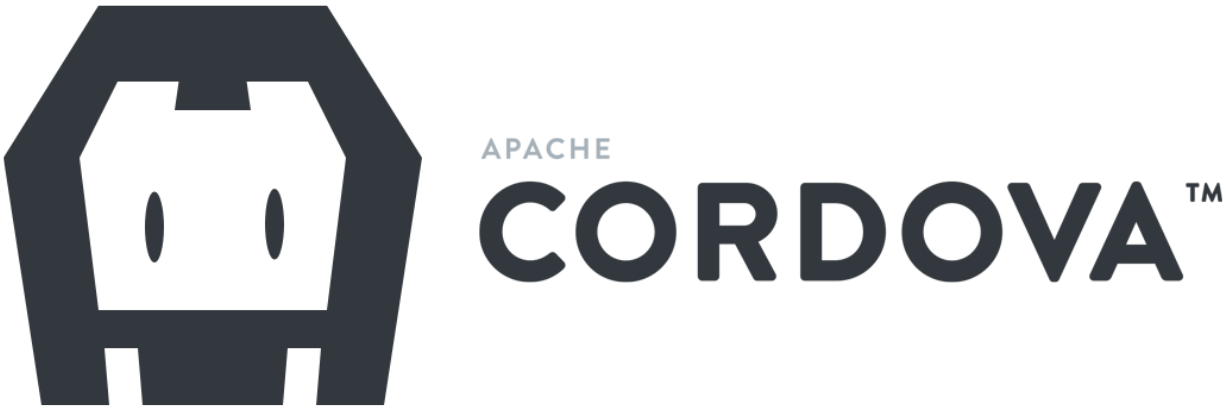
Formation Programmation Multiplateformes (Approche Hybride)

Rossi Oddet

Table des matières

Apache Cordova



Une plateforme permettant de construire des applications natives à l'aide des technologies du Web (HTML, CSS, JavaScript).

Quel intérêts ?

- Réutiliser la même base de code pour différentes plateformes mobiles (Android, iOS, Windows Phone, Blackberry, …)
- Trouver plus facilement des compétences pour réaliser des applications mobiles.
- Eviter d'apprendre dans le détail les technologies de tous les éditeurs

Historique

- `2008` : Phonegap créé par Nitobi lors d'un événement iPhoneDevCamp à San Francisco. Il devient un projet Open Source. Apple confirmera qu'une application Phonegap peut être installée sur iOS via la version 4.0 de la licence développeur
- `2011` : Adobe annonce l'acquisition de Nitobi le 4 octobre 2011. Adobe garde le nom Phonegap. Au même moment, le code open source de Phonegap est reversé à la fondation Apache sous le nom d'Apache Callback puis d'Apache Cordova
- `2012` : Phonegap annonce la sortie de PhoneGap Build en septembre 2012.

Principe

- Développer une application ou une partie d'application avec HTML 5, CSS et JavaScript
- Accéder aux fonctionnalités matériels ou logiciels du mobile (contacts, géolocalisation, appareil photo, …) via une API JavaScript fournie par Cordova
- Gérer plusieurs packages `natifs` qui seront déployés dans les différents stores des éditeurs.

Installation

- Installer le SDK Android et configurer un émulateur. Veillez à définir la variable d'environnement `ANDROID_HOME`.
- Télécharger et Installer NodeJS (http://nodejs.org/)
- Télécharger et Installer un client Git (http://git-scm.com/)
- Installer Cordova

```
npm install -g cordova
```

Créer un projet

cordova create

La commande `cordova create` génère un projet Cordova.

Syntaxe :

```
cordova create NOM_REPERTOIRE ID_PROJET NOM_AFFICHE_DE_L_APPLICATION
```

Exemple :

```
cordova create conference fr.conference Conference
```

Projet généré

```
/conference
    config.xml  // fichier de configuration du projet Cordova
    /hooks      // emplacement de scripts enrichissant les actions des commandes
Cordova
    /platforms  // fichiers liés aux plateformes (Android, iOS, ...)
    /plugins    // les plugins Cordova
    /www        // répertoire hébergeant l'application web
        index.html
        /css
            index.css
        /js
            index.js
        /img
            logo.png
```

Plateforme

cordova platform

La commande `cordova platform` permet de gérer les plateformes supportées par un projet Cordova.

Pour ajouter une plateforme d'exécution :

```
cordova platform add PLATEFORME_A_AJOUTER
```

Exemple :

```
cordova platform add android
cordova platform add ios
```

> Attention : ajouter une plateforme, implique souvent d'installer le kit de développement de la plateforme (Android SDK, iOS SDK, …).

Fichiers générés

- `assets` : les fichiers statiques du projet Android (*.html, *.css, *.js, *.png, …). Le répertoire « www » utilisé pour développer l'application y est copié.
- `cordova` : scripts de cordova
- `CordovaLib` : les librairies Android apportées par Cordova (okHttp & API Cordova pour Android)
- `libs` : Librairies à ajouter au Classpath du projet Android
- `platform_www` : fichier cordova.js
- `res` : répertoire de ressources Android
- `src` : Sources du projet Android. Il contient l'activité CordovaApp qui est l'unique activité utilisé par le projet. *
  - les fichiers habituels d'un projet Android (AndroidManifest.xml, build.gradle, settings.gradle, …)

Déployer

Pour construire un livrable, utiliser la commande `cordova build`.

```
cordova build
cordova build android
cordova build ios
```

Pour lancer l'application dans un émulateur / simulateur :

```
cordova emulate android
cordova emulate ios
```

Pour lancer l’application sur un appareil branché :

```
cordova run android
cordova run ios
```

TP #03-cordova

Objectifs

- Installer un environnement de développement Apache Cordova.
- Déployer une application simple sur plusieurs cibles.

Installer Apache Cordova

- Installer NPM et NodeJS : https://nodejs.org/ (version 5+).
    - A l'issue de l'installation, vérifier que la commande *npm* fonctionne. Exécuter les commandes suivantes :

```
node -v
npm -v
```

- Installer Apache Cordova via NPM

```
npm install -g cordova
```

- Visualiser la version installée de Cordova.

```
cordova -v
```

Installer le SDK Android

- Télécharger et installer Android Studio : http://developer.android.com/sdk/index.html.
- Lancer Android Studio.
    - Effectuer une installation standard.

Attention Android Studio requiert l'installation du JDK 7 minimum.

Création d'un projet Cordova

- Depuis la racine de votre dépôt Git, exécuter la commande :

```
cordova create 03-cordova fr.demo.cordova DemoCordova
```

Ajouter la plateforme Browser

La commande *cordova platform* permet de manipuler les plateformes supportées par le projet.

Lancer la commande *cordova platform list* qui établit la liste des plateformes installées et celle des plateformes disponibles.

```
Installed platforms:

Available platforms:
  android ^9.0.0
  browser ^6.0.0
  electron ^1.0.0
  ios ^6.1.0
  osx ^6.0.0
```

La liste affichée dépend de l'environnement d'exécution. Par exemple, sur Mac OSX, la plateforme Windows n'est pas disponible.

Pour installer une plateforme, utiliser la commande *cordova platform add*.

Exemple pour la plateforme `browser`.

```
cordova platform add browser --save
```

Le répertoire *platforms* est mise à jour.

```
/03-cordova
    ...
    /platforms
        /browser
            /cordova
            /platform_www
            /www
            browser.json
            config.xml
```

Lancer l'application dans un navigateur.

```
cordova run browser
```

Pour choisir le navigateur, utiliser l'option *--target*. Exemple :

```
cordova run browser --target=firefox
```

Ajouter la plateforme Android

```
cordova platform add android
```

Le répertoire *platforms* est mise à jour.

```
/03-cordova
    ...
    /platforms
        /android ①
            /assets ②
            /cordova ③
            /CordovaLib ④
            /libs ⑤
            /platform_www ⑥
            /res ⑦
            /src ⑧
            android.json
            AndroidManifest.xml
            build.gradle
            settings.gradle
            project.properties
```

① Ajout d'un répertoire *android* qui contient un projet Android (*natif*) représentant le projet Cordova actuel.

② Les fichiers statiques du projet Android (*.html, *.css, *.js, *.png, ...). Le répertoire « www » utilisé pour développer l'application y est copié.

③ Scripts de cordova.

④ Les librairies Android apportées par Cordova (okHttp & API Cordova pour Android).

⑤ Librairies à ajouter au Classpath du projet Android.

⑥ Où se trouve généré le fichier cordova.js.

⑦ Répertoire contenant les ressources Android.

⑧ Sources du projet Android. Il contient l'activité CordovaApp qui est l'unique activité utilisé par le projet.

Construire le projet Android

- Créer une variable d'environnement

La commande permet *cordova build* permet de construire le projet pour les différentes plateformes installées.

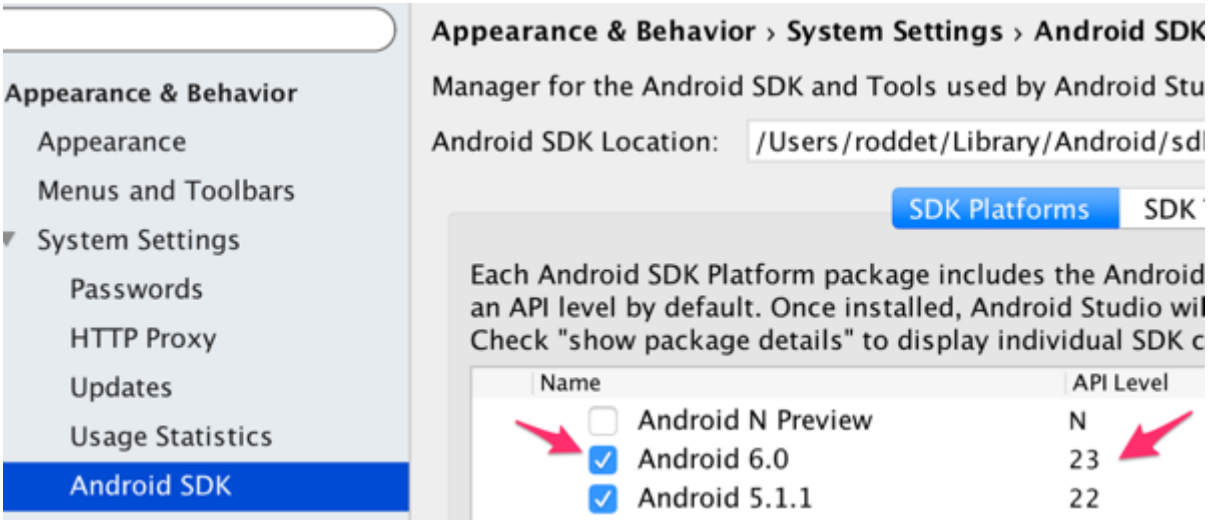
```
cordova build
```

Suivant votre installation d'Android Studio, vous pouvez rencontrer des erreurs liées au besoin de configuration d'une variable d'environnement `ANDROID_SDK_ROOT` ou au besoin d'installation de Gradle. Suivez alors les instructions.

- Si vous obtenez l'erreur suivante :

```
Error: Please install Android target: "android-XX".
```

- Lancer Android Studio.
- Cliquer sur le menu .
- Installer la cible Android demandée.



Déployer le projet dans un émulateur

La commande *cordova emulate* permet de déployer l'application dans un émulateur d'une platforme installée.

Déployer l'application sous Android.

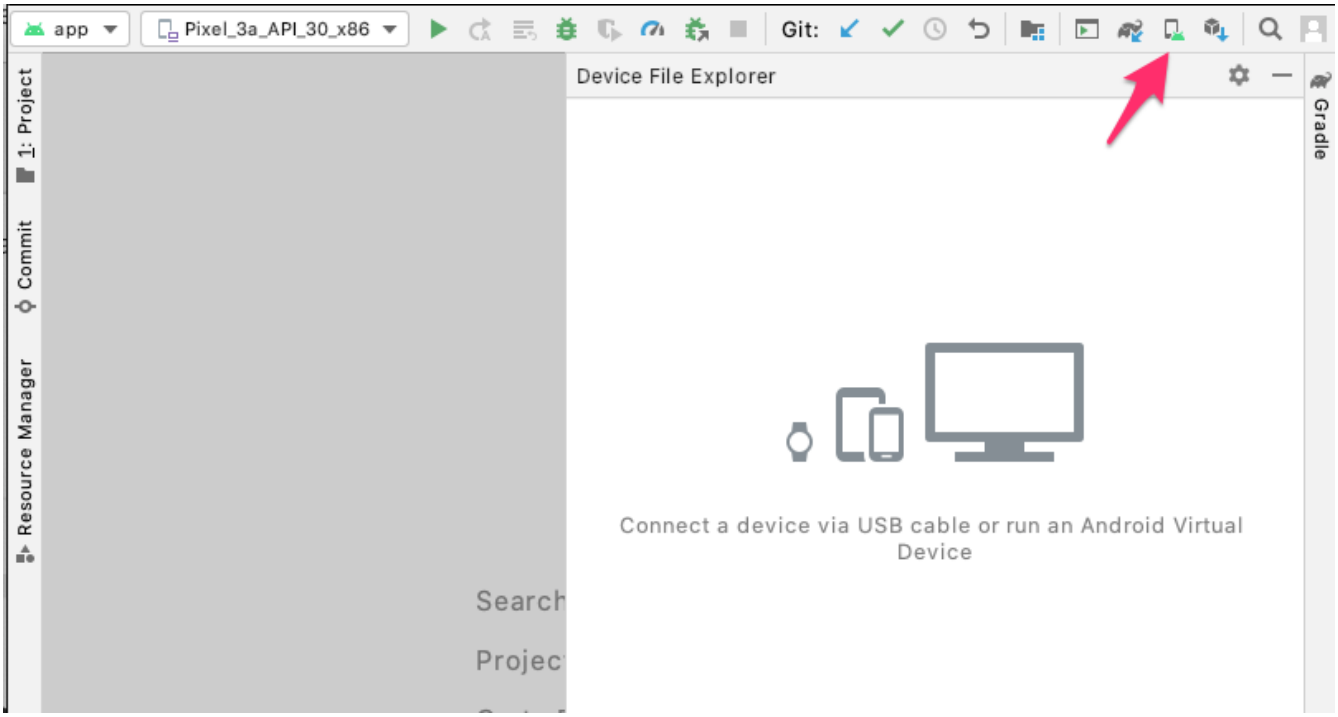
```
cordova emulate android
```

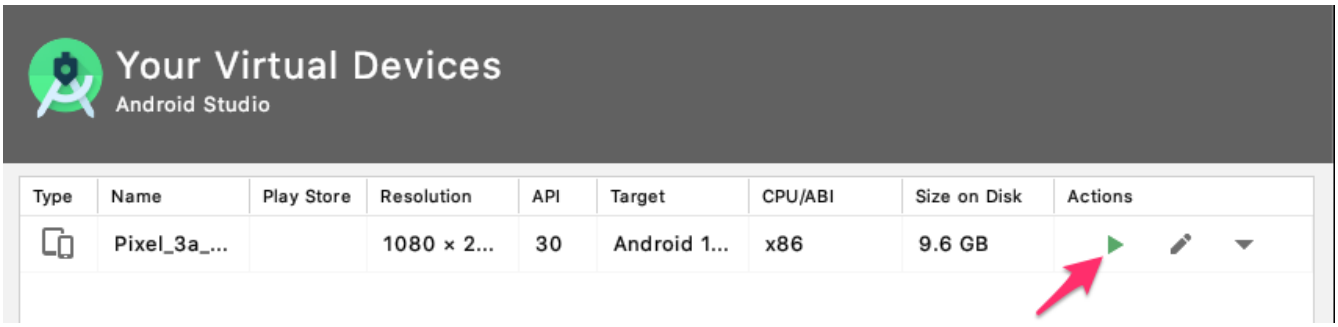
- Si l'émulateur ne se lance pas.

Les symptômes : le message ci-dessous s'affiche et l'émulateur ne se lance pas.

```
No emulator specified, defaulting to Nexus_5
Waiting for emulator...
```

Lancer l'émulateur depuis Android Studio.





Déployer le projet dans un appareil (optionnel)

La commande *cordova run* permet de déployer l'application dans un appareil d'une plateforme installée.

Brancher un appareil Android à votre ordinateur via un cable USB.

Lancer la commande.

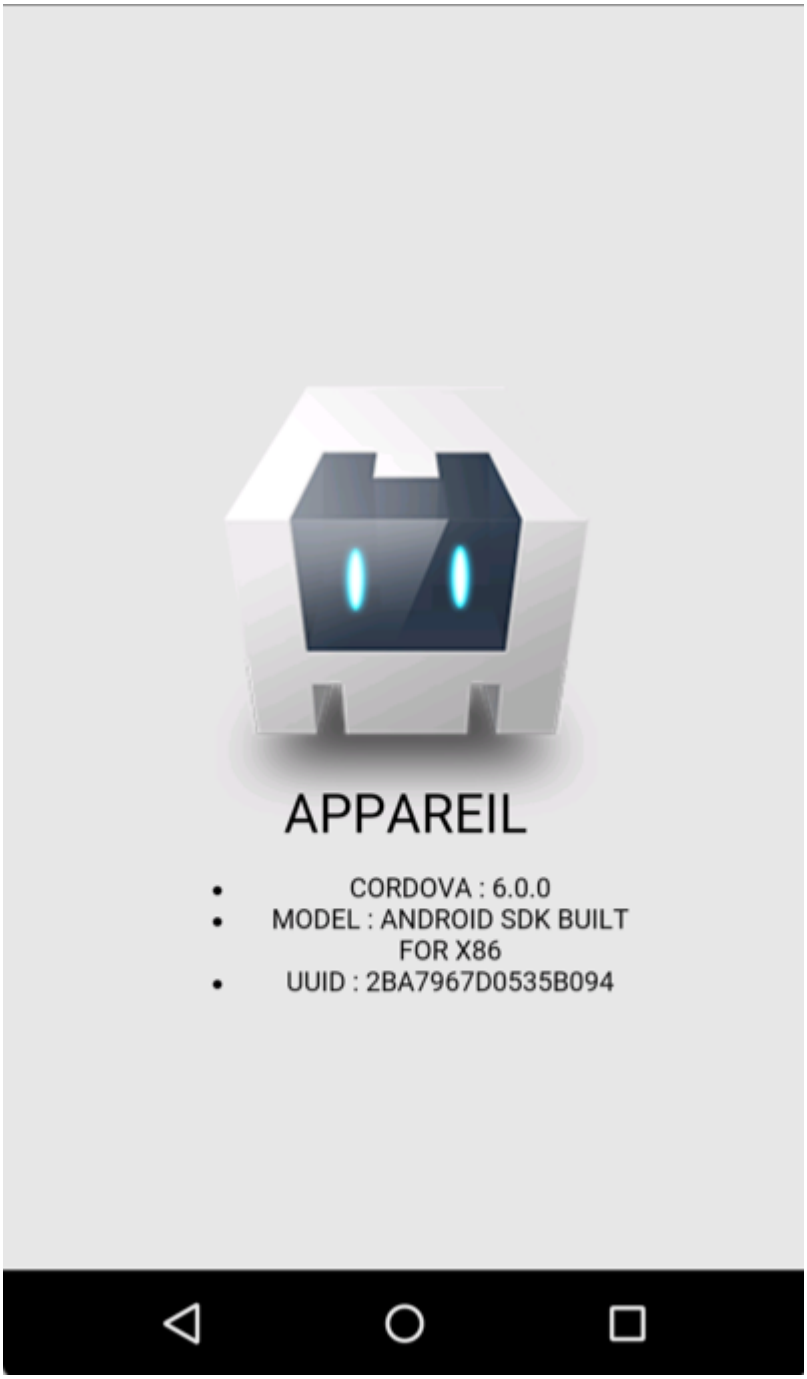
```
cordova run android
```

Plugins

- Ajouter le plugin `cordova-plugin-device` au projet

```
cordova plugin add cordova-plugin-device --save
```

- Compléter le projet pour afficher les informations du matériel.



Documentation officielle du plugin : https://cordova.apache.org/docs/en/latest/reference/cordova-plugin-device/

- Utiliser le plugin `cordova-plugin-network-information` pour afficher les informations de connexion re□seau.

Documentation officielle du plugin : https://cordova.apache.org/docs/en/latest/reference/cordova-plugin-network-information/

Capacitor

Capacitor permet à une application web d'utiliser les capacités natives d'un appareil (Android, iOS, Navigateur, ...) à partir d'une même base de code (JavaScript).

Caractéristiques de `Capacitor` :

- `Open Source` : licence MIT
- `Cross Platform` : une même base de code qui fonctionne sous iOS, Android, Electron et PWA.
- `Accès Natif` : un accès au kit de développement de chaque plateforme, y compis le déploiement dans les *stores*.

`Capacitor` se positionne comme le successeur d'Apache Cordova et offre une compatibilité avec les plugins Cordova.

Couplé aux PWA Elements, Capacitor permet à des plugins de fonctionner dans un navigateur (ou Electron).

La documentation officielle : https://capacitor.ionicframework.com/docs/.

Prérequis

- Node >= 8.6.0
- NPM >= 5.6.0
- Android 5.0+ (Lollipop) (89% du marché)

Pour le support d'Android, installer Android Studio.

Pour le support d'iOS : * Xcode >= 10 * CocoaPods (`xcode-select --install` puis `pod repo update`).

Initialiser Capacitor

Pour un projet web basé sur NPM, ajouter les dépendances :

```
npm i --save @capacitor/core @capacitor/cli
```

Puis initialiser Capacitor avec la commande `init`.

```
npx cap init
```

Cette commande génère un fichier `capacitor.config.json` de la forme suivante :

```
{
  "appId": "fr.devfest.blog",
  "appName": "App",
  "bundledWebRuntime": true,
  "npmClient": "npm",
  "webDir": "www"
}
```

Le paramètre `webDir` indique où se trouve les sources web à déployer.

Plateformes

Installation

Les plateformes à supporter peuvent être ajoutées via la commande `npx cap add` :

```
npx cap add android
npx cap add ios
npx cap add electron
```

> Attention : `npx cap add ios` ne fonctionne que sur Mac OS.

Ouverture des projets

La commande `npx cap open` ouvre le projet correspondant à la plateforme :

- `npx cap open ios` : ouvre le projet XCode.
- `npx cap open android` : ouvre le projet Android Studio.

Synchronisation des projets

En développant avec Capacitor, nous nous retrouvons avec un projet par plateforme.

Les projets peuvent être synchronisés avec la commande `npx cap copy`.

```
npx cap copy
npx cap copy android
```

Exécution des projets

- Projet Web

```
npx cap serve
```

- Projet Electron

```
npx cap open electron
```

- Projet Android

```
npx cap open android
```

Puis exécuter l'application depuis Android Studio.

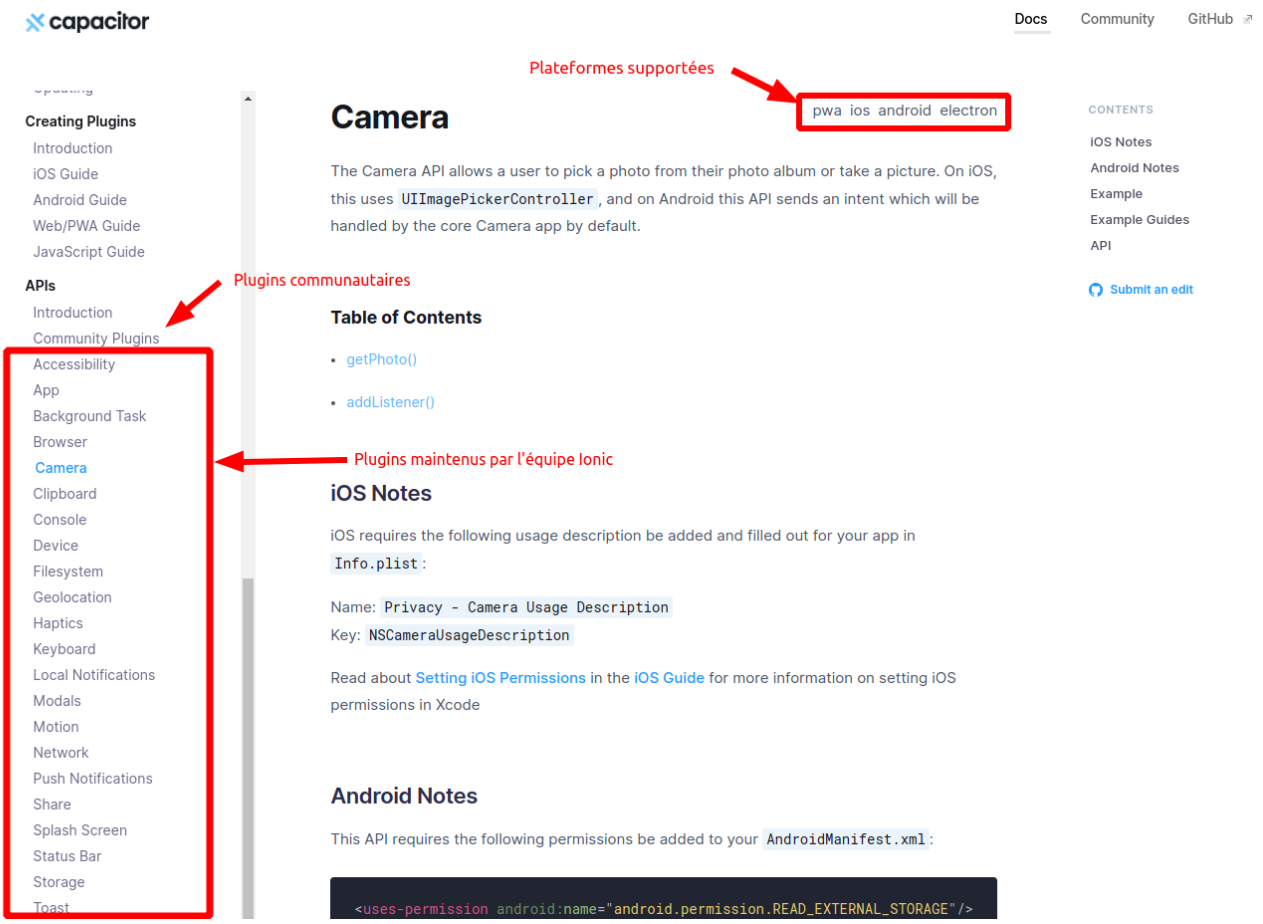
- Projet iOS

```
npx cap open ios
```

Puis exécuter l'application depuis Xcode.

Plugins

Capacitor vient avec un ensemble de plugins : https://capacitor.ionicframework.com/docs/apis



Cas des PWA

Projet avec modules ES2015

Si votre projet web est configuré pour gérer les modules ES2015, les plugins s'utilisent via des imports ES2015 :

```
import { Plugins } from '@capacitor/core';

const position = await Plugins.Geolocation.getCurrentPosition();
```

Projet sans système de module

Sans utiliser le système de module, il faut inclure programmatiquement le script `capacitor.js`, en suivant le mode opératoire suivant :

- passer la propriété `bundledWebRuntime` à `true` dans le fichier `capacitor.config.json`
- exécuter la commande `npx cap copy web`
- inclure le script `capacitor.js` avant votre script

```
<script src="capacitor.js"></script>
<script src="ton.js"></script>
```

PWA Elements

Pour une application Web ou Electron, des composants s'approchant du comportement natif mobile ont été développés.

Ils se regroupent dans le projet PWA Elements

Projet sans frameworks

Intégrer les *PWA Elements* consiste à alimenter la page HTML avec :

```
<script type="module" src="https://unpkg.com/@ionic/pwa-
elements@latest/dist/ionicpwaelements/ionicpwaelements.esm.js"></script>
<script nomodule src="https://unpkg.com/@ionic/pwa-
elements@latest/dist/ionicpwaelements/ionicpwaelements.js"></script>
```

Projet React

Mettre à jour le fichier index.js ou index.tsx :

```
import { defineCustomElements } from '@ionic/pwa-elements/loader';

ReactDOM.render(<App />, document.getElementById('root'));

// Invoquer le chargement des éléments après l'affichage de l'application
defineCustomElements(window);
```

Projet Angular

Mettre à jour le fichier `main.ts` :

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

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

import { defineCustomElements } from '@ionic/pwa-elements/loader';

if (environment.production) {
  enableProdMode();
}

platformBrowserDynamic().bootstrapModule(AppModule)
  .catch(err => console.log(err));

// Call the element loader after the platform has been bootstrapped
defineCustomElements(window);
```

Cas des plugins Apache Cordova

Capacitor offre une intégration *naturelle* avec les plugins Apache Cordova.

Deux étapes avant d'utiliser un plugin Apache Cordova :

- installer la dépendance vers plugin (exemple : `npm i cordova-plugin-device`)
- lancer la synchronisation (`npx cap sync`). Capacitor va automatiquement détecter les plugins Apache Cordova et va les intégrer au projet.

TP #04-capacitor

- Créer un répertoire `04-hellocapacitor`.
- Générer un fichier `package.json` pour le projet :

```
cd 04-capacitor
npm init -y
```

- Créer un répertoire `www` et copier y les fichiers du projet `02-ionic-core`.
- Ajouter Capacitor au projet :

```
npm i --save @capacitor/core @capacitor/cli
```

Puis initialiser Capacitor avec la commande `init`.

```
npx cap init
```

- Compléter le fichier `index.html` avec :

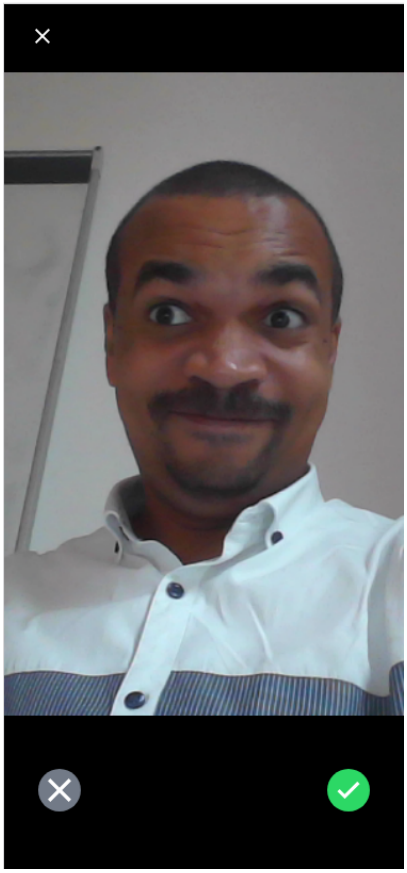
```
<script type="module" src="https://unpkg.com/@ionic/pwa-
elements@latest/dist/ionicpwaelements/ionicpwaelements.esm.js"></script>
<script nomodule src="https://unpkg.com/@ionic/pwa-
elements@latest/dist/ionicpwaelements/ionicpwaelements.js"></script>
```

- Passer la propriété `bundledWebRuntime` à `true` dans le fichier `capacitor.config.json`
- Exécuter la commande `npx cap copy web`
- Inclure le script `capacitor.js` avant votre script

```
<script src="capacitor.js"></script>
<script src="app.js"></script>
```

Prise de Photo

- Compléter le projet pour qu'au clic sur le bouton, une prise de photo soit déclenchée.

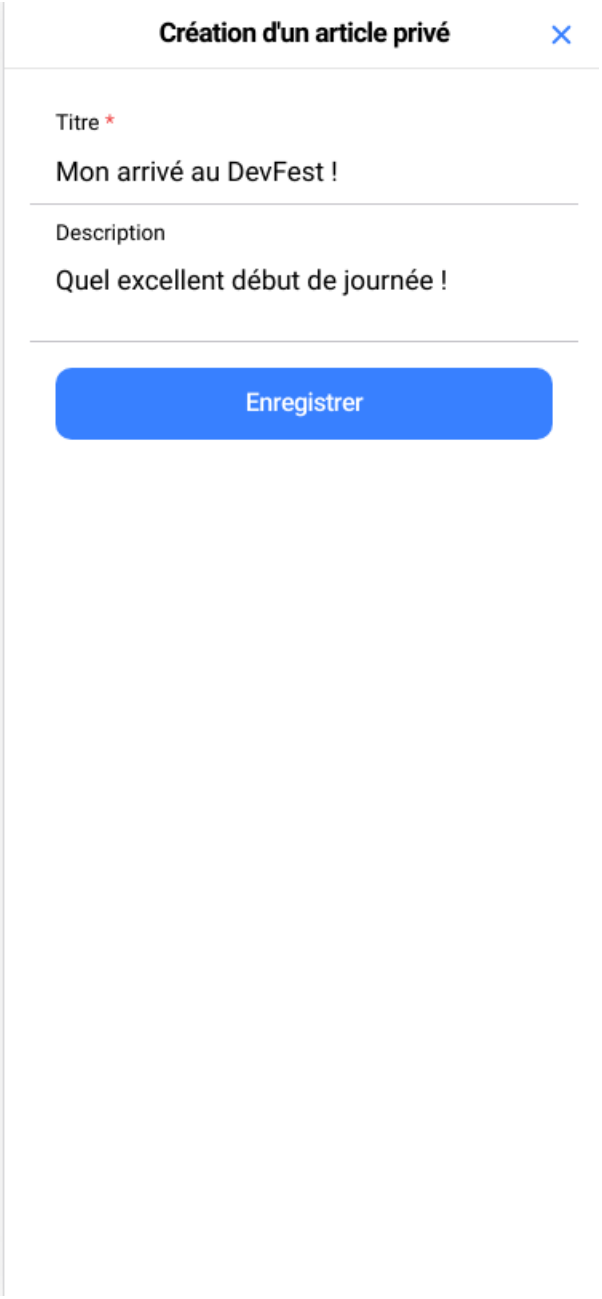


Utiliser l'API Camera : https://capacitorjs.com/docs/apis/camera.

le script `capacitor.js` inclut

Article privé

- Une fois la photo effectuée, la modale suivante s'affiche.



Utiliser le composant ion-modal.

- Puis une nouvelle entrée s'affiche dans la page d'accueil.

Stockage

Les données (photo, titre, description) doivent être stockées `en local`.

Utiliser l'API Storage : https://capacitor.ionicframework.com/docs/apis/storage.

Concernant la photo, pour simplifier l'implémentation, le stockage pourra s'effectuer au format `Base64Url`.

Electron

- Ajouter la plateforme Electron.

```
npx cap add electron
```

- Tester le projet sous Electron.

```
npx cap open electron
```

Android

- Installer Android Studio.
- Ajouter la plateforme Android :

```
npx cap add android
```

- Ouvrir le projet sous Android Studio.

```
npx cap open android
```

- Dans Android Studio, lancer le gestionnaire d'émulateur : `AVD Manager`.
- Créer et lancer un émulateur.
- Exécuter le projet dans l'émulateur Android.

Ionic Framework

Ionic Framework est un outil de construction d'application mobile hybride.

Le projet est open source.

Jusqu'à la version 3, Ionic était construit autour de Cordova et Angular.

Depuis la version 4, le framework est désormais agnostique :

- Ionic donne la possibilité d'utiliser différents frameworks JS (Angular, React, Vue) :
  - Angular : framework sur lequel Ionic est basé jusqu'à la version 3.
  - React
  - Vue
- L'utilisation des ressources matérielles peuvent se faire avec Cordova ou Capacitor.

Jusqu'à présent, nous avons exploré les composants graphiques `Ionic Core`.

Ionic CLI

Ionic CLI est un outil permettant de générer un projet Ionic et dispose de commandes utiles pour les développeurs.

En plus de la génération de projet, Ionic CLI fournit :

- un serveur de développement
- un processus de construction de livrable
- ...

Pour installer Ionic CLI :

```
npm install -g @ionic/cli cordova
```

Créer un projet Ionic

```
ionic start
```

Exemple d'exécution :

```
$ ionic start
Pick a framework! 

Please select the JavaScript framework to use for your new app. To bypass this prompt
next time, supply a value for the
--type option.

? Framework: Angular

Every great app needs a name! 

Please enter the full name of your app. You can change this at any time. To bypass
this prompt next time, supply name,
the first argument to ionic start.

? Project name: hello

Let's pick the perfect starter template! 

Starter templates are ready-to-go Ionic apps that come packed with everything you need
to build your app. To bypass this
prompt next time, supply template, the second argument to ionic start.

? Starter template: tabs
[INFO] Existing git project found (/Users/rossioddet/Dev/trainings). Git operations
are disabled.
 Preparing directory ./hello in 1.35ms
...
```

Pour utiliser Capacitor :

- à l'initialisation avec l'option `--capacitor`

```
ionic start monApp tabs --capacitor
```

- sur un projet Ionic existant :

```
ionic integrations enable capacitor
npx cap init
```

Serveur web de développement

Pour démarrer un serveur web de développement et visualiser l'application dans un navigateur :

```
ionic serve
```