

Progressive Web Apps : Où en sommes-nous aujourd'hui ?

Alexandra Janin



MAKINA **CORPUS**

Sommaire

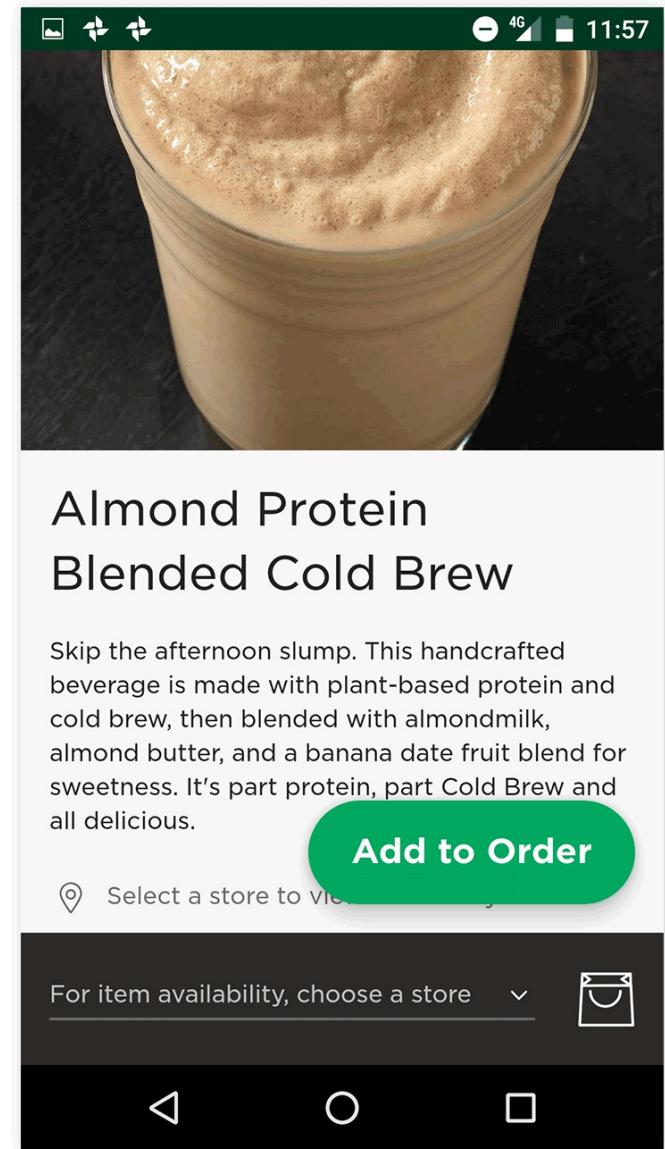
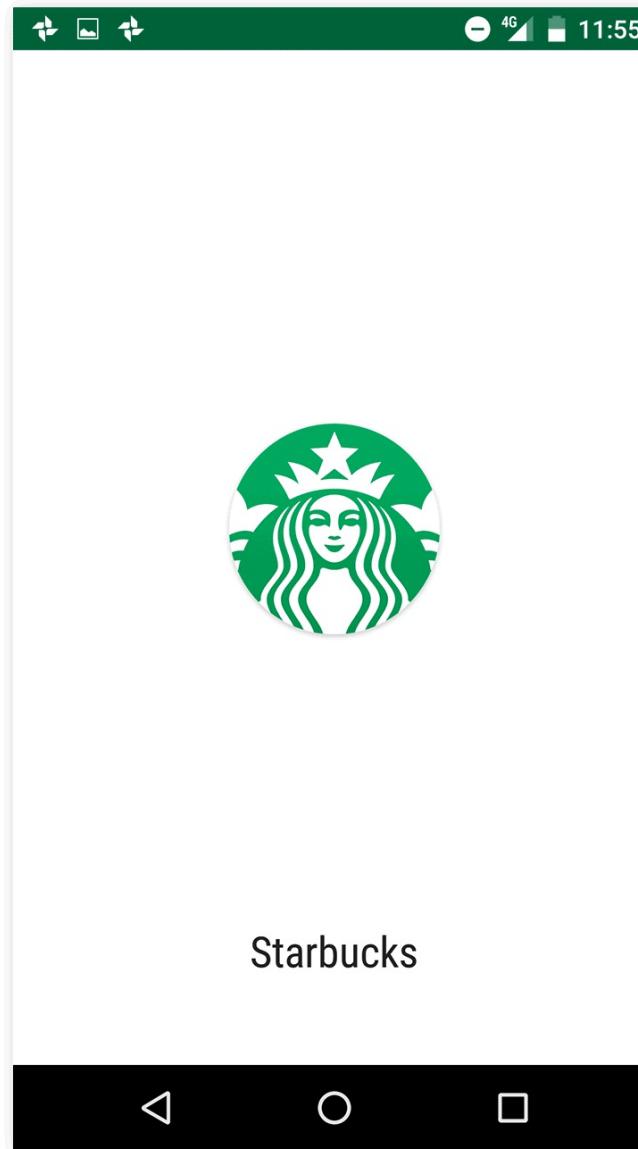
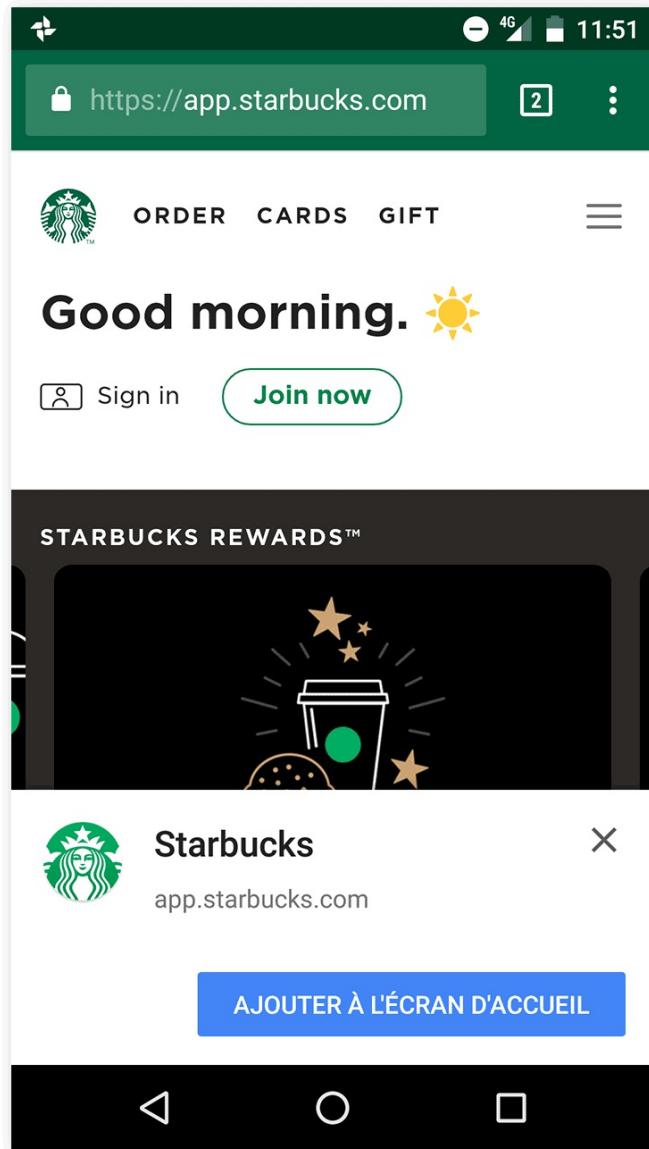
- Principe des PWA
- Apps hybrides / natives / web
- Compatibilité et limitations
- Icône et splash screen
- Service Worker
- Stockage des données
- Workbox
- Notifications push



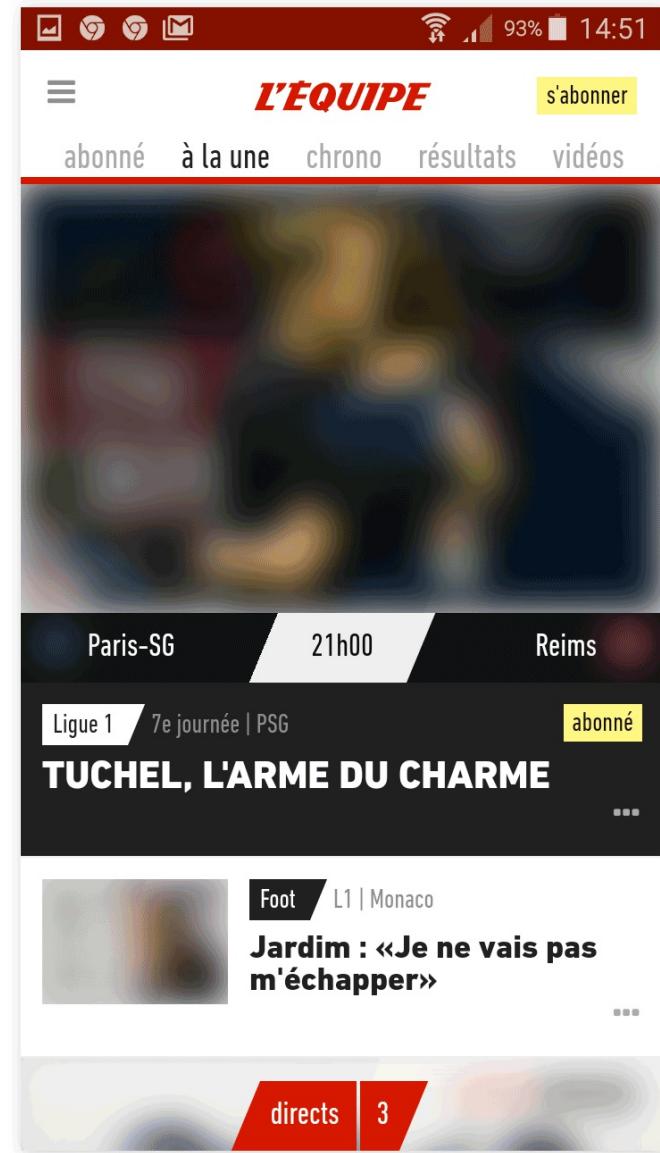
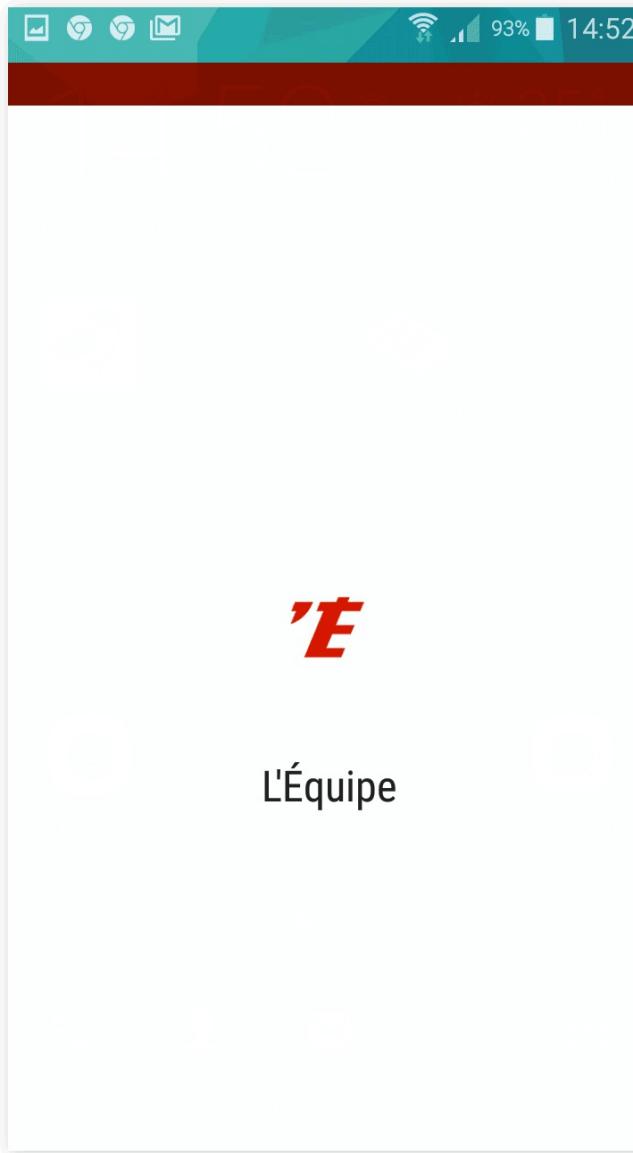
Présentation générale



Une application accessible par navigateur



Disponible hors ligne



Principe des PWA

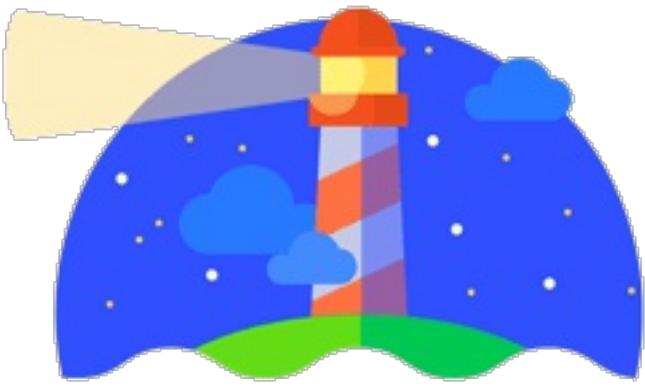
- Site répondant aux standards du web
- Accessible depuis un navigateur
- Amélioration progressive
- Mais qui répond à certains critères...

(PWA n'est PAS estampillé Google. Même si les Google Developers contribuent beaucoup à la documentation sur le sujet.)

Doit répondre à ces critères

- **Performance : fonctionnelle quelque soit la performance de l'appareil ou du réseau.**
- **Disponibilité : doit toujours être accessible, indépendamment de l'appareil ou de la qualité du réseau.**
- **Expérience similaire aux applications natives :**
 - Mode plein écran
 - Possibilité de l'installer
 - Push notifications

Comment savoir si mon app est une PWA ?



Lighthouse est un outil disponible dans les devtools de Chrome dans l'onglet Audit. Lighthouse permet de mesurer :

- Performance
- Accessibilité
- Bonnes pratiques
- SEO

Comparatif

Applications natives

- Développement spécifique Android (Java) / iOS (Swift)
- JS compilé en natif : React Native, Native Script

Applications hybrides

- Navigateur web "encapsulé" : Ionic, Cordova

Applications web

- Progressive Web App

Applications
Natives



Applications
Hybrides



Applications
Web



Les avantages des PWA

- Comportement similaire à une app native
- Sans les contraintes des apps mobiles (soumission aux stores, coût en stockage / mémoire)
- Coûts de développement moindres (pas de développements spécifiques à chaque plateforme)

Les inconvénients des PWA

- Encore mal supporté par Safari
- Pas le plus adapté pour les applications lourdes (cartographie, audio,...)

Compatibilité et limitations des PWA

navigator.serviceWorker

Namespace for page-side service worker API. Spec. Test.

The compatibility chart shows the following browser support levels:

- Chrome: 40+
- Firefox: 44+
- Opera: 27+
- Microsoft Edge: 4+ (Note: Requires "Enable service workers" in about:flags)
- Safari: 11.1+
- Internet Explorer: 16+

Edge: Requires "Enable service workers" in about:flags.

Is Service Worker Ready ?

Test si le navigateur supporte les SW :

```
if ('serviceWorker' in navigator) {  
  ...  
}
```

Principe de l'amélioration progressive

- **Doit fonctionner si le navigateur ne supporte pas les SW.**
- **Ne jamais faire en sorte qu'une requête ne fonctionne que si un SW est présent.**
- **Une PWA, c'est un confort en plus pour l'utilisateur**

Icône et splash screen

manifest.json



Rôle du manifest.json

Le fichier manifest.json permet la personnalisation des éléments suivant :

- **L'icône de l'application**
- **Le nom de l'application**
- **Le style du splash screen**
- **Le type d'affichage (plein écran, orientation, couleur du bandeau android,...)**

Il est déclaré via le html

```
<link rel="manifest" href="/manifest.json">
```

Le fichier manifest.json ressemble généralement à ceci :

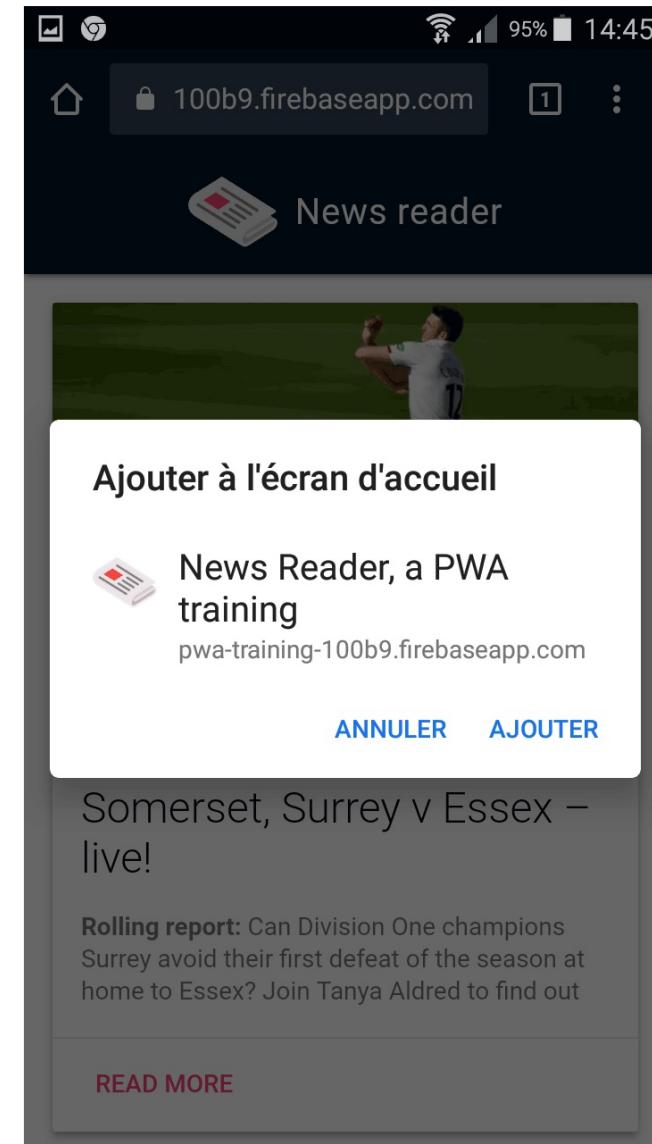
```
{  
  "name": "News Reader, a PWA training",  
  "short_name": "News Reader",  
  "start_url": "/",  
  "icons": [  
    {  
      "src": "/android-chrome-192x192.png",  
      "sizes": "192x192",  
      "type": "image/png"  
    },  
    {  
      "src": "/android-chrome-512x512.png",  
      "sizes": "512x512",  
      "type": "image/png"  
    }  
  ]  
}
```

[Référence du Web App Manifest \[en\], MDN](#)

Définir le nom et le point d'entrée de l'application



name



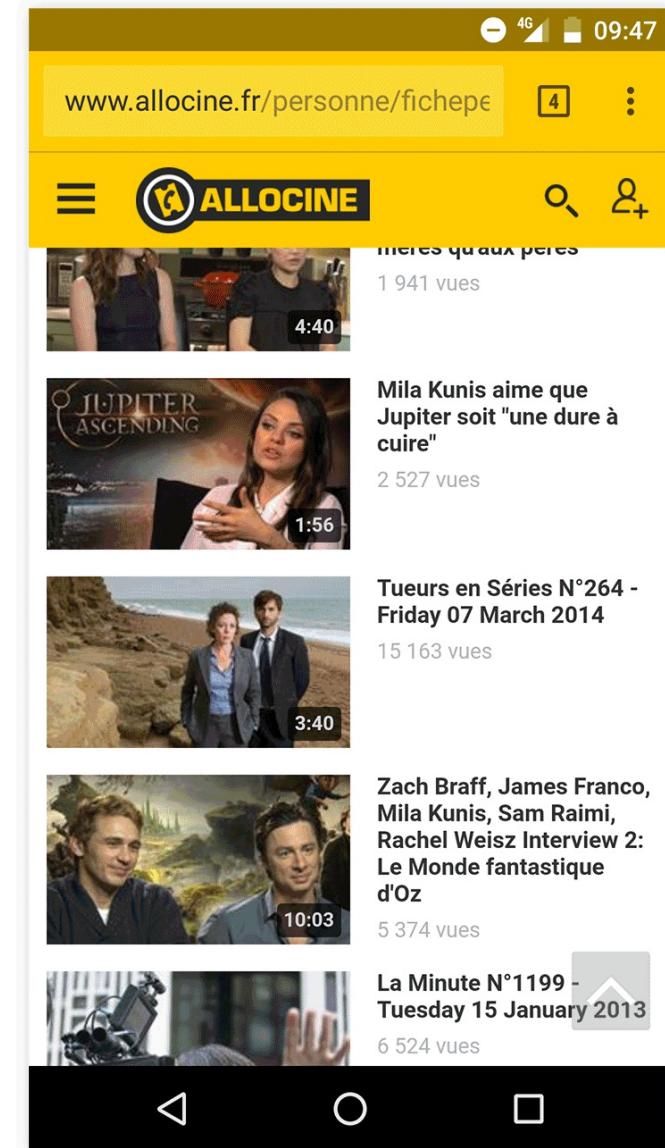
short_name



- **start_url : point d'entrée de l'application**

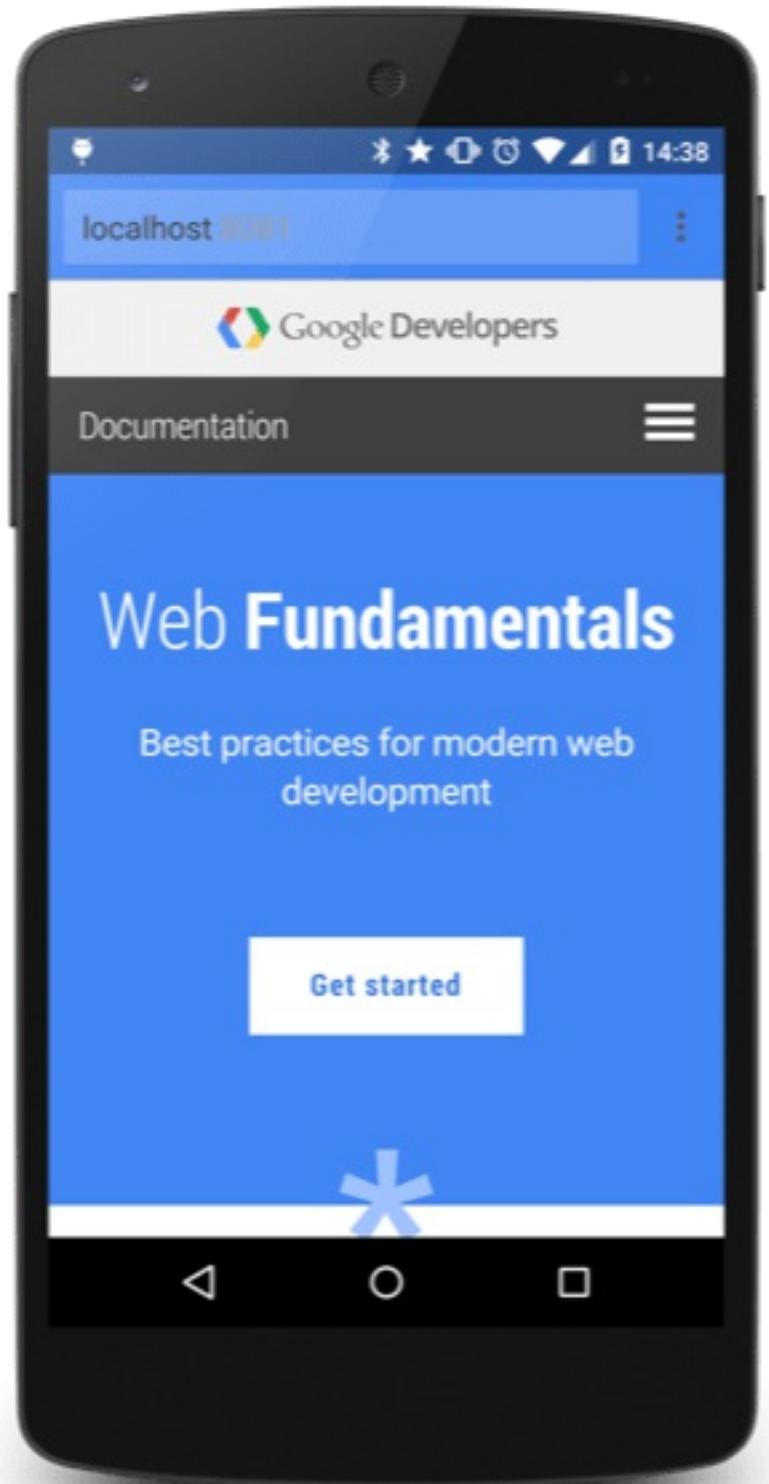
Définir la couleur principale : theme_color

Détecté automatiquement par Chrome, personnalisable.

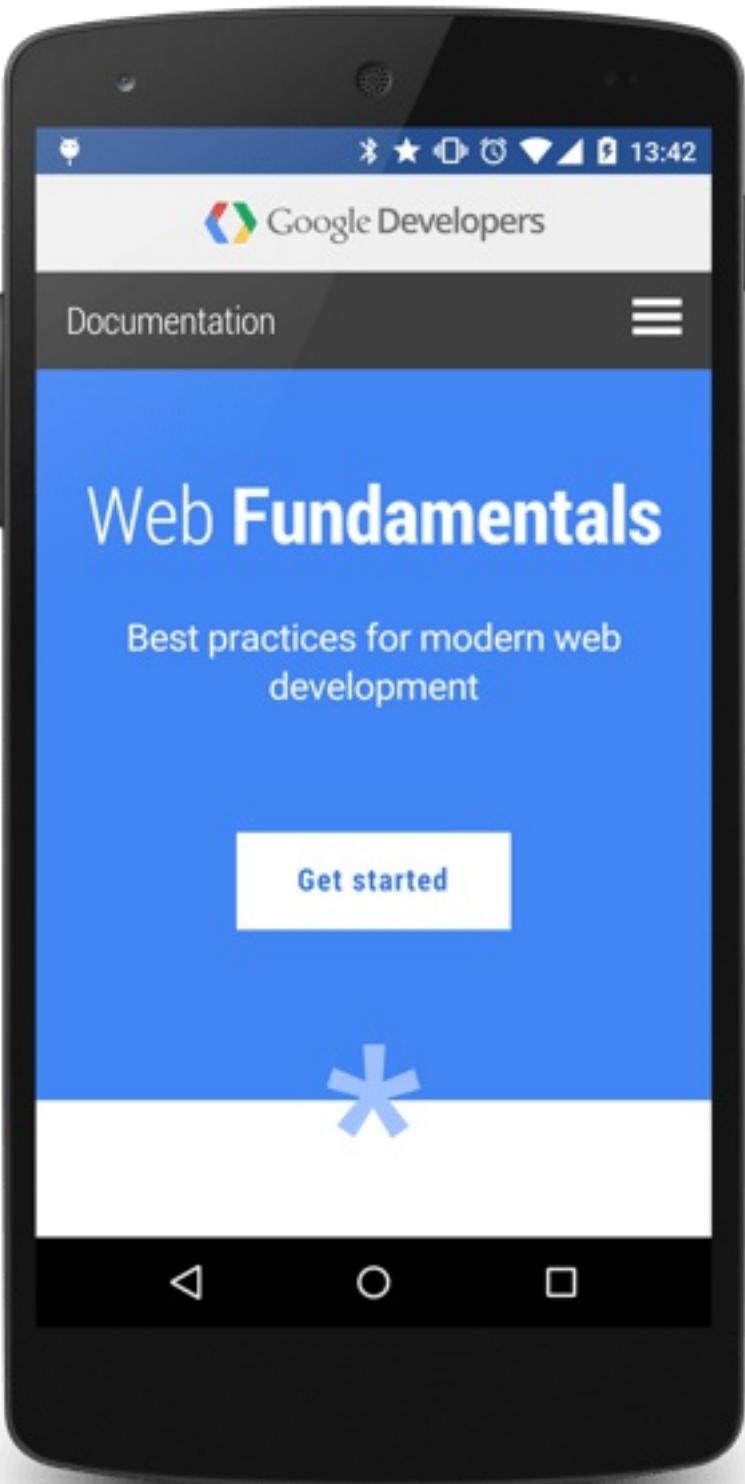


Définir le type d'affichage

- "display": "browser" (**à gauche**)
- "display": "standalone" (**à droite**)



"display": "browser"



"display": "standalone"

Vérifier le manifest.json

On peut vérifier le manifest et les icônes en allant dans l'onglet Application > Manifest.



Elements

Console

Sources

Network

Application

>>



Application

Manifest

Service Workers

Clear storage

Storage

Local Storage

Session Storage

IndexedDB

Web SQL

Cookies

Cache

Cache Storage

Application Cache

Frames

top

App Manifest

[:8080/manifest.json](#)

Identity

[Add to homescreen](#)

Name News Reader, a PWA training

Short name News Reader

Presentation

Start URL

Theme color #001529

Background color #001529

Orientation

Display standalone

Icons

192x192

image/png

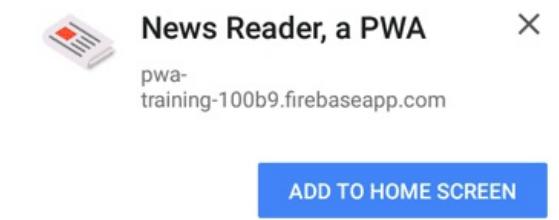
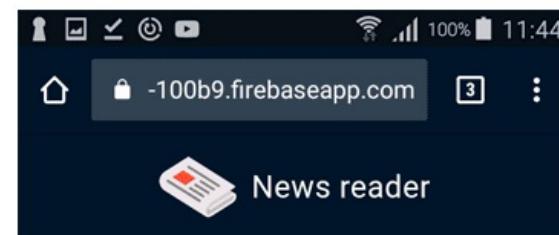


Bandéau d'installation de l'app

Le bandeau d'installation de l'app s'affiche automatiquement si tous les critères sont bien remplis.

The screenshot shows the 'Application' tab of the Service Worker registration interface. It includes sections for 'Manifest', 'Service Workers', and 'Clear storage'. The 'Identity' section shows the app name as 'News Reader, a PWA training'. A prominent red box highlights the 'Add to homescreen' button.

Vérifier l'affichage de l'app banner



Les Service Worker



Principe du Service Worker

Aident à répondre au critère de disponibilité

3 concepts à retenir :

- Thread en arrière-plan (car c'est le principe d'un Worker)
- Rôle de proxy
- Communication avec le Javascript du site

Ils fonctionnent uniquement sur HTTPS pour des raisons de sécurité.

Les outils

- **Service Worker Precache**
- **Service Worker Toolbox**
- **Workbox**

Les framework (React, Vue, Angular,...) disposent en général déjà de Workbox ou d'une implémentation pour gérer les SW.

Déclaration du Service Worker avec register

```
// index.js

if ('serviceWorker' in navigator) {
  navigator.serviceWorker
    .register('/sw-test/service-worker.js', {
      scope: '/sw-test/'
    })
    .then(registration => {
      // Le Service Worker est déclaré !
      console.log('Registration OK');
    })
    .catch(error => {
      // Il y a eu un problème
      console.error('Erreur: ', error);
    });
}
```

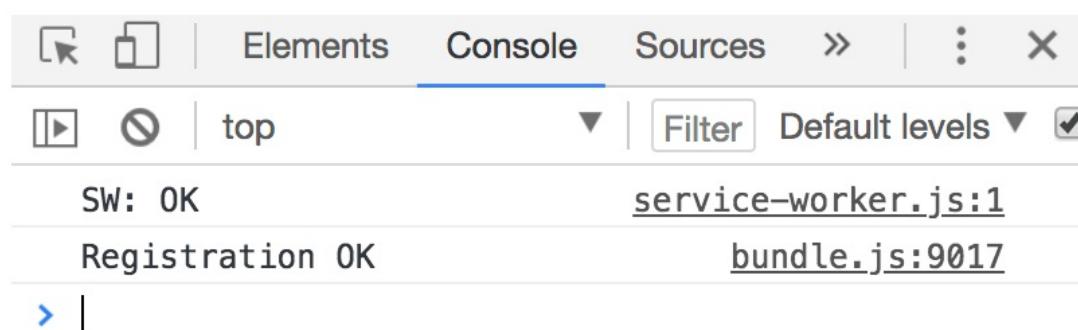
```
// service-worker.js

console.log('SW: OK');
```

Le cycle de vie d'un Service Worker

- Le SW s'active à la première consultation du site.
- Il restera en tâche de fond (idle) tant qu'il ne sera pas mis à jour

Première consultation :

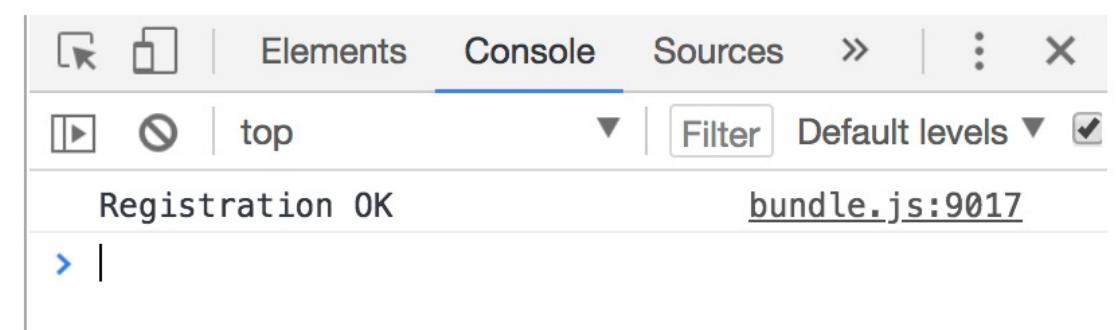


The screenshot shows the Chrome DevTools Console tab. The logs are as follows:

```
SW: OK
Registration OK
```

service-worker.js:1
bundle.js:9017

SW déjà en cours d'exécution :

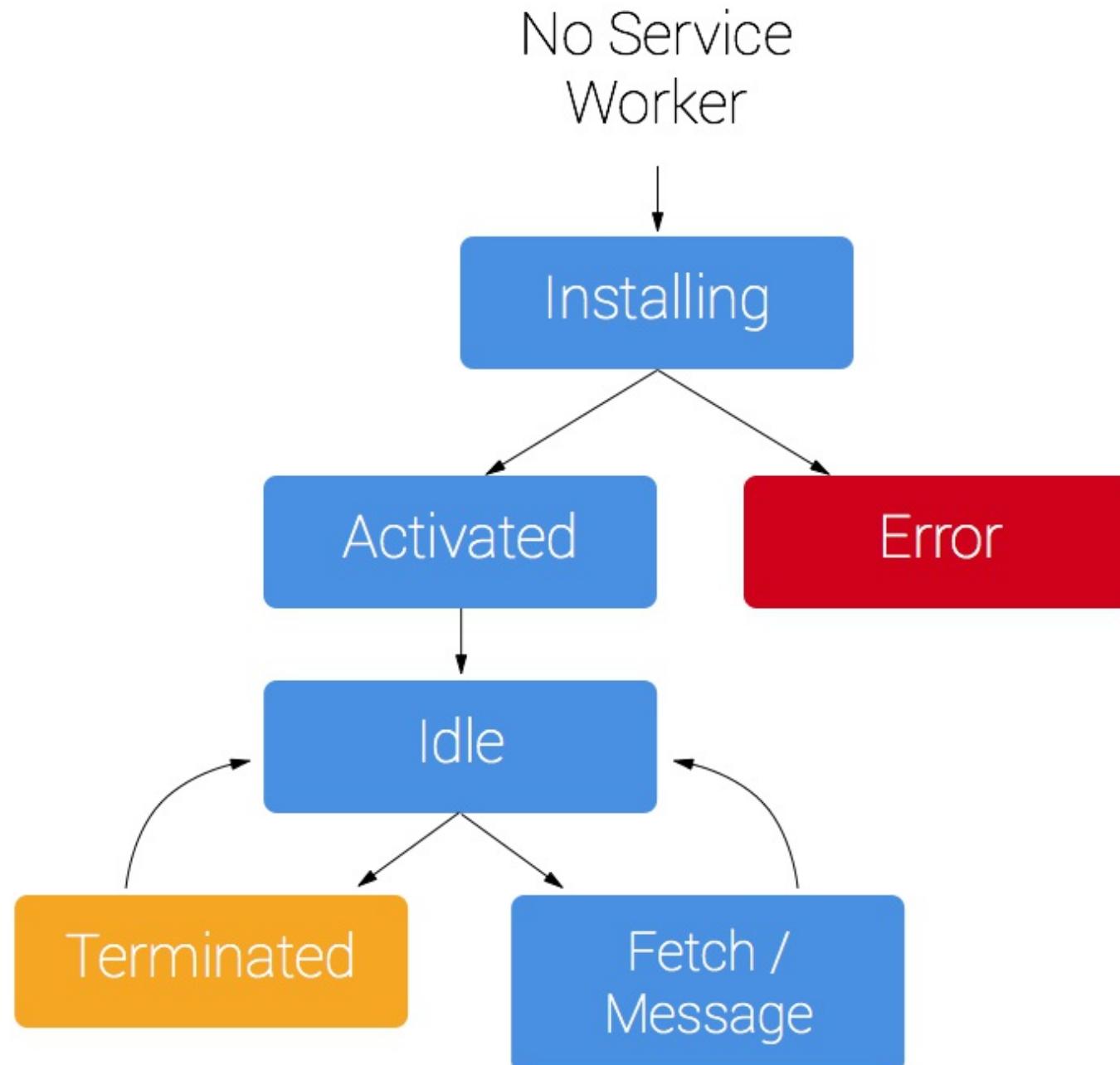


The screenshot shows the Chrome DevTools Console tab. The logs are as follows:

```
top
Registration OK
```

bundle.js:9017

Le cycle de vie d'un Service Worker



Inspecter les Services Workers

Dans Chrome, onglet Application, on peut visualiser l'état des Service Worker.

Service Workers

Offline Update on reload Bypass for network

localhost

[Update](#) [Unregister](#)

Source [service-worker.js](#)

Received 14/03/2018 à 16:14:14

Status ● #17429 activated and is running [stop](#)

● #17435 waiting to activate [skipWaiting](#)

Received 14/03/2018 à 16:16:29

Écouter les événements du cycle de vie

```
// service-worker.js

self.addEventListener('install', function (event) {
    // SW en cours d'installation
    event.waitUntil(
        // Ici on va pouvoir mettre en cache des données
        // On devra retourner une promesse
    );
}) ;

self.addEventListener('activate', function (event) {
    // En général, on se sert de cet événement pour
    // supprimer les anciennes versions du cache
}) ;
```

Intercepter les requêtes avec fetch

Exemple : le SW intercepte toutes les requêtes et renvoie à la place une nouvelle réponse grâce à respondWith().

```
// service-worker.js

self.addEventListener("fetch", event => {
  event.respondWith(new Response('Bonjour'))
});
```

`event.respondWith()` peut prendre en paramètre :

- **Un objet Response**

```
event.respondWith(new Response(  
    '<p>Hello <strong>World</strong></p>',  
    headers: { 'Content-Type': 'text/html' }  
)) ;
```

- **Un élément du cache**

```
event.respondWith(  
    caches.match(event.request);  
) ;
```

- **Une promise `fetch()`**

```
event.respondWith(  
    fetch('images/404.png');  
) ;
```

Communiquer avec le JS du site avec message

```
// index.js

function sendMessage(message) {
    // navigator.serviceWorker.controller est égal à l'objet du SW
    // ou null, s'il n'y en a pas
    if (navigator.serviceWorker.controller) {
        navigator.serviceWorker.controller.postMessage('Bonjour');
    }
}
```

```
// service-worker.js

self.addEventListener("message", event => {
    // Ici on récupère les messages émis par le JS du site
    console.log("Message reçu : " + event.data);
});
```

Stockage des données



Les APIs de stockage navigateur

API	Data Model	Persistence	Browser Support	Transactions	Sync/Async
File system	Byte stream	device	52%	No	Async
Local Storage	key/value	device	93%	No	Sync
Session Storage	key/value	session	93%	No	Sync
Cookies	structured	device	100%	No	Sync
WebSQL	structured	device	77%	Yes	Async
Cache	key/value	device	60%	No	Async
IndexedDB	hybrid	device	83%	Yes	Async
cloud storage	byte stream	global	100%	No	Both

Recommendations :

- Cache API pour les assets
- IndexedDB pour les données et state de l'application

Espace disponible

Browser	Limit
Chrome	<6% of free space
Firefox	<10% of free space
Safari	<50MB
IE10	<250MB
Edge	Dependent on volume size

- **Espace total par domaine, toutes API confondues**
- **Quota Management API (sur Chrome uniquement)**

Utilisation de la Cache API

- L'**interface Cache** permet de stocker les réponses des requêtes effectuées par notre application.
- disponible sur l'objet window et via le SW

Le point d'entrée est caches.

```
caches.open('mysite-static-v3').then(function (cache) {  
    // Ici on peut ajouter ou supprimer des éléments à cache  
})
```

Stratégies de cache

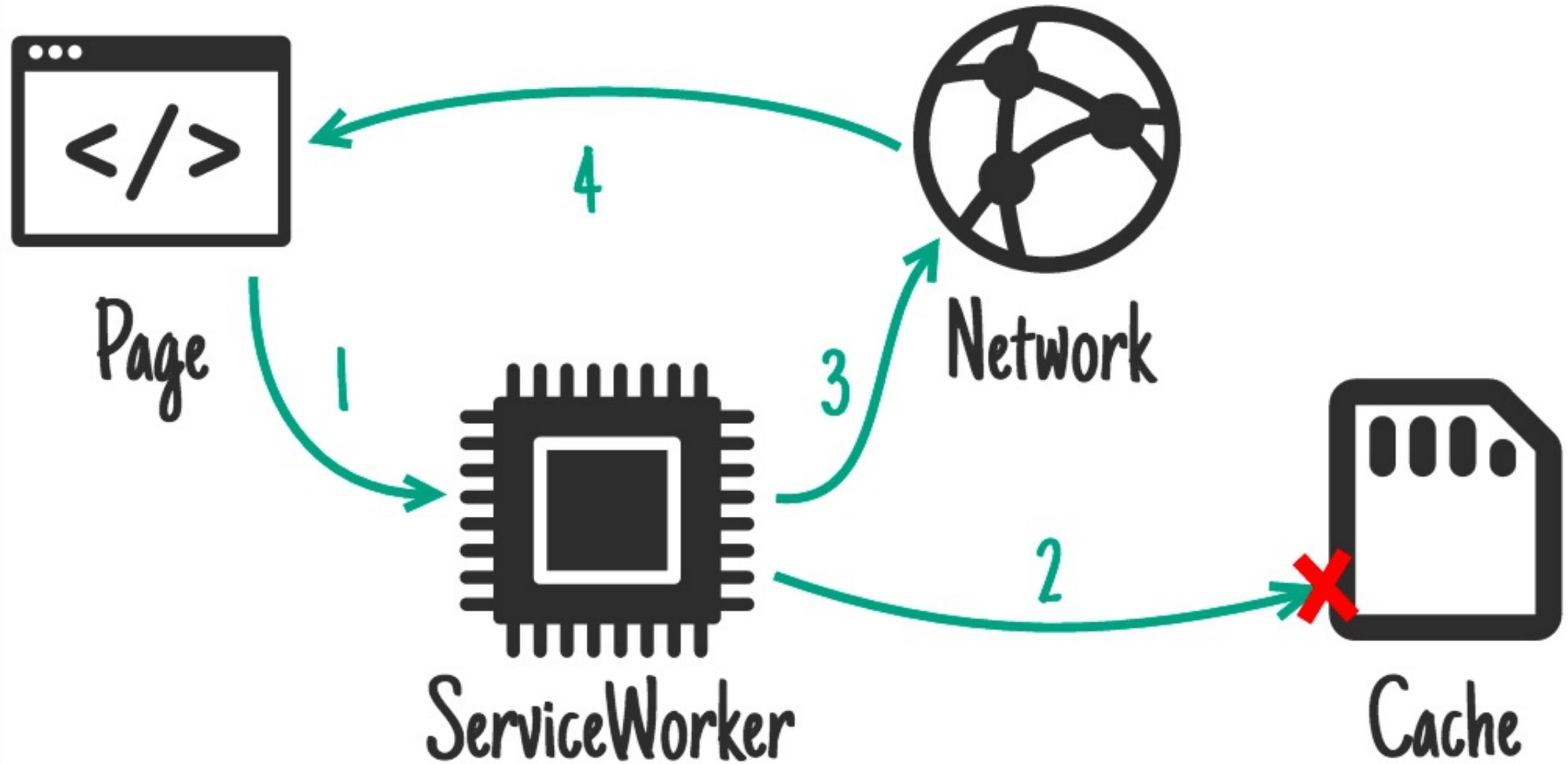
On retiendra principalement :

- Cache First
- Network First
- Stale While Revalidate : on récupère d'abord le cache.
Ensuite on fait la requête sur le réseau et on met en cache la réponse pour la prochaine fois.

cf. [The Offline Cookbook](#)

Cache first

Pour du offline-first



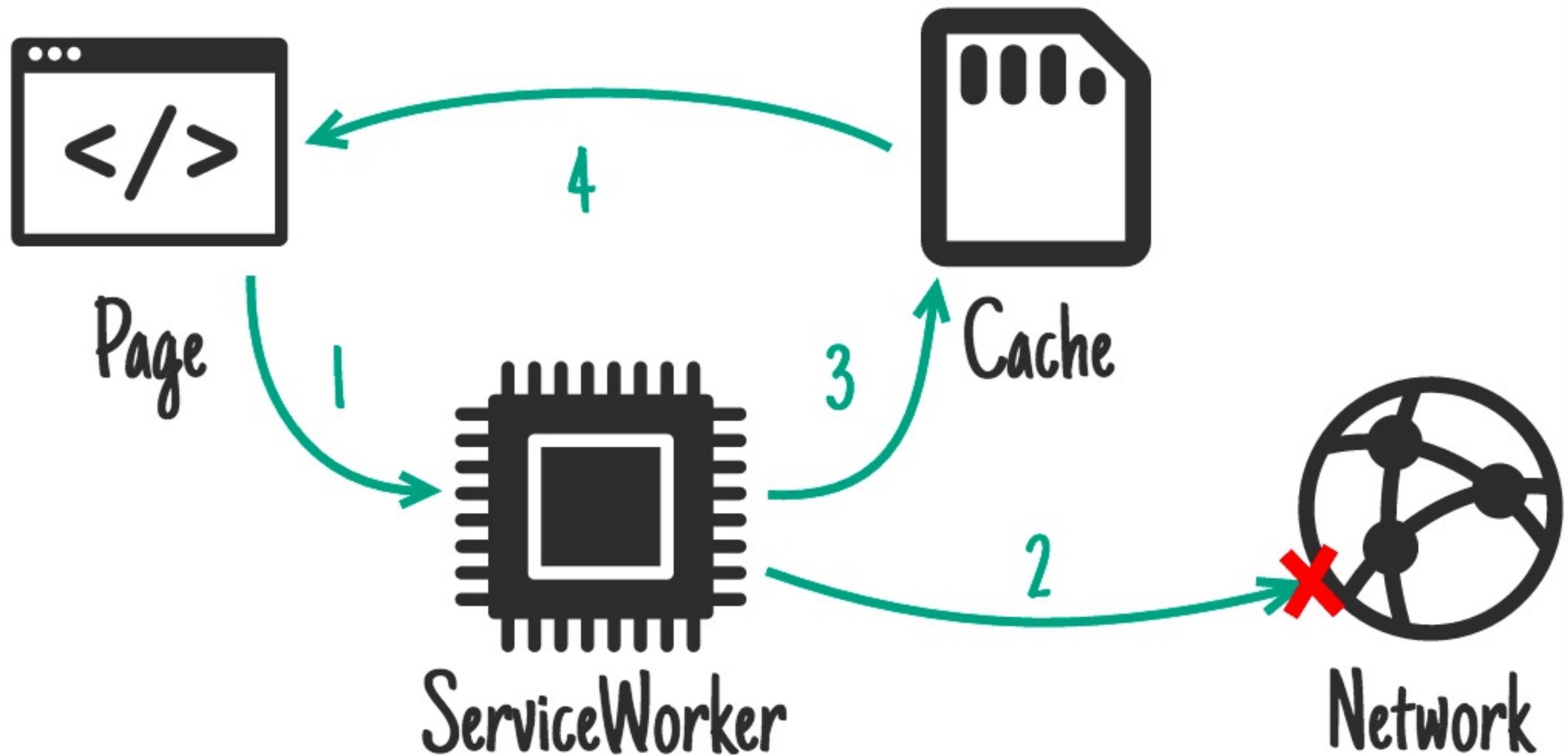
Cache first

```
// service-worker.js

self.addEventListener('fetch', function (event) {
  event.respondWith(
    // On demande au cache si il existe une entrée correspondante
    caches.match(event.request).then(function (response) {
      // Si oui, on renvoie les données du cache
      // Si non, on fetch la requête via le réseau
      return response || fetch(event.request);
    })
  );
}) ;
```

Network first

Pour ce qui doit être mis à jour très fréquemment : articles, timeline de réseau social, avatar...



Network first

```
// service-worker.js

self.addEventListener('fetch', function (event) {
  event.respondWith(
    // On fait une requête réseau
    fetch(event.request).catch(function () {
      // Si elle échoue, on cherche la ressource en cache
      caches.match(event.request).then(function (response) {
        // Si elle existe, on renvoie les données du cache
        // Si non, on retourne un message
        return response || new Response('Ressource non disponible')
      })
    })
  );
});
```

Conclusion

Pour répondre aux critères de disponibilité (offline) et de performance :

Service Worker

- Intercepte des événements
- Pilote la mise en cache et la restitution des données

Stockage des données

- Cache API pour les assets
- IndexedDB pour les données (state, data),...
- Limité en taille

Workbox



Workbox

Workbox est une librairie qui facilite la conception d'applications hors ligne :

- Precaching
- Stratégies de cache
- Versionning de cache
- Génération automatique de la liste des ressources
- Background Sync

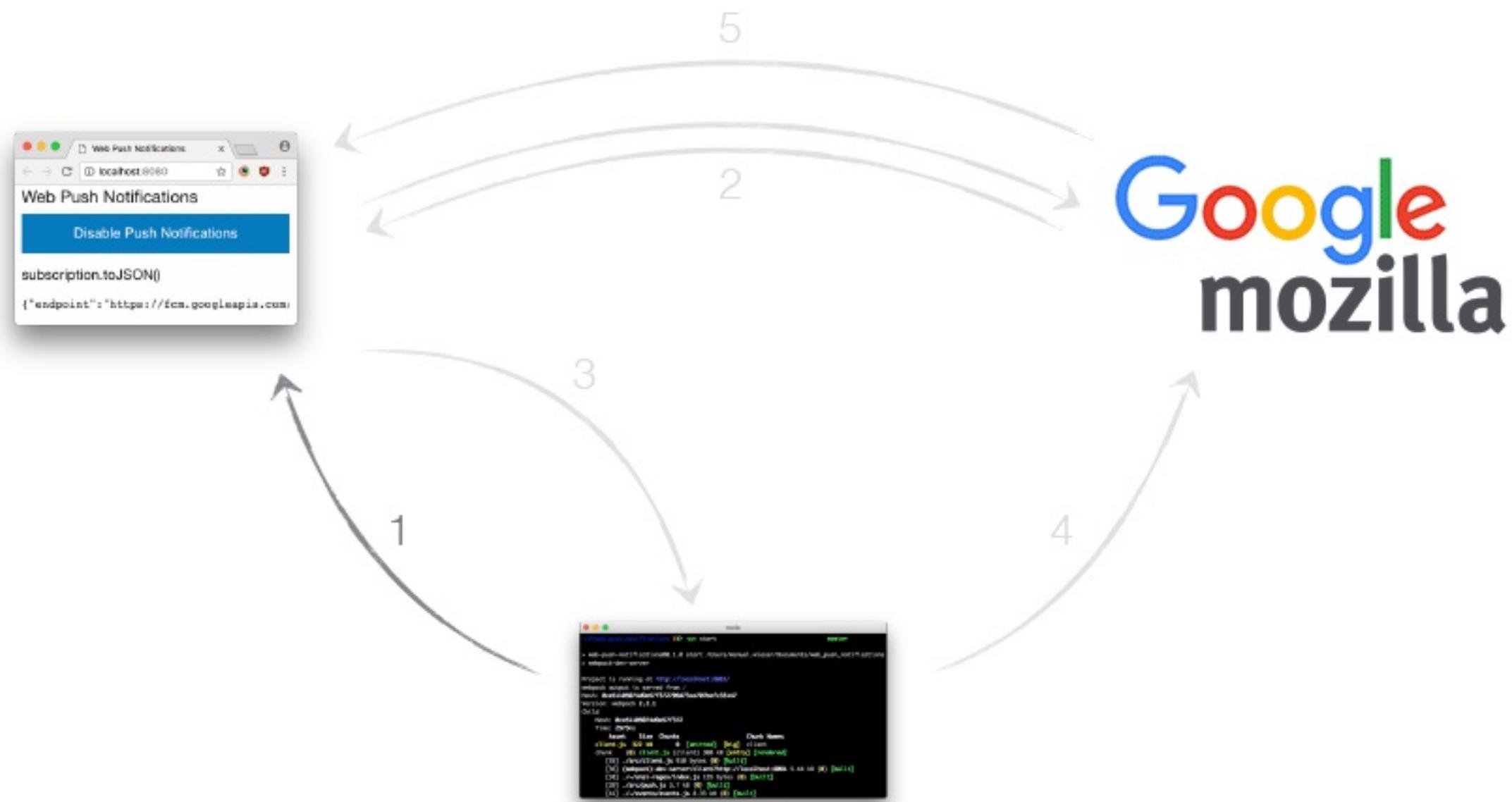
Workbox config

```
module.exports = {
  globDirectory: "public/",
  globPatterns: [
    "**/*.{js,png,xml,ico,svg,html,json,css}",
  ],
  swDest: "public/service-worker.js",
  runtimeCaching: [ {
    urlPattern: new RegExp( '^https://media\\.guim\\.co\\.uk/' ),
    handler: 'staleWhileRevalidate'
  },
  {
    urlPattern: new RegExp( '^https://content\\.guardianapis\\.com/' )
    handler: 'staleWhileRevalidate'
  } ]
}.
```

Notifications push



Mise en place de Push Notification



Définitions

Push API

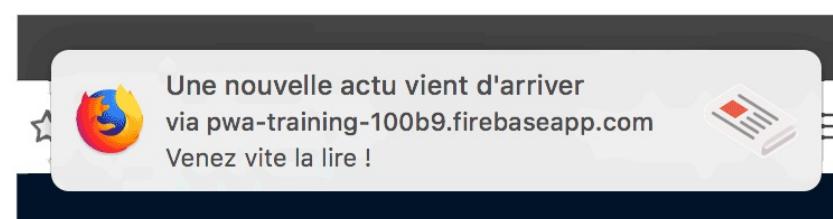
Permet au service worker de recevoir des messages reçus du push service

Notification API

Message visible par l'utilisateur (piloté par le SW)



Android



Firefox MacOS



Chrome MacOS

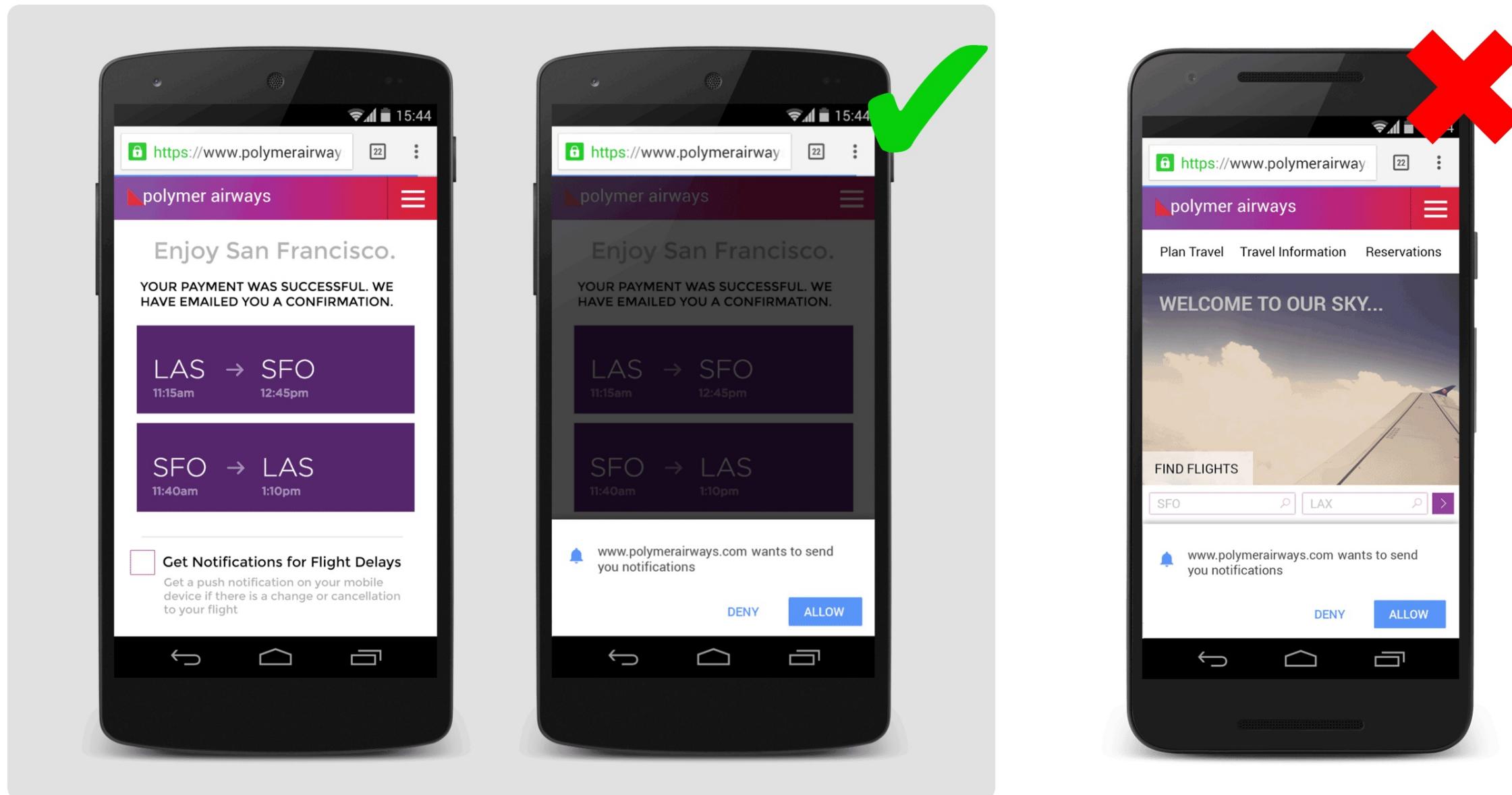
Service de messagerie (push service)

Transmet la notification au Service Worker

- **Firebase Cloud Messaging (FCM, anciennement GCM) pour Chrome**
- **Autopush pour Firefox**
- **Microsoft Notification Hub pour Windows Mobile**
- **Apple Push Notification service (APNs) pour Safari**

UX : demande de permission

- Expliquer en quoi les notifications seront utiles à l'utilisateur
- Moins intrusif



Serveur d'application (push provider)

Le serveur d'application sert à piloter l'envoi de message push.

- **Librairie Node.js Web Push**
- **PyFCM**
- **Pusher Server API**

Envoi d'un message push avec PyFCM

```
from pyfcm import FCMNotification

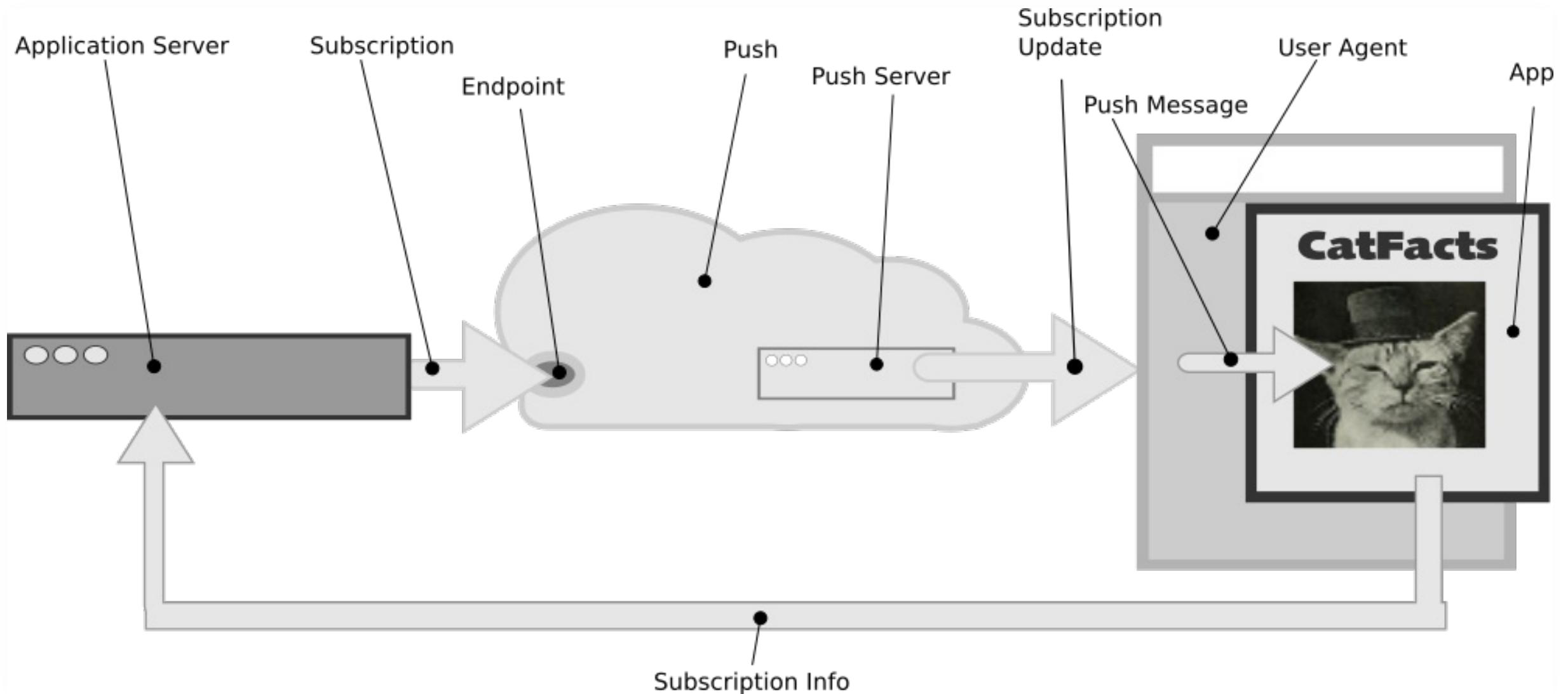
push_service = FCMNotification(api_key="")

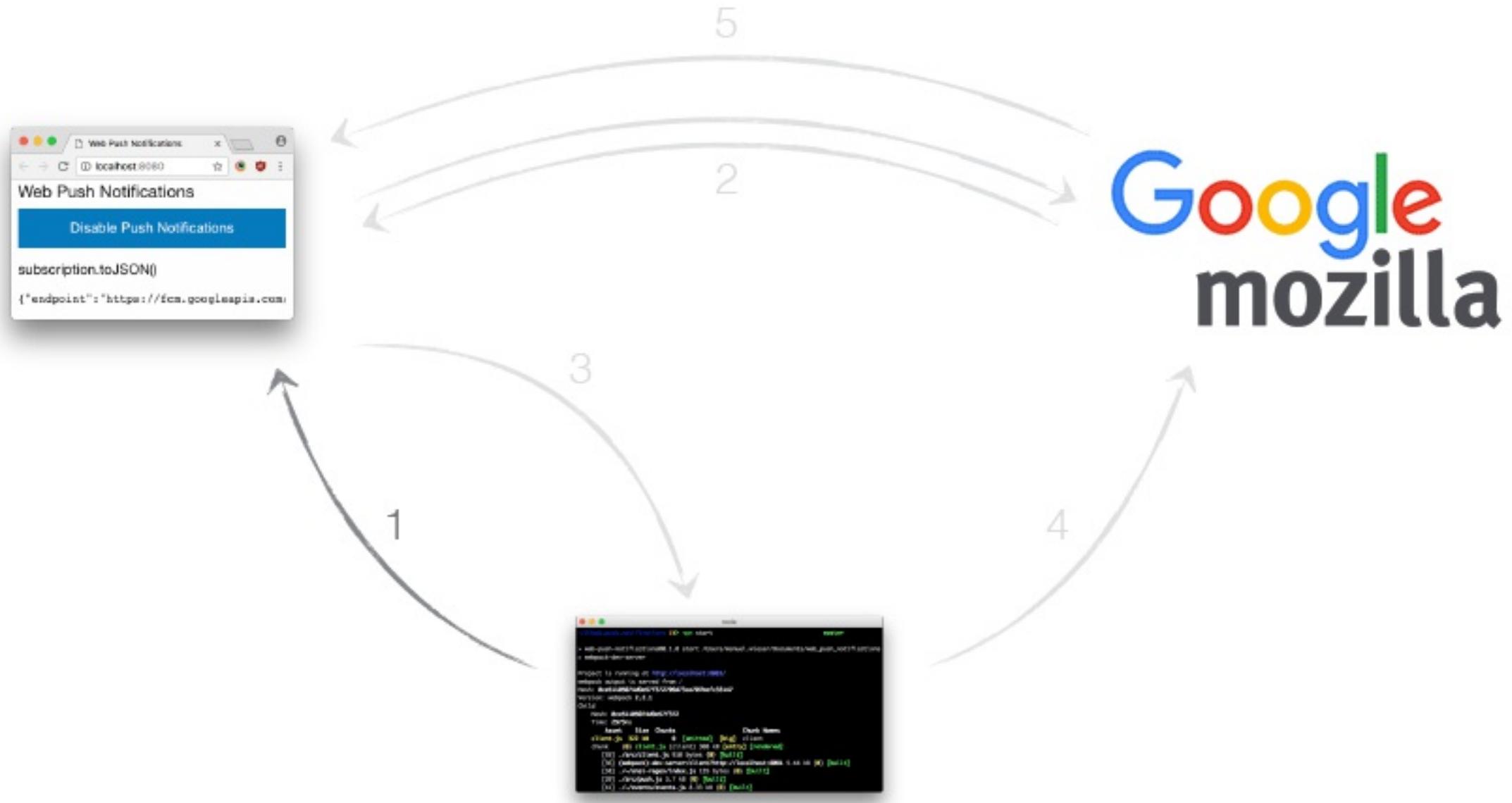
registration_id = "<device registration_id>"
message_title = "Uber update"
message_body = "Hi john, your customized news for today is ready"
result = push_service.notify_single_device(
    registration_id=registration_id,
    message_title=message_title,
    message_body=message_body
)

print result
```

VAPID

- Voluntary Application Server Identification for Web Push (application server key)
- Clé unique permettant de chiffrer les messages qui transitent à travers le push service.





Services de push et de notification "tout-en-un"

Services	Limitations	
One Signal	Très complet, intégrations CMS, planification, A/B tests, segmentation	Gratuit, illimité
Pushcrew	Simple à utiliser, segmentation, ne supporte pas Safari	Gratuit jusqu'à 2000 abonnés
Pusher	Abstraction de la Push API et intégration d'APNs (Safari), envoi des messages via une CLI ou intégration à un serveur	Gratuit jusqu'à 100 abonnés
Pushpad	Abstraction de la Push API et intégration d'APNs (Safari)	À partir de 5\$/mois

Exemple OneSignal

7 Add Code to Site

If you haven't already, add this code to the `<head>` section on all pages of your site that users can subscribe to.

 COPY CODE

```
<head>
...
<link rel="manifest" href="/manifest.json" />
<script src="https://cdn.onesignal.com/sdks/OneSignalSDK.js" async="">
</script>
<script>
  var OneSignal = window.OneSignal || [];
  OneSignal.push(function() {
    OneSignal.init({
      appId: "10c53cd8-e933-41e6-a995-95a473393c07",
    });
  });
</script>
...
</head>
```

ADD CODE TO YOUR SITE

Depending on how your site is hosted, you may need to contact someone to help you add this code to your site.

 READ OUR DOCUMENTATION
Add Code to Your Site

Envoi de messages Push avec OneSignal

Paramétrer l'envoi de messages push

ENGLISH FRENCH 

TITLE

Bonjour 

MESSAGE

Comment allez-vous ? 

ENGLISH FRENCH 

TITLE

Hello 

MESSAGE

How are you? 

Conclusion

Doit-on utiliser les PWA aujourd'hui ?

Pour

- Tout type d'applications : amélioration progressive
- Proposer une alternative légère (qui ne nécessite pas d'installation) à des applications natives
- Peu coûteux à mettre en place

Contre

- Si votre cible principale est iOS ou IE
- Pour les applications lourdes (cartographie, audio,...)

Pour aller plus loin

TP

<https://github.com/makinacorus/pwa-training>

Ressources

Articles en français

- Découvrir le Service Worker, Makina Corpus
- Série d'articles de sur les PWA, Julien Pradet
- Application web progressives, MDN

Pour en savoir plus sur le Service Worker

- Cours interactif gratuit [en], Jake Archibald
- Introduction aux Web Workers, Synbioz
- Service Worker Cookbook [en], MDN
- Service Worker API, MDN
- We built a PWA from scratch - This is what we learned [en], 14islands

Ressources

Push Notification

- Push Notification et VAPID
- Why Doesn't Push Work when the Browser is Closed?
- Using Web Push Notifications with VAPID

Background Sync

- A Basic Guide to BackgroundSync

Des questions ?

