Architecture et langages Web
(NFA040)

Contenu audio et vidéo (tutoriel)

Remarque. Ce tutoriel est extrait de la section de MDN intitulée : Contenu audio et vidéo

Maintenant que nous sommes à l'aise pour ajouter de simples images dans une page web, nous passons à l'étape suivante : ajouter de la vidéo et un lecteur audio à vos documents HTML. Dans cet article, nous nous contenterons de le faire avec les éléments `<video>` et `<audio>`. Nous terminerons en apprenant comment ajouter des légendes et des sous-titres à vos vidéos.

Audio et vidéo sur le web

Les développeurs ont toujours voulu utiliser la vidéo et l'audio sur le web et ce, dès le début des années 2000, quand nous avons commencé à disposer d'une bande passante suffisamment rapide pour supporter toutes sortes de vidéos (les fichiers vidéo étant beaucoup plus lourds que du texte ou des images). Au départ, les technologies embarquées telles que HTML n'avaient pas la capacité d'intégrer de la vidéo ou de l'audio, donc, les solutions « propriétaires » (ou basées sur des greffons) comme Flash(puis, plus tard, Silverlight) sont devenues très populaires pour gérer ce type de contenu. Ces technologies fonctionnaient bien mais avaient de nombreux inconvénients, comme une relation aléatoire avec les fonctionnalités HTML/CSS, des problèmes de sécurité et d'accessibilité.

Une solution embarquée devrait résoudre la plupart de ces problèmes. Heureusement, après quelques années, la spécification HTML5 apportait ces améliorations avec les éléments `<video>` et `<audio>` et des APIs JavaScript flambants neufs pour les contrôler. Nous ne verrons pas JavaScript ici — nous poserons juste les fondamentaux qui peuvent être obtenus avec HTML.

Nous ne vous apprendrons pas à produire des fichiers audio ou vidéo — cela demande des compétences totalement différentes. Nous vous conseillons ce lien Github fichiers d'échantillons audio et vidéo et exemples de code pour votre expérience personnelle, au cas où vous ne pourriez pas y accéder par vous-même.

L'élément `<video>`

L'élément `<video>` vous permet d'intégrer de la vidéo très facilement. En voici un exemple :

```
<video src="rabbit320.webm" controls>
  <p>Votre navigateur ne prend pas en charge les vidéos HTML5. Voici, à la
place, un <a href="rabbit320.webm">lien sur la vidéo</a>.</p>
</video>
```

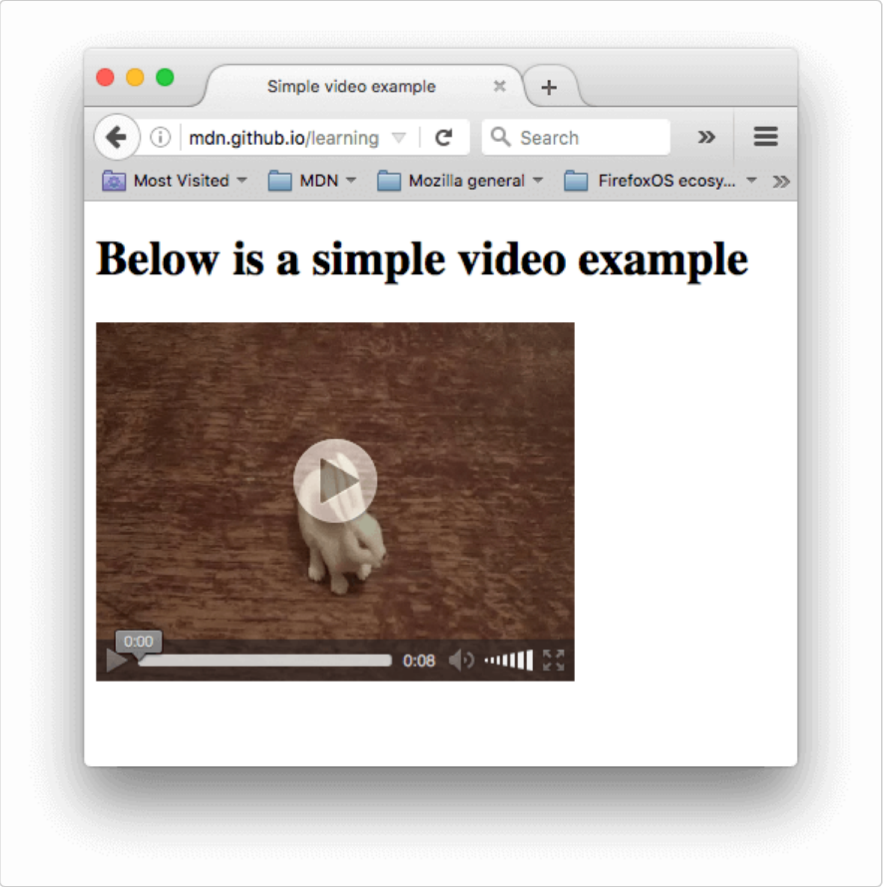
Les fonctionnalités de ce code sont :

`src`. De la même manière que pour l'élément `<img>`, l'attribut `src` (source) contient le chemin vers la vidéo que vous voulez intégrer. Cela fonctionne de la même manière.

`controls`. Les utilisateurs doivent avoir un contrôle sur la lecture de la vidéo ou de l'audio. (c'est particulièrement crucial pour les gens ayant de l'épilepsie.) Vous devez vous servir de l'attribut `controls` pour appeler l'interface de contrôle du navigateur ou construire votre propre interface en utilisant l'API JavaScript adéquat. Au minimum, l'interface doit avoir un contrôle de démarrage et d'arrêt (start/stop) du média et un pour ajuster le volume.

Le paragraphe dans la balise `<video>`. Cela peut s'appeler solution de repli ou contenu de secours (fallback content) — si le navigateur accédant à la page ne supporte pas l'élément `<video>`, cela offre un texte alternatif qui peut être ce que vous voulez ; dans ce cas nous avons mis un lien direct au fichier vidéo, afin que l'utilisateur puisse au moins y accéder sans avoir à se soucier du navigateur qu'il utilise.

La vidéo intégrée donnerait quelque chose comme ça :



Faites un essai avec l'exemple ici. (voyez aussi le code source.)

Gestion de différents formats

Il y a un problème avec l'exemple au-dessus que vous avez dû rencontrer si vous avez accédé au lien « exemple ici » avec un navigateur comme Safari ou Internet Explorer. La vidéo ne se lancera pas ! Ceci parce que les navigateurs acceptent des formats différents de video et d'audio.

Voyons-en rapidement la terminologie. Les formats comme le MP3, MP4 et le WebM sont appelés des formats conteneurs. Ils contiennent plusieurs parties qui, ensemble, donnent l'intégralité de la chanson ou de la vidéo — comme une piste audio, une piste vidéo et les métadonnées qui décrivent le média qui est lu.

Les pistes audio et vidéo sont aussi de différents formats, par exemple :

- Un conteneur WebM empaquette de l'audio Ogg Vorbis avec de la vidéo VP8/VP9. Firefox et Chrome, en particulier, assurent sa prise en charge.

– Un conteneur MP4 assemble de l'audio AAC ou MP3 en audio avec de la vidéo H.264. Internet Explorer et Safari, principalement, le prennent en charge.

– L'ancien conteneur Ogg rassemblait de l'audio Ogg Vorbis et de la vidéo Ogg Theora. Il était essentiellement pris en charge par Firefox and Chrome, mais il a été supplanté par le format WebM qui est de meilleure qualité.

Un lecteur audio peut jouer directement une piste audio, par ex. un fichier MP3 ou Ogg. Elles ne nécessitent pas de conteneur.

Note : Ce n'est pas si simple, comme vous pouvez le voir dans le tableau de compatibilité des codecs audio-vidéo (en-US). En outre, de nombreux navigateurs de plateforme mobile peuvent lire un format non pris en charge en le transférant au lecteur multimédia du système sous-jacent. Mais pour l'instant nous nous contenterons de ce qui précède.

Les formats ci-dessus ont été créés pour compresser la vidéo et l'audio dans des fichiers gérables (les fichiers vidéo et audio bruts sont très volumineux). Les navigateurs contiennent différents Codecs, comme Vorbis ou H.264, utilisés pour convertir le son et la vidéo compressés en binaire et inversement. Comme indiqué ci-dessus, les navigateurs ne supportent malheureusement pas tous les mêmes codecs, vous devrez donc fournir plusieurs fichiers pour chaque production de média. S'il vous manque le bon codec pour décoder le média, il ne pourra pas être lu.

Note : Vous êtes peut-être surpris de l'existence d'une telle situation. Les formats **MP3** (pour l'audio) et **MP4**/**H.264** (pour la vidéo) sont tous deux largement pris en charge et de bonne qualité. Cependant, ils sont également grevés de brevets — les brevets américains couvrent le MP3 jusqu'en 2017 au moins et le H.264 jusqu'en 2027 au plus tôt, ce qui signifie que les navigateurs ne détenant pas de licence doivent payer d'énormes sommes d'argent pour pouvoir utiliser ces formats. En outre, beaucoup de personnes évitent, par principe, les logiciels propriétaires et leur préférent des formats ouverts. C'est pourquoi nous devons fournir plusieurs formats pour une prise en charge par différents navigateurs.

Alors, comment faire ? Jetez un coup d'œil à l'exemple qui suit, mis à jour, (essayez-le directement ici aussi) :

```
<video controls>
  <source src="rabbit320.mp4" type="video/mp4">
  <source src="rabbit320.webm" type="video/webm">
  <p>Votre navigateur ne prend pas en charge les vidéos HTML5. Voici, à la
place, un <a href="rabbit320.mp4">lien sur la vidéo</a>.</p>
</video>
```

Ici, nous avons retiré l'attribut `src` de la balise <video> et inclus des éléments `<source>`séparés qui pointent vers des sources appropriées. Dans ce cas, le navigateur parcourra les éléments `<source>` et jouera le premier dont il peut prendre en charge le codec. Inclure des sources WebM et MP4 devraient suffire pour lire votre vidéo sur la plupart des plateformes et navigateurs de nos jours.

Chaque élément <source> possède également un attribut de type. C'est facultatif, mais il est conseillé de les inclure — ils contiennent le type MIME des fichiers vidéo, et les navigateurs peuvent le lire et sauter immédiatement les vidéos qu'ils ne comprennent pas. Si le type n'est pas indiqué, le navigateur va charger et essayer de lire chaque fichier jusqu'à ce qu'il en trouve un qu'il prenne en charge, ce qui demande du temps et des ressources.

Autres fonctionnalités de `<video>`

Il y a possibilité d'inclure d'autres fonctionnalités dans une vidéo HTML5. Regardez notre troisième exemple :

```
<video controls width="400" height="400"
       autoplay loop muted
       poster="poster.png">
  <source src="rabbit320.mp4" type="video/mp4">
  <source src="rabbit320.webm" type="video/webm">
  <p>Votre navigateur ne prend pas en charge les vidéos HTML5. Voici, à la
place, un <a href="rabbit320.mp4">lien à la vidéo</a>.</p>
</video>
```

Cela produit une sortie du type suivant :

Voici les nouvelles fonctionnalités :

`width` et `height`. Il est possible de contrôler la taille de la vidéo soit avec ces attributs, soit avec le CSS. Dans les deux cas, les vidéos conservent le rapport largeur-hauteur natif — désigné sous le vocable **rapport de proportions**. Si ce dernier ne correspond pas aux tailles indiquées, la vidéo occupera tout l'espace horizontal et l'espace non rempli sera de la couleur d'arrière plan unie par défaut.

`autoplay`. Cet attribut permet de lancer immédiatement la lecture de l'audio ou de la vidéo pendant que le reste de la page se charge. Nous vous déconseillons d'utiliser la lecture automatique de vidéos (ou audio) sur vos sites, car les utilisateurs peuvent trouver cela vraiment ennuyeux.

`loop`. Cet attribut permet de relancer en boucle la lecture de la vidéo (ou de l'audio). Cette façon de procéder pouvant être mal perçue, ne l'utilisez que si c'est vraiment nécessaire.

`muted`. Cet attribut coupe le son de la vidéo par défaut.

`poster`. Cet attribut prend comme valeur l'URL d'une image affichée avant la lecture de la vidéo. Il s'utilise en tant que logo de démarrage ou de publicité.

`preload`. Cet attribut s'utilise pour mettre en tampon les gros fichiers. Il peut prendre 3 valeurs :

- `"none"` : ne pas mettre le fichier dans un tampon

- `"auto"` : mettre le fichier média dans un tampon
- `"metadata"` : ne mettre que les métadonnées dans le tampon

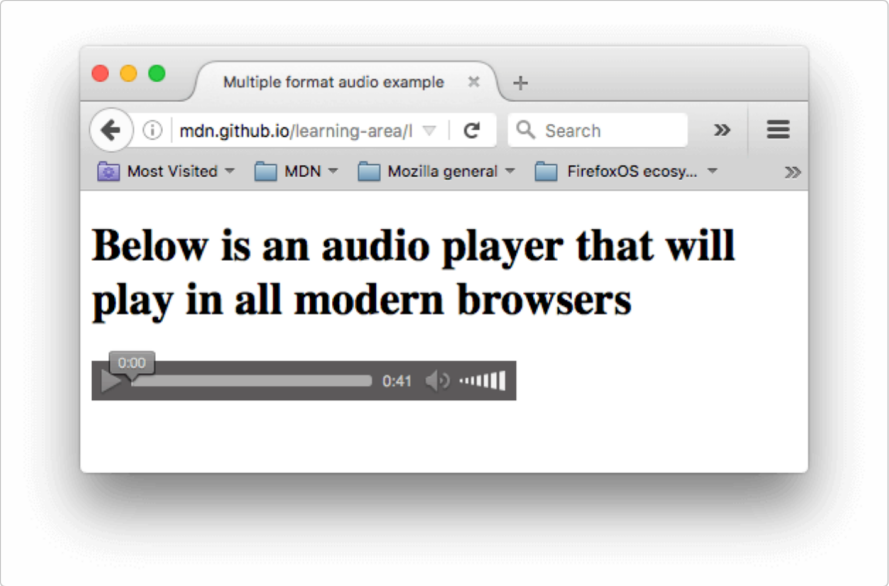
Vous trouverez cet exemple prêt pour l'interprétation sur Github (voir aussi le code source). Notez que nous n'avons pas inséré l'attribut `autoplay` dans la version en direct — si la vidéo débute dès le chargement de la page, vous ne pourrez pas voir le poster !

L'élément `<audio>`

L'élément `<audio>` fonctionne exactement de la même manière que l'élément `<video>`, mais avec quelques menues différences décrites plus bas. Un exemple classique ressemble à ceci :

```
<audio controls>
  <source src="viper.mp3" type="audio/mp3">
  <source src="viper.ogg" type="audio/ogg">
  <p>Votre navigateur ne prend pas en charge l'audio HTML5. Voici, à la place, un
<a href="viper.mp3">lien sur l'audio</a>.</p>
</audio>
```

Vous verrez quelque chose de ce genre dans un navigateur :



Cela prend moins de place qu'une vidéo, et il n'y a pas de composante visuelle — il est juste nécessaire d'afficher les contrôles de lecture de l'audio. Voici les autres différences avec les vidéos HTML5 :

- L'élément `<audio>` ne prend pas en charge les attributs `width`/`height` — redisons-le, il n'y a pas de composant visuel, il n'y donc pas lieu d'assigner une largeur ou une hauteur.
- Il ne prend pas en charge non plus l'attribut `poster` — toujours pas de composant visuel.

Excepté ce qui précéde, `<audio>` accepte les mêmes fonctionnalités que `<video>` — revoyez les sections ci-desssus pour plus d'informations à ce propos.

Afficher du texte dans une vidéo

Nous allons maintenant parler d'un concept un peu plus avancé vraiment utile à connaître. Beaucoup de gens ne peuvent pas ou ne veulent pas entendre le contenu audio/vidéo qu'ils trouvent sur le Web, du moins à certains moments. Par exemple :

- De nombreuses personnes sont mal-entendantes (dures d'oreille ou sourdes), et ne peuvent donc pas entendre le son.
- D'autres ne veulent pas de son, soit parce qu'ils sont dans un environnement bruyant (comme un bar avec une foule pendant une retransmission de compétition sportive) et ne peuvent donc pas entendre, soit parce qu'ils sont dans un environnement silencieux (comme une bibliothèque) et ne veulent donc pas déranger.
- Des personnes qui ne parlent pas la langue d'une vidéo peuvent souhaiter un sous-titrage ou une traduction pour les aider à comprendre le contenu du média.

Ne serait-il pas agréable de pouvoir fournir à ces personnes la transcription des paroles prononcés dans l'audio ou la vidéo ? Eh bien, avec des vidéos HTML5 vous le pouvez, à l'aide du format WebVTT et de l'élément `<track>`.

Note : « Transcrire » signifie écrire des paroles sous forme de texte, et « transcription » est l'action correspondante.

WebVTT est un format d'écriture de fichiers texte ; il contient nombre de chaînes de texte avec des métadonnées comme l'instant dans la vidéo où vous souhaitez l'affichage du texte, et même une information succinte sur le style et la position de celui-ci. Ces chaînes textuelles sont appelées des marqueurs, les plus courants étant :

les sous-titres (`subtitles`). Traductions d'éléments d'une langue étrangère pour les personnes ne comprenant pas les paroles de l'audio.

les légendes (`captions`). Transcriptions synchrones de dialogues ou de descriptions de sons significatifs, pour permettre aux personnes ne pouvant entendre le son de comprendre ce qui se passe.

les descriptions programmées (`descriptions`). Textes convertis en audio, pour aider les personnes avec des défauts de vision.

Un fichier WebVTT typique ressemblera à :

```
WEBVTT

1
00:00:22.230 --> 00:00:24.606
Ceci est le premier sous-titre.

2
00:00:30.739 --> 00:00:34.074
Ceci est le deuxième.

  ...
```

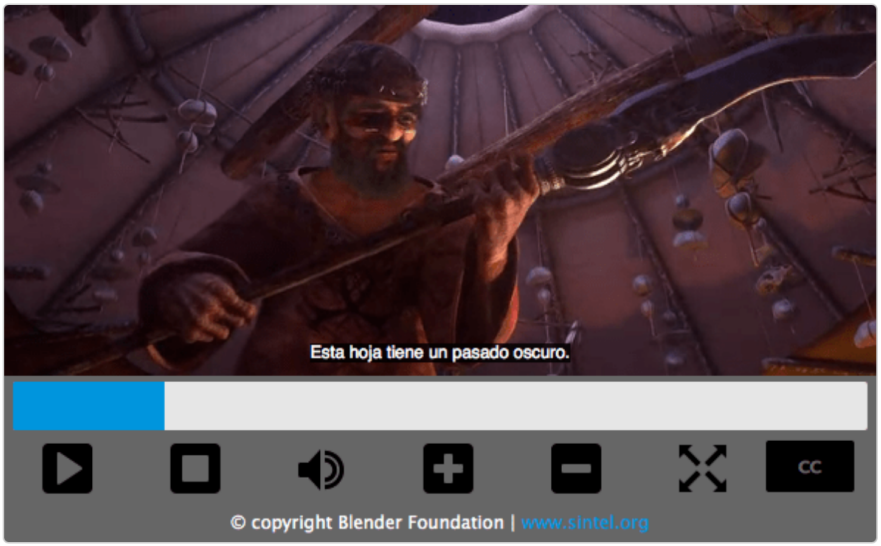
Pour qu'il soit affiché avec la diffusion du média HTML, il faut :

1. Enregistrer le fichier avec l'extension `.vtt` dans un endroit sensé.
2. Lier le fichier `.vtt` avec l'élément `<track>`. `<track>` doit être placé entre les balises `<audio>` ou `<video>`, mais après tous les éléments `<source>`. Utilisez l'attribut `kind`pour préciser si les marqueurs sont `subtitles`, `captions` ou `descriptions`. Plus loin, utilisez l'attribut `srclang` pour indiquer au navigateur la langue dans laquelle sont écrit les sous-titres.

Voici un exemple :

```
<video controls>
    <source src="example.mp4" type="video/mp4">
    <source src="example.webm" type="video/webm">
    <track kind="subtitles" src="subtitles_en.vtt" srclang="en">
</video>
```

Il en résultera une vidéo dont les sous-titres seront affichés un peu comme ceci :



Pour plus de détails, lisez Ajouter des légendes et des sous-titres aux vidéos HTML5. Vous trouverez un exemple, écrit par Ian Devlin, accompagnant cet article sur Github (voyez le code source aussi). Cet exemple utilise un peu de JavaScript pour permettre à l'utilisateur de choisir entre différents sous-titres. Notez que pour activer les sous-titres, vous devez presser le bouton « CC » et selectionner une option — English, Deutsch ou Español.