L'Institut Supérieur des Technologies de l'Information & de la Communication

Technologies & programmation web

Chapitre 5

Progressive Web application (PWA)

Introduction

- Le terme **Progressive Web Application - PWA** est né en 2015 de l'association des deux termes « **web app** » (application web) et « **progressive enhancement** » (principe de l'amélioration progressive)

- Ce n'est pas une technologie comme pourrait l'être un framework ou une librairie JavaScript

- C'est avant tout un terme qui met l'accent sur le principe de l'amélioration progressive
- Aussi, c'est une manière de penser les applications web qui est mise en avant

Avantages des PWA

- **Installation sur le périphérique**: PWA est utilisable sur différents périphériques (ordinateur, tablette et smartphone)
- C'est une application web **responsive** qui s'adapte aux différentes résolutions des supports sur lesquels elle peut être installée
- Accessibles hors ligne: Une PWA est utilisable hors ligne grâce à la technologie de « services worker » qui permet de sauvegarder les fichiers au niveau du navigateur
- Mise à jour automatique: PWA comporte un contenu mis à jour régulièrement grâce au processus de mise à jour du service worker
- SÉCURITÉ: PWA est sécurisée puisqu'elle est accessible uniquement en HTTPS grâce au certificat SSL

Stockage sur client

Principe :

-L'API WebStorage fournit un mécanisme de stockage de l'information côté client plus évolué que les simples cookies :

- La manipulation des données stockées est plus aisée
- La taille maximale des informations stockables est plus grande
- Les objets stockés dans le WebStorage sont simplement des paires clef/valeur, comme les cookies mais les données peuvent être plus structurées, puisqu'il s'agit d'objets JavaScript

- Les données stockées ne sont pas cryptées

Stockage sur client

Les deux principaux mécanismes internes du Stockage Web sont :

- **sessionStorage** qui maintient un espace de stockage, séparé pour chaque origine différente, disponible le temps de la session de navigation ; autrement dit, les données sont conservées jusqu'au moment où la dernière fenêtre ou le dernier onglet ouvert du navigateur est fermé, sessionStorage est réinitialisé à chaque redémarrage du navigateur.
- **localStorage** permet de conserver les données d'une session de navigation à une autre. Cela permet par exemple de conserver sur la machine du client les paramètres de personnalisation d'un site Web.

Stockage sur client :

Enregistrer une valeur dans le stockage

- **Storage.itemKey = itemValue**

```
localStorage.prenom = "Foulen";
localStorage.nom = "Ben Foulen";
```

- **Storage.setItem()** : utilisée pour la création d'une donnée, que pour la modification d'une donnée existante (si cette donnée existe déja). Elle prend deux arguments — la clé de l'élément à créer/modifier, et la valeur associée à stocker.

```
var myObj = {};
myObj.prenom = "Foulen";
myObj.nom = "Ben Foulen";
var chaineJSON = JSON.stringify(myObj)
localStorage.setItem('etudiant', chaineJSON);
```

Stockage sur client :

Récupérer des données du stockage

- **Storage.itemKey;**

```
var prenom = localStorage.prenom;
alert(prenom);
```

- **Storage.getItem()** : prend un seul argument, la clé de l'élément que vous souhaitez récupérer de l'objet de stockage pour le domaine.

```
var student = JSON.parse(localStorage.getItem("etudiant"));
alert(student.nom);
```

Stockage sur client :

Supprimer des données du stockage

l'API de Stockage Web fournit aussi un couple de méthodes simples pour supprimer des données :

- **Storage.removeItem()** : prend un seul argument, la clé de l'élément que vous souhaitez supprimer, et le supprime de l'objet de stockage pour le domaine.

- **Storage.clear()** : ne prend pas d'argument, et vide l'ensemble des données de l'objet de stockage pour le domaine.

Stockage sur client :

Application

On suppose qu'on va donner la possibilité à l'utilisateur de choisir sa couleur de fond d'une page Web.
On va enregistrer le choix fait par l'utilisateur en utilisant **localStorage** afin que celui-ci soit conservé pour ses prochaines visites.
Côté HTML, on va utiliser un élément de formulaire pour permettre à l'utilisateur de taper sa couleur préférée.

```
<form>
<div>
    <label for="bgtheme">Choisissez un thème (hexa)</label>
    <input class="text" id="bgtheme" value="FAFAFA"
    pattern='[a-fA-F0-9]{6}'><br><br>
</div>
</form>
```

Question:

Donner le code JS nécessaire pour **sauvegarder** la couleur saisie par l'utilisateur dans la base de données du navigateur **localStorage.**
Vérifier si la donnée a bien été sauvegarder en utilisant l'Outils de développement de Google Chrome.

Stockage sur client :

Application

```
let bgColor = document.getElementById('bgtheme');

if(localStorage.getItem('bgtheme')){
    updateBg();
}else{
    setBg();
}
function updateBg(){
    let bg = localStorage.getItem('bgtheme');
    // ou let bg = localStorage.bgtheme;
    document.body.style.background = '#'+ bg;
}
function setBg(){
    localStorage.setItem('bgtheme', bgColor.value);
    // ou localStorage.bgtheme = bgColor.value;
    updateBg()
}
bgColor.addEventListener('change', setBg);
```

Web Manifest

C’est quoi ?

Le manifeste d'une application web :

- est un fichier JSON

-fournit des informations sur l’application web : son nom, son auteur, une icône, une description

Pourquoi il est utilisé?

Le but du manifeste est d'installer des applications web progressives sur le bureau ou l'écran d'accueil d'un appareil mobile pour offrire aux utilisateurs un accès plus rapide.

Web Manifest

Déploiement

Les manifestes des applications PWA sont déployés dans des pages HTML en utilisant une balise lien <link> dans l'entête <head> de la page HTML

Le nom du fichier doit être : manifest.webmanifest ou manifest.json

Exemple

```
<link rel="manifest" href="/manifest.webmanifest">
```

Web Manifest

Exemple d'un fichier Web Manifest

```
{
    "name": "Offline SODOKU",
    "short_name": "OfflineSodoku",
    "scope": "./",
    "icons": [{
            "src": "img/sodoku144.png",
            "sizes": "144x144",
            "type": "image/png"
    },
        {
            "src": "img/sodoku192.png",
            "sizes": "192x192",
            "type": "image/png"
    },
        {
            "src": "img/sodoku512.png",
            "sizes": "512x512",
            "type": "image/png"

    }],
    "background_color": "#3367D6",
    "theme_color": "#3367D6",
    "display": "standalone",
    "start_url": "./"
}
```

Web Manifest

Propriétés du Web Manifest

- **name**: affiché sur l'écran de démarrage de l'application
- **short_name**: affiché en dessous du raccourci sur le bureau ou l'écran d'accueil
- **description**: une description générale de l'application
- **start_url**: l'URL qui est chargée en premier quand on ouvre l'application depuis son raccourci sur le bureau ou l'écran d'accueil
- **background_color**: La couleur d'arrière-plan de l'écran de démarrage de l'application
- **theme_color**: la couleur de thème général de l'application, utilisée notamment dans les barres de statut si elles sont affichées

Web Manifest

Propriétés du Web Manifest

- **display**: spécifie le mode d'affichage. Voici les différents modes disponibles triés par ordre de fallback :
  - **fullscreen**: toute la zone d'affichage disponible est utilisée et aucun agent utilisateur n'est montré.
  - **standalone**: comportement similaire a une application native. Cela peut signifier que l'application a sa propre fenêtre, sa propre icône dans le lanceur d'applications, etc. Dans ce mode, l'agent utilisateur va exclure les éléments d'interface qui permettent de contrôler la navigation mais peut inclure d'autres éléments comme une barre de statut par exemple.
  - **minimal-ui**: l'application va ressembler et se comporter comme une application autonome, mais elle aura quelques élements d'interface permettant de contrôler la navigation. Les éléments varient en fonction du navigateur et du système.
  - **browser** (par défaut): l'application s'ouvre dans un nouvel onglet ou une nouvelle fenêtre du navigateur, en fonction du navigateur et de la plateforme
- **icons**: liste d'icônes de l'application de différentes résolutions, utilisées pour le raccourci et l'écran de démarrage. L'appareil choisira la meilleure icône automatiquement selon les cas.

Web Manifest

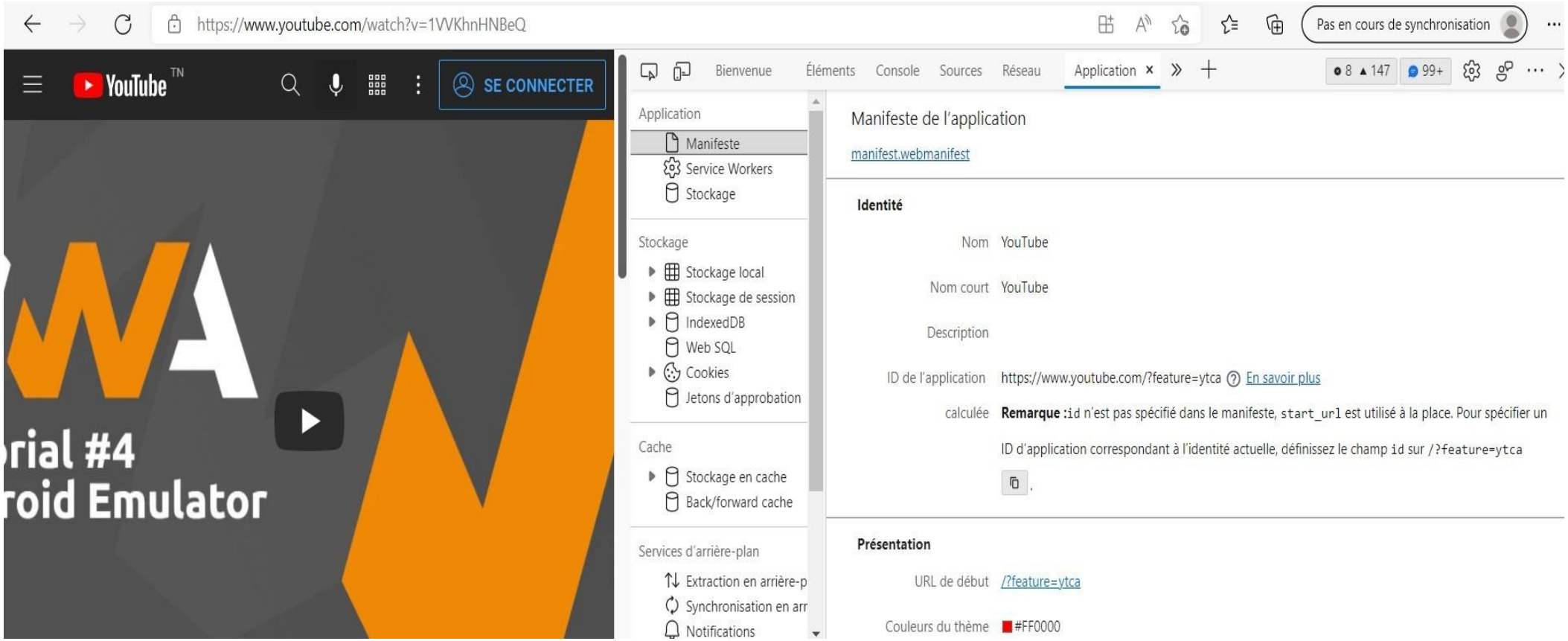
Test de fonctionnement

Il est possible de vérifier la prise en compte du manifeste en regardant dans l'onglet ***Applications*** des **Developer Tools** du navigateur.

Si l’application est basée sur l’utilisation du Web Manifest, la liste des propriétés du manifeste est alors affichée.

Web Manifest

Test de fonctionnement



Web Manifest

Application :

1- Créer un dossier ApplicationPWA contenant:

- une page d'accueil index.html
- un fichier manifest.json avec les propriétés de votre choix (nom, surnom, description, affichage, couleur d'arrière-plan et couleur de thème, URL de démarrage, icônes, etc.)
- un dossier img contenant les sources des icones à utiliser dans le fichier manifeste

2- Relier le manifeste à la page index.html

3- Déployer l'application sur votre serveur web local (par exemple http-server)

4- Utiliser votre navigateur web pour visualiser l'application et l'installer sur votre machine