PWA

Concept

PWA

- PWA est l'acronyme de **P**rogressive **W**eb **A**pplication

- Cela vous permet de déclarer au browser que votre site web est en fait une application web, utilisable comme tel

- L'expérience utilisateur est alors similaire à une application native sans la contrainte des application stores que l'on connait

https://fr.wikipedia.org/wiki/Progressive_web_app

Concept

PWA

- Quelques avantages bien pratiques…
  - Le look&feel d'une vrai app
  - Des mécanismes de caching et de tâches de fond
  - Economie en coûts de développement
  - Updates à la volée !
  - Disponible offline (moyennant configuration)

https://fr.wikipedia.org/wiki/Progressive_web_app

Concept

PWA

- Accès à certaines API
- …mais des inconvénients
  - Limitations quand à l'utilisation des fonctionnalités avancées de l'appareil (accès au hardware, par ex…)

https://fr.wikipedia.org/wiki/Progressive_web_app

Concept

PWA

- Selon Google, une PWA doit disposer des caractéristiques suivantes:
    - Progressive
    - Sécurisée
    - Engageante
    - Installable
    - Rapide
    - Optimisée pour le référencement
    - Indépendante de la connexion

https://fr.wikipedia.org/wiki/Progressive_web_app

Concept

PWA

- Quels sont les prérequis pour déclarer une PWA et la rendre installable ?
    - Un fichier manifest (qui décrit l'application)
    - Site sécurisé (via HTTPS) ou en localhost
    - Une icône
    - Un Service Worker (pour gérer le caching et autre tâche de fond)

https://fr.wikipedia.org/wiki/Progressive_web_app

Fichier manifest

Concept

Fichier manifest

- Les fichiers manifest sont des fichiers formatés en JSON qui finissent par “.webmanifest”
- Ils disposent de plusieurs clés prédéfinies pour permette de décrire l’application
- Ils doivent au minimum certains champs obligatoires
- Il s’inclut grâce à une balise <link> dans le <head>

https://fr.wikipedia.org/wiki/Progressive_web_app

Concept

Fichier manifest

- Les valeurs typiques:
  - name - Nom complet de l'application (ex: "Spotlified - Enjoy the music")
  - short_name - Nom court de l'application (ex: "Spotlified")
  - background_color - La couleur de fond
  - display - Le type d'affichage (plein écran, standalone, ...)
  - icons - Un tableau d'icônes (minimum 1)
  - start_url - L'url de départ de l'application (ex. "/" ou "/#home")

https://fr.wikipedia.org/wiki/Progressive_web_app

La valeur display

Fichier manifest

- Arrêt sur la valeur display : Elle permet de modifier la manière dont l'application va s'afficher lors de son lancement
    - fullscreen - Plein écran, rien d'autre que l'app
    - standalone - Plein écran, mais avec la barre de statut de l'os (le + utilisé)
    - minimal-ui - Dépend du système, mais typiquement les touches précédent/suivant de l'historique
    - browser - Version classique, comme le browser

https://fr.wikipedia.org/wiki/Progressive_web_app

Autres valeurs intéressantes

Fichier manifest

- theme_color - En opposition à background_color, permet de définir la couleur des éléments OS, comme la barre de statut
- orientation - Orientation par défaut
- lang - La langue de l'application
- related_applications - Un tableau d'applications natives qu'il est possible d'installer en fonction de l'os

https://fr.wikipedia.org/wiki/Progressive_web_app

Exemple basique

Fichier manifest

```
{
  "short_name": "Spotlified",
  "name": "Spotlified - Unleash the JS",
  "icons": [
    {
      "src": "images/logo_spotlified.png",
      "type": "image/png",
      "sizes": "512x512"
    }
  ],
  "start_url": "/",
  "background_color": "#121212",
  "display": "standalone",
  "scope": "/",
  "theme_color": "#121212"
}
```

Intégrer au HTML

Fichier manifest

Dans le <head> :

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

https://fr.wikipedia.org/wiki/Progressive_web_app

Icônes

Fichier manifest

Possible de télécharger un icône d'exemple ici:

https://github.com/lgavillet/webmobui/blob/main/Ressources/assets/logo_spotlified.png

Concept

PWA

- Quels sont les prérequis pour déclarer une PWA et la rendre installable ?
  - ✓ Un fichier manifest (qui décrit l'application)
  - ✓ Site sécurisé (via HTTPS) ou en localhost
  - ✓ Une icône
  - Un Service Worker (pour gérer le caching et autre tâche de fond)

https://fr.wikipedia.org/wiki/Progressive_web_app

Détection online/offline

Concept

Détection online/offline

- Il est possible de détecter si la connexion réseau est activée ou non
- Cela peut servir par exemple pour informer l'utilisateur de limitations sur l'application (lister ok, chercher… non.)
- En plus de le tester, il est possible d'être averti de changement via un Event Listener

https://developer.mozilla.org/en-US/docs/Web/API/Navigator/onLine

Propriété window.navigator.onLine

Détection online/offline

- Pour savoir si le browser est en ligne ou non, on peut utiliser l'attribut suivant :

```
window.navigator.onLine

// true si online, false autrement
```

https://developer.mozilla.org/en-US/docs/Web/API/Navigator/onLine

Events online/offline

Détection online/offline

- L'attribut onLine, ne permet que de savoir ponctuellement l'état du navigateur

- Pour en être averti, il y a les évents `online` et `offline` sur `window`

```
window.addEventListener('offline',(e) => console.log('offline'))

window.addEventListener('online', (e) => console.log('online'))
```

https://developer.mozilla.org/en-US/docs/Web/API/Navigator/onLine

Events online/offline

Détection online/offline

- On pourrait par exemple s'en servir pour changer la couleur du logo en rouge

- Ou alors, pour désactiver le bouton de recherche, si offline. Les requêtes de base seront cachées, comme la liste d'artistes ou des chansons, mais une requête de recherche est tellement spécifique qu'il est impossible de toutes les cacher

https://developer.mozilla.org/en-US/docs/Web/API/Navigator/onLine

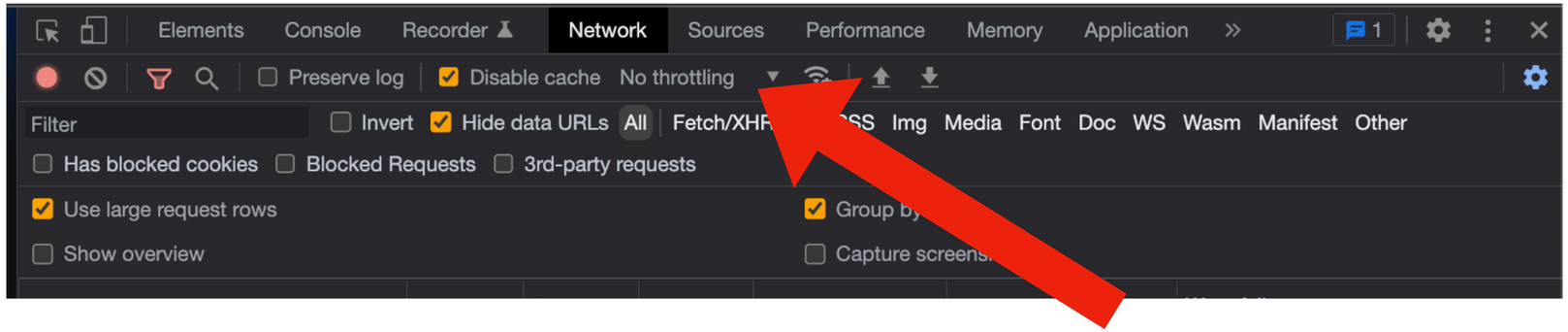
Comment tester ?

Détection online/offline

- Chaque navigateur dispose, dans l'inspecteur, d'une tab "Réseau"

- Il est possible dans cette tab de simuler des problèmes réseaux, soit lenteur, soit simplement désactivé

- Exemple avec chrome (Network Throttling) :



Service Worker API

Aperçu

Service Worker API

- Il existe des dizaines d'API web, permettant d'interagir avec le browser...
- https://developer.mozilla.org/fr/docs/Web/API
- Nous allons nous focuser sur les suivantes :
    - Service Worker API
    - Cache API

Aperçu

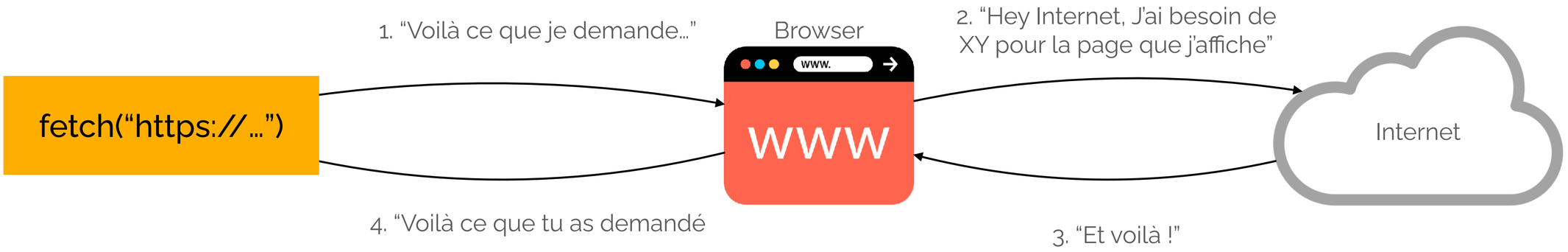
Service Worker API

- Un service worker est un fichier javascript asynchrone qui s'exécute en arrière plan
- Il peut réaliser toute sorte d'opérations, mais est principalement utilisé comme proxy pour mettre les requêtes en cache, grâce à l'API Cache
- En gros, il intercepte ce que fait le code principal et décide ou non de laisser passer
- On peut également lui envoyer des messages push, par exemple

https://developer.mozilla.org/fr/docs/Web/API/Service_Worker_API/Using_Service_Workers
https://developer.mozilla.org/fr/docs/Web/API/Cache

Fonctionnement – Cas classique

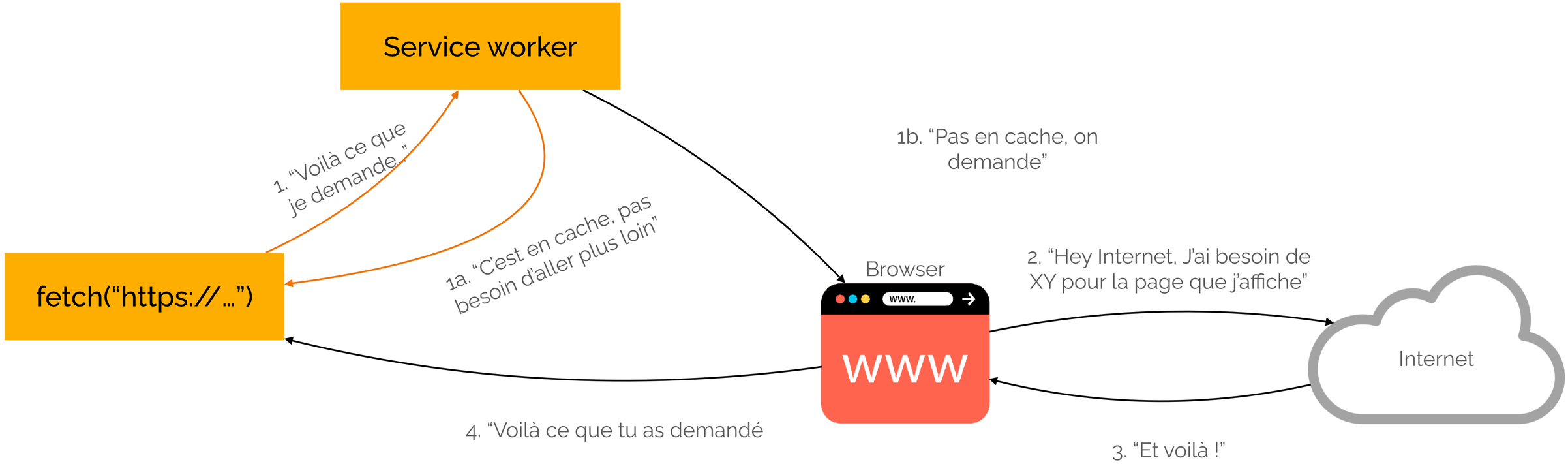
Service Worker API



https://developer.mozilla.org/fr/docs/Web/API/Service_Worker_API/Using_Service_Workers

Fonctionnement – Cas service worker

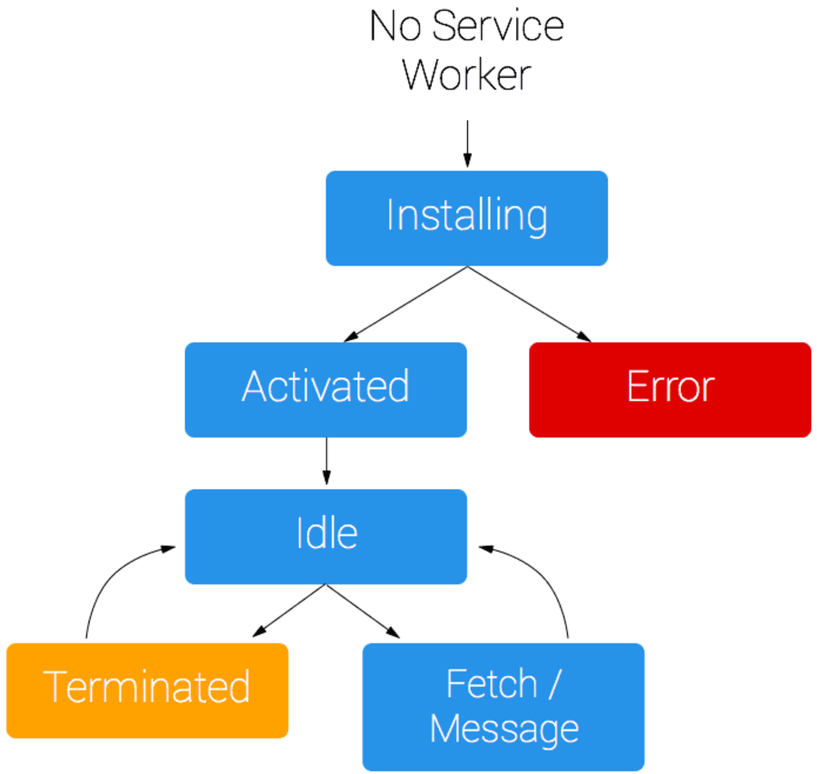
Service Worker API



https://developer.mozilla.org/fr/docs/Web/API/Service_Worker_API/Using_Service_Workers

Cycle de vie

Service Worker API



https://developer.mozilla.org/fr/docs/Web/API/Service_Worker_API/Using_Service_Workers

Cycle de vie et event listeners

Service Worker API

- L'anatomie typique d'un service worker
- Event 'install' - A l'installation du service worker
- Event 'activate' - Lorsque il est activé
- Event 'fetch' - Lorsqu'une requête est envoyée par l'application

https://developer.mozilla.org/fr/docs/Web/API/Service_Worker_API/Using_Service_Workers

Cycle de vie et event listeners

Service Worker API

- `self.addEventListener('install', (event) => {…})`
- `self.addEventListener('activate', (event) => {…})`
- `self.addEventListener('fetch', (event) => {…})`

https://developer.mozilla.org/fr/docs/Web/API/Service_Worker_API/Using_Service_Workers

Dans la pratique

Service Worker API

- Service worker mis à dispo par M. Chabloz

- src/libraries/workerCacheFetched.js

Intégration

Service Worker API

- L'intégration d'un service worker se fait via l'appel à la méthode suivante, lors du chargement de la page

- Cela indique qu'un service worker se trouve à l'url passée en paramètre

  - Exemple:
    ```
    navigator.serviceWorker.register('/monworker.js')
    ```

https://developer.mozilla.org/fr/docs/Web/API/Service_Worker_API/Using_Service_Workers

Intégration – Projet

Service Worker API

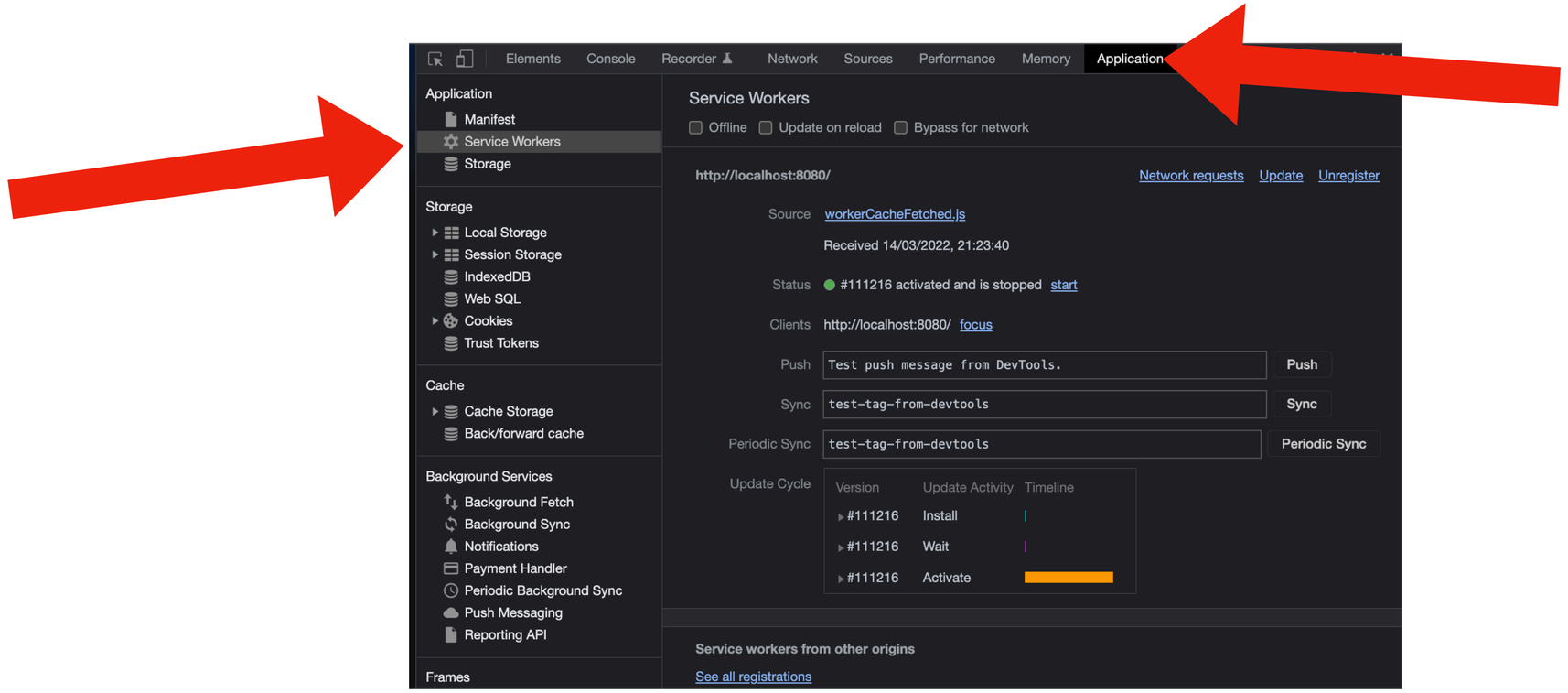
- A cause du fonctionnement de Parcel JS, l'intégration d'un service worker est un peu différente de la syntaxe de base (à ajouter en fin de fichier par ex.)

```
navigator.serviceWorker.register(
    new URL('workerCacheFetched.js', import.meta.url)
)
```

Intégration – Browser

Service Worker API

- Le navigateur va alors détecter le service worker et il est possible d'interagir avec lui via l'inspecteur. Sur Chrome, tab Application :



Intégration – Browser

Service Worker API

- Toute les requêtes sont alors mises en cache par le browser

- Hint: Cela peut poser des problèmes lors du développement, n'hésitez pas à désinstaller/réinstaller le worker pour afficher les nouveaux changements ou utiliser Cmd+Shift+R pour rafraichir en vidant le cache

Concept

PWA

- Quels sont les prérequis pour déclarer une PWA et la rendre installable ?
  - ✓ Un fichier manifest (qui décrit l'application)
  - ✓ Site sécurisé (via HTTPS) ou en localhost
  - ✓ Une icône
  - ✓ Un Service Worker (pour gérer le caching et autre tâche de fond)

https://fr.wikipedia.org/wiki/Progressive_web_app

Intégration – Projet

Service Worker API

- Déplacer le fichier workerCacheFetched.js de src/lib/ vers src/

- A cause du fonctionnement de Parcel JS, l'intégration d'un service worker est un peu différente de la syntaxe de base (à ajouter en fin de fichier par ex.).

```
navigator.serviceWorker.register(
    new URL('workerCacheFetched.js', import.meta.url)
)
```

Notifications Push

Qu’est-ce que c’est ?

Notifications Push

- Les notifications Push sont des messages utilisés par les applications pour vous avertir spontanément d’une information précise

- Elles s’appellent “Push”, car elles servent à “Pusher” du contenu depuis un serveur, vers l’utilisateur final

- Permet de relancer très facilement l’interaction avec l’application

Avantages

Notifications Push

- Pour les utilisateurs, les notifications push sont un moyen de recevoir des informations opportunes, pertinentes et précises

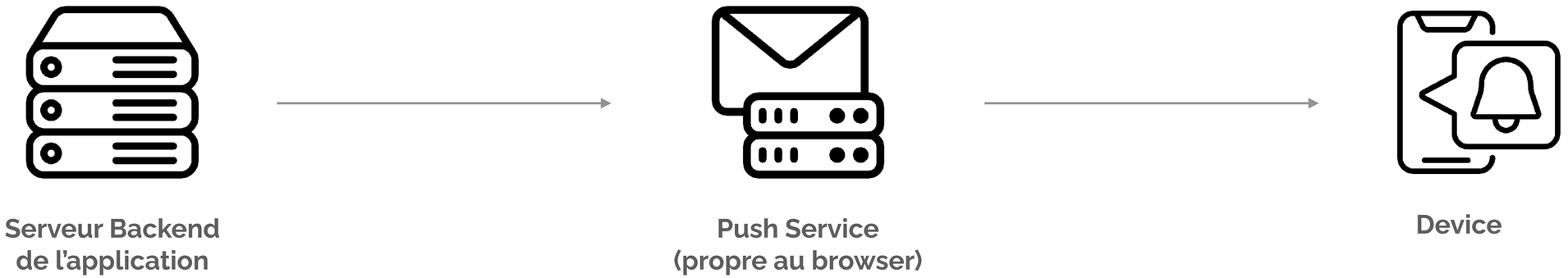
- Pour vous (propriétaire de site web), les notifications push sont un moyen d'accroître l'engagement des utilisateurs

- Typiquement utilisé pour informer d'un événement _précis_ dans le temps, comme un rabais sur une durée limitée ou “teaser” du contenu

Trop de notifications peuvent énerver vos utilisateurs

Comment ça marche ?

Notifications Push



Comment ça marche ? – Subscription

Notifications Push

1. Demander l'autorisation au device (sécurité)

2. Créer une Subscription

3. La transmettre au serveur Backend et la stocker pour l'utiliser

Comment ça marche ? – Envoi

Notifications Push

1. Le serveur envoie un requête Push à un Push Service via un Web Push Protocol

2. Le Push Service l'authentifie et vérifie la validité de la requête

3. Le Push Service envoie la requête au device et l'application l'affiche, grâce à sa logique interne

Comment ça marche ?

Notifications Push

- Il s'agit ici d'une explication simplifiée du mode de fonctionnement

- Il y a plusieurs mécanismes de sécurité à chaque étape qui assure la validité et la sécurité des messages transmis

- Excellent article sur le fonctionnement des notifications Push sur: https://web.dev/push-notifications-overview/

Web Push

Notifications Push

- Les notifications push pour le web sont gérées par les browsers, pour la partie device

- Chaque browser choisi lui-même son Push service, pas d'incidence sur le code (implémentation interne)

- Les browser implémentent 2 Web APIs (les fameuses!) pour les intégrer et cela est ***très*** fortement lié aux Services Workers

Web Push

Notifications Push

- En gros:
  - Push API pour le “transport” des notifications depuis le service worker
  - Notification API pour les afficher du coté browser

Web Push – Autorisation

Notifications Push

- Avant d'utiliser les webpush, il faut d'abord ***demander l'autorisation*** à l'utilisateur, via le browser

- Cela se fait via:
  `Notification.requestPermission()`

- Ou presque…

Web Push – Autorisation

Notifications Push

- Le browser peut dans certains cas simplement **ignorer** la demande de permission

- Pour des questions de sécurité, il peut ignorer cette demande, tant que l'utilisateur n'a pas fait un certain nombre d'opérations

- On appelle cela le "Media Engagement Index" - tant qu'une certaine valeur n'est pas atteinte, pas possible de les utiliser

Egalement valable pour l'installation d'une PWA

Web Push – A l'aide…

Notifications Push

- Tout cela devient compliqué…
- Heureusement, il existe des services de push notification qui s'occupe de faire cela pour vous !
- Exemple : onesignal.com

Web Push – OneSignal

Notifications Push

- Ce type de service vous permet d'écrire des notifications à la main comme des emails
- Il gère l'affichage des demandes d'autorisation, selon la logique du navigateur
- Il vous fournit un service worker clé en main pour afficher les notifications !
- Permet plein d'autres choses…! (Plugin wordpress, par exemple !!)

Web Push – OneSignal

Notifications Push

- Ce type de service vous permet d'écrire des notifications à la main comme des emails

- Il gère l'affichage des demandes d'autorisation, selon la logique du navigateur

- Permet plein d'autres choses…! (Plugin wordpress, par exemple !!)

Web Push – OneSignal

Notifications Push

1. Déployer sur netlify

2. Créer un compte OneSignal

3. Créer une app OneSignal avec des Web Push

4. Suivre les instructions

Web Push – OneSignal

Notifications Push

5. NB: Placez OneSignalSDKWorker.js dans src/

6. Modifiez votre package.json pour que les deux scripts ressemblent à cela:

```
"scripts": {
    "start": "parcel index.html OneSignalSDKWorker.js",
    "build": "parcel build src/index.html src/OneSignalSDKWorker.js"
 },
```

7. Supprimez votre serviceWorker actuel (dans index.js + via la Tab “Application” dans les devs tool)