HACKATHON ‘PLAYER AUDIO’

VEILLE TECHNOLOGIQUE

Manuel Durand

La réalisation de ce lecteur audio s’est faite de A à Z, conception graphique, storyboard et codage HTML, CSS et JavaScript.

Pour la conception, la demande étant de faire un lecteur original, je me suis appuyé sur le design d’un site qui présente une entreprise du secteur numérique éco-responsable, au design minimaliste : www.iocean.fr

Pour la partie codage en JavaScript, plusieurs concepts me manquaient : connaissant la balise HTML <audio>, je ne savais pas comment charger plusieurs morceaux de musique, je me doutais cependant qu’il n’était pas nécessaire de créer plusieurs balises <audio> mais plutôt d’utiliser un tableau d’objets afin de pouvoir les appeler et les lire. Ensuite je ne savais pas comment reconstruire le style d’un bouton de type « range », qui permet par exemple de changer le volume de sortie audio.

Une recherche internet «coder un player audio avec JavaScript» m’a permis de découvrir un tutoriel simple que j’ai pu ensuite adapter à mon modèle.

Sources principales : - https://www.youtube.com/watch?v=BzB0Jxu_xJQ
Auteur : la minute de Code.

- mdm webdocs : API HTMLMediaElement

Etape 1 : en HTML créer « l’interface » contenant la balise <audio> et les contrôleurs (boutons) permettant de commander la lecture, la balise <audio> sera ensuite cachée (display : none).

Etape 2 : JavaScript :

Créer un tableau d’objets contenant les morceaux à lire.
Chaque objet contient les propriétés src, (source),et nom.
Grâce à l’API HTMLMediaElement, le lecteur audio peut être contrôlé avec JavaScript.
Pour cela, nous disposons de deux méthodes :

- « play() » lance la lecture de la source audio
- « pause() » met la lecture en pause.

NB : Il n’y a pas de méthode stop, pour faire cela nous devons donc utiliser la méthode pause et remettre le « compteur » de lecture à zéro grâce à la propriété « currentTime ».

Etape 3 : Indiquer la source à lire.

Puisque nous avons créé un tableau, il faut sélectionner la source à lire. Nous commençons donc par un index i à 0 en spécifiant la source avec le code audio.src = tableau[i].src. Le premier morceau du tableau est sélectionné.
L’index nous permet donc de commander la navigation dans le tableau et ainsi de choisir différentes sources audio. Pour cela il faut mettre en place l’écoute d’un événement ‘clic’ sur un bouton « suivant » ou « retour » par exemple et incrémenter ou décrémenter l’index pour charger le morceau voulu.

Important à mettre en place pour la navigation dans le tableau : la condition surveillant la longueur du tableau pour remettre l'index à zéro s'il dépasse une des bornes.

Concernant les propriétés dont j'ai eu besoin et que j'ai utilisées dans mon programme :

- **currentTime** : spécifie le temps actuel de la lecture du média en secondes

- **duration** : permet de déterminer la durée en secondes du média, et ainsi par exemple de calculer et afficher le temps passé et restant

- **ended** : permet de savoir si la lecture est terminée, et ainsi de passer par exemple au morceau suivant dans le tableau.

- **volume :** permet de paramétrer le volume de lecture, couplé à un événement sur l'entrée type « range » en HTML.

Pour les besoins de mon programme, j'ai rajouté des propriétés dans mon tableau d'objets, notamment le nom d'artiste et surtout la durée, (propriété « duration ») qui me permet de calculer et afficher le temps passé et restant.

Pour les besoins de ce programme j'ai dû aussi rechercher une méthode pour changer le style des « input type range ».
Pour cela le site www.cssportal.com m'a été utile :

J'ai utilisé la règle CSS -webkit-appearance : none ; cela permet dans un premier temps de ne pas afficher la version du navigateur.
Ensuite il y a différentes propriétés en fonction des navigateurs.
J'ai utilisé -moz-range-track et -moz-range-thumb pour avoir une compatibilité avec Firefox.
Pour connaître les différentes propriétés à utiliser en fonction des navigateurs, j'ai trouvé les informations complémentaires sur le site https://nikitahl.com/style-range-input-css :

En anglais : la partie de ce type d'entrée se décompose en piste et « pouce » (le « bouton »)
En utilisant ces 3 propriétés on on sûr d'avoir la compatibilité avec tous les navigateurs

The range input widget consists of two parts the **thumb** and the **track**. Each one of these parts has its own pseudo-class selector for styling with a vendor suffix for cross-browser support.

Thumb:

```
input[type=”range”]::-webkit-slider-thumb
input[type=”range”]::-moz-range-thumb
input[type=”range”]::-ms-thumb
```

Track:

```
input[type=”range”]::-webkit-slider-runnable-track
input[type=”range”]::-moz-range-track
input[type=”range”]::-ms-track
```