LES BALISES AUDIO ET VIDEO DU HTML5

Le HTML5 a apporté un ensemble de nouveautés relatives à la présentation des contenus multimédias.

Les balises dédiées à ces usages sont :

1. <audio> : pour les contenus audio uniquement
2. <video> : pour les contenus audio et vidéo

Les deux éléments sont de type block et sont écrits de la manière suivante :

```
<audio>
< !- - informations sur la piste audio- ->
</audio>
```

```
<video>
< !- - informations sur la vidéo- ->
</video>
```

I. LES ATTRIBUTS INPORTANTS DES BALISES AUDIO ET VIDEO

La source : L'attribut src

Savoir afficher une balise audio/vidéo c'est bien, mais si on ne lui donne rien à afficher, on n'est pas plus avancé ! Il va donc falloir donner une source à afficher. Tout comme pour une image, elle peut être relative ou absolue. Il existe deux moyens pour la spécifier.

Via l'attribut src

Il faut spécifier l'attribut src pour donner un lien vers la vidéo ou le flux audio à lire.

```
<video src="http://masource.com/lavideo.avi">
</video>
```

Via la balise <source>

Il peut être intéressant de proposer plusieurs formats à l'utilisateur. En effet, tous les navigateurs ne savent pas lire tous les formats vidéo. On propose donc la même vidéo dans des formats différents et le navigateur choisira ! Pour cela, on utilise la balise <source> *dans* la balise audio/vidéo.

```
<video>
   <source src="chemin/vers/masource.mp4" type="video/mp4">
   <source src="chemin/vers/masource.ogg" type="video/ogg">
   <source src="chemin/vers/masource.webm" type="video/webm">
</video>
```

Les formats audio

Pour diffuser de la musique ou n'importe quel son, il existe de nombreux formats. La plupart d'entre eux sont compressés (comme le sont les images JPEG, PNG et GIF) ce qui permet de réduire leur poids :

- **MP3** : vous ne pouvez *pas* ne pas en avoir entendu parler ! C'est l'un des plus vieux, mais aussi l'un des plus compatibles (tous les appareils savent lire des MP3), ce qui fait qu'il est toujours très utilisé aujourd'hui.
- **AAC** : utilisé majoritairement par Apple sur iTunes, c'est un format de bonne qualité. Les iPod, iPhone et autres iPad savent les lire sans problème.
- **OGG** : le format Ogg Vorbis est très répandu dans le monde du logiciel libre, notamment sous Linux. Ce format a l'avantage d'être libre, c'est-à-dire qu'il n'est protégé par aucun brevet.
- **WAV (format non compressé)** : évitez autant que possible de l'utiliser car le fichier est très volumineux avec ce format. C'est un peu l'équivalent du Bitmap (BMP) pour l'audio.

La compatibilité dépend des navigateurs.

Les formats vidéo

Le stockage de la vidéo est autrement plus complexe. On a besoin de trois éléments :

- **Un format conteneur** : c'est un peu comme une boîte qui va servir à contenir les deux éléments ci-dessous. On reconnaît en général le type de conteneur à l'extension du fichier : AVI, MP4, MKV…
- **Un codec audio** : c'est le format du son de la vidéo, généralement compressé. Nous venons de les voir, on utilise les mêmes : MP3, AAC, OGG…
- **Un codec vidéo** : c'est le format qui va compresser les images. C'est là que les choses se corsent, car ces formats sont complexes et on ne peut pas toujours les utiliser gratuitement. Les principaux à connaître pour le Web sont :

- **H.264** : l'un des plus puissants et des plus utilisés aujourd'hui… mais il n'est pas 100% gratuit. En fait, on peut l'utiliser gratuitement dans certains cas (comme la diffusion de vidéos sur un site web personnel), mais il y a un flou juridique qui fait qu'il est risqué de l'utiliser à tout va.
- **Ogg Theora** : un codec gratuit et libre de droits, mais moins puissant que H.264. Il est bien reconnu sous Linux mais, sous Windows, il faut installer des programmes pour pouvoir le lire.
- **WebM** : un autre codec gratuit et libre de droits, plus récent. Proposé par Google, c'est le concurrent le plus sérieux de H.264 à l'heure actuelle.

Remarque

Pour convertir une vidéo dans ces différents formats, on peut utiliser le logiciel gratuit Miro Video Converter.

Il vous suffit de glisser-déposer votre vidéo dans la fenêtre du programme et de sélectionner le format de sortie souhaité. Cela vous permettra de créer plusieurs versions de votre vidéo !

Insertion d'un élément audio

En théorie, il suffit d'une simple balise pour jouer un son sur notre page :

```
<audio src="musique.mp3"></audio>
```

En pratique, c'est un peu plus compliqué que cela.
Si vous testez ce code… vous ne verrez rien ! En effet, le navigateur va seulement télécharger les informations générales sur le fichier (on parle de **métadonnées**) mais il ne se passera rien de particulier.

Vous pouvez compléter la balise des attributs suivants :

- **controls**: pour ajouter les boutons « Lecture », « Pause » et la barre de défilement. Cela peut sembler indispensable, et vous vous demandez peut-être pourquoi cela n'y figure pas par défaut, mais certains sites web préfèrent créer eux-mêmes leurs propres boutons et commander la lecture avec du JavaScript.
- **width**: pour modifier la largeur de l'outil de lecture audio.
- **loop**: la musique sera jouée en boucle.
- **autoplay**: la musique sera jouée dès le chargement de la page. Évitez d'en abuser, c'est en général irritant d'arriver sur un site qui joue de la musique tout seul !
- **preload**: indique si la musique peut être préchargée dès le chargement de la page ou non. Cet attribut peut prendre les valeurs :
  - auto(par défaut) : le navigateur décide s'il doit précharger toute la musique, uniquement les métadonnées ou rien du tout.
  - metadata: charge uniquement les métadonnées (durée, etc.).
  - none: pas de préchargement. Utile si vous ne voulez pas gaspiller de bande passante sur votre site.

On ne peut pas forcer le préchargement de la musique, c'est toujours le navigateur qui décide. Les navigateurs mobiles, par exemple, ne préchargent jamais la musique pour économiser la bande passante (le temps de chargement étant long sur un portable).

Ajoutons les contrôles et ce sera déjà mieux !

```
<audio src="hype_home.mp3" controls></audio>
```

L'apparence du lecteur audio change en fonction du navigateur. La figure suivante représente par exemple le lecteur audio dans Google Chrome.

Et si le navigateur ne gère pas le MP3, comment faire ?

Il faut proposer plusieurs versions du fichier audio. Dans ce cas, on va construire notre balise comme ceci :

```
<audio controls>
   <source src="hype_home.mp3">
   <source src="hype_home.ogg">
</audio>
```

Le navigateur prendra automatiquement le format qu'il reconnaît.

II. Insertion d'une vidéo

```
<video src="sintel.webm"></video>
```

Rajoutons quelques attributs (la plupart sont les mêmes que pour la balise<audio>) :

- **poster**: image à afficher à la place de la vidéo tant que celle-ci n'est pas lancée. Par défaut, le navigateur prend la première image de la vidéo mais, comme il s'agit souvent d'une image noire ou d'une image peu représentative de la vidéo, je vous conseille d'en créer une ! Vous pouvez tout simplement faire une capture d'écran d'un moment de la vidéo.
- **controls**: pour ajouter les boutons « Lecture », « Pause » et la barre de défilement. Cela peut sembler indispensable, mais certains sites web préfèrent créer eux-mêmes leurs propres boutons et commander la lecture avec du JavaScript. En ce qui nous concerne, ce sera largement suffisant !
- **width**: pour modifier la largeur de la vidéo.

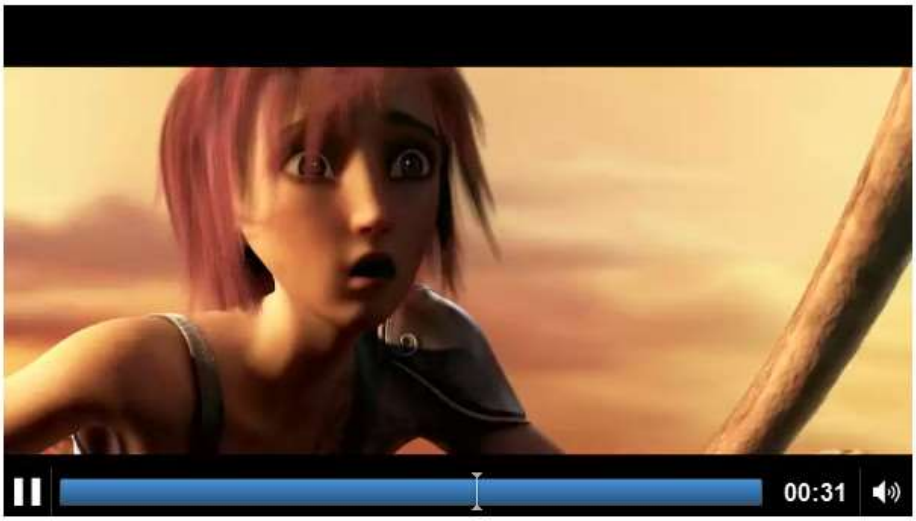
- **height**: pour modifier la hauteur de la vidéo.
- **loop**: la vidéo sera jouée en boucle.
- **autoplay**: la vidéo sera jouée dès le chargement de la page.

- **preload**: indique si la vidéo peut être préchargée dès le chargement de la page ou non. Cet attribut peut prendre les valeurs :
    - auto(par défaut) : le navigateur décide s'il doit précharger toute la vidéo, uniquement les métadonnées ou rien du tout.
    - metadata: charge uniquement les métadonnées (durée, dimensions, etc.).
    - none: pas de préchargement. Utile si vous souhaitez éviter le gaspillage de bande passante sur votre site.

On ne peut pas forcer le préchargement de la vidéo, c'est toujours le navigateur qui décide.
Les proportions de la vidéo sont toujours conservées. Si vous définissez une largeur et une hauteur, le navigateur fera en sorte de ne pas dépasser les dimensions indiquées mais il conservera les proportions.

Exemple

```
<video src="sintel.webm" controls poster="sintel.jpg" width="600"></video>
```



Une vidéo avec les options de lecture et une taille définie

Pour afficher un message ou d'utiliser une technique de secours (en Flash) si le navigateur ne reconnaît pas la balise :

```
<video src="sintel.webm" controls poster="sintel.jpg" width="600">  Il est temps de mettre à jour votre navigateur !</video>
```

Comment contenter tous les navigateurs, puisque chacun reconnaît des formats vidéo différents ?

Vous utiliserez la balise<source>à l'intérieur de la balise<video>pour proposer différents formats. Le navigateur prendra celui qu'il reconnaît :

```
<video controls poster="sintel.jpg" width="600">

   <source src="sintel.mp4">

   <source src="sintel.webm">

   <source src="sintel.ogv">

</video>
```

Contrôler (simplement) le média

Maintenant que la vidéo est présente, ajoutons un peu d'interactivité à cette dernière…

Les options « natives »

Les balises multimédia possèdent par défaut quelques attributs bien pratiques. En effet, voici une liste non exhaustive de celles que j'estime être les plus utiles dans l'immédiat :

- **controls** : permet de rajouter des boutons de contrôle de lecture standards (lecture/pause, barre de progression, plein-écran…) ;
- **autoplay** : (plutôt évident…) la lecture est lancée automatiquement dès que la vidéo commence à se charger ; n'en abusez pas, cela peut être assez gênant pour la navigation ;
- **poster** : lien vers une image d'illustration si la vidéo n'est pas disponible à l'adresse spécifiée ;
- **loop** : relance la lecture quand cette dernière est terminée, encore et encore ;
- **height** et width * : pour spécifier une hauteur et une largeur au lecteur ;
- **muted** : coupe le son.

Exemple : vidéo avec des contrôles, dont le son est coupé, qui jouera en boucle et dont la taille a été limitée à 320x240 pixels.

```
<video width="320" height="240" controls muted loop>
   <source src="chemin/vers/masource.mp4" type="video/mp4">
   <source src="chemin/vers/masource.ogg" type="video/ogg">
   <source src="chemin/vers/masource.webm" type="video/webm">
</video>
```

Des sous-titres pour les vidéos

A l'aide de la balise **<track>** qui sera placée dans une balise vidéo, cette dernière proposera des sous-titres au lecteur.

La balise *track* a besoin des informations suivantes :

- **src** : la source (relative ou absolue) du fichier de sous-titres (au format WebVTT .vtt (WEB Video Text Track)) ;
- **kind="subtitles"** : pour préciser que l'on parle de sous-titres ;
- **srclang** : le code international de la langue (en, de, fr…) ;
- **label** : le nom littéral de la piste de sous-titres.

Par exemple :

```
<video controls>
   <source src="ma-super-video.mp4" type="video/mp4">
   <source src="ma-super-video.ogg" type="video/ogg">
   <track src="subtitles  en.vtt" kind="subtitles" srclang="en" label="English">
   <track src="subtitles_fr.vtt" kind="subtitles" srclang="fr" label="Francais">
</video>
```

Remarque : pour créer des fichiers .VTT : https://support.office.com/fr-fr/article/cr%C3%A9er-des-sous-titres-pour-une-vid%C3%A9o-b1cfb30f-5b00-4435-beeb-2a25e115024b

III. Interagir avec les médias

Imaginons que nous voulions proposer un lecteur sans contrôles natifs, mais uniquement avec nos boutons HTML que nous pourrions styliser via du CSS. Il faudrait alors que ces boutons interagissent avec la vidéo correctement. Admettons que nous voulions ajouter les contrôles suivants :

- **Lecture** : lit la vidéo ;
- **Pause** : met la vidéo en pause ;
- **Stop** : arrête la vidéo ;
- **-10s** : recule la vidéo de 10 secondes ;
- **+10s** : avance la vidéo de 10 secondes ;

Structure de base

Voici la structure de base que nous allons respecter :

```
<video id="mavideo" controls>
   <source src="http://clips.vorwaerts-gmbh.de/VfE_html5.mp4" type="video/mp4">
   <source src="http://clips.vorwaerts-gmbh.de/VfE.webm" type="video/webm">
   <source src="http://clips.vorwaerts-gmbh.de/VfE.ogv" type="video/ogg">

   <p class="alert">
      Votre navigateur ne supporte pas la balise vidéo ! Mettez-vous à jour !
   </p>
</video>
<div id="controles" hidden>
</div>

function lecture() {
   // Lit la vidéo
}

function pause() {
   // Met la vidéo en pause
}

function stop() {
   // Arrête la vidéo
}

function avancer(duree) {
   // Avance de 'duree' secondes
}

function reculer(duree) {
   // Recule de 'duree' secondes
}

function creerBoutons() {
   // Crée les boutons de gestion du lecteur
}
```

Mettre nos contrôleurs

Comme vous pouvez le voir dans le squelette précédent, pour l'instant, aucun bouton personnalisé n'est présent sur notre page et la vidéo possède l'interface par défaut. En effet, nous allons créer les boutons dynamiquement en JavaScript dans la fonction creerBoutons() qui sera exécutée à la fin du chargement de la page. De cette manière, un utilisateur désactivant le JavaScript pourra tout de même utiliser le navigateur avec l'interface standard.

Voici comment nous allons créer nos boutons.

```
        var lecteur;

function creerBoutons() {
   // Crée les boutons de gestion du lecteur
   var btnLecture = document.createElement("button");
   var btnPause = document.createElement("button");
   var btnStop = document.createElement("button");
   var btnReculer = document.createElement("button");
   var btnAvancer = document.createElement("button");

   var controlesBox = document.getElementById("controles");
   lecteur = document.getElementById("mavideo");

   // Ajoute un peu de texte
   btnLecture.textContent = "Lecture";
   btnPause.textContent = "Pause";
   btnStop.textContent = "Stop";
   btnReculer.textContent = "-10s";
   btnAvancer.textContent = "+10s";

   // Ajoute les boutons à l'interface
   controlesBox.appendChild(btnLecture);
   controlesBox.appendChild(btnPause);
   controlesBox.appendChild(btnStop);
   controlesBox.appendChild(btnReculer);
   controlesBox.appendChild(btnAvancer);

   // Lie les fonctions aux boutons
   btnLecture.addEventListener("click", lecture, false);
   btnPause.addEventListener("click", pause, false);
   btnStop.addEventListener("click", stop, false);
   btnReculer.addEventListener("click", function(){reculer(10)}, false);
   btnAvancer.addEventListener("click", function(){avancer(10)}, false);

   // Affiche les nouveaux boutons et supprime l'interface originale
   controlesBox.removeAttribute("hidden");
   lecteur.removeAttribute("controls");
}

// Crée les boutons lorsque le DOM est chargé
document.addEventListener('DOMContentLoaded', creerBoutons, false);
```

Interagir avec la vidéo

Maintenant que nous avons un squelette, nous allons devoir faire appel aux propriétés de l'objet vidéo pour interagir avec (son id est « mavideo »). Pour cela, on ira chercher dans la référence

de l'élément : HTMLMediaElement. Vous y trouverez les attributs accessibles (dont certains ont été vus plus tôt) ainsi que les méthodes que nous pouvons appeler.

Ainsi nous trouverons par exemple les éléments suivants :

- play() : pour lire la vidéo ;
- pause() : pour la mettre en pause ;
- currentTime : attribut représentant le minutage actuel de la vidéo (*position* dans la vidéo).

Les méthodes JavaScript :

```
function lecture() {
   // Lit la vidéo
   lecteur.play();
}

function pause() {
   // Met la vidéo en pause
   lecteur.pause();
}

function stop() {
   // Arrête la vidéo
   // On met en pause
   lecteur.pause();
   // Et on se remet au départ
   lecteur.currentTime = 0;
}

function avancer(duree) {
   // Avance de 'duree' secondes
   // On parse en entier pour être sûr d'avoir un nombre
   lecteur.currentTime += parseInt(duree);
}

function reculer(duree) {
   // Recule de 'duree' secondes
   // On parse en entier pour être sûr d'avoir un nombre
   lecteur.currentTime -= parseInt(duree);
}
```