

L'Institut Supérieur des Technologies de l'Information & de la Communication

# Technologies & programmation web

---

## Chapitre 5

### **Progressive Web application (PWA)**

# Introduction

---

- Le terme **Progressive Web Application - PWA** est né en 2015 de l'association des deux termes « **web app** » (application web) et « **progressive enhancement** » (principe de l'amélioration progressive)
- Ce n'est pas une technologie comme pourrait l'être un framework ou une librairie JavaScript
- C'est avant tout un terme qui met l'accent sur le principe de l'amélioration progressive
- Aussi, c'est une manière de penser les applications web qui est mise en avant

# Avantages des PWA

---

- **Installation sur le périphérique:** PWA est utilisable sur différents périphériques (ordinateur, tablette et smartphone)
- C'est une application web **responsive** qui s'adapte aux différentes résolutions des supports sur lesquels elle peut être installée
- Accessibles hors ligne: Une PWA est utilisable hors ligne grâce à la technologie de « services worker » qui permet de sauvegarder les fichiers au niveau du navigateur
- Mise à jour automatique: PWA comporte un contenu mis à jour régulièrement grâce au processus de mise à jour du service worker
- **SÉCURITÉ:** PWA est sécurisée puisqu'elle est accessible uniquement en HTTPS grâce au certificat SSL

# Stockage sur client

---

## Principe :

-L'API WebStorage fournit un mécanisme de stockage de l'information côté client plus évolué que les simples cookies :

- La manipulation des données stockées est plus aisée
  - La taille maximale des informations stockables est plus grande
  - Les objets stockés dans le WebStorage sont simplement des paires clef/valeur, comme les cookies mais les données peuvent être plus structurées, puisqu'il s'agit d'objets JavaScript
- Les données stockées ne sont pas cryptées

# Stockage sur client

---

Les deux principaux mécanismes internes du Stockage Web sont :

- ❖ **sessionStorage** qui maintient un espace de stockage, séparé pour chaque origine différente, disponible le temps de la session de navigation ; autrement dit, les données sont conservées jusqu'au moment où la dernière fenêtre ou le dernier onglet ouvert du navigateur est fermé, sessionStorage est réinitialisé à chaque redémarrage du navigateur.
- ❖ **localStorage** permet de conserver les données d'une session de navigation à une autre. Cela permet par exemple de conserver sur la machine du client les paramètres de personnalisation d'un site Web.

# Stockage sur client :

---

## Enregistrer une valeur dans le stockage

### ❖ **Storage.itemKey = itemValue**

```
localStorage.prenom = "Foulen";  
localStorage.nom = "Ben Foulen";
```

- ❖ **Storage.setItem()** : utilisée pour la création d'une donnée, que pour la modification d'une donnée existante (si cette donnée existe déjà). Elle prend deux arguments — la clé de l'élément à créer/modifier, et la valeur associée à stocker.

```
var myObj = {};  
myObj.prenom = "Foulen";  
myObj.nom = "Ben Foulen";  
var chaineJSON = JSON.stringify(myObj)  
localStorage.setItem('etudiant', chaineJSON);
```

# Stockage sur client :

---

## Récupérer des données du stockage

❖ **Storage.itemKey;**

```
var prenom = localStorage.prenom;  
alert(prenom);
```

❖ **Storage.getItem()** : prend un seul argument, la clé de l'élément que vous souhaitez récupérer de l'objet de stockage pour le domaine.

```
var student = JSON.parse(localStorage.getItem("etudiant"));  
alert(student.nom);
```

# Stockage sur client :

---

## Supprimer des données du stockage

L'API de Stockage Web fournit aussi un couple de méthodes simples pour supprimer des données :

- ❖ **Storage.removeItem()** : prend un seul argument, la clé de l'élément que vous souhaitez supprimer, et le supprime de l'objet de stockage pour le domaine.
- ❖ **Storage.clear()** : ne prend pas d'argument, et vide l'ensemble des données de l'objet de stockage pour le domaine.



# Stockage sur client :

---

## Application

On suppose qu'on va donner la possibilité à l'utilisateur de choisir sa couleur de fond d'une page Web. On va enregistrer le choix fait par l'utilisateur en utilisant localStorage afin que celui-ci soit conservé pour ses prochaines visites.

Côté HTML, on va utiliser un élément de formulaire pour permettre à l'utilisateur de taper sa couleur préférée.

```
<form>
<div>
  <label for="bgtheme">Choisissez un thème (hexa)</label>
  <input class="text" id="bgtheme" value="FAFAFA"
    pattern=' [a-fA-F0-9]{6} ' ><br><br>
</div>
</form>
```

Question:

Donner le code JS nécessaire pour **sauvegarder** la couleur saisie par l'utilisateur dans la base de données du navigateur **localStorage**.

Vérifier si la donnée a bien été sauvegarder en utilisant l'Outils de développement de Google Chrome.

# Stockage sur client :

---

## Application

```
let bgColor = document.getElementById('bgtheme');

if(localStorage.getItem('bgtheme')){
    updateBg();
}else{
    setBg();
}

function updateBg(){
    let bg = localStorage.getItem('bgtheme');
    // ou let bg = localStorage.bgtheme;
    document.body.style.background = '#' + bg;
}

function setBg(){
    localStorage.setItem('bgtheme', bgColor.value);
    // ou localStorage.bgtheme = bgColor.value;
    updateBg()
}

bgColor.addEventListener('change', setBg);
```

# Web Manifest

---

## C'est quoi ?

Le manifeste d'une application web :

- est un fichier JSON
- fournit des informations sur l'application web : son nom, son auteur, une icône, une description

## Pourquoi il est utilisé?

Le but du manifeste est d'installer des applications web progressives sur le bureau ou l'écran d'accueil d'un appareil mobile pour offrir aux utilisateurs un accès plus rapide.

# Web Manifest

---

## Déploiement

Les manifestes des applications PWA sont déployés dans des pages HTML en utilisant une balise lien `<link>` dans l'entête `<head>` de la page HTML

Le nom du fichier doit être : **manifest.webmanifest** ou **manifest.json**

## Exemple

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

# Web Manifest

---

## Exemple d'un fichier Web Manifest

```
{
  "name": "Offline SODOKU",
  "short_name": "OfflineSudoku",
  "scope": "./",
  "icons": [{
    "src": "img/sudoku144.png",
    "sizes": "144x144",
    "type": "image/png"
  },
  {
    "src": "img/sudoku192.png",
    "sizes": "192x192",
    "type": "image/png"
  },
  {
    "src": "img/sudoku512.png",
    "sizes": "512x512",
    "type": "image/png"
  }
],
  "background_color": "#3367D6",
  "theme_color": "#3367D6",
  "display": "standalone",
  "start_url": "./"
}
```

# Web Manifest

---

## Propriétés du Web Manifest

- **name**: affiché sur l'écran de démarrage de l'application
- **short\_name**: affiché en dessous du raccourci sur le bureau ou l'écran d'accueil
- **description**: une description générale de l'application
- **start\_url**: l'URL qui est chargée en premier quand on ouvre l'application depuis son raccourci sur le bureau ou l'écran d'accueil
- **background\_color**: La couleur d'arrière-plan de l'écran de démarrage de l'application
- **theme\_color**: la couleur de thème général de l'application, utilisée notamment dans les barres de statut si elles sont affichées

# Web Manifest

---

## Propriétés du Web Manifest

- **display:** spécifie le mode d'affichage. Voici les différents modes disponibles triés par ordre de fallback :
  - **fullscreen:** toute la zone d'affichage disponible est utilisée et aucun agent utilisateur n'est montré.
  - **standalone:** comportement similaire à une application native. Cela peut signifier que l'application a sa propre fenêtre, sa propre icône dans le lanceur d'applications, etc. Dans ce mode, l'agent utilisateur va exclure les éléments d'interface qui permettent de contrôler la navigation mais peut inclure d'autres éléments comme une barre de statut par exemple.
  - **minimal-ui:** l'application va ressembler et se comporter comme une application autonome, mais elle aura quelques éléments d'interface permettant de contrôler la navigation. Les éléments varient en fonction du navigateur et du système.
  - **browser** (par défaut): l'application s'ouvre dans un nouvel onglet ou une nouvelle fenêtre du navigateur, en fonction du navigateur et de la plateforme
- **icons:** liste d'icônes de l'application de différentes résolutions, utilisées pour le raccourci et l'écran de démarrage. L'appareil choisira la meilleure icône automatiquement selon les cas.

# Web Manifest

---

## Test de fonctionnement

Il est possible de vérifier la prise en compte du manifeste en regardant dans l'onglet ***Applications*** des **Developer Tools** du navigateur.

Si l'application est basée sur l'utilisation du Web Manifest, la liste des propriétés du manifeste est alors affichée.



# Web Manifest

## Test de fonctionnement

The screenshot shows a web browser displaying a YouTube video player. The video player has a large orange 'WA' logo and the text 'Tutorial #4 Android Emulator'. The browser's address bar shows the URL <https://www.youtube.com/watch?v=1VVKhnHNBQ>. The Chrome DevTools Application panel is open, showing the 'Manifeste de l'application' (Application Manifest) for the current page. The manifest file is <manifest.webmanifest>.

**Identité**

Propriété	Valeur
Nom	YouTube
Nom court	YouTube
Description	
ID de l'application	<a href="https://www.youtube.com/?feature=ytca">https://www.youtube.com/?feature=ytca</a> <a href="#">En savoir plus</a>

calculée **Remarque :** id n'est pas spécifié dans le manifeste, start\_url est utilisé à la place. Pour spécifier un ID d'application correspondant à l'identité actuelle, définissez le champ id sur `/?feature=ytca`.

**Présentation**

Propriété	Valeur
URL de début	<a href="/?feature=ytca">/?feature=ytca</a>
Couleurs du thème	<span style="color: red;">■</span> #FF0000

# Web Manifest

---

## Application :

1- Créer un dossier ApplicationPWA contenant:

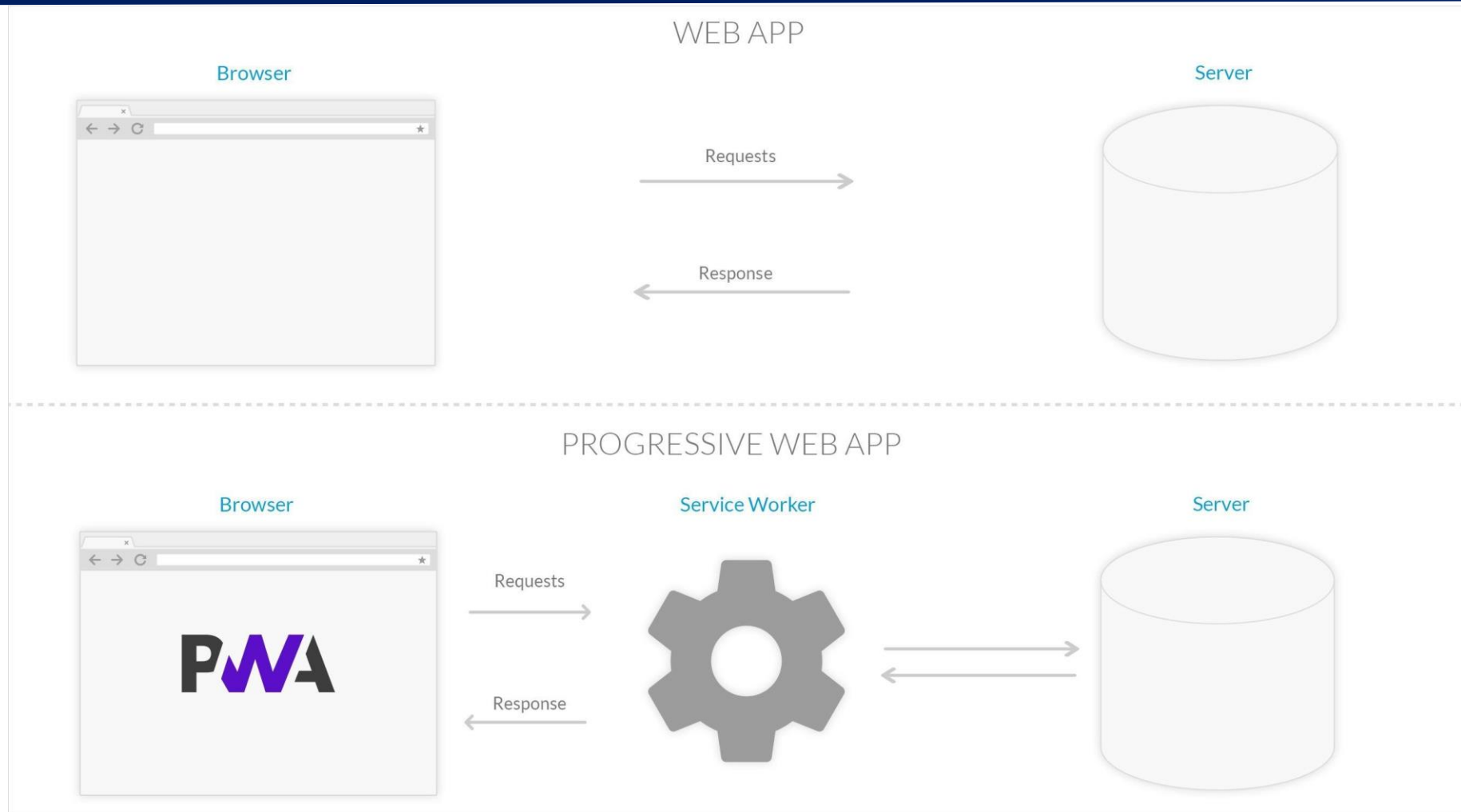
- une page d'accueil index.html
- un fichier manifest.json avec les propriétés de votre choix (nom, surnom, description, affichage, couleur d'arrière-plan et couleur de thème, URL de démarrage, icônes, etc.)
- un dossier img contenant les sources des icones à utiliser dans le fichier manifeste

2- Relier le manifeste à la page index.html

3- Déployer l'application sur votre serveur web local (par exemple http-server)

4- Utiliser votre navigateur web pour visualiser l'application et l'installer sur votre machine

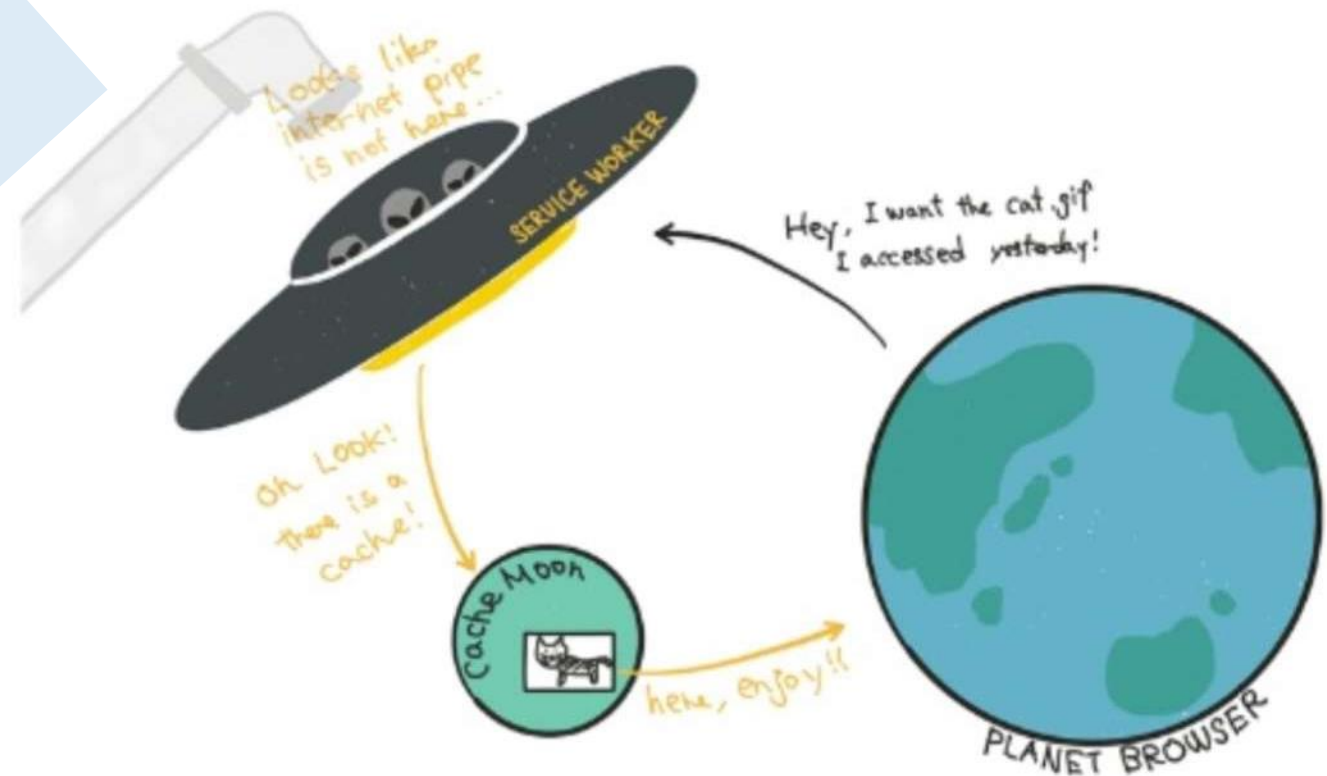
# Service Worker : C'est quoi?



# Service Worker : A quoi sert?

---

Servir des ressources en mode déconnecté



# Service Worker : Définition et Principe

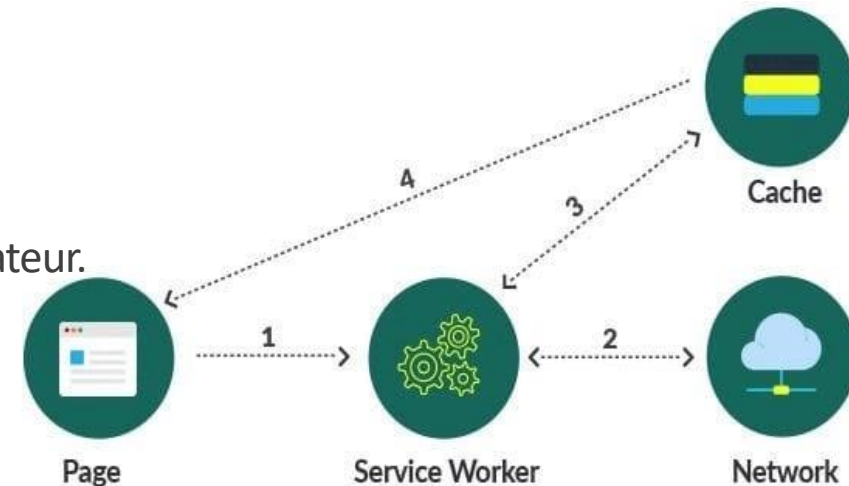
---

## Définition:

- Un service worker prend souvent la forme d'un fichier JavaScript
- Il joue le rôle de **proxy** entre une application, le réseau et le navigateur.

## Principe de fonctionnement:

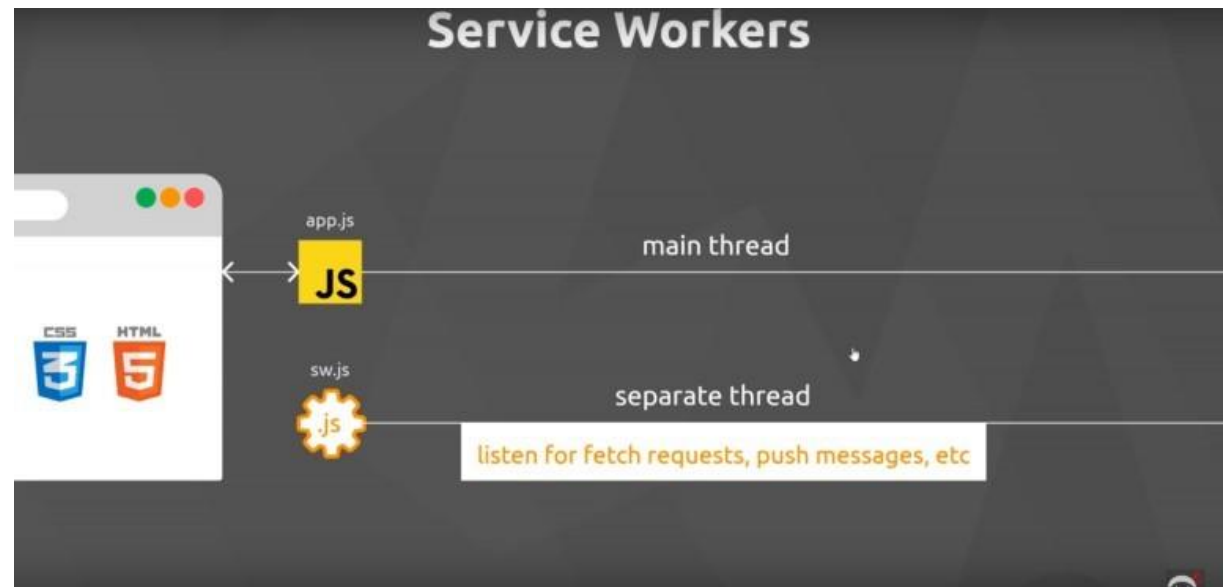
Il permet de contrôler l'application web auquel il est associé, en interceptant les requêtes de ressources, et en mettant en cache ces ressources pour donner une maîtrise complète de la manière dont doit se comporter l'application en cas d'indisponibilité (problème réseau, problème au niveau du serveur, ...)



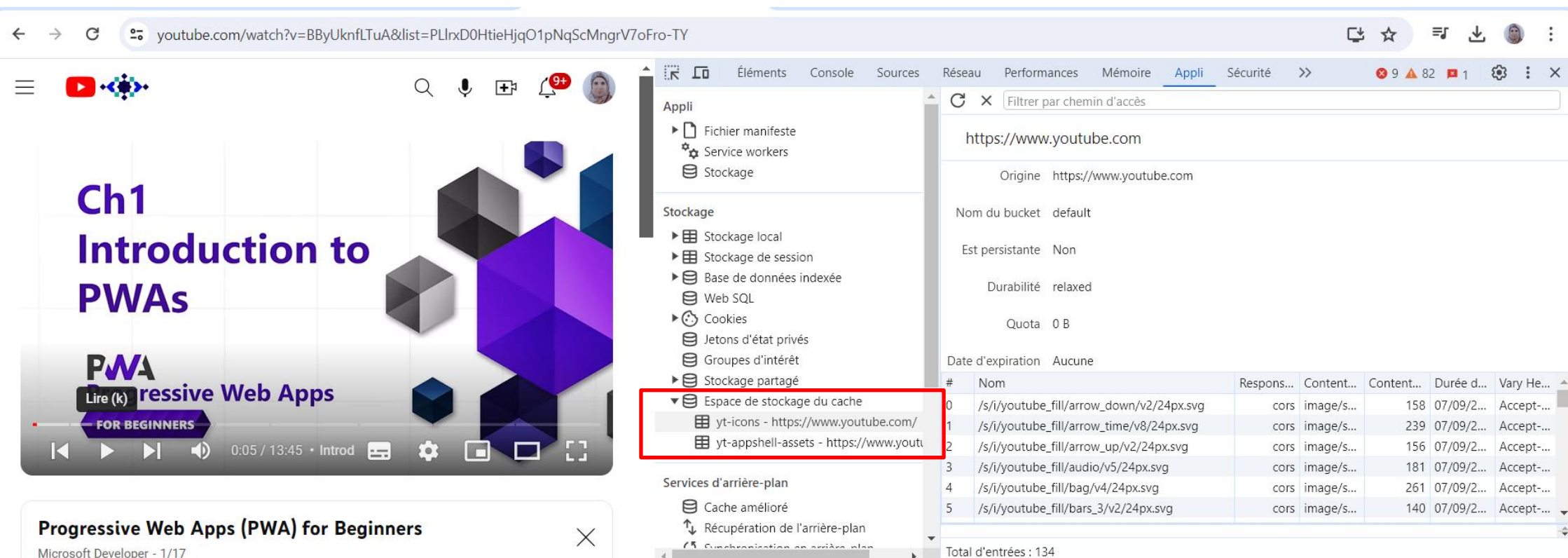
# Service Worker : Définition et Principe

---

- Il n'est pas lié au DOM du document
- Il s'exécute sur un thread distinct du thread principal du navigateur.
- > Il n'est pas bloquant



# Service Worker : Principe



The screenshot illustrates the principle of Service Workers by showing a web browser (Chrome) displaying a YouTube video titled "Ch1 Introduction to PWAs" and "Progressive Web Apps (PWA) for Beginners". The video player interface includes a progress bar, volume control, and a "FOR BEGINNERS" badge.

The Chrome DevTools interface is open, showing the "Storage" panel. The "Cache" storage area is highlighted, listing the following items:

- yt-icons - <https://www.youtube.com/>
- yt-appshell-assets - <https://www.youtube.com/>

The "Network" panel is also open, displaying a list of resources loaded from the origin <https://www.youtube.com>. The table below shows the details of these resources:

#	Nom	Respons...	Content...	Content...	Durée d...	Vary He...
0	/s/i/youtube_fill/arrow_down/v2/24px.svg	cors	image/s...	158	07/09/2...	Accept-...
1	/s/i/youtube_fill/arrow_time/v8/24px.svg	cors	image/s...	239	07/09/2...	Accept-...
2	/s/i/youtube_fill/arrow_up/v2/24px.svg	cors	image/s...	156	07/09/2...	Accept-...
3	/s/i/youtube_fill/audio/v5/24px.svg	cors	image/s...	181	07/09/2...	Accept-...
4	/s/i/youtube_fill/bag/v4/24px.svg	cors	image/s...	261	07/09/2...	Accept-...
5	/s/i/youtube_fill/bars_3/v2/24px.svg	cors	image/s...	140	07/09/2...	Accept-...

Total d'entrées : 134

# Service Worker : Comment le déployer?

---

1- Vérifier si le navigateur supporte le ServiceWorker

```
if('serviceWorker' in navigator) {  
    // Supported  
} else {  
    // Not supported  
}
```



# Service Worker : Comment le déployer?

---

## 2- Enregistrer le ServiceWorker (cas du serviceWorker supporté)

Afin d'enregistrer le ServiceWorker:

- Le fichier du ServiceWorker doit être accessible
- Le fichier du ServiceWorker ne doit contenir que du code relatif au service worker
- Il faut

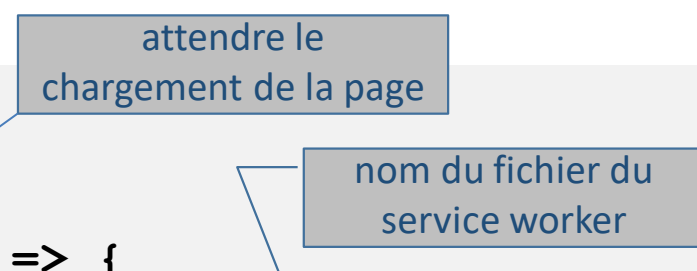
- ❖ attendre le chargement de la page
- ❖ ensuite transmettre le chemin du fichier du service worker à la méthode  
**`navigator.serviceWorker.register ()`**

# Service Worker : Comment le déployer?

---

## 2- Enregistrer le ServiceWorker (cas du serviceWorker supporté)

```
if ('serviceWorker' in navigator) {  
  window.addEventListener('load', () => {  
    navigator.serviceWorker.register('/service-worker.js')  
      .then(registration => {  
        console.log('Service Worker is registered', registration);  
      })  
      .catch(err => {  
        console.error('Registration failed:', err);  
      });  
  });  
}
```



attendre le chargement de la page

nom du fichier du service worker

# Service Worker : Comment le déployer?

---

## 2- Enregistrer le ServiceWorker (cas du serviceWorker supporté)

### Remarques:

- Le service est enregistré avec une **portée** (scope)
- La portée (scope) du service worker détermine le chemin à partir duquel le service worker intercepte les requêtes.
- La portée par défaut est le chemin vers le script du service worker et s'étend sur tous ses sous-répertoires.
- Le service worker est fonctionnel via **https**. L'utilisation du protocole https garantit qu'il n'y a pas eu d'usurpation lors de l'installation du service worker. Si la page est servie en http; le service worker ne sera pas installé.



# Service Worker : Comment le déployer?

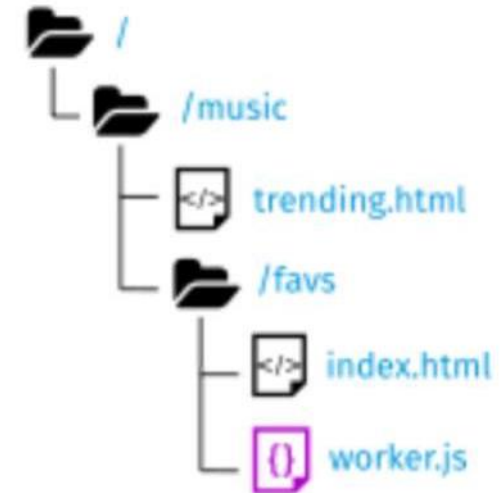
---

## 2- Enregistrer le ServiceWorker (cas du serviceWorker supporté)

### Exemple :

Supposons que worker.js est le fichier java script du service worker

- La portée par défaut du service est : .....
- La ressource que le service worker pourra servir est : .....
- La ressource que le service worker ne pourra pas servir est : .....



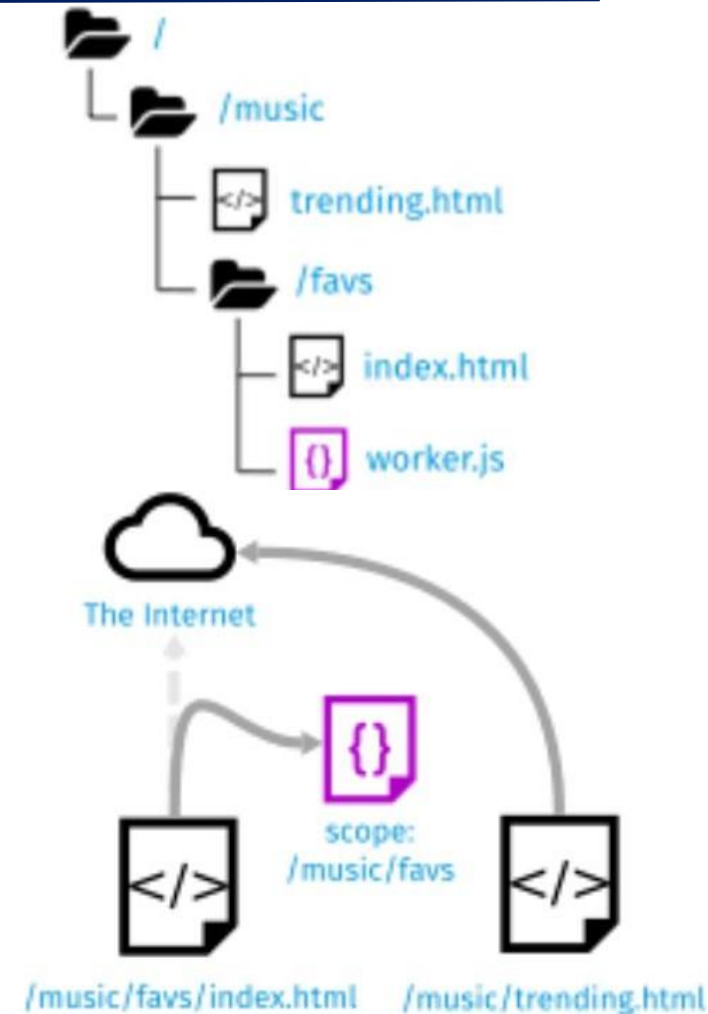
# Service Worker : Comment le déployer?

## 2- Enregistrer le ServiceWorker (cas du serviceWorker supporté)

### Exemple :

Supposons que `worker.js` est le fichier java script du service worker

- La portée par défaut du service est : `/music/favs/`
- La ressource que le service worker pourra servir est : `index.html`
- La ressource que le service worker ne pourra pas servir est : `trending.html`



# Service Worker : Son cycle de vie

---

Suite à son enregistrement, le service worker passe par un cycle de vie qui comprend les trois étapes suivantes:



Télécharger

Installer

Activer



# Service Worker : Son cycle de vie

---

## 1- Télécharger

Le service worker est immédiatement téléchargé lorsqu'un utilisateur accède pour la première fois à une page ou à un site contrôlé par un service worker.

## 2- Installer

Une fois téléchargé par le navigateur, le service worker peut être installé grâce à l'évènement **install**. Cet évènement sera déclenché si le navigateur considère le service worker comme nouveau (première installation ou une mise à jour).

L'évènement **install** est utilisé pour définir les instructions qui s'exécuteront durant l'installation, principalement les instructions de mise en cache des ressources.

# Service Worker : Son cycle de vie

---

Exemple :

```
var cacheName = 'first-test';
var filesToCache = ['/FirstApplication/index.html'];

self.addEventListener('install', function (e) {
  console.log('[ServiceWorker] Install');
  e.waitUntil(
    caches.open(cacheName).then(function (cache) {
      console.log('[ServiceWorker] Caching app files', );
      return cache.addAll(filesToCache);
    })
  );
});
```



# Service Worker : Son cycle de vie

---

## 3- Activer

Suite à la première installation d'un nouveau service worker (intallation avec succès), on peut passer à l'étape d'après : l'activation du service worker grâce à l'événement **activate**

Cet événement permet éventuellement de nettoyer les vieux caches associés avec la version précédente du service worker.

# Service Worker : Son cycle de vie

---

```
var cacheName = 'first-test';
var filesToCache = ['/FirstApplication/index.html'];

// ici gestion de l'évènement install
// ...

self.addEventListener('activate', function (e) {
  console.log('[ServiceWorker] Activate');
  e.waitUntil(
    caches.keys().then(function (keyList) {
      return Promise.all(keyList.map(function (key) {
        if (key !== cacheName) {
          console.log('[ServiceWorker] Removing old cache', key);
          return caches.delete(key);
        }
      }));
    })
  );
  return self.clients.claim();
});
```

# Service Worker : recherche d'une ressource

---

```
var cacheName = 'first-testt';  
var filesToCache = ['/FirstApplication/index.html'];  
  
// ici gestion des évènements install et activate  
// ....  
  
self.addEventListener('fetch', function (e) {  
  console.log('[ServiceWorker] Fetch', e.request.url);  
  e.respondWith(  
    caches.match(e.request).then(function (response) {  
      return response || fetch(e.request);  
    })  
  );  
});
```

# Hébergement PWA :

---

- Hébergement local sur un serveur web local
  - Hébergement sur Tomcat
  - Test du service worker
  - Test du stockage en cache
- Hébergement distant sur un serveur d'hébergement en ligne
  - À voir en TP

# Hébergement PWA : Local sur un serveur web

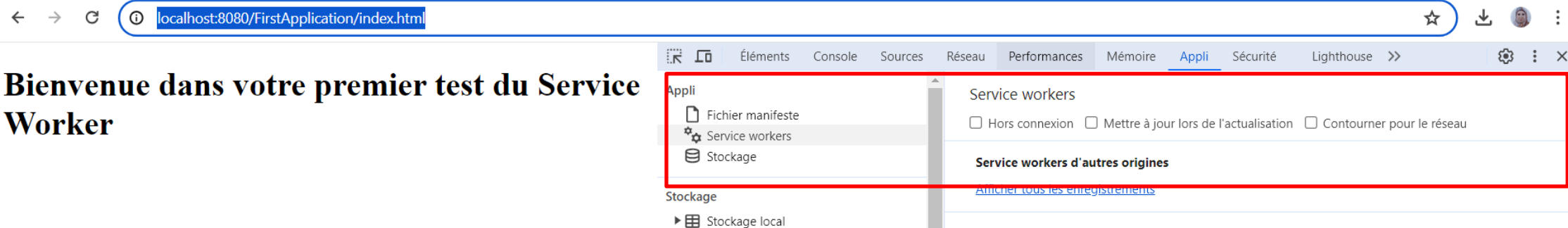
---

- Pré-requis : Installation du Serveur Tomcat
- Création d'un répertoire 'FirstApplication' contenant :
  - Dossier : src
  - Fichiers html basiques : index.html , message.html
  - Fichiers javascript : istic-sw.js
- Placer le répertoire 'FirstApplication' sous le serveur Tomcat au niveau du dossier webapps (dossier du serveur tomcat contenant toutes les applications web à déployer sur le serveur)
- Modifier les fichiers index.html et message.html de sorte qu'ils affichent des messages différents
- Démarrer Tomcat
- Accéder à l'adresse <http://localhost:8080/FirstApplication/index.html>

# Hébergement PWA : Local sur un serveur web

- index.html

```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <meta charset="utf-8">
    <title>Service Worker- Test APPLICATION</title>
  </head>
  <body>
    <h1>Bienvenue dans votre premier test du Service Worker</h1>
  </body>
</html>
```



# Hébergement PWA : Local sur un serveur web

---

- Au niveau du fichier index.html ajouter un script qui permet d'enregistrer le service worker istic-sw.js

```
<script type="text/javascript">
  if ('serviceWorker' in navigator) {
    window.addEventListener('load', function() {
      navigator.serviceWorker.register('/FirstApplication/istic-sw.js').then(function(registration) {
        // Registration was successful
        console.log('ServiceWorker registration successful with scope: ', registration.scope);
      }, function(err) {
        // registration failed :(
        console.log('ServiceWorker registration failed: ', err);
      });
    });
  }
</script>
```

# Hébergement PWA : Local sur un serveur web

- Actualiser la page dans le navigateur

localhost:8080/FirstApplication/index.html

## Bienvenue dans votre premier test du Service Worker

**Service workers**

☐ Hors connexion ☐ Mettre à jour lors de l'actualisation ☐ Contourner pour le réseau

**http://localhost:8080/FirstApplication/** [Requêtes réseau](#) [Mettre à jour](#) [Annuler l'enregistrement](#)

Source [istic-sw.js](#)

Reçu : 15/04/2024 17:29:05

État ● #315 activé et est en cours d'exécution [arrêter](#)

Push  [Push](#)

Synchronisation  [Synchronisation](#)

Synchronisation  [Synchronisation périodique](#)

Cycle de mise à jour

Version	Mettre à jour l'activité	Chronologie
▶ #315	Install	<div></div>
▶ #315	Wait	
▶ #315	Activate	

**Console** Nouveautés

ServiceWorker registration successful with scope: <http://localhost:8080/FirstApplication/> [Requêtes réseau](#) [Mettre à jour](#) [Annuler l'enregistrement](#) [index.html:14](#)



# Hébergement PWA : Local sur un serveur web

---

- Modifier le service worker afin d'installer et activer le cache des ressources

```
var cacheName = 'first-testtt';
var filesToCache = ['/FirstApplication/index.html'];

self.addEventListener('install', function (e) {
  console.log('[ServiceWorker] Install');
  e.waitUntil(
    caches.open(cacheName).then(function (cache) {
      console.log('[ServiceWorker] Caching app files', );
      return cache.addAll(filesToCache);
    })
  );
});
```

```
self.addEventListener('activate', function (e) {
  console.log('[ServiceWorker] Activate');
  e.waitUntil(
    caches.keys().then(function (keyList) {
      return Promise.all(keyList.map(function (key) {
        if (key !== cacheName) {
          console.log('[ServiceWorker] Removing old cache', key);
          return caches.delete(key);
        }
      }));
    })
  );
  return self.clients.claim();
});

self.addEventListener('fetch', function (e) {
  console.log('[ServiceWorker] Fetch', e.request.url);
  e.respondWith(
    caches.match(e.request).then(function (response) {
      return response || fetch(e.request);
    })
  );
});
```

# Hébergement PWA : Local sur un serveur web

- Modifier le service worker afin d'installer et activer le cache des ressources

The screenshot shows a web browser at `localhost:8080/FirstApplication/index.html`. The page content reads: **Bienvenue dans votre premier test du Service Worker**.

The DevTools interface is open, showing the **Service workers** panel. The left sidebar lists various storage and service worker options. The main panel shows the service worker `http://localhost:8080/FirstApplication/` with source `istic-sw.js`. It indicates that the worker is active and running.

The **Console** panel at the bottom shows the following messages:

```
ServiceWorker registration successful with scope: http://localhost:8080/FirstApplication/
[ServiceWorker] Install
[ServiceWorker] Caching app files
```

Red boxes highlight the 'Espace de stockage du cache' in the sidebar, the 'Service workers' panel, and the console messages.

# Hébergement PWA : Local sur un serveur web

- Arrêter le serveur Tomcat
- Accéder au deux ressources index.html et message.html

localhost:8080/FirstApplication/index.html

## Bienvenue dans votre premier test du Service Worker

The screenshot shows the Chrome DevTools interface with the 'Service workers' panel open. The left sidebar lists various storage and service worker options. The main panel shows the registration details for a service worker at the URL 'http://localhost:8080/FirstApplication/'. The status is 'État #316 activé et est en cours d'exécution' (State #316 active and running). The 'Push' button is labeled 'Testez le message push dans les outils de développement.' (Test the push message in the development tools). The 'Synchronisation' (Synchronization) section shows a tag 'test-tag-from-devtools' and a 'Synchronisation périodique' (Periodic synchronization) option. The 'Cycle de mise à jour' (Update cycle) table shows the status of the service worker across different versions.

Version	Mettre à jour l'activité	Chronologie
▶ #316	Install	
▶ #316	Wait	██████████
▶ #316	Activate	

Console: ServiceWorker registration successful with scope: <http://localhost:8080/FirstApplication/> index.html:14  
[ServiceWorker] Fetch <http://localhost:8080/favicon.ico> istic-sw.js:24

# Hébergement PWA : Local sur un serveur web

The screenshot shows a web browser at the address `localhost:8080/FirstApplication/message.html`. The page displays an error message: "Ce site est inaccessible" (This site is inaccessible), explaining that the page might be temporarily unavailable or moved. Below this, the error code `ERR_FAILED` is shown.

The Chrome DevTools interface is open, with the "Service workers" panel selected. The panel shows the service worker for `http://localhost:8080/FirstApplication/` with source `istic-sw.js`. The status is "État: #316 activé et est en cours d'exécution" (Status: #316 activated and is running). The "Cycle de mise à jour" (Update cycle) section shows a table of updates:

Version	Mettre à jour l'activité	Chronologie
#316	Install	
#316	Wait	<div></div>
#316	Activate	

The "Console" panel at the bottom shows several error messages:

- `was rejected.`
- `Uncaught (in promise) TypeError: Failed to fetch at istic-sw.js:25:72`
- `[ServiceWorker] Fetch http://localhost:8080/FirstApplication/message.html`
- `The FetchEvent for "http://localhost:8080/FirstApplication/message.html" resulted in a network error response: the promise was rejected.`
- `Uncaught (in promise) TypeError: Failed to fetch at istic-sw.js:25:72`

# Hébergement PWA : Local sur un serveur web

---

- Modifier le fichier `istic-sw.js` afin d'activer le cache pour la ressource `message.html`
- Démarrer le serveur et faire les tests nécessaires
- Déplacer le service worker vers le dossier ***src*** et vérifier si vous pouvez toujours accéder aux ressources `index.html` et `message.html`