— ReadMe —

Mathieu Gauthier & Etienne Jézéquel

———

Projet HTML5 - SnapPhoto

———

Introduction :

L'objectif de ce projet est de développer une application web de prises de photo via les API HTML 5. L'utilisateur doit pouvoir prendre une photo grâce à sa webcam. Il est possible de récupérer la position de l'utilisateur et la date lors de la prise de cette photo. Enfin, on peut sauvegarder cette dernière localement et restaurer l'état de l'application pour prendre une nouvelle photo.

Afin de rendre l'application la plus agréable à l'utilisation, nous avons mis en place certaines fonctionnalités provenant directement des API HTML 5. Il est fortement conseillé d'utiliser l'application sur des navigateurs modernes (dernières versions de Chrome et Firefox). Toutefois, pour des raisons de performance mais aussi d'expérience utilisateur, nous vous recommandons d'utiliser **Google Chrome**.

L'application est accessible via l'URL suivante : https://snapphoto-322a3.firebaseapp.com/

Afin de l'utiliser en local, il est indispensable d'intégrer un serveur Apache (Wampserver par exemple) ou NodeJS. Pour NodeJS (v6++), il suffit de lancer la commande *node server.js.*

Fonctionnalités utilisées :

1 - Appareil photo (viseur)

L'appareil photo est géré via une balise vidéo prenant la webcam comme flux vidéo. Nous utilisons les méthodes provenant de l'API Canvas afin de dessiner la photo prise à l'instant t. Cette photo est stockée dans un canvas.

2 - Notification

En utilisant l'API Notification HTML 5, nous avons intégré un système de notifications lorsqu'une photo est prise par l'utilisateur. Ces notifications renvoient l'information suivante : « Une nouvelle photo a été ajoutée à la Google Map ».

3 - Géolocalisation

La géolocalisation est demandée lorsqu'une photo est prise par l'utilisateur. Elle est utilisée afin de stocker les coordonnées de l'utilisateur dans la base de données locale (**IndexedDB**).

4 – Google Map API

Nous avons intégré l'API de google afin de traiter les données de géolocalisation.
Nous pouvons ainsi voire sur une carte où nous avons capturé nos photos.

5 - Filtres

Une fois que la photo est capturée, nous avons accès à un outil d'édition.
Nous pouvons modifier certains paramètres qui permettent d'appliquer des filtres à la photo.
(Exemple : sépia, noir et blanc, gérer la lumière ou la saturation)

6 - Download

En appuyant sur le bouton de téléchargement, nous avons la possibilité de sauvegarder sur notre périphérique (Ordinateur, téléphone, tablette, etc..) la photo capturé.
Si des filtres ont été appliqués, ceux-ci seront sauvegardés également.
De plus, si vous avez utilisé l'outil de dessin, les traits seront sauvegardés sur l'image.

7 - Dessin

En utilisant l'API dédiée au Canvas, nous avons pu intégrer un système permettant de dessiner sur la photo prise par l'utilisateur. Il suffit tout simplement de maintenir le clic gauche de la souris sur la photo. Chaque trait peuvent être sauvegardé localement lors de l'enregistrement de la photo.

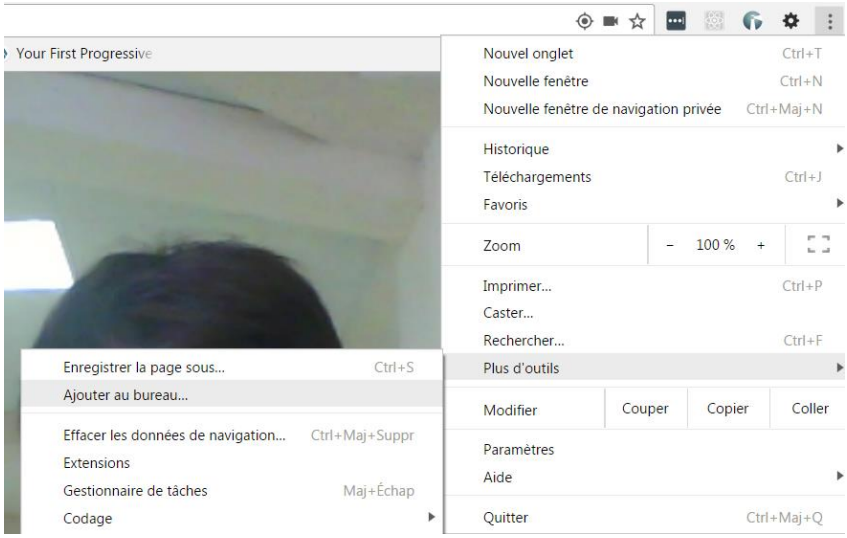
9 - IndexedDB

Via la librairie DexieJS, nous avons pu intégrer IndexedDB permettant de stocker les informations de géolocalisation de l'utilisateur, la photo (encodée en base 64) et la date de la prise de la photo. Toutes ces informations sont ensuite mises en cache et intégrées en tant que marqueurs sur la Google Maps. Si on clique sur l'un de ces marqueurs, on accède aux informations de la photo.

10 - Progressive Web App

Le principe de **Progressive Web App (PWA)** est utilisé pour cette application. Les PWAs prennent les avantages des nouvelles technologies web utilisées pour les sites mobiles et ceux de l'expérience utilisateur pour les applications natives. Elles sont sécurisées, rapides et performantes même en hors-ligne. En d'autres termes, elles sont capables de générer une vue selon l'état du réseau de l'utilisateur (d'où la provenance du terme Progressive).

Pour cela, les PWA exploitent plusieurs technologies clés comme **les services workers**. Un service worker (SW) est un script qui est lancé dans le navigateur et qui permet de gérer le cache de l'application de manière à supporter les expériences hors-ligne. Nous utilisons aussi un *manifest.json* permettant de gérer la vue de la Web App sous la forme d'une application native (Splash screen, android et IOS icônes…). Pour utiliser l'application comme une PWA, il suffit d'utiliser Chrome et d'ajouter l'application sur le bureau (Voir capture d'écran ci-dessous). Cela fonctionne de la même manière sur un mobile.



Plus d'outils -> Ajouter au bureau…