Offline Apps et Compilation Native

Programmation Web avancée et mobile – M1if13

Aurélien Tabard

La direction :

des applications Web accédant aux API natives

Aurélien Tabard - Université Claude Bernard Lyon 1

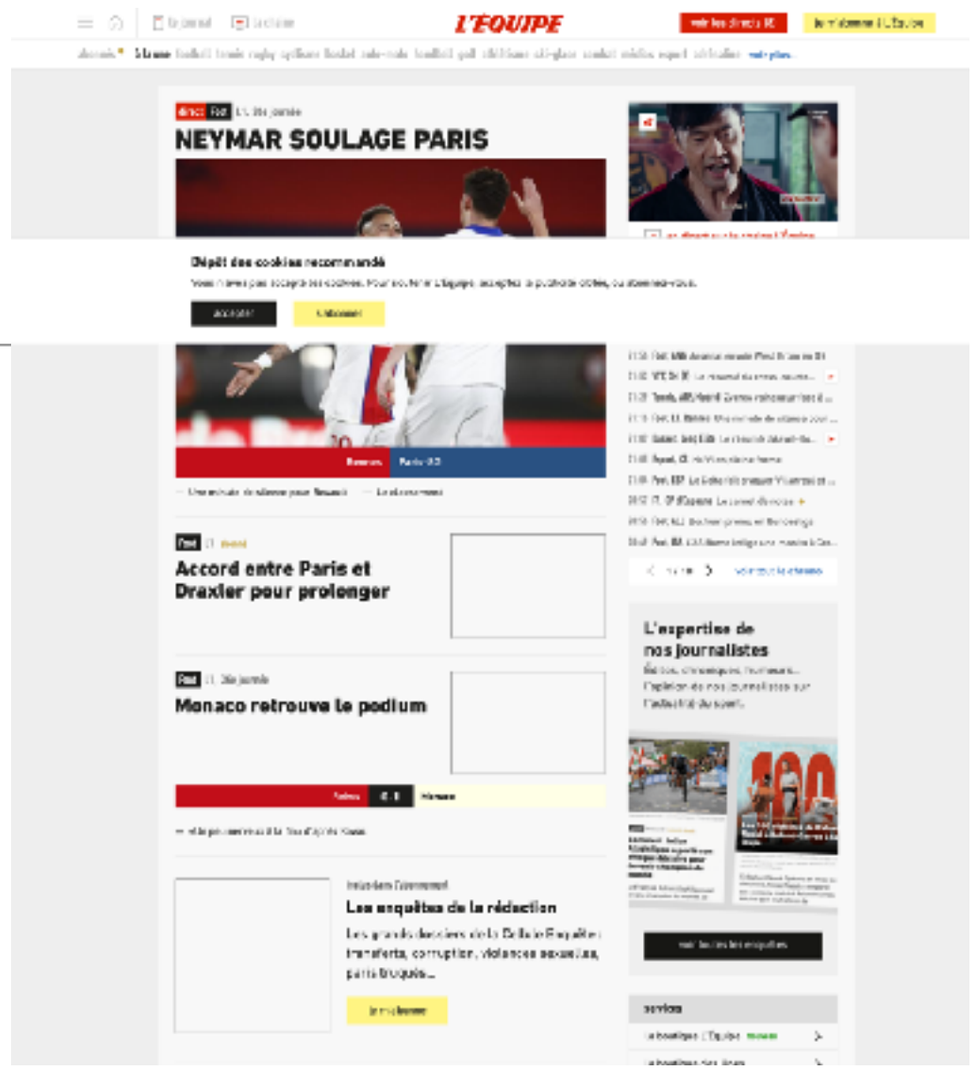
Les alternatives techniques pour le développement mobile

- **HTML 5**
  - Progressive apps + responsive + device APIs
- Natif mobile
  - iOS, Android, Windows 10 Mobile, Tizen, Sailfish OS…
- Cross-platform mobile
  - Flutter (Dart),
  - Xamarin Studio (C#)
  - Unity…
- Web embarqué
  - Cordova
  - Electron
  - Ionic (Angular + les autres), React Native, Quasar (Vue)
  - Capacitor

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



Trois principes

- Fiable
- Rapide
- Engageant

Les principes

- **Progressif** - Pour n'importe quel utilisateur indépendamment du navigateur : on améliore progressivement.
- **Responsif** - S'adapte au dispositif.
- Indépendant de la connectivité - **Fonctionne hors-ligne** ou en mauvaises conditions grace aux services-workers
- **App-like** - Impression d'utiliser une application, avec un modèle d'*app shell* qui séparer les fonctionnalisés de l'application du contenu.
- **À jour** - Toujours à jour grace au processus d'update des service worker.

Les principes

- **Sûr** - Servi en HTTPS.
- **Découvrable** - comme une “app” grace au manifest (standard du W3C) et le *registration scope* des service worker qui permettent aux moteurs de recherche de les trouver.
- (ré-)Engageant- Facilite sur le retour sur l’application via les **notifications push**.
- **Installable** - S’installe facilement sur le “home screen” des utilisateurs mobiles
- **Liable** - Permet de facilement partager une URL.

Les briques de base

- Design responsif
- Manifest
- Service-workers
- App-shell

Manifest

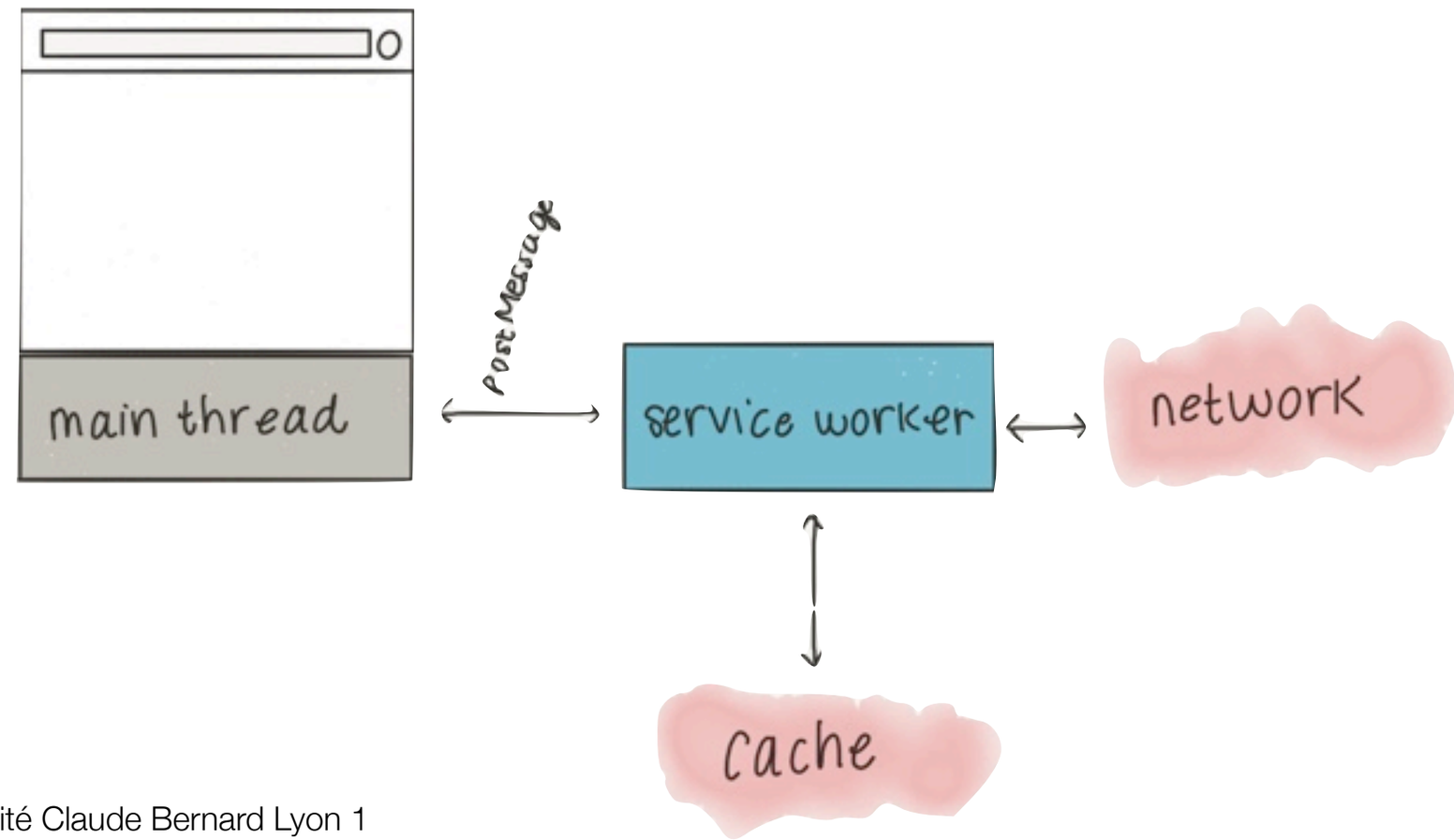
Permet de décrire l'application Web :

- Nom
- Icônes
- Url de départ
- Affichage
- Couleurs de thème
- Splash-screen
- ...

```
{
  "name": “Mon application",
  "short_name": “appli-pwa",
  "icons": [
    {
      "src": "/static/img/mySmallIcon.png",
      "sizes": "192x192",
      "type": "image/png"
    },
    {
      "src": "/static/img/myBigIcon.png",
      "sizes": "512x512",
      "type": "image/png"
    }
  ],
  "start_url": "/index.html",
  "display": "standalone",
  "background_color": "#000000",
  "theme_color": "#4DBA87"
}
```

Mise en cache et Service Worker

- Concept général : un proxy permettant de contrôler les requêtes http depuis venant des pages web.
- Pas d'accès au DOM, communication via passage de messages, mais accès à l'API IndexedDB



Service workers pour les PWA

- Gestion des notifications Push (mécanisme de souscription au serveur pour recevoir des notifications) -> thread séparé.
- Mécanismes de caching variés
  - Cache du App-shell
  - Offline app complète
  - Cache à la demande (du serveur via push, des utilisateurs)
  - Utilisation de workbox pour abstraire cela.
  - https://developers.google.com/web/tools/workbox/modules/workbox-strategies
- Voir les onglets Application et Lighthouse des dev-tools

Les alternatives techniques pour le développement mobile

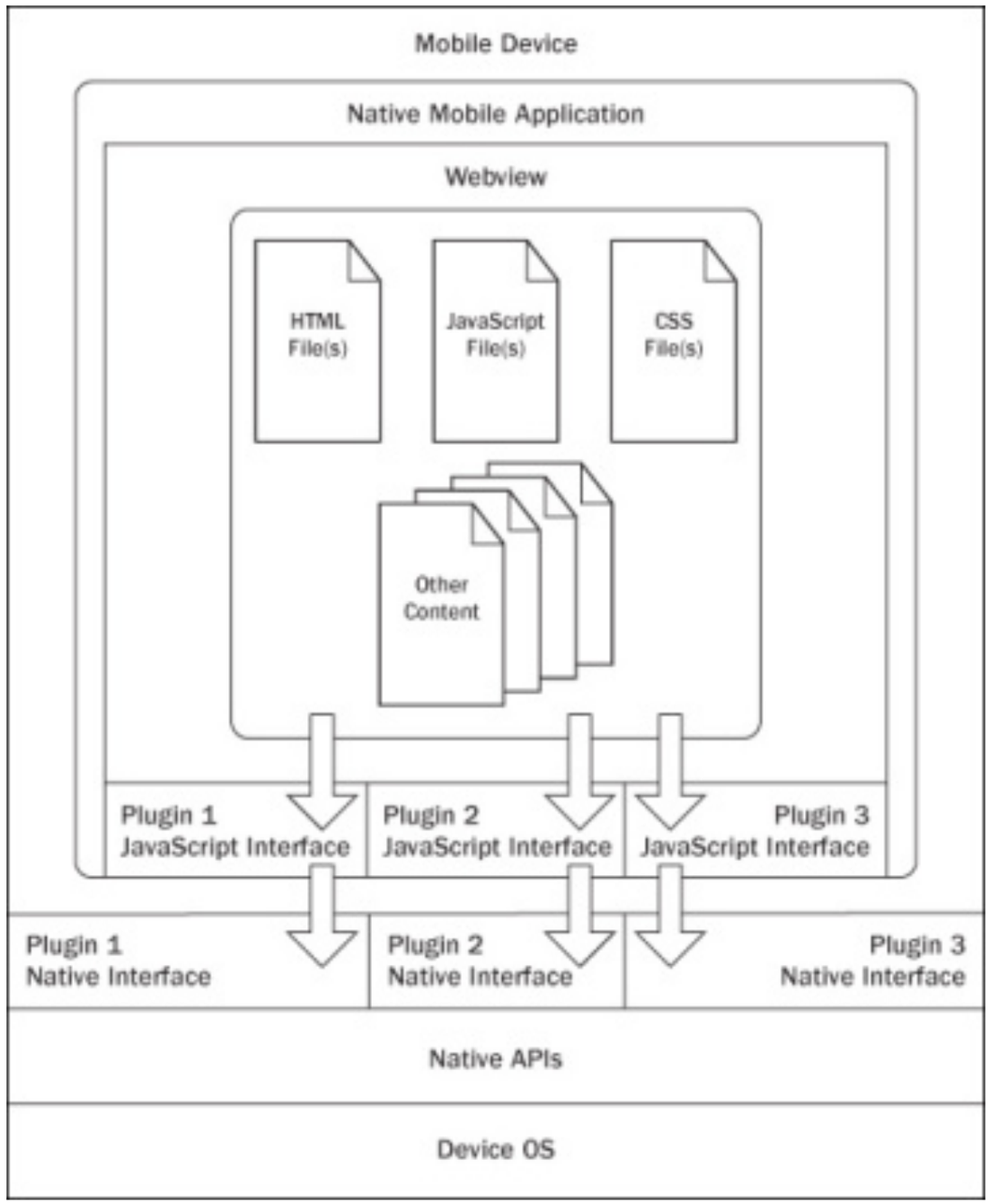
- Reponsive Web
  - Progressive apps + responsive + device APIs
- Natif mobile
  - iOS, Android, Tizen…
- Cross-platform mobile
  - Flutter (Dart),
  - Xamarin Studio (C#)
  - Unity…
- **Web embarqué**
  - **Cordova**
  - Electron
  - Ionic (Angular + les autres), React Native, Quasar (Vue)
  - Capacitor

Cordova : détails pratiques

- Principe :
  - écrire une application Web et la faire compiler pour plusieurs OS mobiles
- OS pris en charge :
  - Android, iOS, BlackBerry, Windows Phone, Tizen, Firefox OS
- Permet d'accéder 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)
- Exécution en plein écran dans le navigateur natif
  - le coeur est toujours une WebView.

- PhoneGap (http://phonegap.com) est la surcouche de Cordova la plus populaire. Racheté en 2011 par Adobe.

Architecture



Cordova : détails 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

Ionic, React Native, et Quasar

- Ionic : historiquement pour Angular basé sur Cordova
- React Native : compilation et génération de code Natif iOS/ Android
- Quasar : plus récent, pour vue, plus haut niveau (pwa, ou natif via cordova ou capacitor…)

Capacitor

Version “moderne” de Cordova par les personnes d’Ionic (moins de legacy)

Centré sur iOS et Android

Délègue la configuration et une partie du build aux plateformes (plutôt que d’abstraire) :

- Permet un contrôle plus fin et transparent.
- Intégration de code natif plus facile
- Meilleur support des pwa, peut cibler Electron