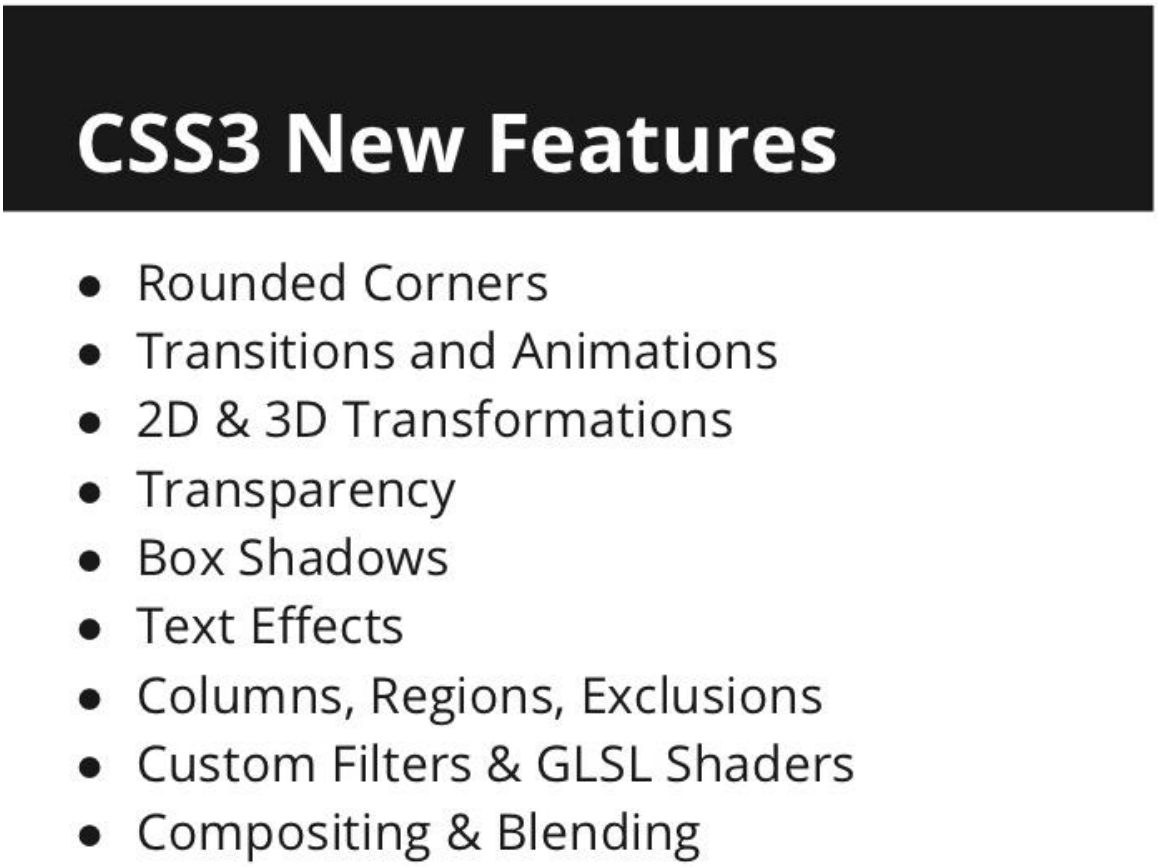
- Ecrire dans un seul et unique langage et générer en natif : 1→N
  a. Exemple Xamarin

2. Développement d'applications Web : HTML 5, CSS3 et JavaScript

10. HTML5



11. CSS3



12. Quelques liens utiles

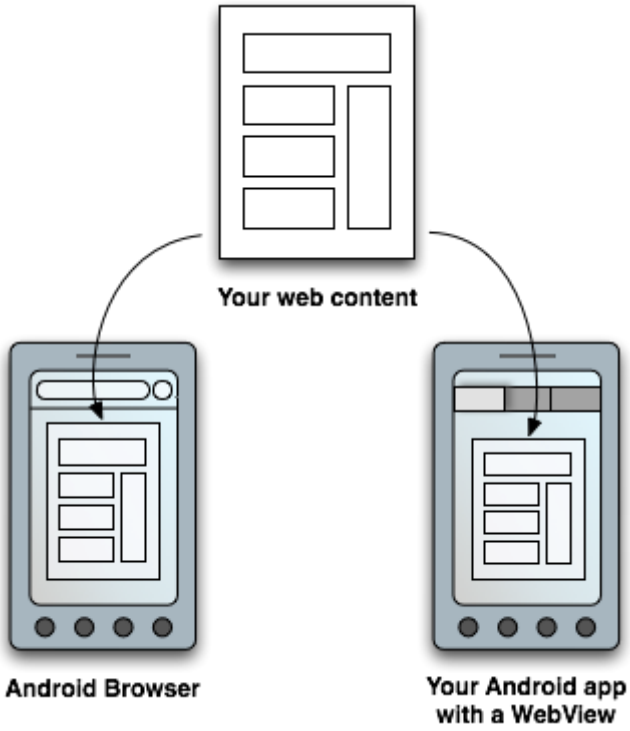
Lien 1 : http://pierre-giraud.com/html-css/cours-complet/cours-html-css-presentation.php

Lien 2 : https://openclassrooms.com/courses/apprenez-a-creer-votre-site-web-avec-html5-et-css3

Lien 3 : http://41mag.fr/apprendre-le-html5-tutoriel-complet

Lien 4 : https://openclassrooms.com/courses/dynamisez-vos-sites-web-avec-javascript

3. Building Web Apps



13. Building Web Apps in WebView

1. If you want to deliver a web application (or just a web page) as a part of a client application, you can do it using `WebView`.

2. The `WebView` class is an extension of Android's `View` class that allows you to display web pages as a part of your activity layout.

3. It does *not* include any features of a fully developed web browser, such as navigation controls or an address bar. All that `WebView` does, by default, is show a web page.

4. A common scenario in which using `WebView` is helpful is when you want to provide information in your application that you might need to update, such as an end-user agreement or a user guide. Within your Android application, you can create an `Activity` that contains a `WebView`, then use that to display your document that's hosted online.

5. Another scenario in which `WebView` can help is if your application provides data to the user that always requires an Internet connection to retrieve data, such as email. In this case, you might find that it's easier to build a `WebView` in your Android application that shows a web page with all the user data, rather than performing a network request, then parsing the data and rendering it in an Android layout. Instead, you can design a web page that's tailored for Android devices and then implement a `WebView` in your Android application that loads the web page.

14. Adding a WebView to Your Application

1. To add a WebView to your Application (basic WebView that displays a web page), simply include the <WebView> element in your activity layout. For example, here's a layout file in which the WebView fills the screen:

```
<?xml version="1.0" encoding="utf-8"?>
<WebView  xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/webview"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
/>
```

2. To load a web page in the WebView, use loadUrl(). For example:

```
WebView myWebView = (WebView) findViewById(R.id.webview);
myWebView.loadUrl("http://www.example.com");
```

1. Before this will work, however, your application must have access to the Internet. To get Internet access, request the INTERNET permission in your manifest file. For example:

```
a. <manifest ... >
       <uses-permission android:name="android.permission.INTERNET" />
       ...
   </manifest>
```

That's all you need for a basic WebView that displays a web page.

15. Using JavaScript in WebView

If the web page you plan to load in your `WebView` use JavaScript, you must enable JavaScript for your `WebView`. Once JavaScript is enabled, you can also create interfaces between your application code and your JavaScript code.

3. *Enabling JavaScript*

1. JavaScript is disabled in a `WebView` by default. You can enable it through the `WebSettings` attached to your `WebView`. You can retrieve `WebSettings` with `getSettings()`, then enable JavaScript with `setJavaScriptEnabled()`.

    a. For example:

```
WebView myWebView = (WebView) findViewById(R.id.webview);
WebSettings webSettings = myWebView.getSettings();
webSettings.setJavaScriptEnabled(true);
```

    b. `WebSettings` provides access to a variety of other settings that you might find useful. For example, if you're developing a web application that's designed specifically for the `WebView` in your Android application, then you can define a custom user agent string with `setUserAgentString()`, then query the custom user agent in your web page to verify that the client requesting your web page is actually your Android application.

4. *Binding JavaScript code to Android code*

1. When developing a web application that's designed specifically for the `WebView` in your Android application, you can create interfaces between your JavaScript code and client-side Android code. For example, your JavaScript code can call a method in your Android code to display a `Dialog`, instead of using JavaScript's `alert()` function.

2. To bind a new interface between your JavaScript and Android code, call `addJavascriptInterface()`, passing it a class instance to bind to your JavaScript and an interface name that your JavaScript can call to access the class.

3. For example, you can include the following class in your Android application:

```
public class WebAppInterface {
    Context mContext;

    /** Instantiate the interface and set the context */
    WebAppInterface(Context c) {
        mContext = c;
    }

    /** Show a toast from the web page */
    @JavascriptInterface
```

```
    public void showToast(String toast) {
        Toast.makeText(mContext, toast, Toast.LENGTH_SHORT).show();
    }
}
```

Caution: If you've set your `targetSdkVersion` to 17 or higher, **you must add the `@JavascriptInterface` annotation** to any method that you want available to your JavaScript (the method must also be public). If you do not provide the annotation, the method is not accessible by your web page when running on Android 4.2 or higher.

In this example, the `WebAppInterface` class allows the web page to create a `Toast` message, using the `showToast()` method.

You can bind this class to the JavaScript that runs in your `WebView` with `addJavascriptInterface()` and name the interface `Android`. For example:

```
WebView webView = (WebView) findViewById(R.id.webview);
webView.addJavascriptInterface(new WebAppInterface(this), "Android");
```

This creates an interface called `Android` for JavaScript running in the `WebView`. At this point, your web application has access to the `WebAppInterface`class. For example, here's some HTML and JavaScript that creates a toast message using the new interface when the user clicks a button:

```
<input type="button" value="Say hello" onClick="showAndroidToast('Hello
Android!')" />

<script type="text/javascript">
    function showAndroidToast(toast) {
        Android.showToast(toast);
    }
</script>
```

There's no need to initialize the `Android` interface from JavaScript. The `WebView` automatically makes it available to your web page. So, at the click of the button, the `showAndroidToast()` function uses the `Android` interface to call the `WebAppInterface.showToast()` method.

Pour avoir le cours complet, consulter :

https://developer.android.com/guide/webapps/index.html

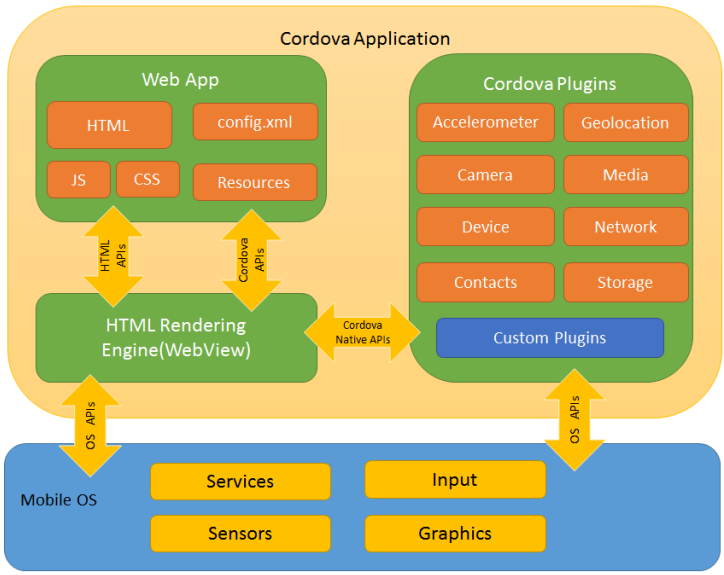
4. Cordova : Généralités

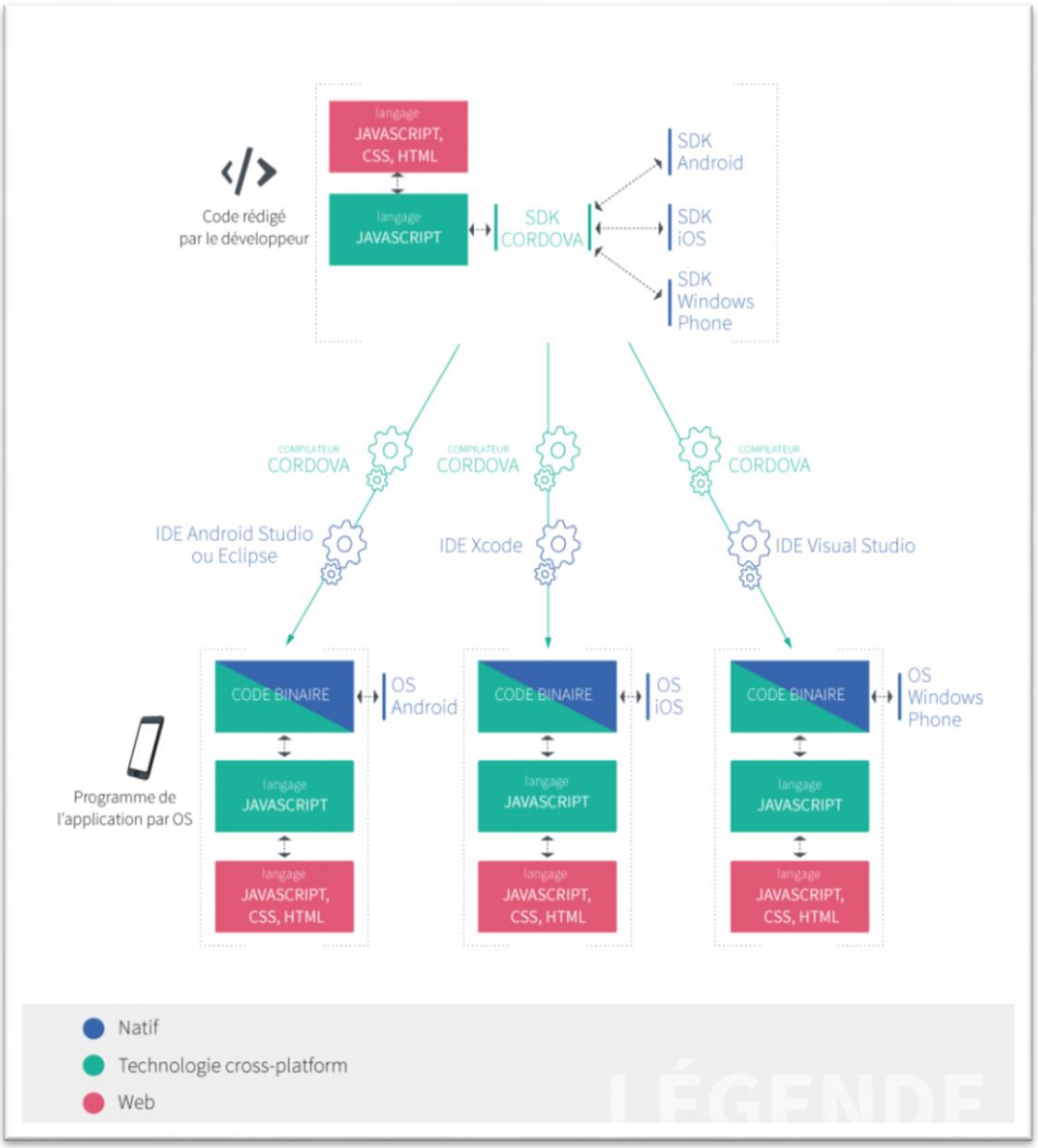
16. Les genèses

- Est un projet permis grâce à Adobe qui a fourni le code source à la fondation Apache.
- Framework de développement d'application SPA pour plateformes mobiles
- L'idée : compiler une coquille utilisant un module « webview » chargeant notre application.
    - C'est un peu comme si on encapsulait notre application dans un navigateur très allégé.
- PhoneGap existe toujours et peut permettre via une offre « cloud » de générer ses applications pour Android mais aussi iOS, Windows Phone, BlackBerry…
- Pour information, il permet également de compiler vers des plateformes desktop : Windows 8 et Ubuntu.
    - L'adresse : https://build.phonegap.com/
- Adresse Cordova : https://cordova.apache.org/docs/en/latest/guide/overview/

17. L'architecture

- Bâtie au-dessus de NodeJS
- Permet de développer en HTML / CSS /JavaScript
- S'occupe de déployer sur les plateformes
- Permet d'utiliser des plugins pour accéder aux fonctionnalités natives


Cordova Application
Web App
HTML
config.xml
JS
CSS
Resources
Cordova Plugins
Accelerometer
Geolocation
Camera
Media
Device
Network
Contacts
Storage
Custom Plugins
HTML APIs
Cordova APIs
HTML Rendering Engine(WebView)
Cordova Native APIs
OS APIs
OS APIs
Mobile OS
Services
Input
Sensors
Graphics


langage JAVASCRIPT, CSS, HTML
Code rédigé par le développeur
langage JAVASCRIPT
SDK CORDOVA
SDK Android
SDK iOS
SDK Windows Phone
COMPILATEUR CORDOVA
COMPILATEUR CORDOVA
COMPILATEUR CORDOVA
IDE Android Studio ou Eclipse
IDE Xcode
IDE Visual Studio
CODE BINAIRE
OS Android
CODE BINAIRE
OS iOS
CODE BINAIRE
OS Windows Phone
Programme de l'application par OS
langage JAVASCRIPT
langage JAVASCRIPT, CSS, HTML
Natif
Technologie cross-platform
Web
LÉGENDE

5. Cordova: La pratique

18. Create your first Cordova app

This guide shows you how to create a JS/HTML Cordova application and deploy them to various native mobile platforms using the `cordova`command-line interface (CLI). For detailed reference on Cordova command-line, review the CLI reference

5. *Installing the Cordova CLI*

The Cordova command-line tool is distributed as an npm package.

To install the `cordova` command-line tool, follow these steps:

1. Download and install Node.js. On installation you should be able to invoke `node` and `npm` on your command line.

2. (Optional) Download and install a git client, if you don't already have one. Following installation, you should be able to invoke `git` on your command line. The CLI uses it to download assets when they are referenced using a url to a git repo.

3. Install the `cordova` module using `npm` utility of Node.js. The `cordova` module will automatically be downloaded by the `npm` utility.

- on OS X and Linux:

```
$ sudo npm install -g cordova
```

On OS X and Linux, prefixing the `npm` command with `sudo` may be necessary to install this development utility in otherwise restricted directories such as `/usr/local/share`. If you are using the optional nvm/nave tool or have write access to the install directory, you may be able to omit the `sudo` prefix. There are more tips available on using `npm` without `sudo`, if you desire to do that.

- on Windows:

```
C:\>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.

6. *Create the App*

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

```
$ cordova create hello com.example.hello HelloWorld
```

This creates the required directory structure for your cordova app. By default, the `cordova create` script generates a skeletal web-based application whose home page is the project's `www/index.html` file.

See Also

- Cordova create command reference documentation
- Cordova project directory structure
- Cordova project templates

7. *Add Platforms*

All subsequent commands need to be run within the project's directory, or any subdirectories:

```
$ cd hello
```

Add the platforms that you want to target your app. We will add the 'ios' and 'android' platform and ensure they get saved to `config.xml`:

```
$ cordova platform add ios --save
$ cordova platform add android --save
```

To check your current set of platforms:

```
$ cordova platform ls
```

Running commands to add or 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.

See Also

- Cordova platform command reference documentation

8. *Install pre-requisites for building*

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

Requirements check results for android:

Java JDK: installed .

Android SDK: installed

Android target: installed android-19,android-21,android-22,android-23
,Google Inc.:Google APIs:19,Google Inc.:Google APIs (x86 System Image
):19,Google Inc.:Google APIs:23

Gradle: installed


Requirements check results for ios:

Apple OS X: not installed

Cordova tooling for iOS requires Apple OS X

Error: Some of requirements check failed
```

See Also

- Android platform requirements
- iOS platform requirements
- Windows platform requirements

9. *Build the App*

By default, `cordova create` script generates a skeletal web-based application whose start page is the project's `www/index.html` file. Any initialization should be specified as part of the deviceready event handler defined in `www/js/index.js`.

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 - 'ios' in this case:

```
$ cordova build ios
```

See Also

- Cordova build command reference documentation

10. Test the App

SDKs for mobile platforms often come bundled with emulators that execute a device image, so that you can launch the app from the home screen and see how it interacts with many platform features. Run a command such as the following to rebuild the app and view it within a specific platform's emulator:

```
$ cordova emulate android
```

Following up with the `cordova emulate` command refreshes the emulator image to display the latest application, which is now available for launch from the home screen:

Alternately, you can plug the handset into your computer and test the app directly:

```
$ cordova run android
```

Before running this command, you need to set up the device for testing, following procedures that vary for each platform.

See Also

- Setting up Android emulator
- Cordova run command reference documentation
- Cordova emulate command reference documentation

11. Add Plugins

You can modify the default generated app to take advantage of standard web technologies, but for the app to access device-level features, you need to add plugins.

A *plugin* exposes a Javascript API for native SDK functionality. Plugins are typically hosted on npm and you can search for them on the plugin search page. Some key APIs are provided by the Apache Cordova open source project and these are referred to as Core Plugin APIs. You can also use the CLI to launch the search page:

```
$ cordova plugin search camera
```

To add the camera plugin, we will specify the npm package name for the camera plugin:

```
$ cordova plugin add cordova-plugin-camera
Fetching plugin "cordova-plugin-camera@~2.1.0" via npm
Installing "cordova-plugin-camera" for android
Installing "cordova-plugin-camera" for ios
```

Plugins can also be added using a directory or a git repo.

NOTE: The CLI adds plugin code as appropriate for each platform. If you want to develop with lower-level shell tools or platform SDKs as discussed in the Overview, you need to run the Plugman utility to add plugins separately for each platform. (For more information, see Using Plugman to Manage Plugins.)
Use `plugin ls` (or `plugin list`, or `plugin` by itself) to view currently installed plugins. Each displays by its identifier:

```
$ cordova plugin ls
cordova-plugin-camera 2.1.0 "Camera"
cordova-plugin-whitelist 1.2.1 "Whitelist"
```

See Also

- Cordova plugin command reference documentation
- Cordova plugin search page
- Core Plugin APIs

12. Using merges to Customize Each Platform

While Cordova allows you to easily deploy an app for many different platforms, sometimes you need to add customizations. In that case, you don't want to modify the source files in various `www` directories within the top-level `platforms` directory, because they're regularly replaced with the top-level `www` directory's cross-platform source.

Instead, the top-level `merges` directory offers a place to specify assets to deploy on specific platforms. Each platform-specific subdirectory within `merges` mirrors the directory structure of the `www` source tree, allowing you to override or add files as needed. For example, here is how you might uses `merges` to boost the default font size for Android devices:

- Edit the `www/index.html` file, adding a link to an additional CSS file, `overrides.css` in this case:

```
<link rel="stylesheet" type="text/css" href="css/overrides.css" />
```

- Optionally create an empty `www/css/overrides.css` file, which would apply for all non-Android builds, preventing a missing-file error.
- Create a `css` subdirectory within `merges/android`, then add a corresponding `overrides.css` file. Specify CSS that overrides the 12-point default font size specified within `www/css/index.css`, for example:

```
body { font-size:14px; }
```

When you rebuild the project, the Android version features the custom font size, while others remain unchanged.

You can also use `merges` to add files not present in the original `www` directory. For example, an app can incorporate a *back button* graphic into the iOS interface, stored in `merges/ios/img/back_button.png`, while the Android version can instead capture backbutton events from the corresponding hardware button.

13. Updating Cordova and Your Project

After installing the `cordova` utility, you can always update it to the latest version by running the following command:

```
$ sudo npm update -g cordova
```

Use this syntax to install a specific version:

```
$ sudo npm install -g cordova@3.1.0-0.2.0
```

Run `cordova -v` to see which version is currently running. To find the latest released cordova version, you can run:

```
$ npm info cordova version
```

To update platform that you're targeting:

```
$ cordova platform update android --save
$ cordova platform update ios --save
...etc.
```

19. Guide pour la plate-forme Android

Ce guide montre comment configurer votre environnement SDK pour déployer des applications de Cordova pour les appareils Android et comment éventuellement utiliser Android-centré des outils de ligne de commande dans votre flux de travail de développement. Vous devez installer le SDK Android indépendamment si vous voulez utiliser ces outils axés sur la plate-forme de shell ou la CLI de Cordova multi-plateforme pour le développement. Pour une comparaison entre les voies de deux développement, consultez la vue d'ensemble. Pour plus d'informations sur la CLI, consultez l'Interface de ligne de commande.

14. Configuration requise et support

Cordova pour Android nécessite le SDK Android qui peut être installé sur le système d'exploitation OS X, Linux ou Windows. Voir du SDK AndroidConfiguration du système requise.

Cordova supporte Android 4.0.x (en commençant par le niveau de l'API Android 14) et plus élevé. En règle générale, les versions Android deviennent non étayées par Cordova comme ils plonger au-dessous de 5 % sur Google dashboard de distribution. Android versions antérieures à la version API de niveau 10 et les versions 3.x (Honeycomb, niveaux API 11-13) tombent nettement au-dessous de ce seuil de 5 %.

15. Installer les outils de Cordova Shell

Si vous souhaitez utiliser les outils de coquille Android-centrée de Cordova conjointement avec le SDK, Télécharger Cordova de cordova.apache.org. Sinon ignorer cette section si vous envisagez d'utiliser l'outil CLI de multi-plateforme décrit dans l'Interface de ligne de commande.

Le téléchargement de Cordova contient des archives distincts pour chaque plate-forme. N'oubliez pas d'élargir l'archive appropriée, `android` dans ce cas, dans un répertoire vide. Les utilitaires les pertinents sont disponibles dans le niveau supérieur `bin` répertoire. (Consultez le fichier **README** si nécessaire pour des directions plus détaillées).

Ces outils de coquille permettent de créer, générer et exécuter des applications Android. Pour plus d'informations sur l'interface de ligne de commande supplémentaire qui active les fonctionnalités de plugin sur toutes les plateformes, voir Plugman à l'aide à gérer les Plugins. Voir Application Plugins pour plus d'informations sur la façon de développer des plugins.

16. Installez le Kit de développement Java (JDK)

Installer Java Development Kit (JDK) 7 ou version ultérieure.

Lors de l'installation sous Windows, vous devez également définir la Variable d'environnement `JAVA_HOME` selon le chemin d'installation de JDK (par exemple, C:\Program Files\Java\jdk1.7.0_75).

17. *Installer le SDK Android*

Installer les outils de Android SDK autonome ou Studio Android. Procceed avec `Android Studio` si vous prévoyez Cordova nouvelle pour Android plugins ou utilisant des outils natifs pour exécuter et déboguer la plateforme Android. Dans le cas contraire, `Outils du SDK Android autonome`suffisent pour créer et déployer des applications Android.

Instructions d'installation détaillées sont disponibles dans le cadre des liens d'installation ci-dessus.

Pour outils de ligne de commande de Cordova pour travailler, ou la CLI qui repose sur eux, vous devez inclure les répertoires de `plate-forme-outils` et `outils` du SDK dans votre `PATH`. Sur un Mac ou Linux, vous pouvez utiliser un éditeur de texte pour créer ou modifier le fichier`~/.bash_profile` , ajoutant une ligne comme ci-dessous, en fonction d'où installe le SDK :

```
    export PATH=${PATH}:/Development/android-sdk/platform-tools:/Development/andro
id-sdk/tools
```

Cette ligne dans `~/.bash_profile` expose ces outils dans windows terminales nouvellement ouverts. Si votre fenêtre de terminal est déjà ouvert dans OSX ou d'éviter une déconnexion/connexion sur Linux, exécutez ceci pour les rendre disponibles dans la fenêtre du terminal actuelle :

```
    $ source ~/.bash_profile
```

Pour modifier l'environnement `PATH` sous Windows :

1. Cliquez sur le menu **Démarrer** dans le coin en bas à gauche du bureau, faites un clic droit sur **ordinateur**, puis sélectionnez **Propriétés**.
2. Sélectionnez **Paramètres système avancés** dans la colonne de gauche.
3. Dans la boîte de dialogue, appuyez sur **Variables d'environnement**.
4. Sélectionnez la variable **PATH** et appuyer sur **modifier**.
5. Ajouter ce qui suit à le `PATH` basé sur lequel vous avez installé le SDK, par exemple :

```
6. ;C:\Development\android-sdk\platform-tools;C:\Development\android-sdk\tools
```

7. Enregistrez la valeur et fermez les deux boîtes de dialogue.

18. Installer les paquets SDK

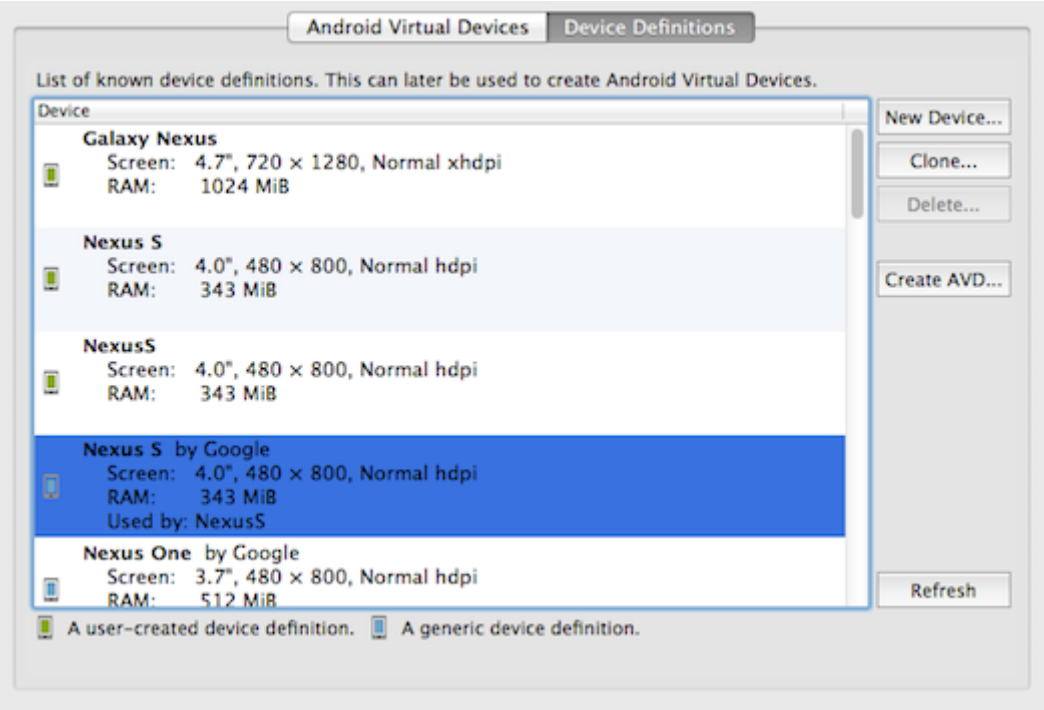
Ouvrez le gestionnaire de SDK Android (par exemple, par l'intermédiaire de borne : `android`) et installer :

1. 5.1.1 Android (API 22) platform SDK
2. Version d'Android SDK Build-tools 19.1.0 ou supérieur
3. Référentiel de prise en charge Android (Extras)

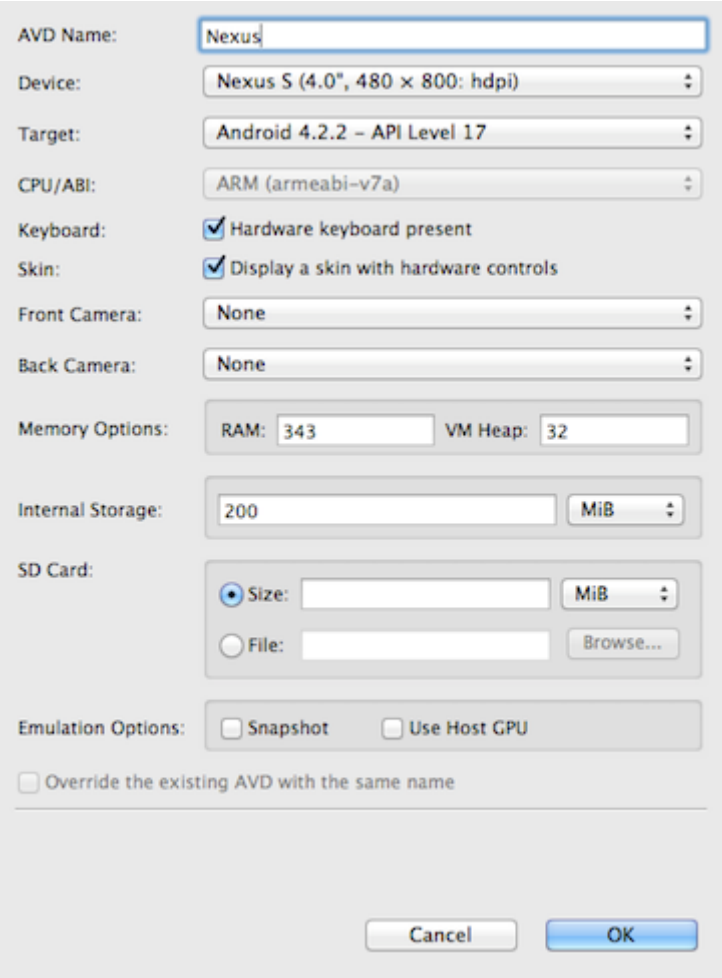
Pour plus de détails, voir Installation de Packages de SDK .

19. Configurer un émulateur

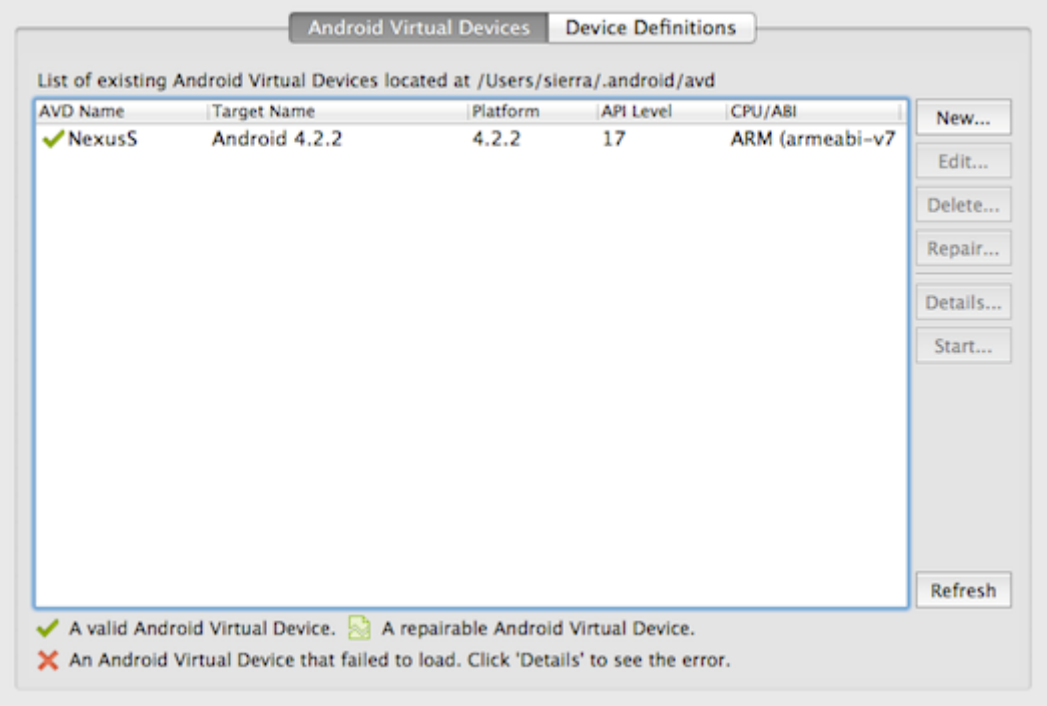
Android sdk ne fournit pas de n'importe quelle instance d'émulateur par défaut par défaut. Vous pouvez créer un nouveau en exécutant `android` sur la ligne de commande. La presse **Outils → gérer AVDs** (périphériques virtuels Android), puis choisissez n'importe quel élément du **Dispositif de définitions** dans la boîte de dialogue :



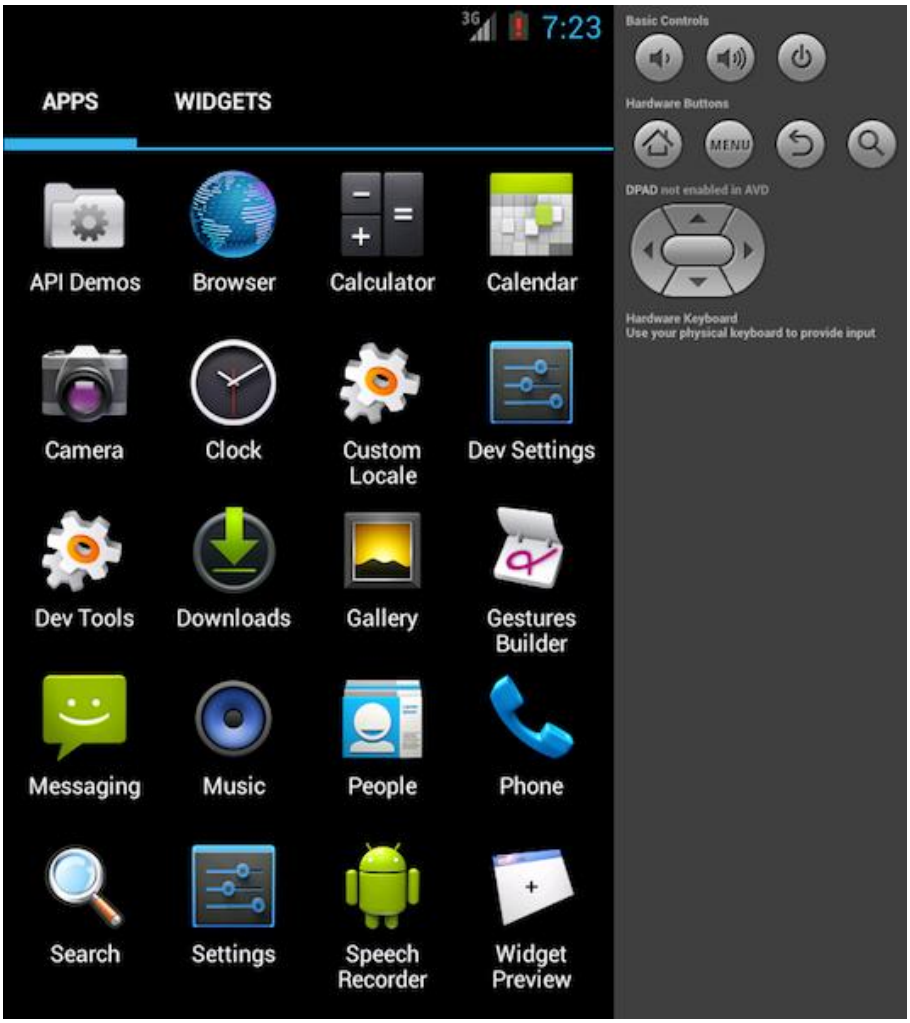
Appuyez sur **Créer AVD**, éventuellement modifier le nom, puis appuyez sur **OK** pour accepter les modifications :



L'AVD apparaît alors dans la liste **Des périphériques virtuels Android** :



Pour ouvrir l'émulateur comme une demande distincte, l'AVD et cliquez sur **Démarrer**. Il lance autant qu'il le ferait sur le dispositif, avec des contrôles supplémentaires disponibles pour les boutons matériels :

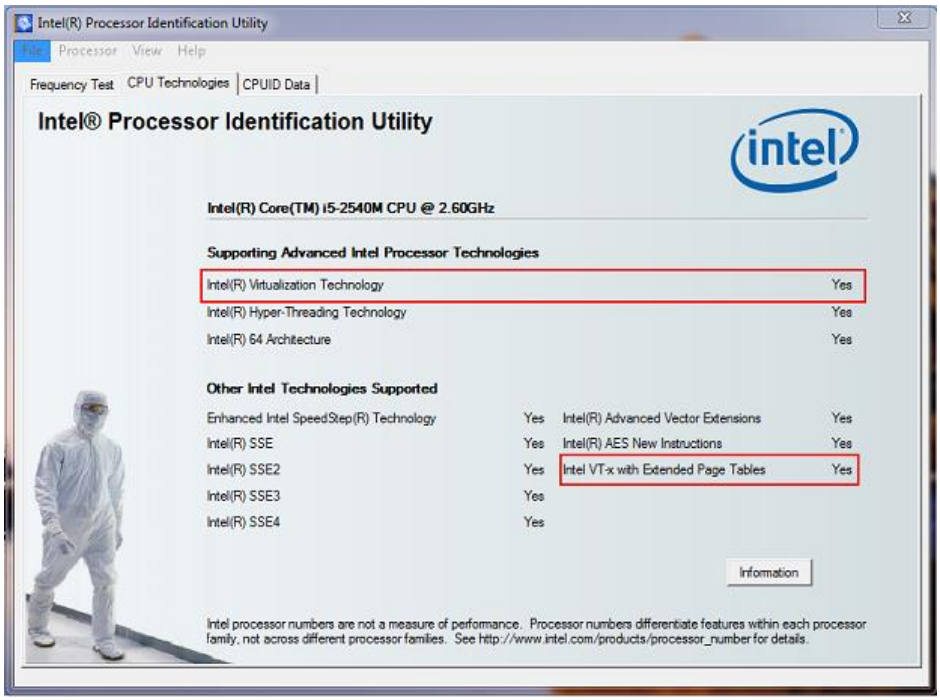


Pour une expérience plus rapide, vous pouvez utiliser l' `Accélération de la Machine virtuelle` pour améliorer la vitesse d'exécution. De nombreux processeurs modernes fournissent des extensions pour exécuter des Machines virtuelles plus efficacement. Avant d'utiliser ce type d'accélération, vous devez déterminer si CPU de votre système actuel de développement, on supporte les technologies de virtualisation suivants :

- **Technologie de virtualisation Intel** (VT-x, vmx) → Intel VT-x pris en charge la liste des processeurs
- **AMD Virtualization** (AMD-V, SVM), pris en charge uniquement pour Linux (depuis mai 2006, tous les processeurs AMD incluent AMD-V, sauf Sempron).

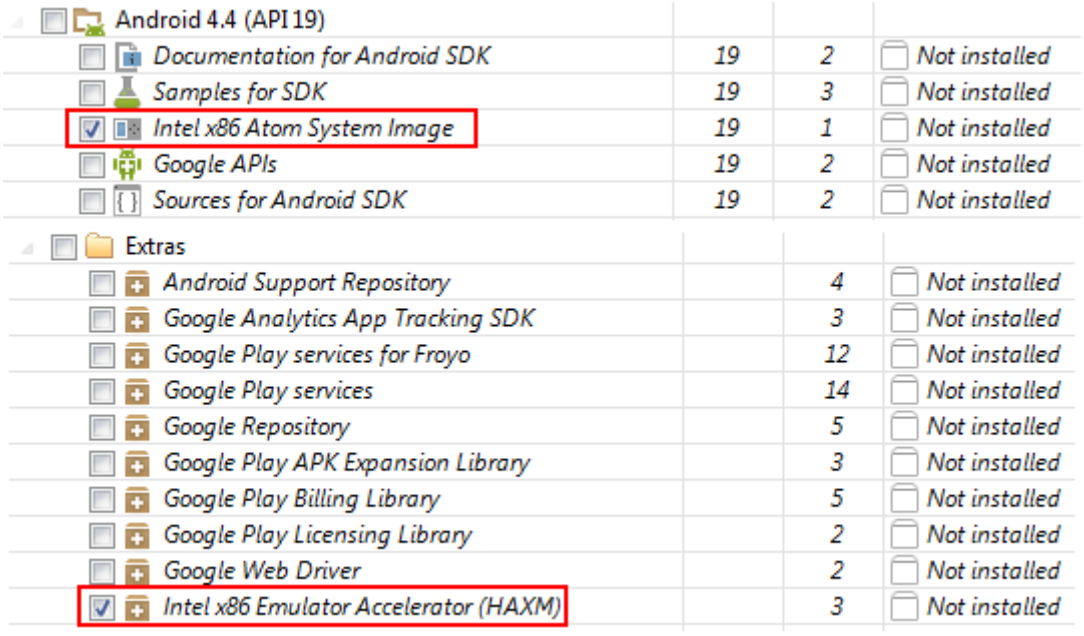
Une autre façon de savoir si votre processeur supporte la technologie de VT-x, c'est en exécutant l' `Utilitaire Intel Processor Identification Utility`, pour `Windows`, vous pouvez le télécharger depuis le Centre de téléchargementde Intel, ou vous pouvez utiliser l' utilitaire booteable, qui est`Indépendant de l'OS`.

Après avoir installer et exécuter `l'Utilitaire d'Identification des processeurs Intel` sur Windows, vous obtiendrez la fenêtre suivante, afin de vérifier si votre processeur supporte les Technologies de virtualisation :



Afin d'accélérer l'émulateur, vous devez télécharger et installer une ou plusieurs Images de système `Atom d'Intel x 86` , ainsi que l' `Intel matériel accéléré l'exécution Manager (HAXM)`.

Ouvrez votre gestionnaire de SDK Android et sélectionnez l'Image du système `Atom d'Intel x 86` , pour quelle que soit la version que vous souhaitez tester. Puis allez à `options` et sélectionnez `Intel x 86 Emulator accélérateur (HAXM)`et installer ces paquets :



Après le téléchargement, exécuter le programme d'installation d'Intel, qui est disponible dans votre Android SDK à`Options/intel/Hardware_Accelerated_Execution_Manager`. **Remarque**:`si vous avez des difficultés pour installer le package, vous pouvez trouver plus d'informations et conseils étape par étape cochez-le` Article Intel.

1. Installez une ou plusieurs Images de système `x 86 d'Intel Atom` ainsi que le `Gestionnaire d'exécution accélérée matériel Intel`, disponible sous **Extras**.
2. Exécutez le programme d'installation d'Intel, qui est disponible dans votre Android SDK à`Options/intel/Hardware_Accelerated_Execution_Manager`.
3. Créer un nouvel AVD avec l'objectif fixé à une image d'Intel.
4. Lorsque vous démarrez l'émulateur, assurez-vous il n'y a aucun message d'erreur indiquant une panne de charger les modules HAX.

20. Créez un nouveau projet

À ce stade, pour créer un nouveau projet, vous pouvez choisir entre l'outil CLI multiplate-forme décrit dans l'Interface de ligne de commande, ou l'ensemble des outils de coquille spécifiques à Android. Partir dans un répertoire de code source, voici l'approche de la CLI :

```
    $ cordova create hello com.example.hello HelloWorld
    $ cd hello
    $ cordova platform add android
    $ cordova prepare              # or "cordova build"
```

Voici l'approche de shell-outil de niveau inférieur correspondant pour Unix et Windows :

```
    $ /path/to/cordova-android/bin/create /path/to/new/hello com.example.hello Hel
loWorld
    C:\path\to\cordova-android\bin\create.bat C:\path\to\new\hello com.example.hel
lo HelloWorld
```

21. Générez le projet

Si vous utilisez l'interface CLI dans le développement, le répertoire de niveau supérieur `www` du répertoire du projet contient les fichiers sources. Courir à chacun d'entre eux dans le répertoire du projet pour reconstruire l'application :

```
    $ cordova build                  # build all platforms that were added
    $ cordova build android          # build debug for only Android
```

```
    $ cordova build android --debug   # build debug for only Android
    $ cordova build android --release # build release for only Android
```

Si vous utilisez les outils de coquille spécifiques à Android en développement, il y a une approche différente. Une fois que vous générez le projet, source de l'application par défaut est disponible dans le sous-répertoire `assets/www` . Les commandes suivantes sont disponibles dans son sous-répertoire de `cordova` .

La commande `build` nettoie les fichiers projet et régénère l'app. Voici la syntaxe pour Mac et Windows. Les deux premiers exemples génèrent des informations de débogage, et le second s'appuie les apps pour diffusion immédiate :

```
    $ /path/to/project/cordova/build --debug
    C:\path\to\project\cordova\build.bat --debug


    $ /path/to/project/cordova/build --release
    C:\path\to\project\cordova\build.bat --release
```

22. Déployer l'application

Vous pouvez utiliser l'utilitaire CLI de `cordova` pour déployer l'application sur l'émulateur ou le dispositif de la ligne de commande :

```
    $ cordova emulate android       #to deploy the app on a default android emulat
or
    $ cordova run android --device  #to deploy the app on a connected device
```

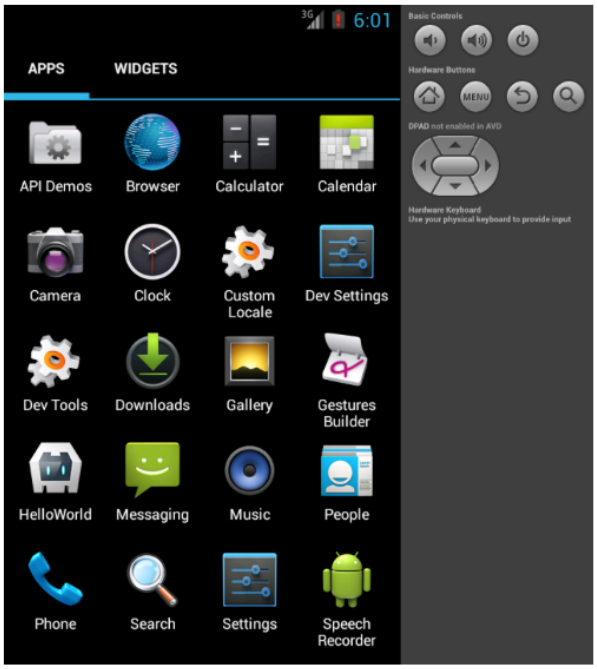
Sinon, utilisez l'interface de coquille alternative :

```
    $ /path/to/project/cordova/run --emulator
    $ /path/to/project/cordova/run --device
```

Vous pouvez utiliser **cordova run android --list** pour voir toutes les cibles disponibles et **cordova run android --target=target_name** pour exécuter l'application sur un émulateur ou un périphérique spécifique (par exemple, `cordova run android --target="Nexus4_emulator"`).

Vous pouvez également utiliser **cordova run --help** pour voir construire supplémentaire et exécuter les options.

Cela pousse l'app à l'écran d'accueil et il lance :



Lorsque vous `run` l'application, vous aussi `build` il. Vous pouvez ajouter supplémentaires `--debug`, `--release`et `--nobuild` drapeaux pour contrôler comment il est construit, ou même si une reconstruction est nécessaire :

```
$ /path/to/project/cordova/run --emulator --nobuild
```

23. Autres commandes

Ce qui suit génère un journal détaillé de l'application en cours d'exécution :

```
$ /path/to/project/cordova/log

C:\path\to\project\cordova\log.bat
```

Le texte suivant nettoie les fichiers de projet :

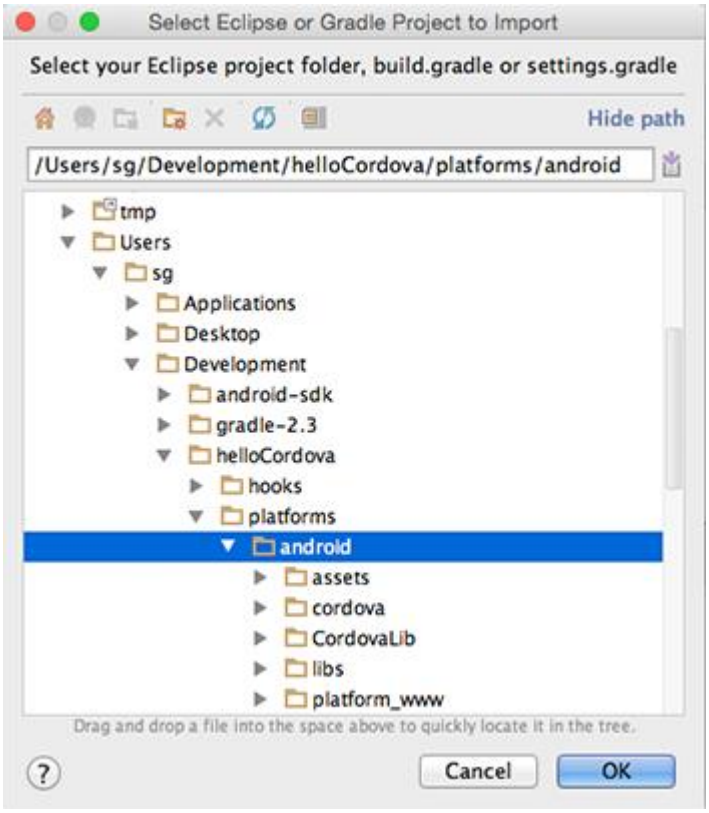
```
$ /path/to/project/cordova/clean

C:\path\to\project\cordova\clean.bat
```

24. Ouvrez un nouveau projet dans le SDK

Une fois que la plateforme android est ajouté à votre projet, vous pouvez l'ouvrir depuis AndroidStudio :

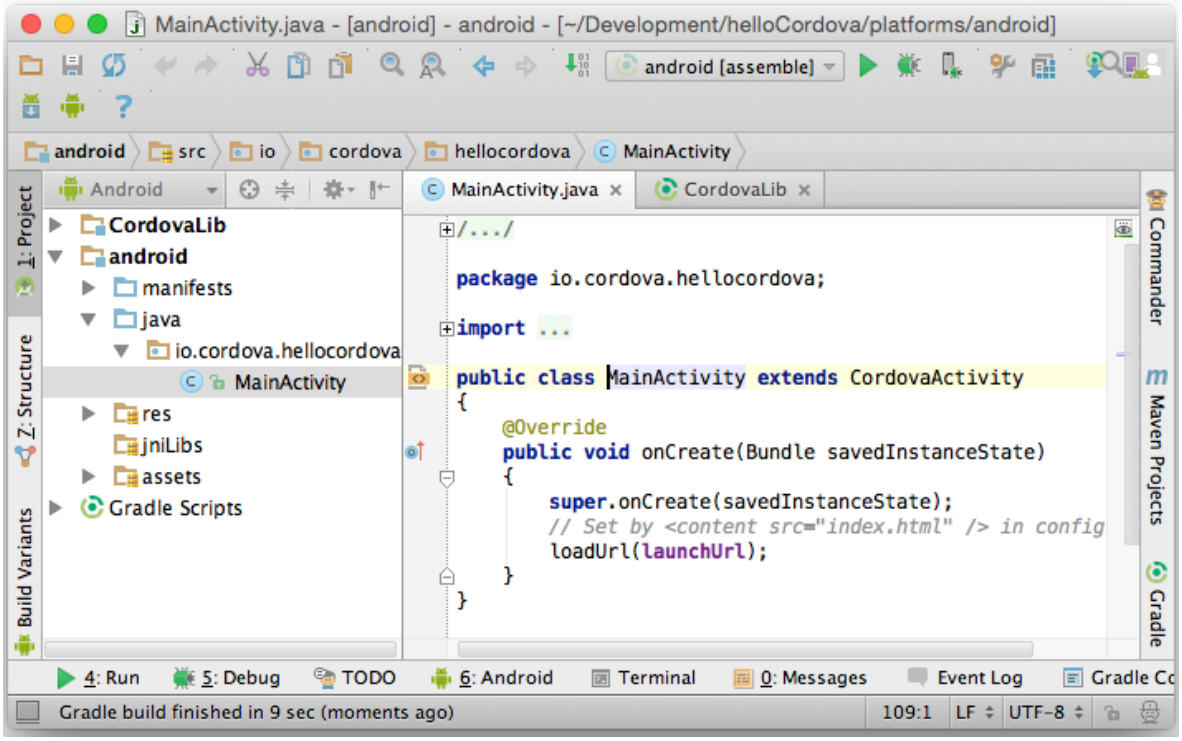
1. Lancez l'application **Android de Studio** .
2. Sélectionnez **Import Project (Eclipse ADT, Gradle, etc.)**.

3. Sélectionnez l'emplacement où la plateforme android est stockée (`votre/projet/platforms/android`).



4. Pour la question `Gradle Sync` vous pouvez simplement répondre **Oui**.

Vous êtes tous ensemble maintenant et pouvez générer et exécuter l'application directement à partir de `Studio Android`.



Consultez Vue d'ensemble Studio de Android et génération et l'exécution de Studio Android pour plus de détails.

20. IDEs et Cordova

25. *Cordova en Ligne de commande*

- https://cordova.apache.org/docs/en/2.5.0/guide/getting-started/android/
- https://cordova.apache.org/docs/fr/3.1.0/guide/platforms/android/
- https://cordova.apache.org/docs/fr/6.x/

26. *Visual Studio-Cordova :*

- https://msdn.microsoft.com/fr-fr/library/dn757057.aspx
- https://msdn.microsoft.com/fr-fr/library/dn771545.aspx
- https://msdn.microsoft.com/fr-fr/library/dn757054.aspx
- https://www.visualstudio.com/fr/vs/cordova/
- https://taco.visualstudio.com/en-us/docs/get-started-first-mobile-app/
- https://taco.visualstudio.com/en-us/docs/install-vs-tools-apache-cordova/
- https://channel9.msdn.com/Series/Visual-Studio-Tools-for-Apache-Cordova/Building-Apache-Cordova-Apps-with-Visual-Studio

27. *Cordova sur Eclipse*

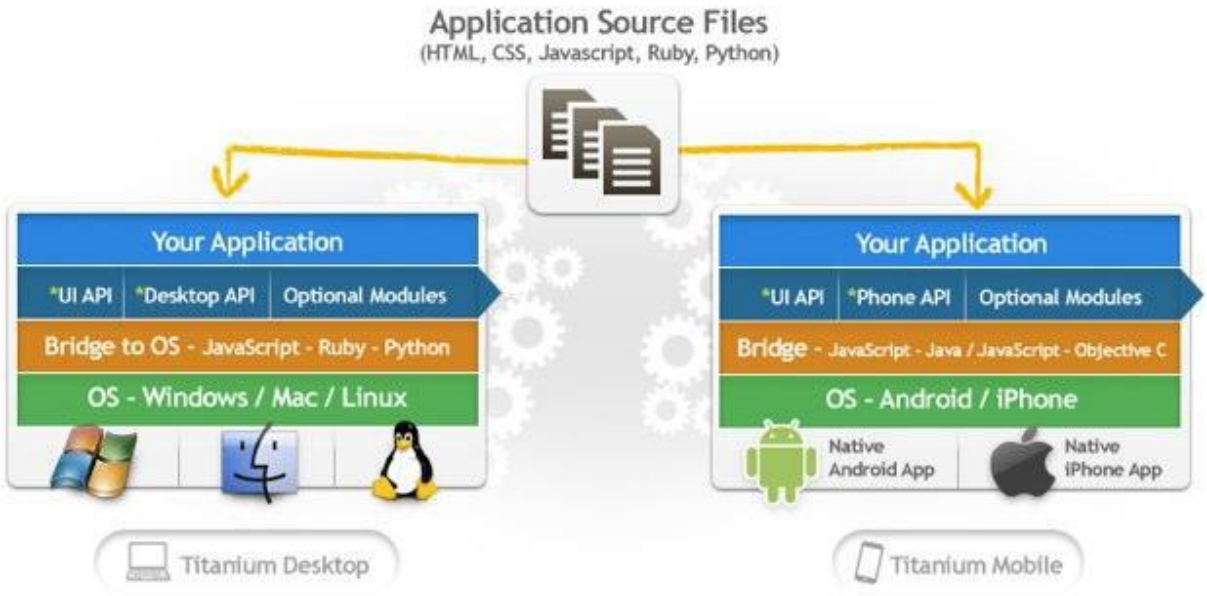
- https://www.eclipse.org/community/eclipse_newsletter/2015/july/article2.php
- https://www.youtube.com/watch?v=H5ry5WpziVw
- https://www.adobe.com/devnet/archive/html5/articles/getting-started-with-phonegap-in-eclipse-for-android.html

- JBOSS
  - http://tools.jboss.org/documentation/howto/hmt_firstapp.html
  - http://www.jboss.org/ticket-monster/hybridui/
  - http://tools.jboss.org/blog/2014-11-24-three-ways-cordova-apps.html
  - https://aerogear.org/docs/guides/aerogear-cordova/CordovaAndroidDevJBDS/

6. Autres approches

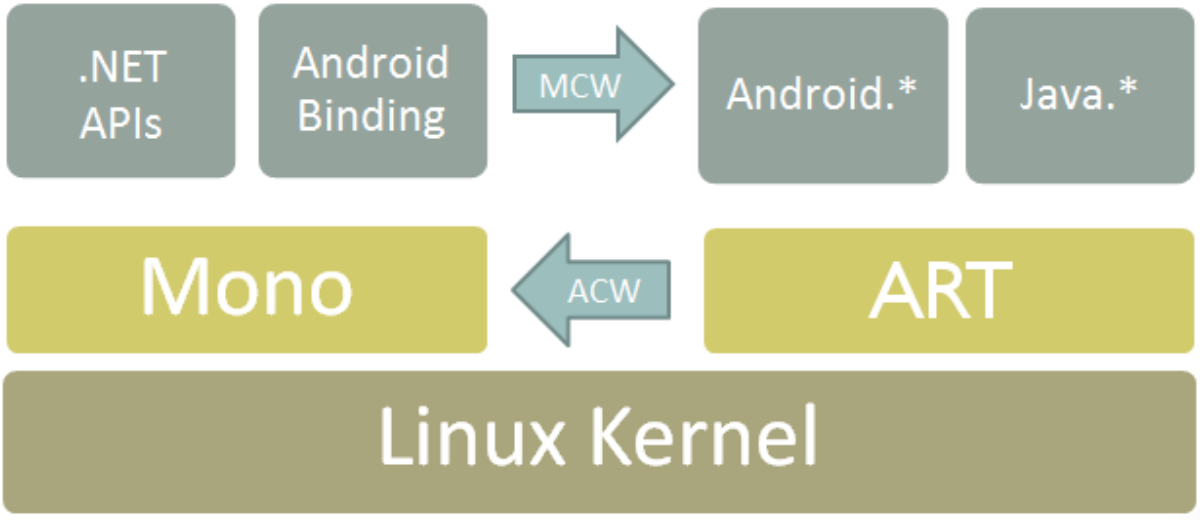
21. Appcelerator Titanium,

https://docs.appcelerator.com/platform/latest/#!/api



22. Xamarin

https://developer.xamarin.com/api/



7. Quelques liens pour exemples de code Cordova

- https://code.tutsplus.com/tutorials/an-introduction-to-cordova-example--cms-25328
- http://imikado.developpez.com/tutoriels/androidCordova/ma-premier-application/
- https://www.toptal.com/mobile/developing-mobile-applications-with-apache-cordova
- https://gist.github.com/dhavaln/2238017
- http://coenraets.org/blog/phonegap-tutorial/

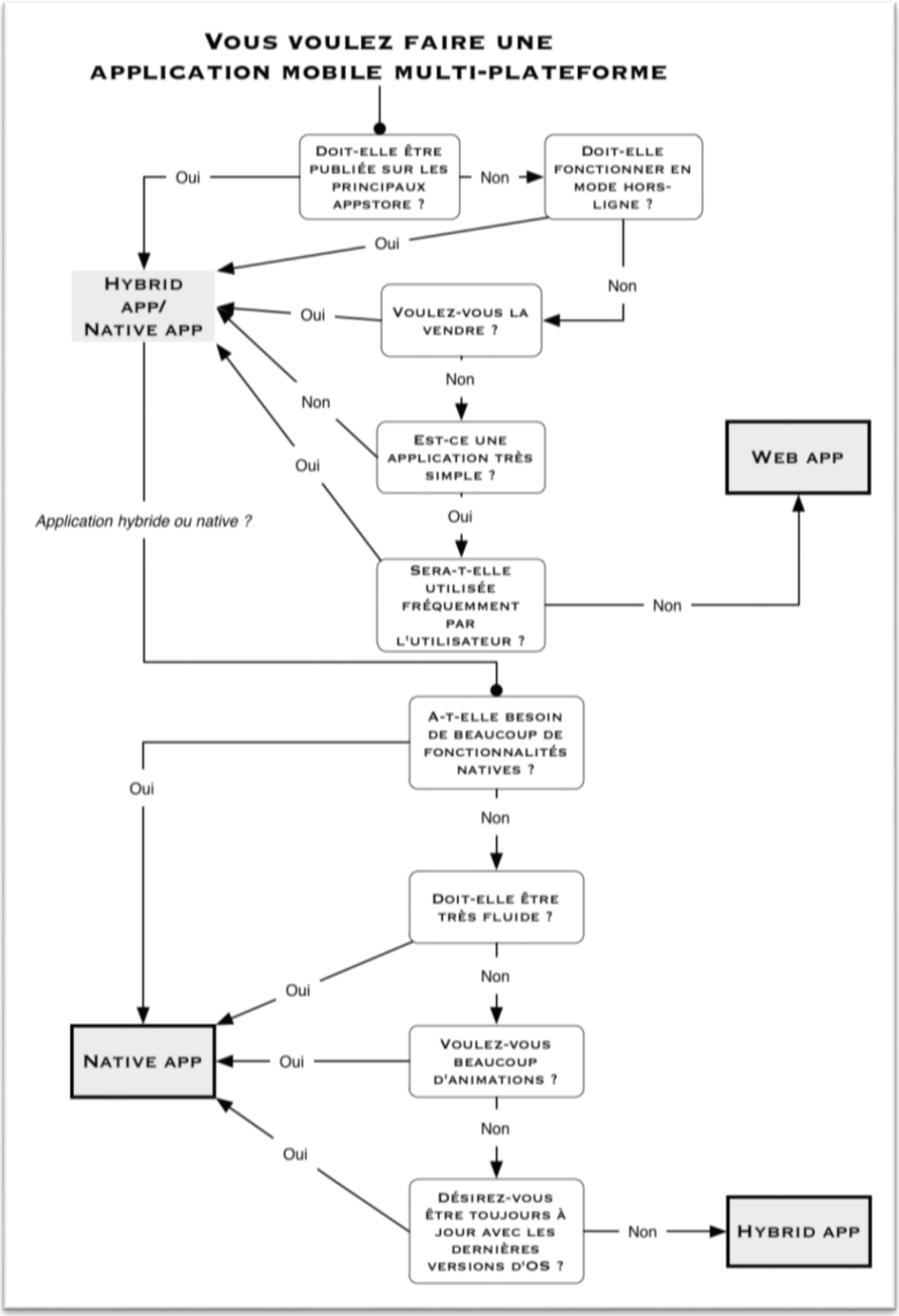
8. Annexes

23. Les Fonctionnalités supportées par PhoneGap par OS

	iOS iPhone / iPhone 3G	iOS iPhone 3GS and newer	Android	OS 5.x	OS 6.0+	WebOS	WP7	Symbian	Bada
ACCELEROMETER	✓	✓	✓	✓	✓	✓	✓	✓	✓
CAMERA	✓	✓	✓	✓	✓	✓	✓	✓	✓
COMPASS	✗	✓	✓	✗	✗	✗	✓	✗	✓
CONTACTS	✓	✓	✓	✓	✓	✗	✓	✓	✓
FILE	✓	✓	✓	✓	✓	✗	✓	✗	✗
GEOLOCATION	✓	✓	✓	✓	✓	✓	✓	✓	✓
MEDIA	✓	✓	✓	✗	✗	✗	✓	✗	✗
NETWORK	✓	✓	✓	✓	✓	✓	✓	✓	✓
NOTIFICATION (ALERT)	✓	✓	✓	✓	✓	✓	✓	✓	✓
NOTIFICATION (SOUND)	✓	✓	✓	✓	✓	✓	✓	✓	✓
NOTIFICATION (VIBRATION)	✓	✓	✓	✓	✓	✓	✓	✓	✓
STORAGE	✓	✓	✓	✓	✓	✓	✓	✓	✗

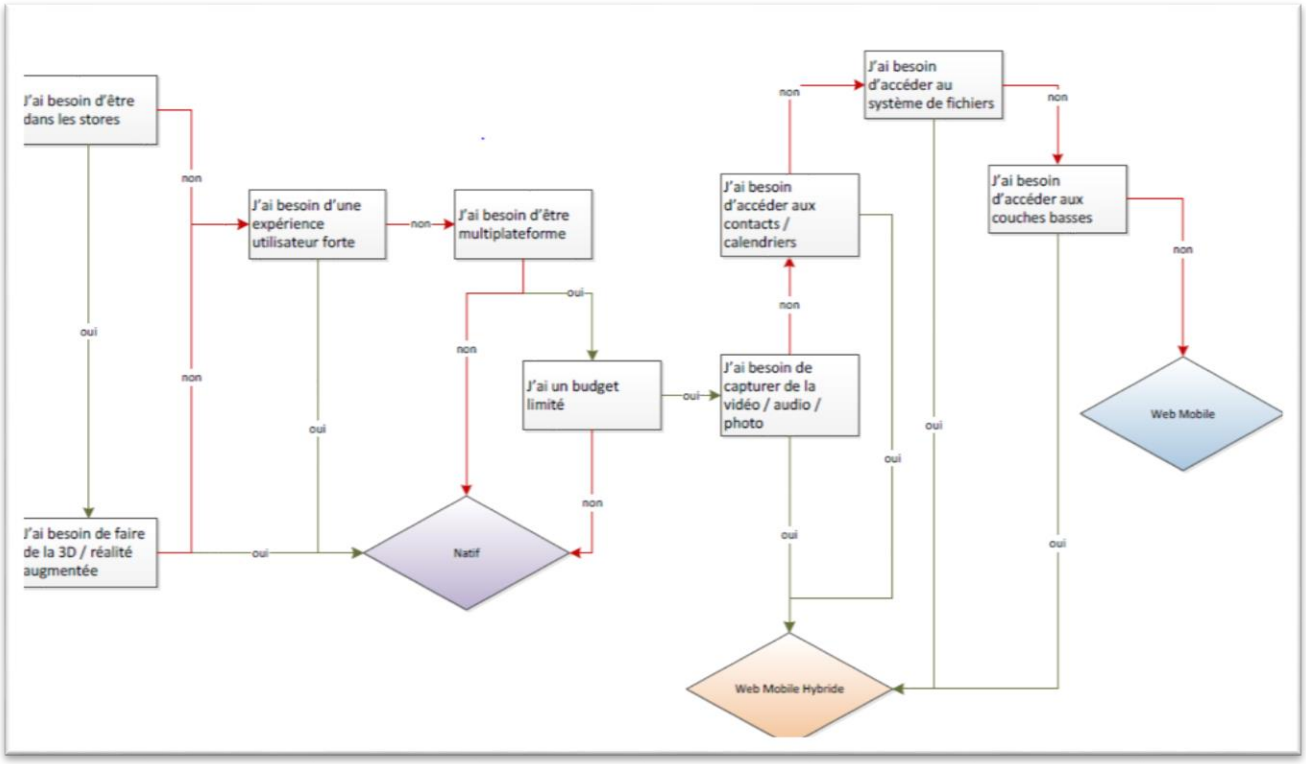
http://www.w3t.ca/nos-services/applications-web-mobiles/

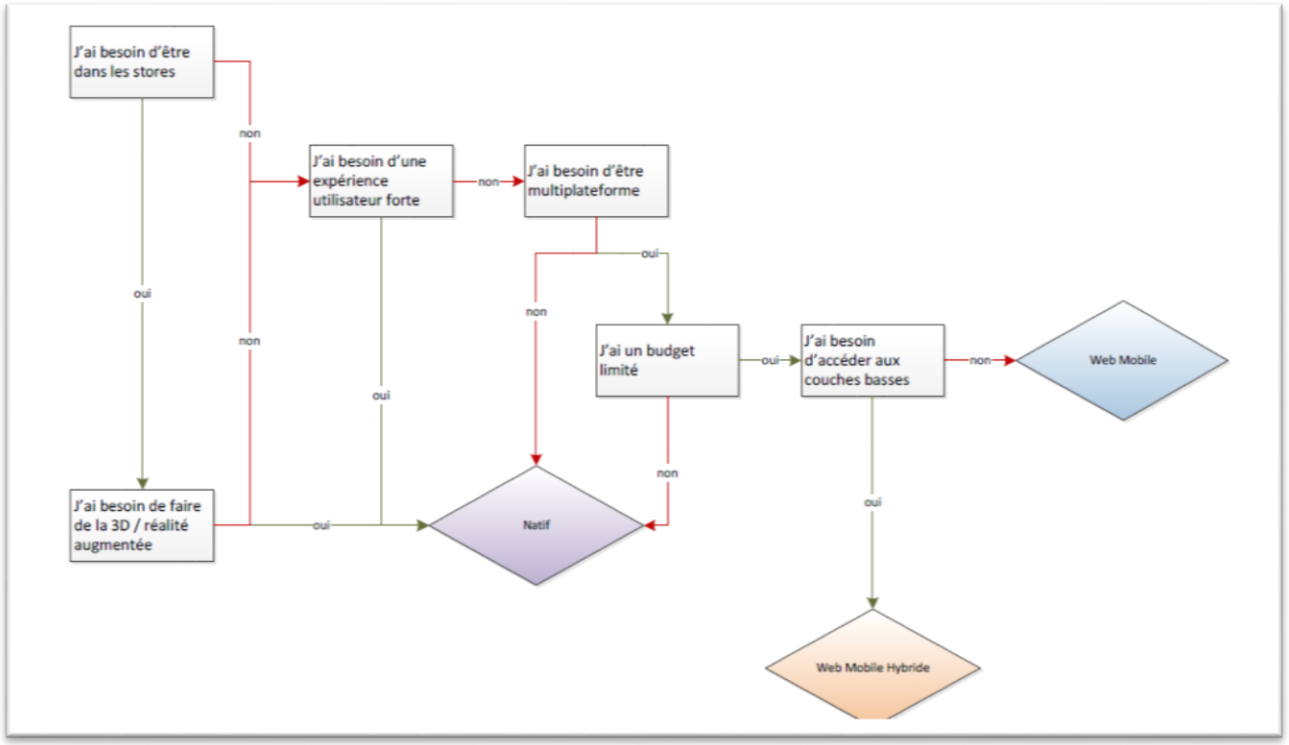
24. Schéma 1 d'aide au choix du type d'application



http://www.nrblog.fr/pepito/2012/04/14/web-application-application-native-ou-application-hybride-choix-cornelien/

1. Schéma 2 d'aide au choix du type d'application





Extrait Objet-Direct