Ma première Progressive Web App

> *Une Progressive Web App utilise les possibilités du web moderne pour délivrer une expérience utilisateur similaire à une application native.*
>
> - *Google*

L'application native peut envoyer des notifications, être ouverte en mode hors ligne, se charger depuis l'écran d'accueil… En somme, les PWA contiennent toute une série de technologies, de concepts d'architecture et d'API Web qui travaillent pour proposer une expérience similaire aux applications natives sur le Web mobile.

Une PWA est :

- **Progressive** : Elle doit marcher pour tout les utilisateurs, quelque soit son navigateur.
- **Responsive** : S'adapte à n'importe quel lecteur : mobile, desktop, tablet…
- **Indépendante** : Elle peut marcher avec une connection de base qualité ou même sans.
- **Fraîche** : Toujours à jour !
- **Fiable** : Servie via HTTPS.
- **Identifiable** : Elle doit être identifié par les moteurs de recherche comme une application.
- **Engageante** : Avec des fonctionnalités ressemblantes à celle des applications native. (hors-ligne, push notifications…).
- **Installable** : Les utilisateurs peuvent la garder sur leurs écran d'accueil sans passer par le app store.
- **Partageable** : Peut être envoyer à d'autres utilisateurs grâce à une URL sans nécéssité d'instalattion.

Comment y répondre ?

Nous nous attarderons pas sur les points suivants: Progressive, Responsive, Ressemblante, Fiable, Identifiable. Ceci pourrais faire l'objet de tout autres tutoriels.

Indépendante/Fraîche :

- Grâce… au service worker !
- Oui mais c'est quoi un service worker ?
- Ah oui bonne question !

Le service worker permet à votre PWA de ce lancer sans aucune connexion internet, il mets en cache vos ressources, les mets à jours si nécessaire et permet d'éliminer les dépendances au réseau. Ceci permet à votre PWA de se charger sans connexion et rapidement !

Engageante/Installable :

- Avec un joli fichier json… le Manifest.json !
- Son rôle !?
- J'y viens !

Le Manifest va permettre de dire au navigateur qu'il s'agit bien d'une PWA, il va vous permettre de lui dire de la lire en fullscreen, de mettre cette icone pour l'écran d'accueil, son thème (couleur) lorsque la PWA se lance…

- *Manifest* : Yop ! Je suis une PWA !
- *Navigateur* : Ok c'est cool mais je te lis comment ?
- *Manifest* : En fullscreen, au chargement tu me mets un fond rouge, et n'oublie pas de demander à l'utilisateur s'il veut me mettre en raccourci sur son écran, s'il veut, tu lui mets ce jolie logo sur son écran d'accueil !
- *Navigateur* : Ok ! Roulez jeunesse !

Are you ready ?

L'environnement de travail :

- Google Chrome 52 ou une version supérieure, ou Chronium.
- Lighthouse: ceci va nous permettre de tester notre PWA, tout au long de sa création.
- Récupérer ceci PWA tuto.

```
git clone git@github.com:jordanlefort/PWA-tuto.git
cd PWA-tuto
npm install
node server.js
```

Normalement, vous pouvez maintenant vous rendre sur localhost:3000.

Vous devriez avoir ceci :

premiére installation

Testons maintenant notre PWA avec Lightouse !

premier test Lightouse

Pas de panique ! Notre bout de code n'est à l'heure actuelle aucunement une PWA, nous verrons un peu plus en détails les demandes de Lighthouse plus tard !

Le services worker :

C'est partie maintenant que tout est bien installé, c'est à vous de jouer !

Avant de faire appel au services worker nous allons devoir dire à notre navigateur, enfin s'il est compatible, d'enregistrer notre services worker. Commençons par le commencement.. demandons à notre navigateur s'il prend en charge les services worker ?

```
// ./index.html

if ('serviceWorker' in navigator) {
  //Si le navigateur prend en charger le service worker !
}
```

Maintenant on peut lui dire où le trouver, pour qu'il puisse l'enregistrer !

```
// ./index.html

if ('serviceWorker' in navigator) {
  // J'enregistre mon service worker sw.js
  // avec comme scope '/' (racine);
  navigator.serviceWorker.register('sw.js', { scope: '/'})
  // Si c'est good..
  .then(function(reg){
    console.log('Registration succeeded. Scope is' + reg.scope);
  })
  // Si c'est bad..
  .catch(function(error){
    console.log('Registration failed with' + error);
  });
};
```

Vous devriez voir votre service worker dans la devTools !

Service worker on devTools

Ok ! Maintenant qu'il l'a enregistré, ce serait bien que notre fichier sw.js fasse quelque chose ! Maintenant lors de la première visite de l'utilisateur sur notre PWA, le service worker va récuperer l'objet cache pour le remplir avec les différents éléments nécessaires au bon fonctionnement de notre PWA.

```
// ./sw.js
var cacheName = 'pwa'; //le nom de mon cache !

// J'assigne un écouteur 'install' à mon service worker
self.addEventListener('install', function(e){
  // Cool il est installé !
  console.log('[ServiceWorker] Install');
  e.waitUntil(
    // Je récupére mon cache du nom de pwa
    caches.open(cacheName).then(function(cache) {
      // Je fais là mise en cache de ma PWA
      console.log('[ServiceWorker] Caching app shell');
      return cache.addAll(filesToCache);
      // filesToCache : tableaux contenant les différents path v
    })
  );
});
```

Bon c'est cool, mais s'il ne stocke rien, ça ne sert à rien ! On va donc stocker tout les fichiers nécessaires au fonctionnement de notre PWA en indiquant les différents path dans notre tableau filesToCache.

```
// ./sw.js

var filesToCache = [
 '/',
 'index.html',
 'css/main.css',
 'main.58fec847.js',
 'Semantic-UI-CSS-master/semantic.min.css',
 'Semantic-UI-CSS-master/themes/default/assets/fonts/icons.eot',
 'Semantic-UI-CSS-master/themes/default/assets/fonts/icons.otf',
 'Semantic-UI-CSS-master/themes/default/assets/fonts/icons.svg',
 'Semantic-UI-CSS-master/themes/default/assets/fonts/icons.ttf',
 'Semantic-UI-CSS-master/themes/default/assets/fonts/icons.woff'
 'Semantic-UI-CSS-master/themes/default/assets/fonts/icons.woff2
]
```

Et on obtient donc ceci ! Je vous invite à vous diriger dans votre cache !

cache devTools

Le service worker peuvent intercepter les demandes faites de notre PWA et les traiter. Cela signifie que nous pouvons déterminer comment nous voulons traiter la requête et potentiellement se servir de notre cache en réponse.

```
// ./sw.js

// J'assigne un écouteur 'fetch' à mon service worker
self.addEventListener('fetch', function(e) {
  // J'intercepte bien une requéte x ou y.
  console.log('[ServiceWorker] Fetch', e.request.url);
  e.respondWith(
    // Jy réponds soit avec une ressource trouvé dans mon cache
    caches.match(e.request).then(function(response) {
      return response || fetch(e.request);
    })
  );
});
```

Alons voir ce qui ce passe du coté de Ligthouse !

Ligthouse with offline

Maintenant que Lightouse trouve notre service worker et réussi simuler la page en offline c'est déjà mieux ! Vous pouvez vous diriger vers votre devTools à l'onglet Network pour simuler votre PWA en offline.

offline simulation

Le Manifest.json

Comme expliqué plus tôt le manifest.json permet de dire au navigateur qu'il s'agit d'une PWA, et de lui expliquer comment là lire !

```
// ./manifest.json
{
  "name": "TodoList", // le nom de votre PWA
  "short_name": "TodoList", // si le nom de votre PWA est trop l
  "icons": [{
    "src": "icon.png",
      "sizes": "128x128",
      "type": "image/png"
    }, {
      "src": "icon.png",
      "sizes": "144x144",
      "type": "image/png"
    }, {
      "src": "icon.png",
      "sizes": "152x152",
      "type": "image/png"
    }, {
      "src": "icon.png",
      "sizes": "192x192",
      "type": "image/png"
    }, {
      "src": "icon.png",
      "sizes": "256x256",
      "type": "image/png"
    }],  // les icons (ecrans d'accueil) !
  "start_url": "index.html", // la page de démarage par default
  "display": "standalone", // la façon de lire votre PWA
  "background_color": "#21BA45", // couleur de fond
  "theme_color": "#21BA45" // couleur du theme
}
```

Maintenant que nous avons crée notre manifest.json nous allons le lier à notre PWA, je vous invite donc à vous rendre sur votre index.html et à rajouter le link juste en dessous dans vos balise head

```
<!-- ./index.html -->
<link rel="manifest" href="manifest.json">
```

Cool on y est presque ! Rendez vous dans votre devTools et dans Application, vous devriez retrouver notre manifest !

manifest devTools

Un petit tour sur Ligthouse voir si il y a du mieux !

Lightouse with manifest

Amélioration !

Si vous avez fouinez un petit peu dans Ligthouse, on peut constater que certaines améliorations sont possibles ! Comme l'ajout de ces méta dans vos balise head :

```
<!-- ./index.html -->
<meta name="viewport" content="width=device-width, initial-scale
<meta name="theme-color" content="#21BA45">
```

Le chargement de vos script de manière asynchrone est également recommandée, de vos fichiers css aussi ! Pour les scripts rien de plus simple, il vous suffit dans vos balise script de rajouter ceci “async”.

```
<!-- ./index.html -->
<script src='..' async></script>
```

Pour les fichier css, je vous renvoie vers LoadCSS un script qui sert à charger vos fichier css de manière asynchrone ! L'intérieur de nos balise head devrais ressembler à ceci :

```
<!-- ./index.html -->
<head>
    <meta charset="UTF-8"/>
    <meta name="viewport" content="width=device-width, initial-s
    <meta name="viewport" content="initial-scale=1.0, user-scala
    <meta name="theme-color" content="#21BA45"/>
    <title>PWA todolist</title>
    <link rel="manifest" href="manifest.json"/>
    <script async>function loadCSS(e,t,n){"use strict";function
    <script async>
      loadCSS( "Semantic-UI-CSS-master/semantic.min.css" );
      loadCSS( "css/main.css");
    </script>
</head>
```

Pour plus de détails Lighthouse vous renvoit vers des tutoriels sur les points à corriger vis à vis de votre PWA, si vous voulez tester sur vos téléphone cette PWA (todoList) je vous invite à aller ici : todoList

Avec ce tutoriel, un peu de travail et un serveur en HTTPS, Lighthouse vous rendra ceci :

Ligthouse 100/100

Bon courage !

Written on January 23, 2017

1 Commentaire blog.simplon-occ 1 S'identifier

♡ Recommander Partager Les meilleurs

Participer à la discussion…

S'IDENTIFIER AVEC OU INSCRIVEZ-VOUS SUR DISQUS ?

Nom

Thomas Ruffier • il y a un an

Merci beaucoup, j'ai visité de nombreux sites et celui-ci est le plus clair !

Répondre • Partager ›