

JsonStorage et les events...

Favoris

Ecouter un changement sur le storage....

```
import JsonStorage from './lib/jsonStorage' // ou autre chemin que vous aurez choisi
...
const favoriteStorage = new JsonStorage({ name: 'favorites', eventName: 'playlist_update' })
...
window.addEventListener('playlist_update', () => { console.log("c'est à jour!") })
```

JsonStorage et les events...

Favoris

Obtenir un tableau étirable facilement

```
import JsonStorage from './lib/jsonStorage' // ou autre chemin que vous aurez choisi

const favoriteStorage = new JsonStorage({ name: 'favorites', eventName: 'playlist_update'})

for(const element of favoriteStorage.values()) { // Problème, .values().length ne fonctionne pas
}
```

... ou

```
const tableau = playlistStorage.toArray().map((e) => e[1])
```

Composants ?

Projet

✓ Projet webpack vide

✓ Squelette HTML

✓ Styles CSS structurels

✓ Icônes

✓ Routeur pour les pages web

✓ Client pour l'API JSON

✓ Lecteur audio

✓ Local storage pour les favoris

- Détection online/offline

- Manifest PWA

- Caching

- Service worker

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 connaît

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)

Concept

PWA

- ...mais des inconvénients
 - Notifications Push pas vraiment natives... Seulement que l'app est ouverte
 - Limitations quand à l'utilisation des fonctionnalités avancées de l'appareil (accès au hardware, par ex...)

Concept

PWA

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

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)

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 permettre de décrire l'application
- Ils doivent au minimum certains champs obligatoires
- Il s'inclut grâce à une balise <link> dans le <head>

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")

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

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

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

- A la différence des autres fichiers, nous n'allons ni éditer celui-ci, ni l'importer dans notre code via des "imports". Il doit donc être servi tel quel, sans passer par webpack
- Vous pouvez le placer dans le répertoire "public" et non "src"
- Il sera alors possible d'y accéder via <http://localhost:8080/manifest.webmanifest>

Intégrer au HTML

Fichier manifest

Dans le <head> :

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

Icônes

Fichier manifest

Possible de télécharger un icône d'exemple ici:

https://github.com/lgavillet/webmobui-22/blob/master/Ressources/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)

Détection online/offline

Concept

Détection online/offline

- Il est possible de détecter si la connection 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

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 événements `online` et `offline` sur `window`

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

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

Events online/offline

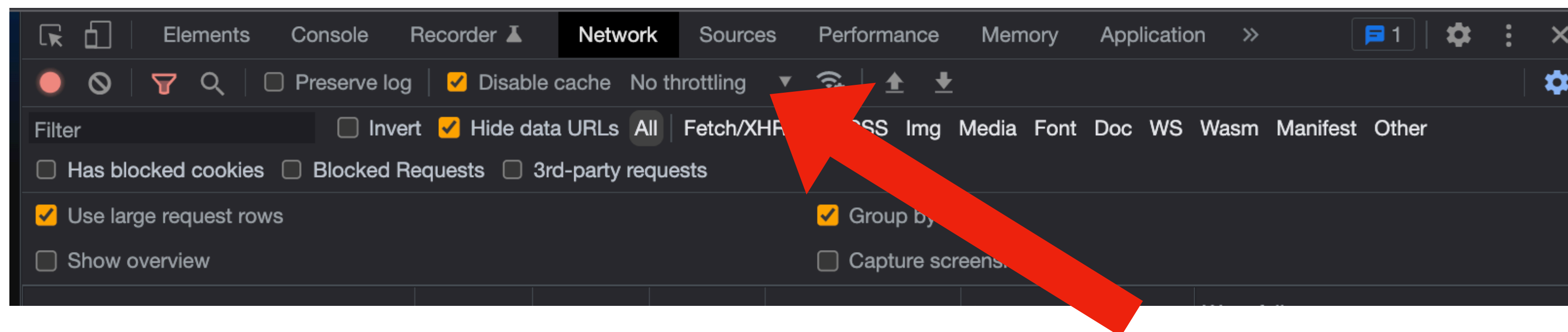
Détection online/offline

- 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

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 focaliser sur les suivantes :
 - Service Worker API
 - Cache API

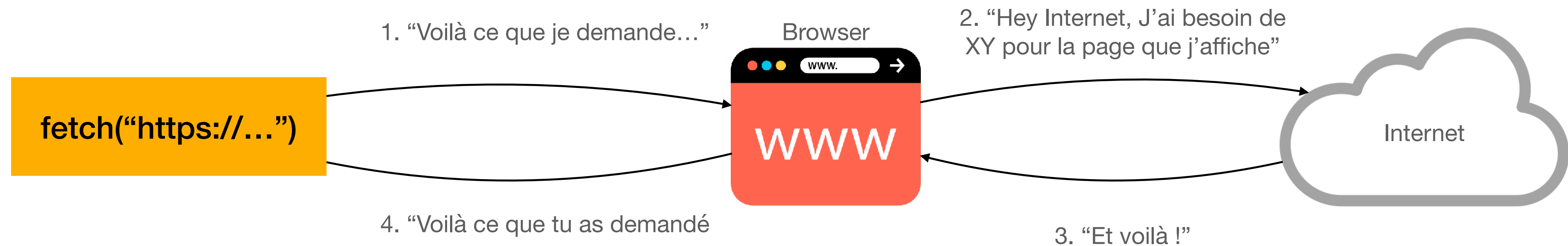
Aperçu

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

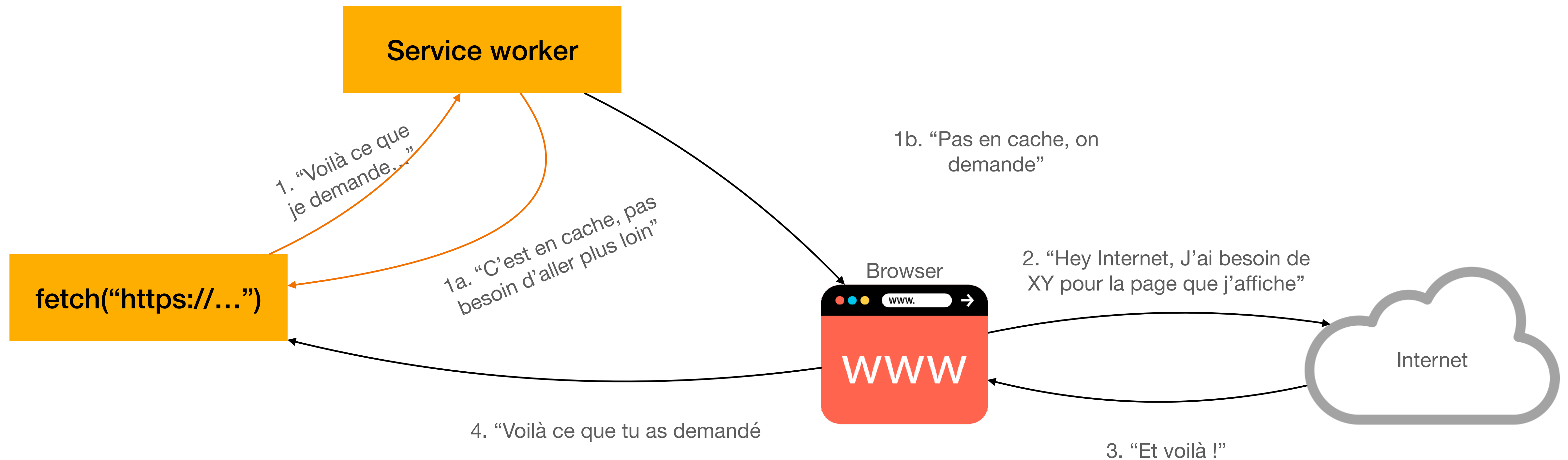
Fonctionnement - Cas classique

Service Worker API



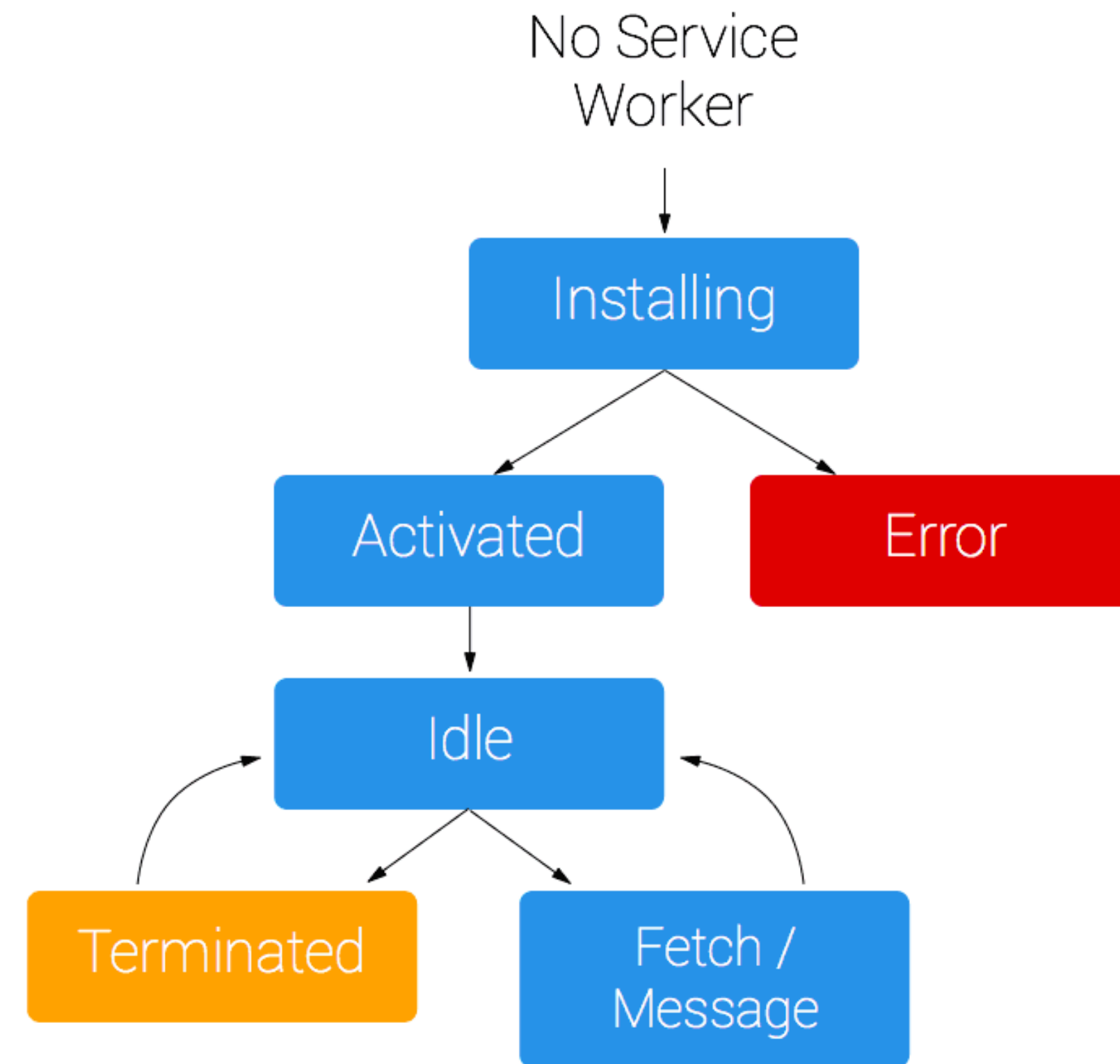
Fonctionnement - Cas service worker

Service Worker API



Cycle de vie

Service Worker API



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

Cycle de vie et event listeners

Service Worker API

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

Dans la pratique

Service Worker API

- Service worker mis à dispo par M. Chabloz

[https://github.com/lgavillet/webmobui-22/blob/master/Ressources/
workerCacheFetched.js](https://github.com/lgavillet/webmobui-22/blob/master/Ressources/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')`

Intégration - Projet

Service Worker API

- Télécharger le fichier <https://github.com/lgavillet/webmobui-22/blob/master/Ressources/workerCacheFetched.js>
- A la différence des autres fichiers, nous n'allons ni éditer celui-ci, ni l'importer dans notre code via des "imports". Il doit donc être servi tel quel, sans passer par webpack
- Vous pouvez le placer dans le répertoire "public" et non "src"
- Il sera alors possible d'y accéder via <http://localhost:8080/workerCacheFetched.js>

Intégration - Projet

Service Worker API

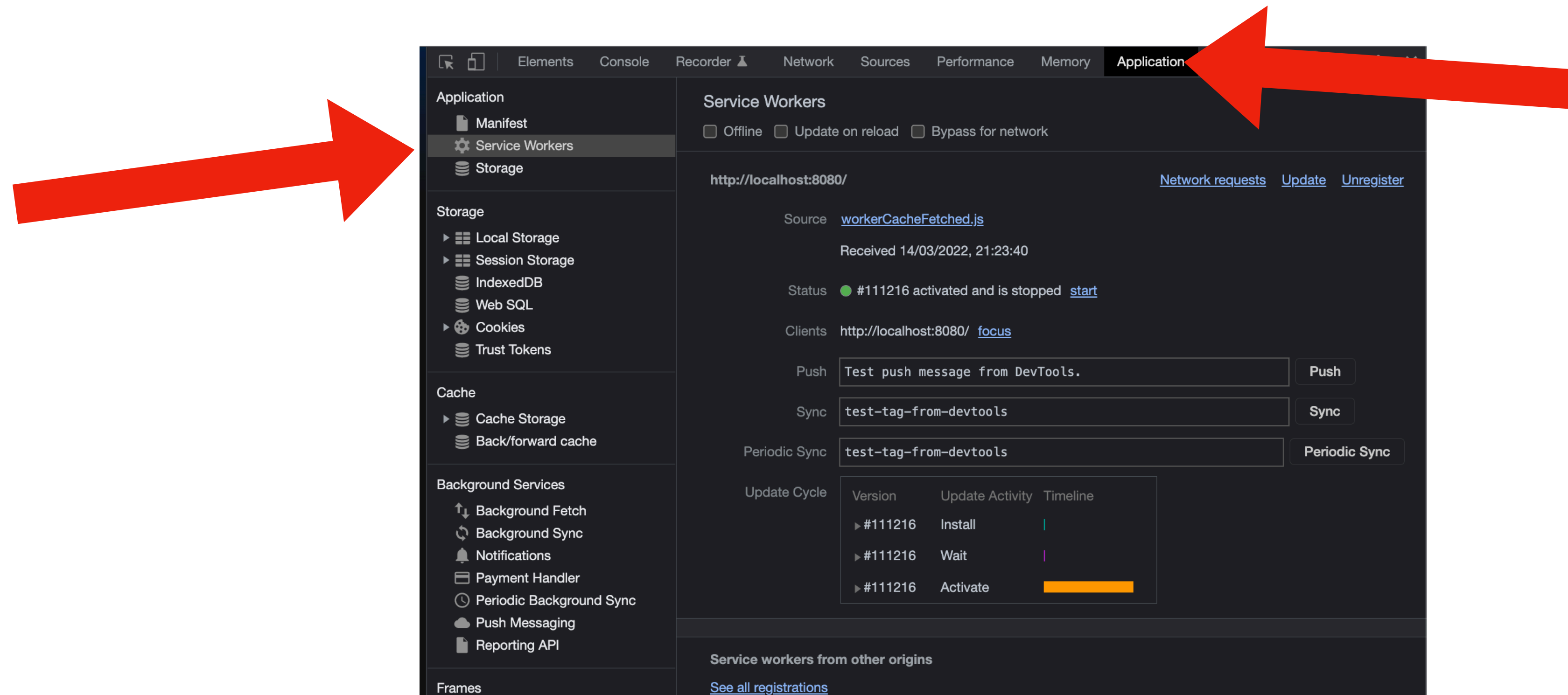
- Il faut ensuite l'enregistrer auprès du browser
- Pour cela, ajouter la ligne que nous avons vu quelque part dans votre fichier index (à la fin, par exemple)

```
navigator.serviceWorker.register( ' /workerCacheFetched.js ' )
```

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 - Projet web pack

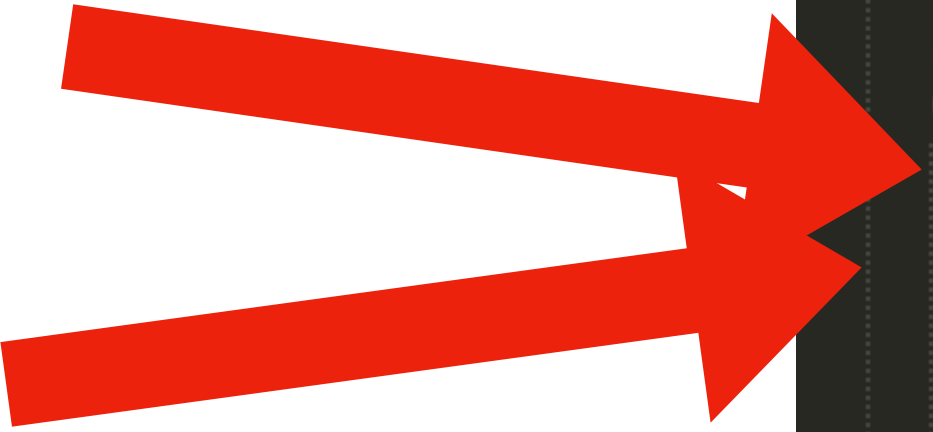
Service Worker API

- Nous utilisons avec webpack la notion de “hot reloading” ou de “live reload”
- Ces outils ajoutent des requêtes dans l’app pour automatiquement rafraichir la page, lorsque l’on modifie des fichiers javascript
- Problème: Ils entrent en conflit avec le service worker et peuvent donner des résultats aléatoires...
- Solution: les désactiver

Intégration - Projet web pack

Service Worker API

- Se rendre dans le fichier webpack.config.js
- Dans “devServer”, s’assure qu’il y ait les deux valeurs “hot” et “liveReload” à false. Redémarrer le serveur pour prendre la nouvelle config (npm run start)



```
// Dev server config
devServer: {
  hot: false,
  liveReload: false,

  // Open browser on server start
  open: true,
  // Statit directory for non-compilable items
  static: {
    directory: path.join(__dirname, 'public'),
  },
},
};
```


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)