

# Workbox

Y. Stroppa  
PSB

# Sommaire

**Introduction**

**Librarie workbox**

**Mise en oeuvre from scratch**

**Site réf :**

**<https://developers.google.com/web/tools/workbox>**

# Introduction

**Workbox est une librairie qui fournit un ensemble de best practices dans l'écriture de service workers.**

Precaching

Runtime Caching

Strategies

Request routing

Background sync

Helpful debugging

# From scratch

## Installation

**Elaboration d'un site from scratch avec nodejs et workbox.**

**Tout d'abord se positionner dans un répertoire et lancez la commande `#npm init`. Complétez les informations sur le projet, l'auteur, version et ligne de commande. Ensuite copier le fichier suivant `server.js` pour initialiser le service web que l'on va démarrer.**

```
const express = require('express');
const app = express();

// This serves static files from the specified directory
app.use(express.static(__dirname));

const server = app.listen(8081, () => {

  const host = server.address().address;
  const port = server.address().port;

  console.log('App listening at http://%s:%s', host, port);
});
```

Fichier `server.js` très simple pour démarrer notre web sur le port 8081

# **From scratch**

## **installation**

**On utilise le module express dans la configuration de notre serveur, donc il est nécessaire de compléter l'installation par la commande**

**#npm install express --save**

**Qui aura pour action d'installer le module express dans node-modules et de compléter le fichier package.json**

**Compléter le fichier index.html avec un contenu très léger.  
Qui sera compléter à l'aide du fichier style main.css.**

# From scratch

## Fichier index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta name="viewport" content="width=device-width, minimum-scale=1.0, initial-scale=1.0,
user-scalable=yes">
    <meta charset="utf-8">
    <title>workbox Lab</title>
    <link rel="stylesheet" href="styles/main.css">
  </head>
  <body>
    <header>
      <h1>PSB Cours PWA Workbox et service workers</h1>
    </header>
    <main>
      <label for="country">Country Name:</label>
      <input id="country" type="text" placeholder="enter country name"><br><br>
      <button id="get-image-name">Get Image Name</button><br><br>
      <div class="img-container" id="img-container">
        <!-- image added dynamically -->
      </div>
    </main>
    <footer>
    </footer>
    <script src="js/index.js"></script>
  </body>
</html>
```

# From scratch

## Fichier styles/main.css

```
body {
  align-items: center;
  display: flex;
  flex-direction: column;
  font-family: 'Roboto', 'Helvetica', 'Arial', sans-serif;
  height: 100%;
  justify-content: space-between;
  margin: 0;
  min-height: 100vh;
  text-align: center;
}

header {
  background-color: #2196f3;
  box-shadow: 0 2px 2px 0 rgba(0, 0, 0, 0.14), 0 3px 1px -2px rgba(0, 0, 0, 0.2), 0 1px 5px 0 rgba(0, 0, 0, 0.12);
  color: white;
  margin-bottom: 10px;
  width: 100%;
}

main {
  width: 100%;
}
```

# From scratch

## Fichier styles/main.css

```
label {
  padding: 5px;
  text-align: left;
}
input {
  border: 1px black solid;
  border-radius: 3px;
  font-size: 14px;
  min-height: 20px;
}
button {
  background-color: #2196f3;
  border: none;
  border-radius: 2px;
  box-shadow: 0 2px 2px 0
    rgba(0, 0, 0, 0.14), 0 3px 1px -2px
    rgba(0, 0, 0, 0.2), 0 1px 5px 0
    rgba(0, 0, 0, 0.12);
  color: white;
  font-size: 16px;
  font-weight: 500;
  height: 36px;
  margin: 5px 10px;
  min-width: 80px;
  overflow: hidden;
  padding: 0 10px;
  text-align: center;
  vertical-align: middle;
}

.img-container {
  background-color: #c5c5c5;
  margin: 5px;
  padding: 5px;
  visibility: hidden;
}

img {
  max-width: 100%;
}

a {
  color: white;
  font-weight: bold;
}

footer {
  align-items: center;
  background-color: #2196f3;
  display: flex;
  height: 10vh;
  justify-content: center;
  width: 100%;
}
```



# From scratch

## Arborescence du projet

```
./
├── index.html
├── js
├── node_modules
├── package.json
├── package-lock.json
├── server.js
├── service-worker.js
├── styles
└── js
    ├── index.js
    └── styles
        └── main.css
```

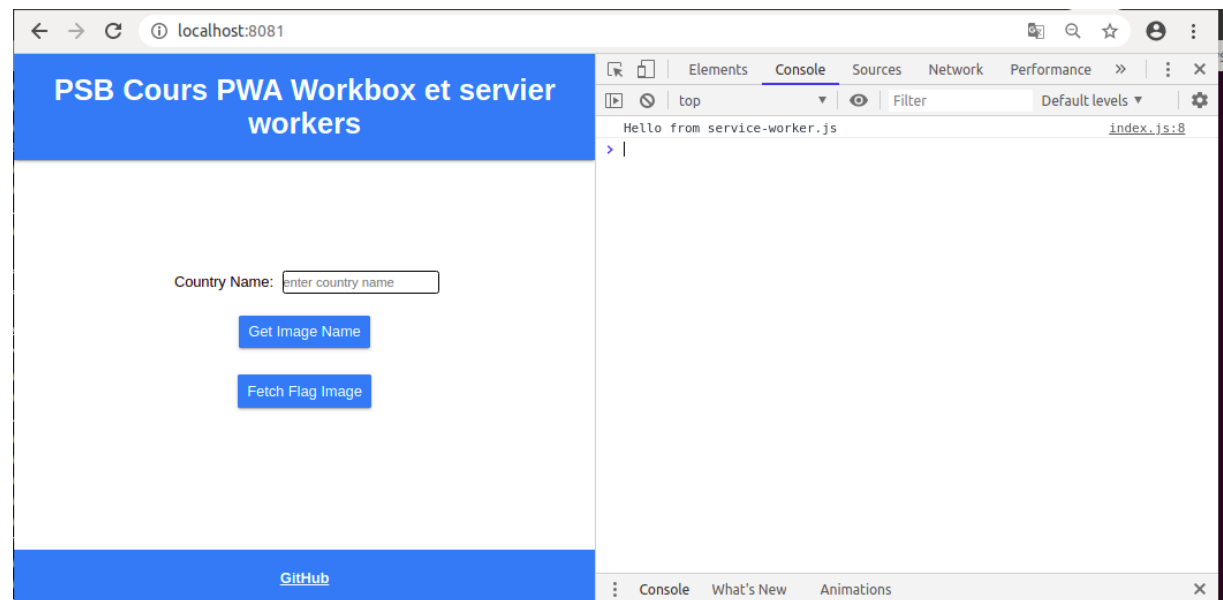
# From scratch

## Fichier js/index.js

Indication au navigateur d'aller charger le fichier service-worker.js à générer sur la racine de votre projet.

```
// Check that service workers are supported
if ('serviceWorker' in navigator) {
  // Use the window load event to keep the page load performant
  window.addEventListener('load', () => {
    console.log('Hello from service-worker.js');
    navigator.serviceWorker.register('/service-worker.js');
  });
}
```

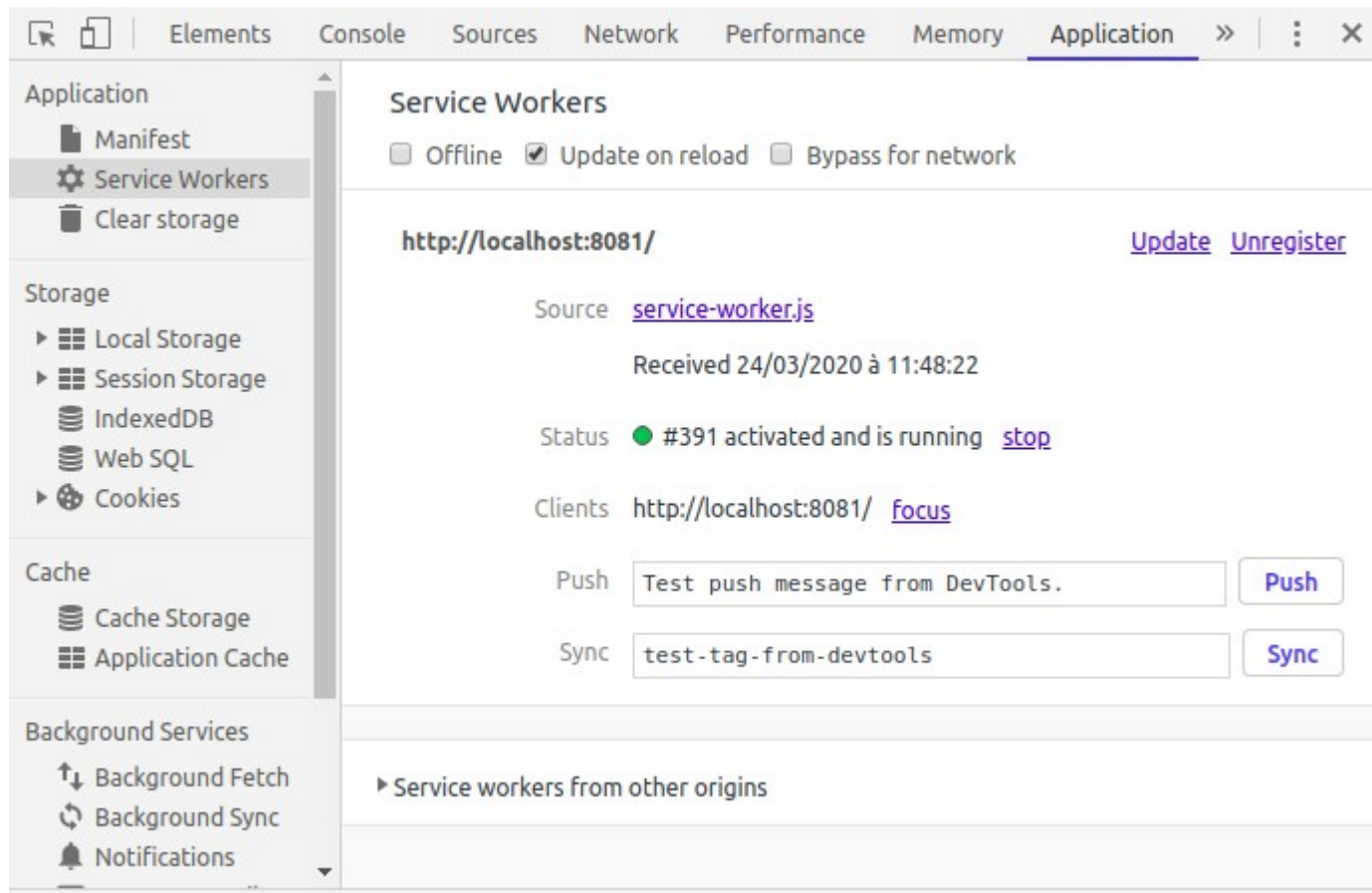
#npm start



# From scratch

## console de dev -- visualisation du SW

Le service workers est démarré



# From scratch

## Importation de workbox

**Pour nous simplifier la tâche, on va pouvoir importer directement un script workbox qui va nous permettre de définir les stratégies à utiliser dans notre web app.**

**On rajoute dans le fichier service-worker.js les lignes suivantes :**

```
// a partir du CDN
importScripts('https://storage.googleapis.com/workbox-cdn/releases/5.0.0/
workbox-sw.js');

if (workbox) {
  console.log(`Youpi! Workbox is loaded 🎉`);
} else {
  console.log(`Oups! Workbox didn't load 😞`);
}
```

# From scratch

## Utilisation de workbox

**Une fois workbox installé nous pouvons définir les stratégies à utiliser en fonction des natures de fichiers et du fonctionnement souhaité. Dans workbox on dispose des éléments suivantes :**

`workbox.routing.registerRoute`

`workbox.strategies`

CacheFirst : on regarde dans la cache en premier

NetworkFirst : on regarde du côté internet en premier

StaleWithRevalidate : on fournit le contenu du cache et on le rafraîchit à partir du réseau.

`workbox.RegExp`

# From scratch

## Utilisation de workbox workbox.routing

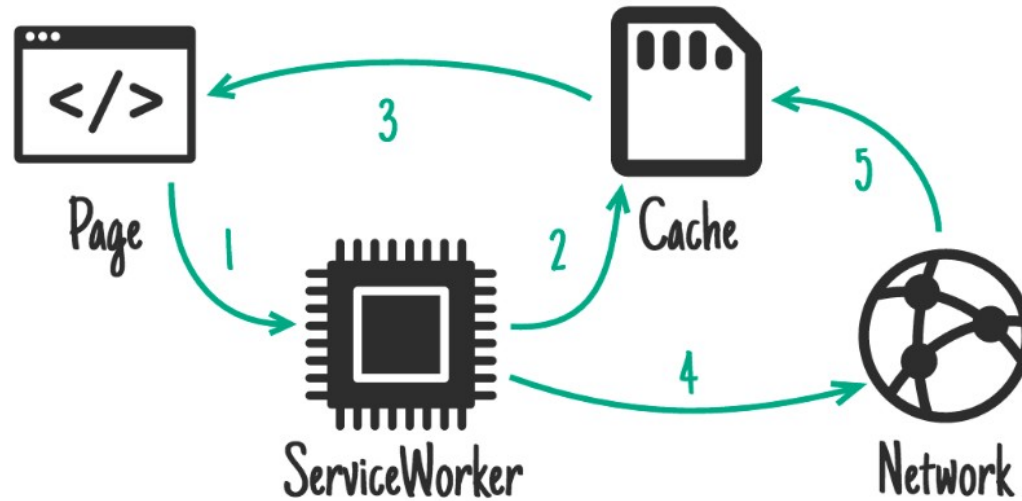
**Le service worker peut intercepter les requêtes pour une page. Il peut répondre au browser avec le contenu du cache, du contenu du réseau ou un contenu généré par le service worker.**

**workbox-routing est le module qui s'occupe de la construction de la route pour fournir les différentes réponses.**

**Comment fonctionne le Routing ?**

# Stratégies

Rappel = Stale-While-Revalidate



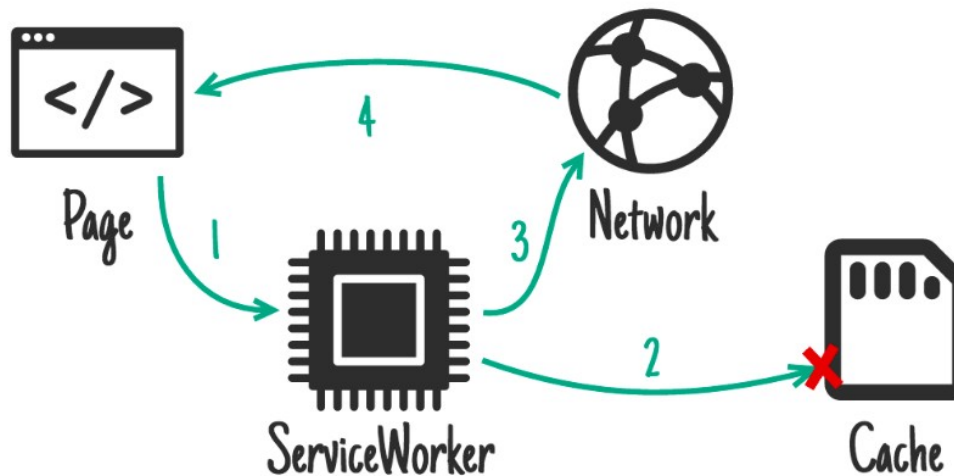
```
import {registerRoute} from 'workbox-routing';
import {StaleWhileRevalidate} from 'workbox-strategies';
```

```
registerRoute(
  new RegExp('/images/avatars/'),
  new StaleWhileRevalidate()
);
```

```
workbox.routing.registerRoute(
  new RegExp('\\.css$'),
  new workbox.strategies.StaleWhileRevalidate()
);
```

# Stratégies

Rappel = Cache First(Cache Falling Back to Network)



```
import {registerRoute} from 'workbox-routing';  
import {StaleWhileRevalidate} from 'workbox-strategies';
```

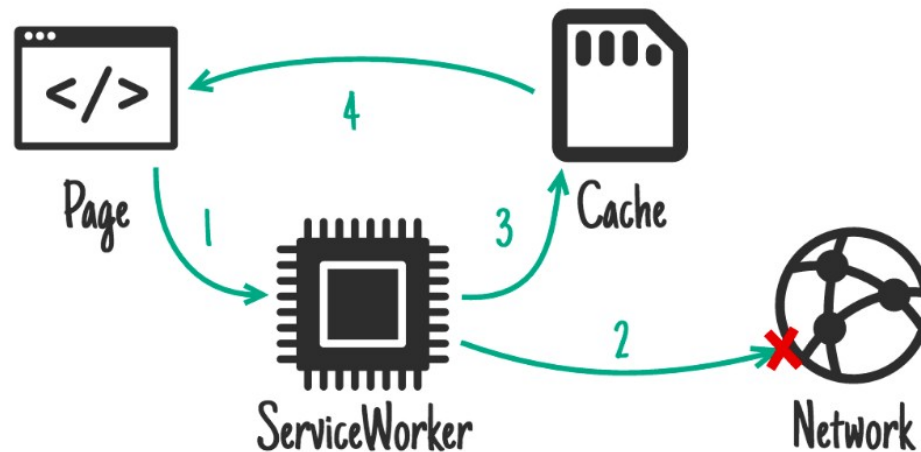
```
registerRoute(  
  new RegExp('/styles/'),  
  new CacheFirst()  
);
```

```
workbox.routing.registerRoute(  
  new RegExp('\\.js$'),  
  new workbox.strategies.CacheFirst()  
);
```



# Stratégies

Rappel = Network First(Network Falling Back to Cache)



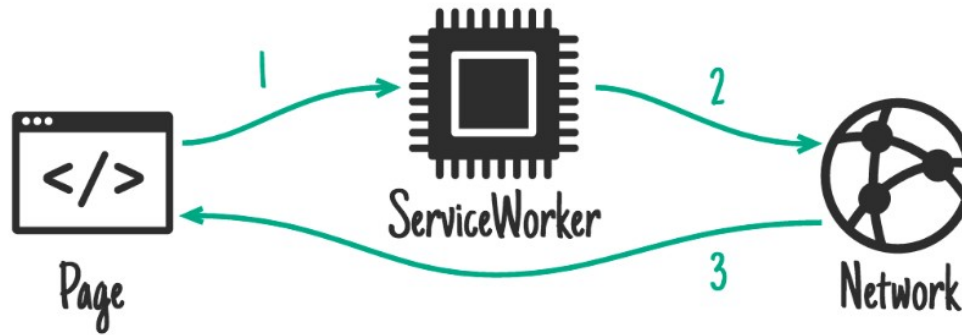
```
import {registerRoute} from 'workbox-routing';  
import {NetworkFirst} from 'workbox-strategies';
```

```
registerRoute(  
  new RegExp('/social-timeline/'),  
  new NetworkFirst()  
);
```

```
workbox.routing.registerRoute(  
  new RegExp('\\\\.js$'),  
  new workbox.strategies.NetworkFirst()  
);
```

# Stratégies

Rappel = Network Only



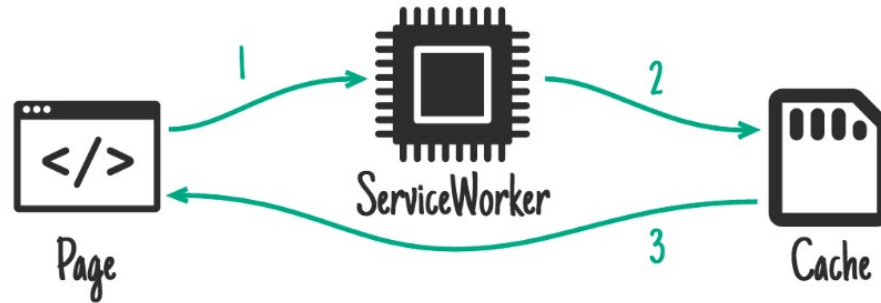
```
import {registerRoute} from 'workbox-routing';  
import {NetworkOnly} from 'workbox-strategies';
```

```
registerRoute(  
  new RegExp('/admin/'),  
  new NetworkOnly()  
);
```

```
workbox.routing.registerRoute(  
  new RegExp('\\.js$'),  
  new workbox.strategies.NetworkOnly()  
);
```

# Stratégies

## Rappel = Cache Only



```
import {registerRoute} from 'workbox-routing';  
import {CacheOnly} from 'workbox-strategies';
```

```
registerRoute(  
  new RegExp('/app/v2/'),  
  new CacheOnly()  
);
```

```
workbox.routing.registerRoute(  
  new RegExp('\\.js$'),  
  new workbox.strategies.CacheOnly()  
);
```

# From scratch

## Utilisation de workbox

**Pour chaque stratégie, on peut préciser des options supplémentaires :**

cacheName: nom du cache utilisé

```
import {registerRoute} from 'workbox-routing';  
import {CacheFirst} from 'workbox-strategies';
```

```
registerRoute(  
  new RegExp('/images/'),  
  new CacheFirst({  
    cacheName: 'image-cache',  
  })  
);
```

# From scratch

## Utilisation de workbox/plugins

**Workbox arrive avec un ensemble de plugins qui peuvent être utilisés avec ces stratégies comme :**

workbox-background-sync  
workbox-broadcast-update  
workbox-cacheable-response  
workbox-expiration  
workbox-range-requests

```
import {registerRoute} from 'workbox-routing';
import {CacheFirst} from 'workbox-strategies';
import {ExpirationPlugin} from 'workbox-expiration';

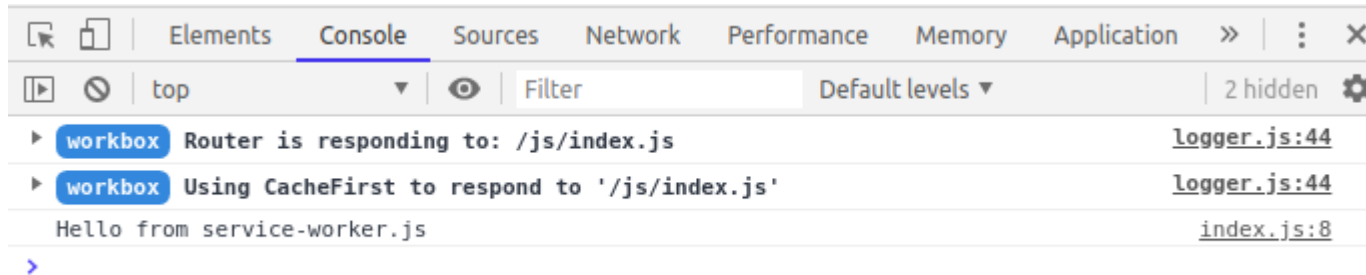
registerRoute(
  new RegExp('/images/'),
  new CacheFirst({
    cacheName: 'image-cache',
    plugins: [
      new ExpirationPlugin({
        // Only cache requests for a week
        maxAgeSeconds: 7 * 24 * 60 * 60,
        // Only cache 10 requests.
        maxEntries: 10,
      }),
    ],
  })
);
```

# From scratch

## Utilisation de workbox

### Stratégie pour les fichiers js par exemple :

```
workbox.routing.registerRoute(  
  new RegExp('\\.js$'),  
  new workbox.strategies.CacheFirst()  
);
```

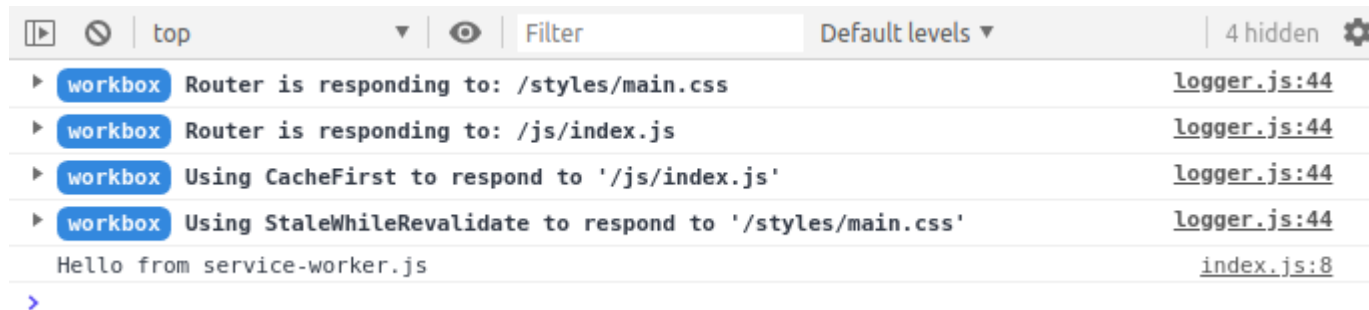


# From scratch

## Utilisation de workbox

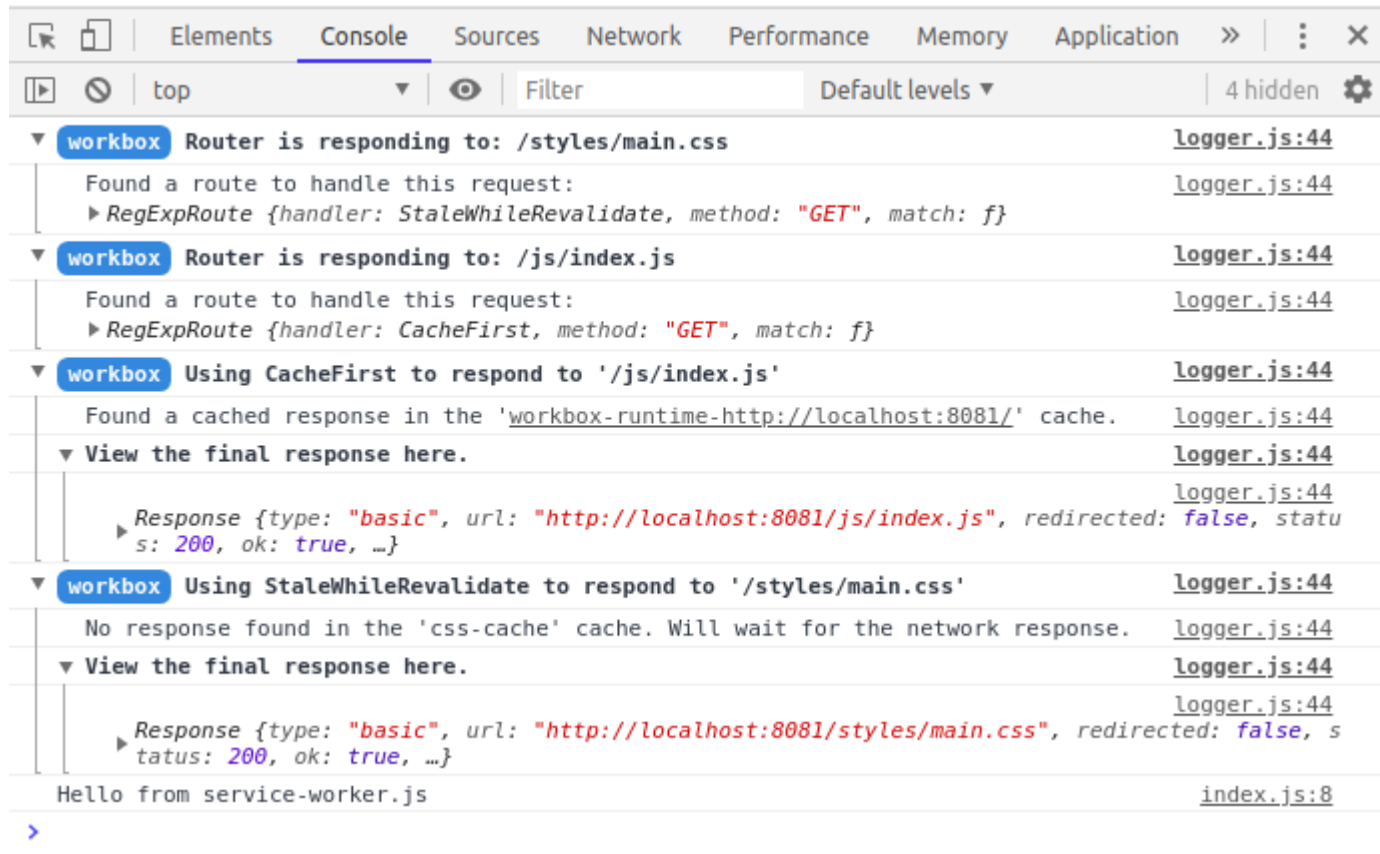
### Stratégie pour les fichiers css par exemple :

```
workbox.routing.registerRoute(  
  new RegExp('\\.css$'),  
  new workbox.strategies.StaleWhileRevalidate({  
    cacheName:"css-cache",  
  })  
);
```



# From scratch

## Utilisation de workbox



The screenshot shows a web browser's developer console with the 'Console' tab selected. The console displays several log messages from the Workbox library, indicating the routing and caching process for two requests: `/styles/main.css` and `/js/index.js`. The logs show that `/js/index.js` was served from the cache using the `CacheFirst` strategy, while `/styles/main.css` was not found in the cache and was therefore fetched from the network using the `StaleWhileRevalidate` strategy. The final response for `/styles/main.css` is shown with a status of 200 and `ok: true`. The console also shows a message 'Hello from service-worker.js' from `index.js:8`.

```
workbox Router is responding to: /styles/main.css logger.js:44
  Found a route to handle this request: logger.js:44
    ▶ RegExpRoute {handler: StaleWhileRevalidate, method: "GET", match: f}
workbox Router is responding to: /js/index.js logger.js:44
  Found a route to handle this request: logger.js:44
    ▶ RegExpRoute {handler: CacheFirst, method: "GET", match: f}
workbox Using CacheFirst to respond to '/js/index.js' logger.js:44
  Found a cached response in the 'workbox-runtime-http://localhost:8081/' cache. logger.js:44
  View the final response here. logger.js:44
    Response {type: "basic", url: "http://localhost:8081/js/index.js", redirected: false, statu
    s: 200, ok: true, ...}
workbox Using StaleWhileRevalidate to respond to '/styles/main.css' logger.js:44
  No response found in the 'css-cache' cache. Will wait for the network response. logger.js:44
  View the final response here. logger.js:44
    Response {type: "basic", url: "http://localhost:8081/styles/main.css", redirected: false, s
    tatus: 200, ok: true, ...}
Hello from service-worker.js index.js:8
```



# From scratch

## Precaching

<https://developers.google.com/web/tools/workbox/modules/workbox-precaching>

**Comment utiliser l'application offline ?**

**Pour le fonction en mode offline, le mécanisme de precaching est la meilleure solution.**

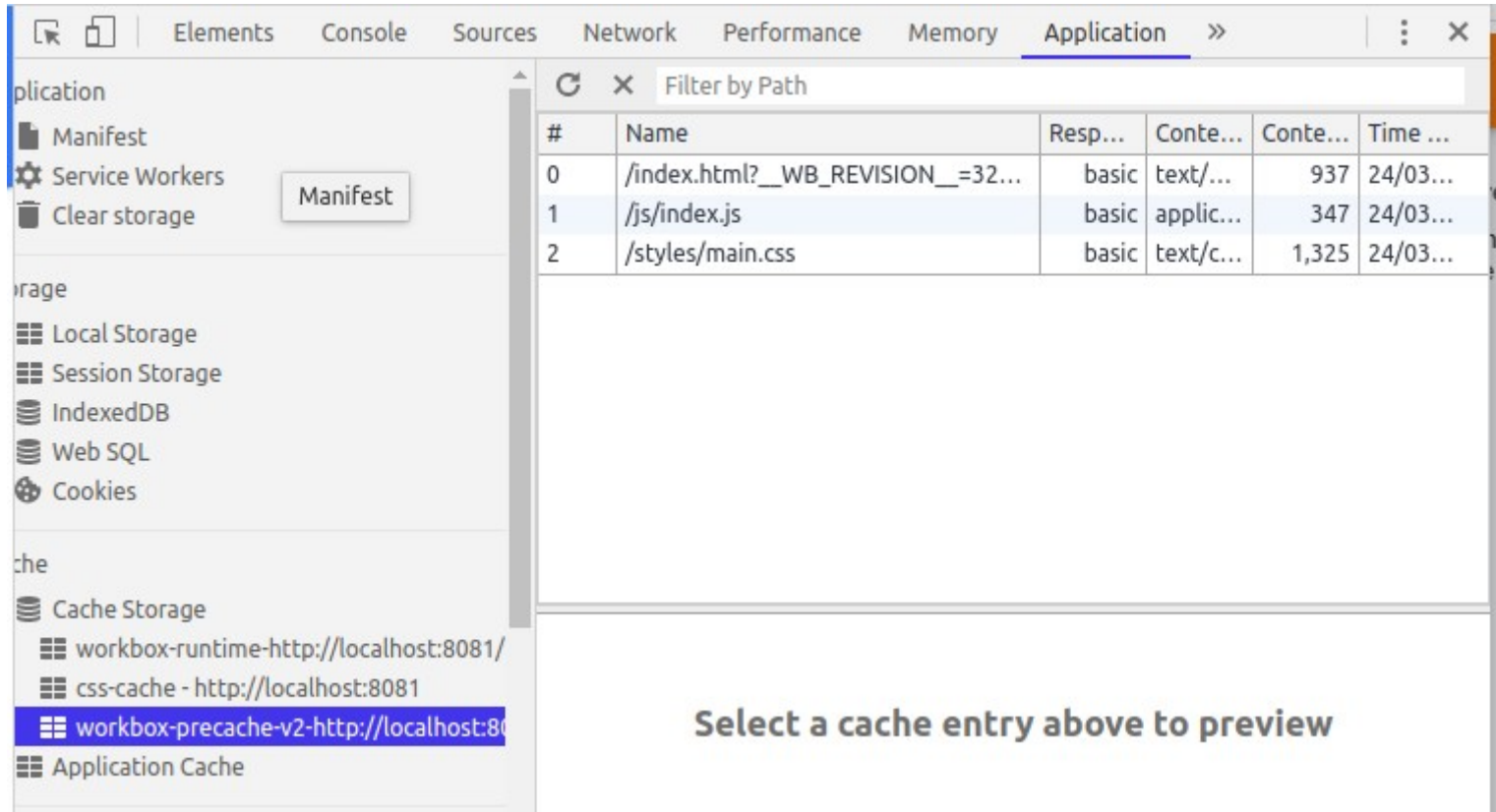
**On va pouvoir utiliser le pre cache pour stocker les éléments importants de notre application pour permettre une utilisation en offline.**

**Pour ce faire,**

```
// mise ne place d'un precaching
workbox.precaching.precacheAndRoute([
  {url: '/index.html', revision:'322154' },
  {url: '/js/index.js', revision:null },
  {url: '/styles/main.css', revision:null }
])
```

# From scratch

## Precaching



The screenshot shows the Chrome DevTools Application tab. On the left, the 'Cache' section is expanded, showing a list of cache entries. The entry 'workbox-precache-v2-http://localhost:8081/' is selected. On the right, the 'Network' tab is active, displaying a table of network resources.

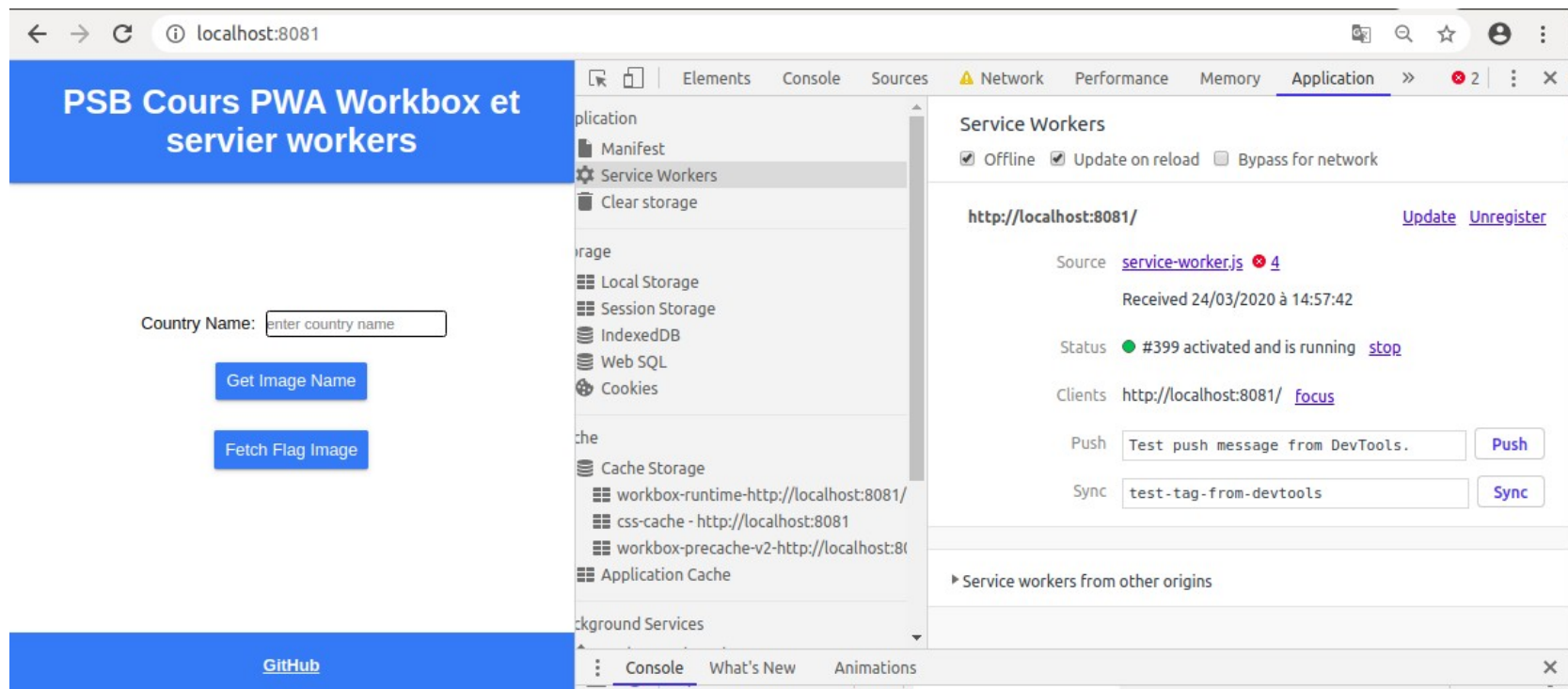
#	Name	Resp...	Conte...	Conte...	Time ...
0	/index.html?__WB_REVISION__=32...	basic	text/...	937	24/03...
1	/js/index.js	basic	applic...	347	24/03...
2	/styles/main.css	basic	text/c...	1,325	24/03...

Select a cache entry above to preview

# From scratch

## Precaching

Et on passe en offline :



# Test de la solution

**Si on veut tester ce type de solution sur un autre poste, plusieurs recommandations sont importantes à suivre dans le cadre de l'utilisation des services workers.**

**On doit être en mode HTTPS donc générer les certificats. De plus dans notre exemple on génère des certificats auto-signés il faudra démarrer votre navigateur avec des options de type `google-chrome --user-data-dir=/tmp/foo --ignore-certificate-errors --unsafely-treat-insecure-origin-as-secure=https://192.168.1.58:8081`**

**Voir exemple de mise en place d'un serveur nodejs en mode https :**

**<https://www.zem.fr/creer-un-serveur-https-nodejs-express/>**

# Autres notions

## Background sync

<https://developers.google.com/web/tools/workbox/modules/workbox-background-sync>

### Background sync

**Dans la situation où lorsqu'on envoie nos données vers le serveur, quelque fois la requête peut échouée suite à des problèmes de connectivité ou des problèmes de serveur. La solution est de réitérer plus tard ...**

**La nouvelle BackgroundSync API est une solution idéale pour ce type de problème. En effet, quand le serveur worker détecte que le réseaux est défectueux, il peut enregistrer et attendre un événement de sync qui lui sera délivré lorsque le navigateur détectera que la connectivité est revenue.**

**On peut noter que l'événement sync peut être délivré même si l'utilisateur a quitté l'application.**

# Autres notions

## Background sync

```
import {BackgroundSyncPlugin} from 'workbox-background-sync';
import {registerRoute} from 'workbox-routing';
import {NetworkOnly} from 'workbox-strategies';

const bgSyncPlugin = new BackgroundSyncPlugin('myQueueName', {
  maxRetentionTime: 24 * 60 // Retry for max of 24 Hours (specified in minutes)
});

registerRoute(
  /\/api\/.*\/*.json/,
  new NetworkOnly({
    plugins: [bgSyncPlugin]
  }),
  'POST'
);
```