Compilation Native et Progressive Web Applications

Web Avancé, Web mobile – M1IF13

Lionel Médini – Aurélien Tabard

Position du problème

- Offrir une expérience d’usage plaisante sur mobile
- Minimiser l’effort de développement
- Réduire le temps de démarrage

Les alternatives

- Natif mobile
- Compilation vers natif
- Hybrides / Web embarqué
- Progressive Web Apps

Les alternatives

- **Natif mobile**
- Compilation vers natif
- Hybrides / Web embarqué
- Progressive Web Apps

Application native (desktop ou mobile)

- Définition (rappel)
  - Application installée sur l'OS de la machine
- Intérêts
  - Accès natif aux fonctionnalités de la machine
  - APIs de développement complètes
  - Accès aux stores, validation des applications
  - Performance
  - Seules limitations : hardware / OS
- Inconvénients
  - Portabilité
  - environnements de développement parfois propriétaires

Application native (desktop ou mobile)

- Exemples
  - iOS SDK
  - Android SDK
  - ...

Les alternatives

- Natif mobile
- **Compilation vers natif**
- Hybrides / Web embarqué
- Progressive Web Apps

Compilation vers du code natif

- Définition
  - Application développée dans un langage non natif et un SDK dédié
- Intérêts
  - Même que le développement natif
    - Y compris : l'accès natif aux fonctionnalités de la machine
  - Possibilité de cibler plusieurs plateformes au moment de la compilation
- Inconvénients
  - SDK potentiellement plus limité
  - Du code spécifique doit de toute façon être écrit pour être proche des standards de chaque plateforme
  - Environnements de développement parfois propriétaires

Compilation vers natif

- Exemples
  - Xamarin (C#) compilation vers iOS ou Android
  - Unity
    - principalement pour les jeux,
    - interprétation du code plutôt que compilation
  - Appcelerator Titanium
  - ...

Les alternatives

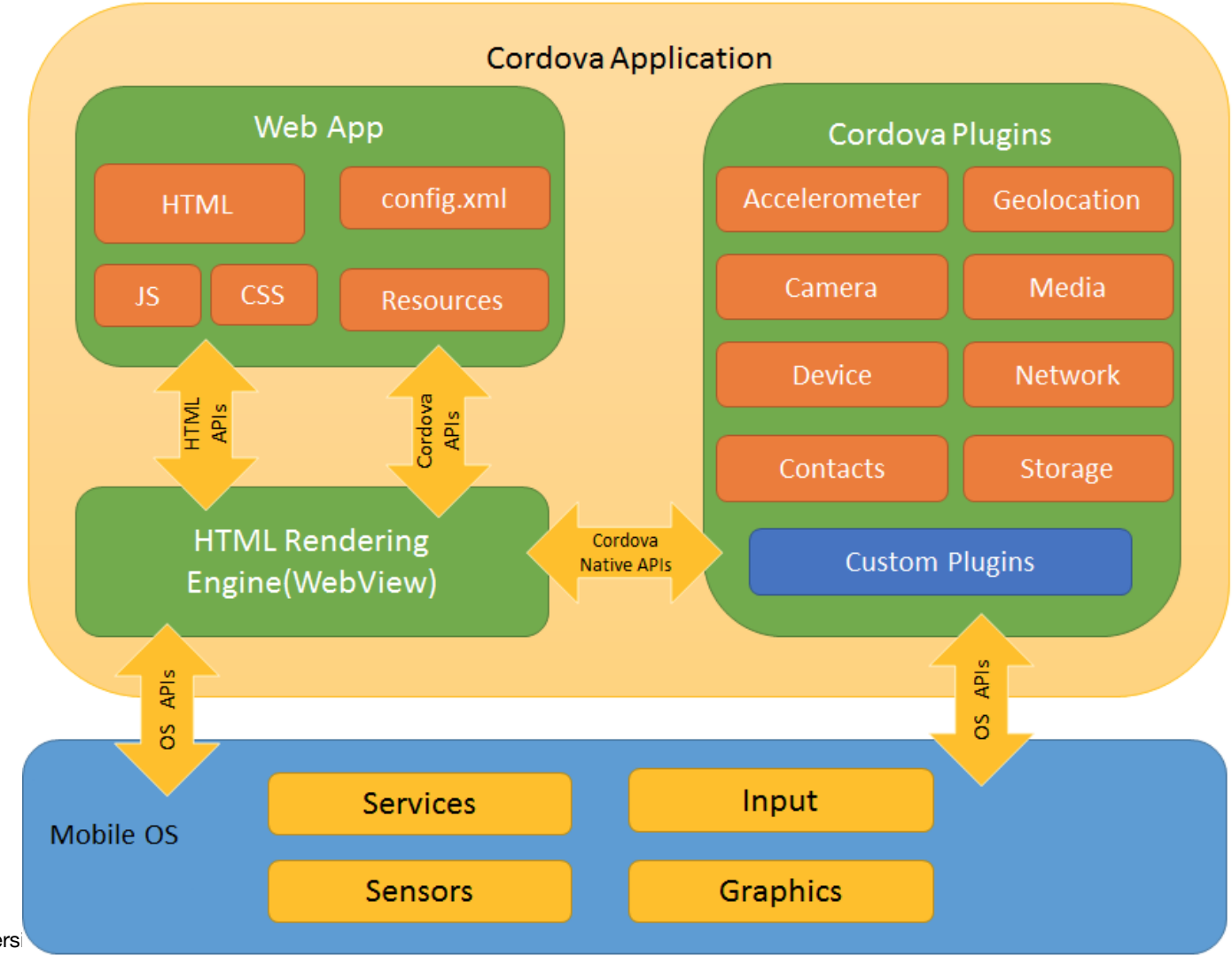
- Natif mobile
- Compilation vers natif
- **Web embarqué / Applications Hybrides**
  - Cordova
  - ReactNative
- Progressive Web Apps

Web embarqué / Applications Hybrides

- Définition
  - Application Web portée sur l'OS de la machine
  - Exécution en plein écran dans le navigateur natif de la machine (WebView)
- Intérêts
  - Un seul process de développement
    - Portage ± "tardif" vers les OS natifs
  - Des outils de portage libres, éventuellement adaptés aux frameworks
  - L'application peut être rendue disponible sur les stores
- Inconvénients
  - Performance par rapport aux applications natives
  - Le portage se complique pour les APIs "bas niveau"
  - La fin de la chaîne de portage peut être propriétaire (platform-dependent)

Cordova

- Principe
  - “Packager” une application Web dans une application native
  - Permettre l’accès à certaines API natives
    - API JS spécifiques (pas toujours conformes aux specs du W3C)
    - Les SDKs sont nécessaires pour accéder aux API (donc les licences associées)
- OS pris en charge
  - Android, iOS, BlackBerry, Windows Phone, Tizen, Firefox OS

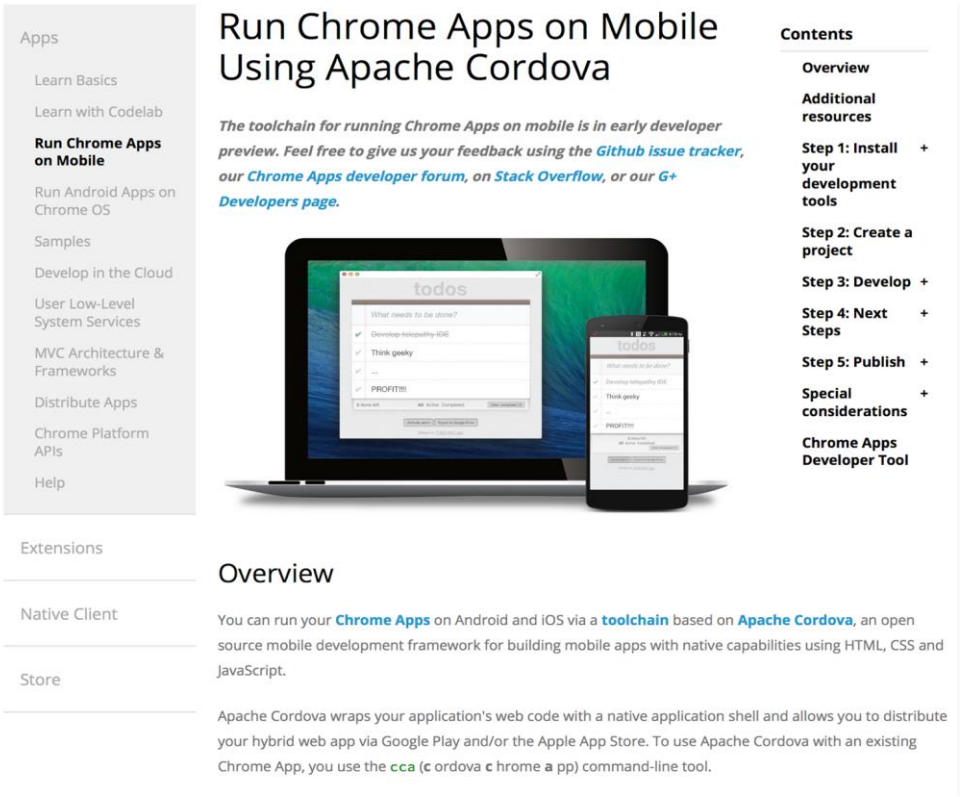

Cordova Application
Web App
HTML
config.xml
JS
CSS
Resources
Cordova Plugins
Accelerometer
Geolocation
Camera
Media
Device
Network
Contacts
Storage
Custom Plugins
HTML APIs
Cordova APIs
HTML Rendering Engine(WebView)
Cordova Native APIs
OS APIs
OS APIs
Mobile OS
Services
Input
Sensors
Graphics

Cordova : aspects pratiques

- Mode CLI (pas d'IDE)
  - Compilation, émulation
  - Utilisation de NPM pour gérer les plugins (console améliorée, accès aux capteurs…)
- Possibilité de faire tourner l'application
  - Sur un appareil connecté en USB (nécessite les drivers)
  - Sur un émulateur (Android Virtual Device) à créer avec le AVD manager

Cordova : exemples de surcouches

- PhoneGap
  - Développée par Adobe
  - Générique
  - Nombreux sous-projets pour accéder à des fonctionnalités spécifiques
- Ionic
  - Développée par Google
  - Dédiée au portage d'applications Angular
  - Permet d'utiliser des styles mobiles (UX plus proche des applications natives)



Les alternatives

- Natif mobile
- Compilation vers natif
- **Web embarqué / Applications Hybrides**
  - Cordova
  - **ReactNative**
- Progressive Web Apps

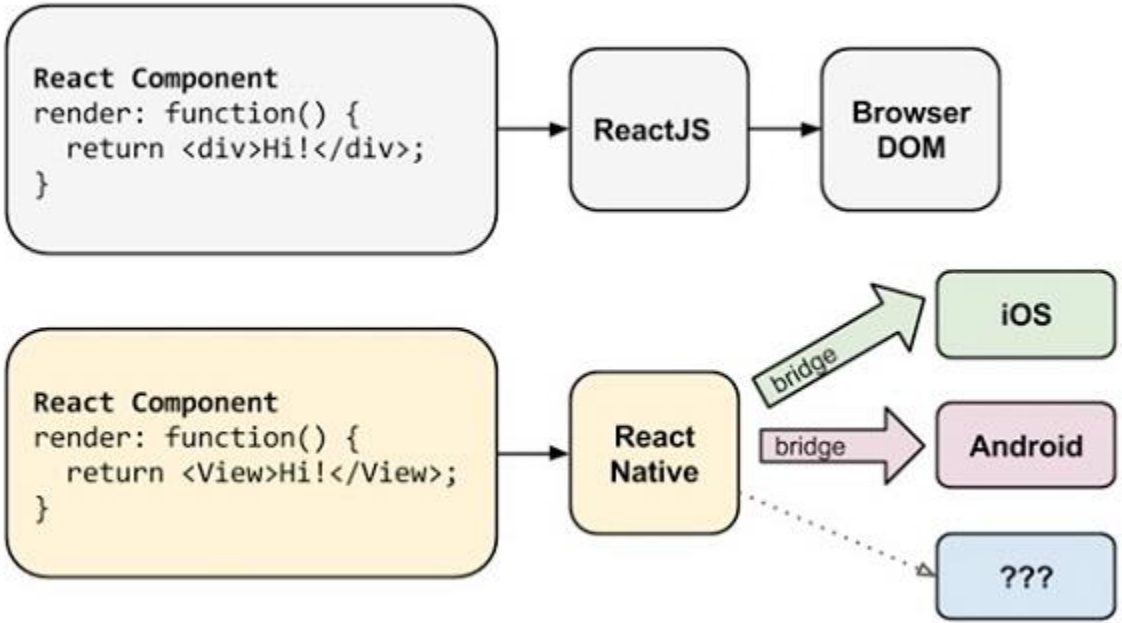
React Native

- Principes
  - Portage de Web apps en React
  - Utilisation de widgets natifs
  - Interprétation du javascript à la volée
    - i.e. pas de compilation en code natif à proprement parler
  - Tourne à une vitesse proche du natif

> “The working principles of React Native are virtually identical to React except that React Native **does not manipulate the DOM** via the Virtual DOM. It **runs in a background process** (which interprets the JavaScript written by the developers) directly on the end-device and communicates with the native platform via a serialisation, asynchronous and batched Bridge.”
> Source : WikiPedia

- OS pris en charge
  - Android, iOS, Universal Windows Platform (UWP)

React vs. ReactNative



https://unbug.gitbooks.io/react-native-training/content/12_how_it_works.html

Les alternatives

- Natif mobile
- Compilation vers natif
- Web embarqué / Applications Hybrides
- **Progressive Web Apps**

Rappel : Web applications

- Initiative du W3C, menée par le WebApp WG
- Différentes spécifications visant à standardiser la notion d'application Web
  - Web IDL
  - Web Components
  - Widgets (interfaces & packages)
  - DOM (Level 4, Events, shadow…)
  - Service Workers
  - Device API
  - Offline
  - …
- Spécifications / implémentations en cours

Progressive Web Apps

- Principes
  - Application Web
  - Utilisable en offline / “Frais” (toujours à jour)
  - Expérience proche des applications natives
    - Jeu de styles
    - Installable
  - Démarrage rapide
  - Peut être signée et déployée sur des stores
- Intérêts :
  - portabilité
  - environnements de développement Web classiques
- Inconvénients :
  - limitations techniques / de sécurité imposées par le navigateur

Progressive Web Apps

- Aspects techniques
  - Progressif (cache des éléments principaux)
  - Responsive
  - Sûr - seulement via HTTPS
  - Découvrable (manifeste)
  - Engageant (support des notifications)
  - Mise à jour en tâche de fond (Service worker)
- Mise en oeuvre
  - Tutoriel sur MDN

Progressive Web Apps

- Structure
  - Structure classique d'une application Web
    - HTML, JS, CSS…
  - Icône du logo
    - Pour démarrer la PWA, une fois installée
  - Service worker
    - Décrit quelles ressources cacher et comment le faire
  - Fichier .webmanifest
    - « Déclare » l'application et la rend découvrable
- Exemple : tuto sur MDN

Service Worker

- Objectif
  - Répondre aux nombreuses limites du App Cache
- Concept général
  - Proxy côté client
- Caractéristiques
  - Permet de contrôler les requêtes HTTPS venant des pages web
  - Synchronisation en tâche de fond, dans son propre thread
  - C’est un Worker : pas d’accès au DOM, communication par messages, API IndexedDB
  - Utilisation des promesses
- Exemple : tuto sur MDN

Conclusion sur les PWA

- Autres possibilités techniques depuis le tuto MDN
  - Installation
  - Notifications
- En règle générale
  - Il faut décider ce qu'on peut cacher et ce qui nécessite d'être en ligne
  - Structure de l'application importante pour séparer les ressources
    - En fonction de leur durée de vie
    - En fonction de leur origine
    - App shell
- Outil de mesure de performance : LightHouse