

Progressive Web App PWA

Intervenant : Jean-Frédéric VINCENT



The background of the slide is a close-up photograph of green leaves covered in numerous water droplets. The droplets are of various sizes and are scattered across the surface of the leaves, which have prominent veins. The lighting creates highlights on the droplets, giving them a three-dimensional appearance.

Module 2

Le Manifest, le service worker et l'API cache

Création du manifest

Création du service worker

Mise en place du cache

Le Manifest, le service worker et l'API cache

Le fichier Manifest JSON



Le **fichier manifest.json** est central dans la création d'une PWA. Il fournit les informations requises au navigateur sur l'application et permet ainsi l'installation de cette dernière sur l'appareil de l'utilisateur. Il fournit entre autre le nom de l'application, les couleurs de l'interface, les icones à utiliser et ainsi de suite.

<https://developer.mozilla.org/fr/docs/Mozilla/Add-ons/WebExtensions/manifest.json>

Le Manifest, le service worker et l'API cache

Le fichier manifest.json

attribut	valeur	description
name		Le nom de l'application
short_name		le nom "court" sous l'icone
start_url		Fichier html
scope	.	page inclues dans le PWA experience
display	standalone	barre de navigation ou pas
background_color	#fff	couleur background au chargement
theme_color	#ccc	
description	lorem ipsum	texte de description
dir	ltr	sens de lecture
lang	fr-FR	
direction	portrait-primary	
icons	[...]	définition des icones sur l'écran du téléphone

Le Manifest, le service worker et l'API cache

Le Manifest

- le fichier est nommé `manifest.json`
- il est placé à la racine du projet
- les données sont au format JSON

les pages HTML l'utilisant doivent le charger grâce à l'utilisation d'une balise :

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

Le Manifest, le service worker et l'API cache

Le Manifest

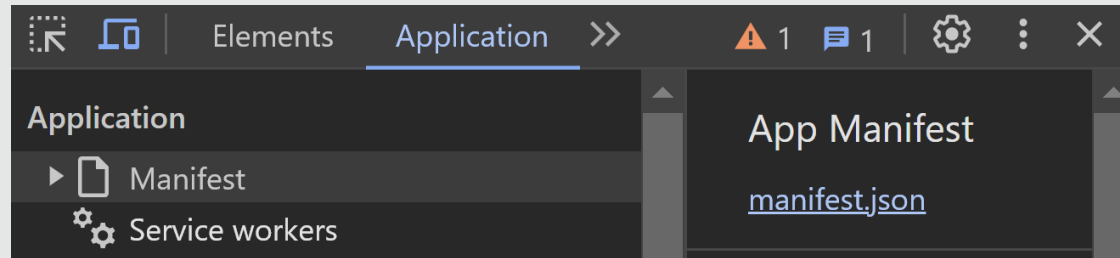
```
{
  "name": "PWA-Demo",
  "short_name": "PWA-Demo",
  "start_url": "index.html",
  "display": "standalone",
  "background_color": "#172554",
  "theme_color": "#172554",
  "orientation": "portrait-primary",
  "icons": [
    {
      "src": "images/pwa_72.png",
      "type": "image/png",
      "sizes": "72x72"
    },
    {
      "src": "images/pwa_128.png",
      "type": "image/png",
      "sizes": "128x128"
    },
    {
      "src": "images/pwa_192.png",
      "type": "image/png",
      "sizes": "192x192"
    }
  ]
}
```

Le Manifest, le service worker et l'API cache

Vérifier le Manifest

On peut vérifier son bon fonctionnement

depuis la console Chrome : "application/manifest".



Le Service Worker

Un service worker est un script indépendant de la page web qui y fait appel et est exécuté de manière **asynchrone** en arrière plan.

Il permet entre autre :

- de gérer et mettre en **cache** les données et fichiers utilisés par l'application
d'intercepter les requêtes réseau et d'y répondre si besoin
de répondre à des événements tels que l'installation ou l'activation de l'application
- de faciliter la mise en œuvre des notifications push
D'une manière globale, il améliore l'expérience utilisateur (sécurité, fluidité, hors ligne, ...) et la rapproche de celle d'une application native.

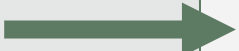
https://developer.mozilla.org/fr/docs/Web/API/Service_Worker_API

Le Manifest, le service worker et l'API cache

Service Worker

NE MANIPULE PAS LE DOM !

création du fichier "sw.js"
à la racine du projet (il contiendra le service worker)



```
> css
> img
✓ js
  JS app.js
  <> about.html
  <> index.html
  {} manifest.json
  JS sw.js
```

Le Manifest, le service worker et l'API cache

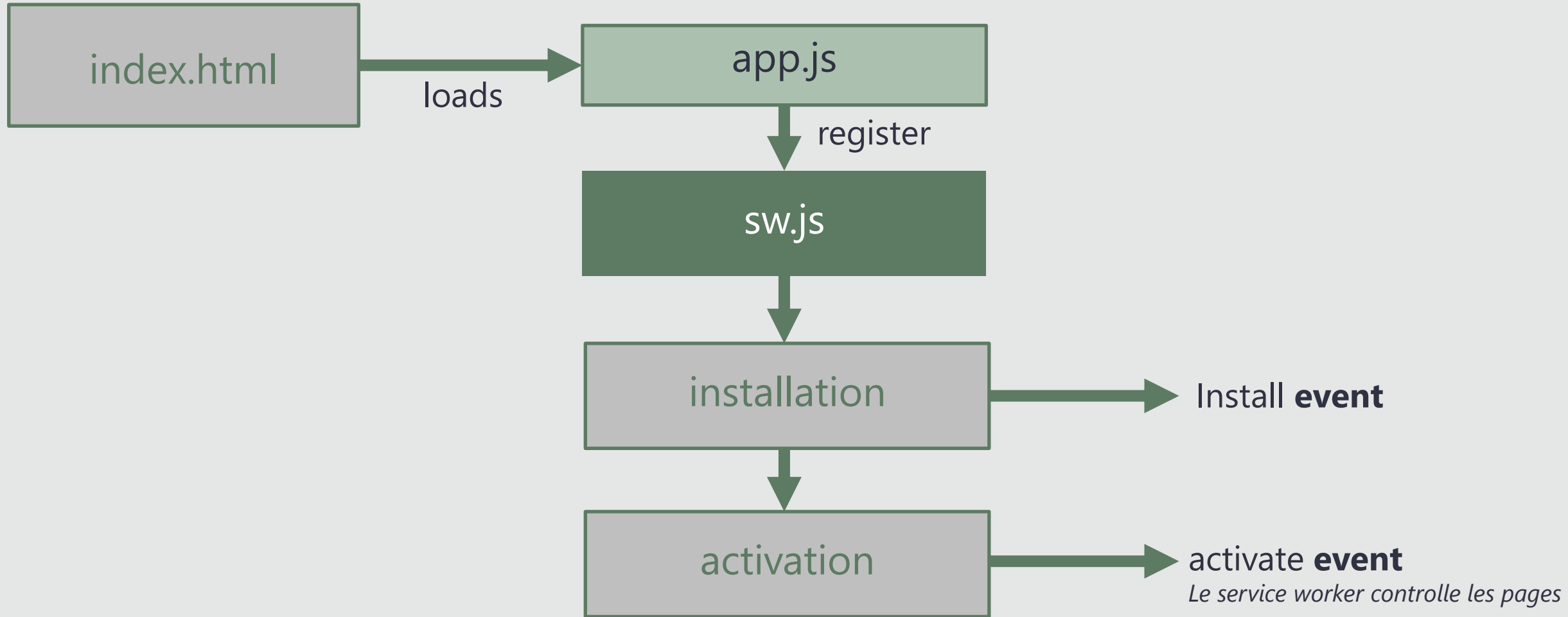
Service Worker – Cycle de vie

Lorsqu'il est mis en place, le service worker suit les étapes suivantes :

1. Téléchargement
2. Installation
3. Activation

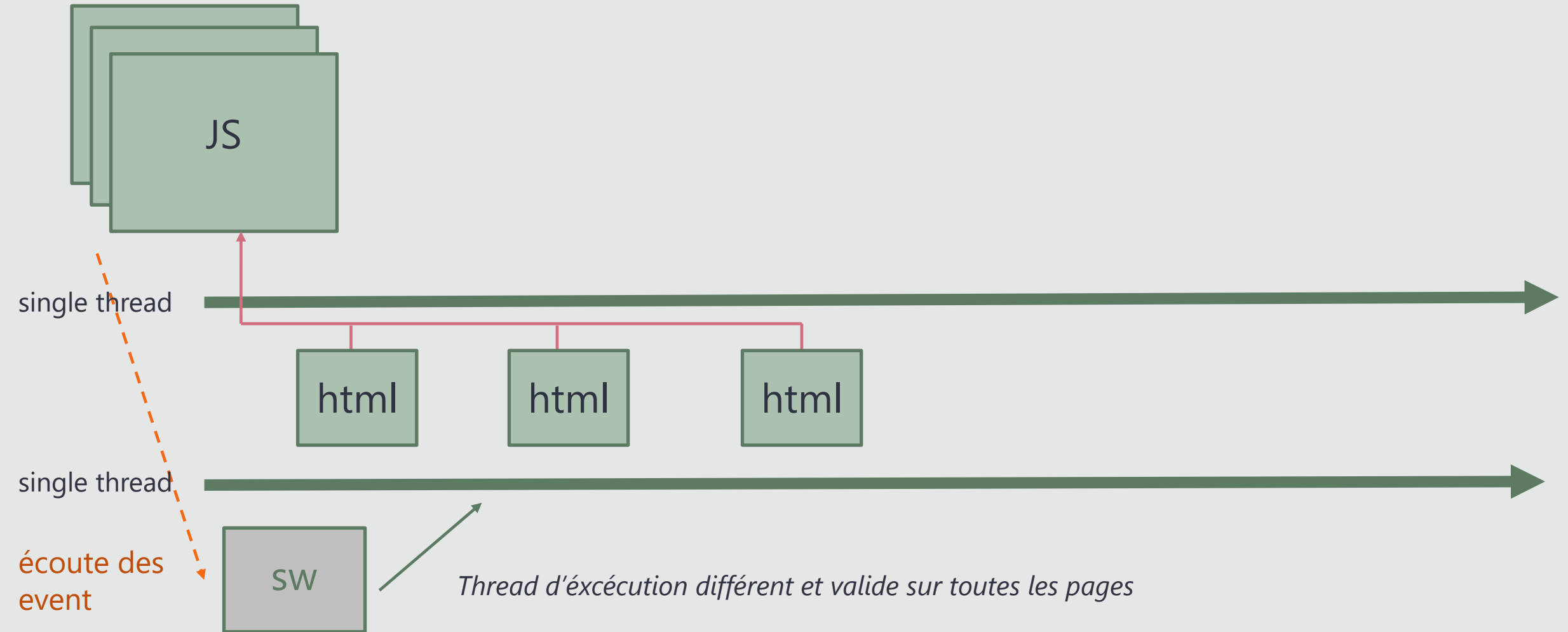
Le Manifest, le service worker et l'API cache

Service Worker – Cycle de vie



Le Manifest, le service worker et l'API cache

Service Worker



Le Manifest, le service worker et l'API cache

Service Worker – Cycle de vie

Le téléchargement (ou mise à jour) aura lieu si :

- à la première connection à une page / application contrôlée par un service worker
- si un événement du service est déclenché et qu'il n'a pas été mis à jour depuis plus de 24h
- si le service worker a été modifié (taille en octet)

Si une autre version du service worker est déjà installée, la nouvelle version sera téléchargée en arrière plan et restera en attente le temps qu'il reste des pages chargées utilisant le précédent.

Après cela, la version actuelle est activée.

(depuis le dev tool, on peut forcer la mise à jour du service worker).

Le Manifest, le service worker et l'API cache

Création du service worker

Démonstration

Le Manifest, le service worker et l'API cache

Cache API – La mise en cache

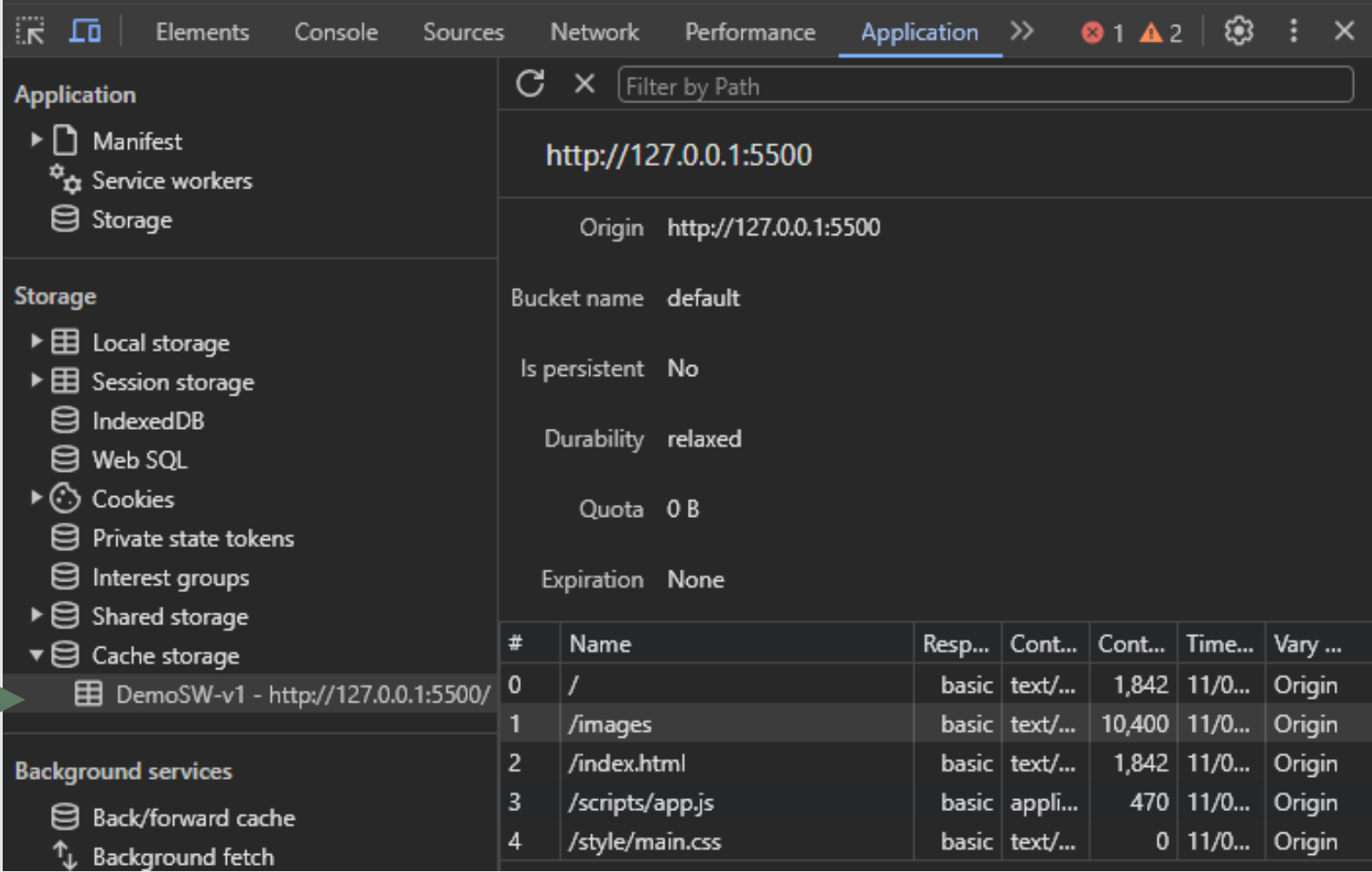
Pour pouvoir fonctionner hors ligne, notre application (par l'intermédiaire du service worker) va devoir dans un premier temps mettre en cache les données et fichiers utilisés puis dans un second les récupérer depuis ce cache.

Ainsi, même sans connexion, images, css, script, ... seront **déjà** sur le poste du client et pourront être distribués rapidement.

Le Manifest, le service worker et l'API cache

Cache API – La mise en cache

On peut vérifier l'état du cache depuis le dev tool :



The screenshot shows the Chrome DevTools Application tab. The left sidebar has a tree view with 'Cache storage' expanded, showing 'DemoSW-v1 - http://127.0.0.1:5500/'. A green arrow points to this entry. The right pane shows the details for this cache, including a table of cached resources.

#	Name	Resp...	Cont...	Cont...	Time...	Vary ...
0	/	basic	text/...	1,842	11/0...	Origin
1	/images	basic	text/...	10,400	11/0...	Origin
2	/index.html	basic	text/...	1,842	11/0...	Origin
3	/scripts/app.js	basic	appli...	470	11/0...	Origin
4	/style/main.css	basic	text/...	0	11/0...	Origin