Partie 2: Introduction aux PWA et Single Page App

Dans cette deuxième partie, nous allons rentrer dans le vif du sujet. Après avoir vu quelques bases de JS qui nous aideront à coder une PWA et que vous pourrez exercer tout au long du cours, nous allons-nous concentrer sur ce qu'est une PWA et comment nous allons en coder une dit en "Single Page App".

1. Qu'est-ce qu'une PWA (théoriquement parlant)
2. Le challenge à réaliser
3. Single Page app

> PWA: **Websites that took all the right vitamins**

Vous avez dit PWA ?

Une PWA (progessive web app) est un site web répondant à différents critères et utilisant différentes web APIs pour le rendre utilisable comme une application mobile et desktop native.

> Progressive Web Apps (PWA) are built to reach **anyone, anywhere, on any device with a single codebase**. (source)

Ressources

Google training PWA

Introduction aux web APIs / MDN

40 exemples de PWAs

Le challenge

Pour ce cours, nous allons réaliser ensemble une PWA. Le challenge est d'arriver en fin de cours avec le même résultat que vous pouvez voir ici même. Nous allons donc créer une cinémathèque. Cette web application nous donnera la possibilité de:

- Ajouter un film => une photo (l'affiche par exemple), une date (date de visionnage par exemple), un nombre d'étoiles (pour lui donner une note par exemple)
- Supprimer un film
- Trier les films => par date, par étoile, par dernier ajout

La PWA que vous réussirez à développer sera fonctionnelle et optimisée, elle pourra donc être déployée (sur Github Pages par exemple) et sera donc installable en tant qu'application sur mobile et desktop.

Joli non ⁉

Single Page App

Bienvenue à notre premier point théorique postbases de JS.
Nous allons voir ce qu'est une Single Page App.

L'idée principale d'une SPA et d'enrichir ou de supprimer les éléments du DOM de manière dynamique à l'interne d'une seule page chargée depuis le serveur. En soi, l'utilisateur fait une requête au serveur, le serveur lui renvoie l'application (ou le site) et, à partir de ce moment, aucune autre requête ne sera faite pour appeler **le design / squelette** de l'app. Il restera toujours le fait de faire appel au serveur de base de données pour appeler **les données / le contenu** de l'application, mais tout le contenant sera déjà chargé et se modifiera dynamiquement sous l'action de l'utilisateur.

Comment ça marche ?

```
<!DOCTYPE html>
<style>
    .hidden {
        display: none;
    }
</style>
<body>
    <button id="toogle">toogle</button>
    <section id="section1">
        <p>Je suis visible dès le début</p>
    </section>
    <section id="section2" class="hidden">
        <p>Je suis caché au début du chargement</p>
    </section>
</body>
<script>
    document.querySelector("#toogle").addEventListener("click", (evt) => {
        document.querySelector("#section2").classList.toggle("hidden");
        document.querySelector("#section1").classList.toggle("hidden");
    });
</script>
</html>
```

Voilà un petit exemple de comment on load tout le squelette de l'app est on le fait voir ou pas à l'utilisateur quand il fait une action ou tout autres type d'événements.

Il reste un petit bémol aux SPA. Comment faire pour revenir en arrière s'il n'y a qu'une seule page ? Les flèches du navigateur ne feront que vous ramener à la *page* précédente et non pas à l'*état de la page*.

Nous allons donc voir dans la suite de ce cours notre toute première Web APi: celle qui nous permettra de revenir en arrière même avec une Single Page App.