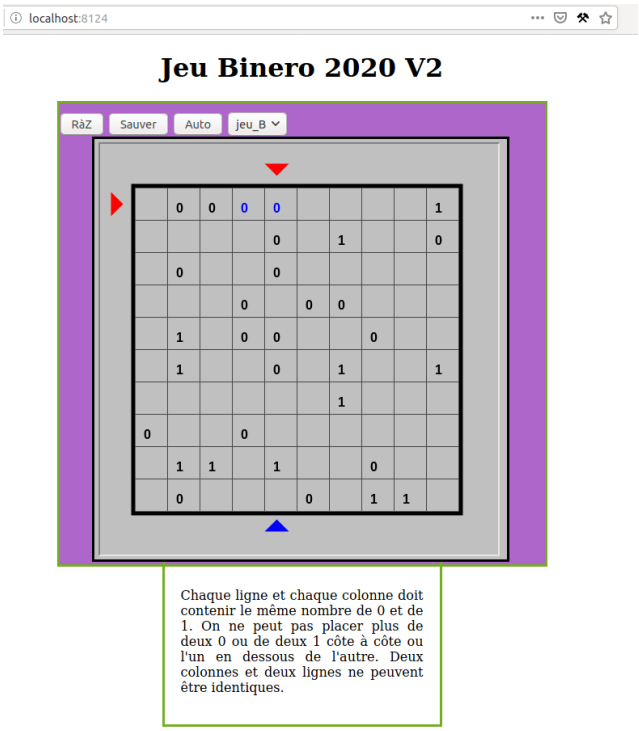
TP de mise en œuvre de PWA sur une application simple
Y. Stroppa
2020
CORRECTION

Exemple de l'application Binero 2020

Voici l'application dans son fonctionnement initial



Objectif de cet exercice est de transformer l'application fournie en application PWA. En résumé que l'application puisse fonctionner en offline et que l'on puisse l'installer sur des médias tel que le téléphone dans l'écran d'accueil.

Veuillez télécharger le projet git à l'aide de la commande
https://github.com/ystroppa/PSB_M2_projet

Commande : git clone https://github.com/ystroppa/PSB_M2_projet

Une fois le projet descendu, veuillez installer les dépendances de Node.js et exécutez l'application afin de vérifier qu'elle fonctionne bien sur votre poste. On cherchera où est paramétré le port d'écoute par défaut configurer dans l'application.

Que faut-il faire pur compléter l'installation et démarre le service

Instructions :
#npm install
#npm start
On pourra rajouter la commande dans app.js
app.get('/favicon.ico', (req, res) => res.status(204)); pour éviter l'erreur de la console

Déterminez dans cette arborescence la partie client et la partie serveur.

Commentaires :
La partie client est sous le répertoire public
Tout le reste est consacré à la partie serveur

A quoi sert le fichier package.json

Commentaires :
Ce fichier sert à décrire et préciser la configuration de l'application node.js er télécharger les dépendances

Il faut passer à l'aide de Chrome ou de Chromium l'outil d'audit afin de regarder la compatibilité de l'application aux technologies PWA et ainsi que d'autres aspects éventuels tels que les performances ...

Utilisation de l'outil Lighthouse intégré par défaut sur ces navigateurs. Examinez le résultat et ensuite procéder à la transformation de l'application et vous pourrez vérifier au fur et à mesure de l'amélioration.

La première étape est de venir insérer le chargement si votre navigateur le supporte d'un service worker ....

A faire dans le main.js

```
// etre sur que le navigateur supporte les services workers
if ('serviceWorker' in navigator) {
   // inscription du service workler
   window.addEventListener('load', () => {
      navigator.serviceWorker
      .register('../sw.js')
      .then (reg => console.log("Service worker : Register"))
      .catch(err=> console.log("Service worker: Error  ${err}"))
   })
 }
```

Une fois la partie complétée veuillez ajouter ou vérifier les autres éléments associés :

Le fichier sw.js

L'inclusion dans index.html

Recharger l'application et rendez vous dans la partie Application de la console pour vérifier la présence du service worker.

Le fichier sw.js est vide pour le moment, et on a bien pensé à ajouter dans le fichier index.html inclusion de main.js

```
<script src="scripts/main.js"></script>
<script src='scripts/config.js'></script>
<script src='scripts/comportement.js'></script>
```

Ensuite on va s'occuper du chargement du cache à partir du fichier sw.js dans lequel il faut déclarer les éléments à mettre en cache et le nom du cache. Ensuite dans la partie associé à l'événement install la copie des éléments en cache. Une fois réaliser veuillez vérifier leur présence.

Eléments/fichiers à mettre en cache :

```
const cacheAssets =[
    "index.html",
    "about.html",
    "/styles/style.css",
    "/scripts/main.js",
    "/scripts/config.js",
    "/scripts/comportement.js",
    "/scripts/jquery-3.4.1.min.js"
];
```

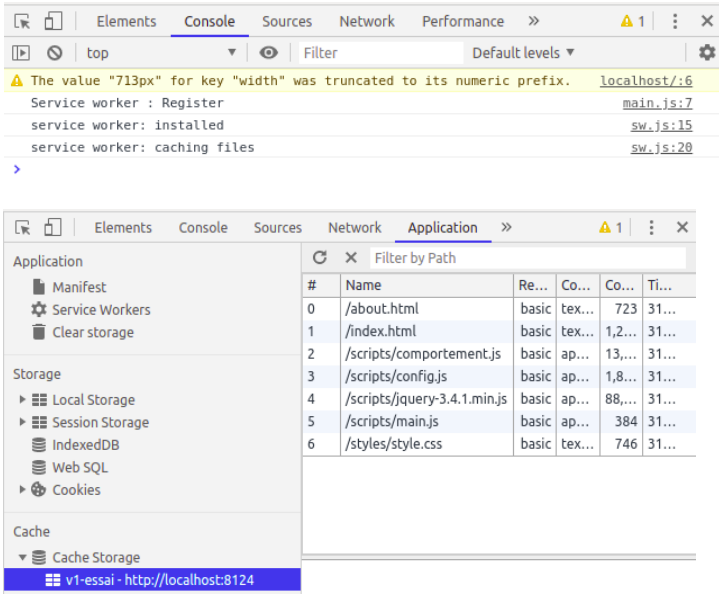
Nom du cache

```
const  cacheName='v1-essai';
```

Bloc javascript événement install :

```
self.addEventListener('install',(e) => {
    console.log('service worker: installed');
    e.waitUntil (
        caches
        .open(cacheName)
        .then(cache => {
            console.log('service worker: caching files');
            cache.addAll(cacheAssets);
        })
        .then(()=> self.skipWaiting())
    )
})
```

Vérification du chargement dans le cache de votre navigateur.



Veuillez changer le nom de votre cache et recharger dans la navigateur, que se passe t-il ?
Veuillez implémenter lors de l'activate un nettoyage du cache ...

Evénement activate :

```
// lors de l'activation du service worker nettoyage des caches
self.addEventListener('activate',(e) => {
    console.log('service worker: activated');
    e.waitUntil(
        caches.keys().then(cacheNames => {
            return Promise.all(
                cacheNames.map(cache => {
                    if (cache !== cacheName) {
                        // si on change le nom du chaque permet de supprimer les anciens
                        console.log('Service Worker : clearing old cache');
                        return caches.delete(cache)
                    }
                })
            );
        })
    );
});
```

Ensuite une fois ces problèmes réglés et vérifiés, on souhaite que notre application puisse fonctionner en mode offline .... que faut-il faire et qu'elles sont les stratégies que l'on peut utiliser ... veuillez en sélectionner une et l'insérer dans votre application ...

Liste des stratégies possibles : cache-only ... stale- ...

Evénement fetch:

```
// call Fetch event
self.addEventListener('fetch',e => {
    console.log('Service Worker: fetching');
    e.respondWith (
        // on va repondre avec les elements du cache
        fetch(e.request).catch(() => caches.match(e.request))
    )
})
```

Veuillez expliquer comment vous avez tester le Offline ...

Commentaires :

Dans la console sur le servie worker, passez le flag à offline et essayer d'accéder à la page.

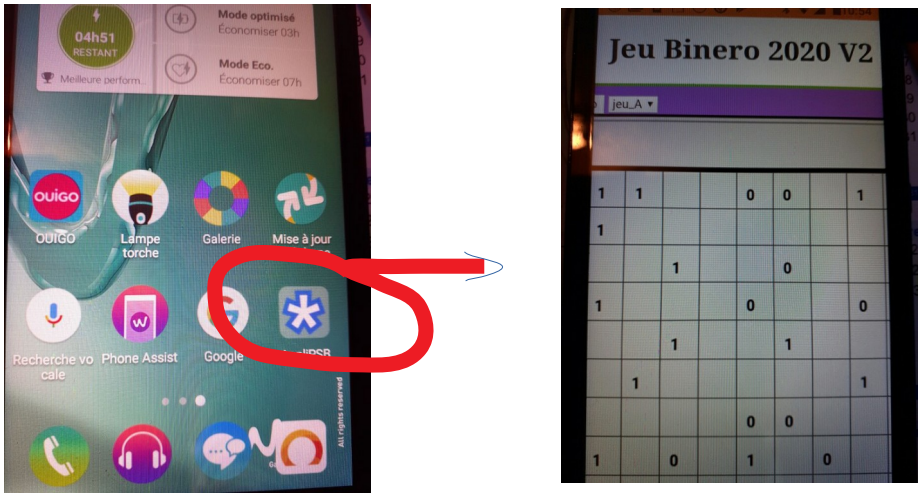
On revérifie la compatibilité de votre application à l'aide de lighthouse de votre navigateur veuillez compléter le fichier index.html des différents meta données nécessaires ....

Ajout des meta-data index.html

```
<meta charset="utf-8">
<meta name="viewport" content="width=713px, minimum-scale=1.0, initial-scale=1.0, user-scalable=yes">
<link rel="stylesheet" type="text/css" href="styles/style.css" />
<link rel="manifest" href="/manifest.json">
<meta name="msapplication-starturl" content="/index.html">
<meta name="mobile-web-app-capable" content="yes">
<meta name="apple-mobile-web-app-capable" content="yes">
<meta name="application-name" content="Space Missions">
<meta name="apple-mobile-web-app-title" content="Space Missions">
<meta name="theme-color" content="#FF9800">
<meta name="msapplication-navbutton-color" content="#FF9800">
<meta name="apple-mobile-web-app-status-bar-style" content="black-translucent">
<link rel="icon" sizes="128x128" href="/images/touch/icon-128x128.png">
<link rel="apple-touch-icon" sizes="128x128" href="/images/touch/icon-128x128.png">
<link rel="icon" sizes="192x192" href="icon-192x192.png">
<link rel="apple-touch-icon" sizes="192x192" href="/images/touch/icon-192x192.png">
<link rel="icon" sizes="256x256" href="/images/touch/icon-256x256.png">
<link rel="apple-touch-icon" sizes="256x256" href="/images/touch/icon-256x256.png">
<link rel="icon" sizes="384x384" href="/images/touch/icon-384x384.png">
<link rel="apple-touch-icon" sizes="384x384" href="/images/touch/icon-384x384.png">
<link rel="icon" sizes="512x512" href="/images/touch/icon-512x512.png">
<link rel="apple-touch-icon" sizes="512x512" href="/images/touch/icon-512x512.png">
```

Ensuite on souhaite avoir la possibilité d'installer l'application sur des médias directement , veuillez définir le contenu du manifest.json et l'insérer dans votre index.html .

Veuillez tester le déploiement sur des machines ... comment se passe l'utilisation ....



Si la version sur la machine de référence évolue que se passe t-il sur le média sur lequel on a installer l'application

Que peut-ton faire ???

Dans le prolongement on souhaite également avertir l'utilisateur et lui envoyer des messages veuillez implémenter la partir push de sw.js pour permettre de type de fonctionnalités :

Pour aller plus loin, il est nécessaire de passer votre service web en ode https pour ce faire il sera nécessaire d'installer quelques éléments indispensables pour générer des certificats de type auto-signé.

```
openssl genrsa -out key.pem
openssl req -new -key key.pem -out csr.pem
openssl x509 -req -days 9999 -in csr.pem -signkey key.pem -out cert.pem

A installer sur le répertoire racine de votre site
```

```
// A modifier votre fichier pour l'adapter au chargement du certificat
const https = require('https');
const fs = require('fs');
const express = require('express');
const app = express();

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

https.createServer({
  key: fs.readFileSync('key.pem'),
  cert: fs.readFileSync('cert.pem')
}, app).listen(8124);
```

Redémarrez l'application en mode https cette fois-ci ... et connectez vous sur l'url
https://localhost:8124
Attention à exécuter votre navigateur avec les options sans contrôles de certificat sinonnça se passe mal ....
#google-chrome --user-data-dir=/tmp/foo --ignore-certificate-errors --unsafely-treat-insecure-origin-as-secure= https://192.168.1.58:8124