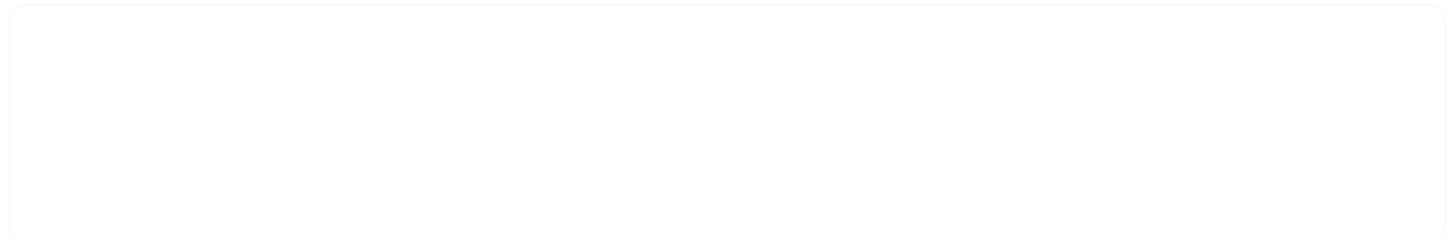


Introduction à la balise video de HTML5

Une des nouveautés majeures de HTML5 et l'une des plus remarquables est la balise `video`.



L'élément `<video>`, cousin de `<audio>` offre en HTML5 une solution simple, native pour les navigateurs pour l'intégration d'une vidéo dans une page web. Elle permet également de proposer une alternative à l'utilisation de Flash pour les plate-formes ne le supportant pas (iOS par exemple avec iPhone, iPod, iPad...)



Balise

Syntaxe générale

La syntaxe de base de la balise video est extrêmement simple :

```
<video controls src="video.ogv">Ici la description alternative</video>
```

L'attribut `src` définit bien entendu l'adresse du fichier vidéo, tout comme pour la balise `img` lorsqu'il s'agit d'une image. Si vous indiquez les dimensions avec les attributs `height` et `width`, c'est encore mieux, et si tout va bien, un élément devrait s'afficher dans le navigateur... pour peu que celui-ci supporte le format de vidéo indiqué dans la source.



Sources multiples

On peut également proposer **plusieurs sources** dans plusieurs formats différents en indiquant les types MIME grâce à l'attribut `type` :

```
<video width="400" height="222" controls="controls">
  <source src="video.mp4" type="video/mp4" />
  <source src="video.webm" type="video/webm" />
  <source src="video.ogv" type="video/ogg" />
  Ici l'alternative à la vidéo : un lien de téléchargement, un message, etc.
</video>
```

Les navigateurs ne pouvant pas lire le MP4/H.264 ni la version WebM nativement (comme Firefox 3.6 par exemple) prendront la version au format Ogg Theora. Cela vous oblige néanmoins à encoder le fichier avec différents codecs.

Particularité de la syntaxe XHTML : il faut ajouter `controls="controls"` (et pas juste `controls` comme vous pourrez le voir sur le premier exemple) pour afficher les possibilités de contrôle de la vidéo. Ceci est valable pour tous les attributs (`autoplay`, etc.).

Attributs

L'attribut `controls` donne accès aux contrôles de lecture (boutons de navigation, volume, etc, selon les possibilités du navigateur), ou les masque s'il est omis.

L'attribut `preload="auto"` permet de spécifier au navigateur de **débuter le téléchargement** de la vidéo tout de suite, en anticipant sur le fait que l'utilisateur lira la vidéo. Attention, cette option est à manier avec prudence (il est préférable que ce soit la seule raison d'être de la page). Note : il s'agit de l'ancien attribut `autobuffer` qu'il vous faudra laisser pour Firefox 3.5 et 3.6.

L'attribut `autoplay="true"` comme son nom l'indique, permet de lancer la **lecture automatiquement**. Cela peut également être problématique avec une connexion à faible bande passante ou sur un terminal mobile. De manière générale, évitez d'imposer vos choix à l'utilisateur... et à sa connexion internet.

L'attribut `poster="image.jpg"` permet d'indiquer une **image à afficher par défaut** dans l'espace réservé par la vidéo, avant que la lecture de celle-ci ne soit lancée.

L'attribut `loop` indique que la lecture doit s'effectuer **en boucle**.

Prérequis

Pensez également à préciser les types MIME dans un fichier `.htaccess` pour être sûr qu'ils soient corrects, les trois lignes suivantes suffisent à s'assurer la tranquillité :

```
AddType video/ogg .ogv
AddType video/mp4 .mp4
AddType video/webm .webm
```

Formats

Plusieurs formats tiennent le devant de la scène : WebM, MP4 et Ogg Theora. Même si le but de ce tutoriel est de proposer une solution d'intégration de la balise `video` compatible sur le plus de navigateurs possible (et pas de discuter du choix des formats dans un interminable débat), faisons quand même une présentation rapide.

H.264/MP4

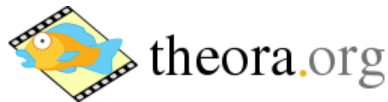


H.264 est supporté par le Moving Picture Experts Group. C'est un format non-libre (soumis à brevets) et non-gratuit. Toutefois, il est gratuit dans certaines utilisations (la diffusion gratuite de vidéos par des sites Web par exemple).

🌐 Pour en savoir plus : [H264 sur Wikipedia](http://fr.wikipedia.org/wiki/H.264) » <http://fr.wikipedia.org/wiki/H.264> .

Les fichiers MP4 utilisant H.264 sont lisibles nativement sur les navigateurs Apple (Safari, Safari Mobile), Opera, Google Chrome et Firefox depuis la version 34.

OGG/Theora



Theora est un format de compression vidéo open-source, sans brevets. Ceci donne le droit à tous d'utiliser Theora (à des fins non commerciales tout comme à des fins commerciales) sans devoir payer de redevance au consortium MPEG.

🌐 Pour en savoir plus : [Theora sur Wikipedia](http://fr.wikipedia.org/wiki/Theora) » <http://fr.wikipedia.org/wiki/Theora> .

OGG/Theora est lisible sur Firefox, Opera, et Google Chrome.




WebM/VP8



WebM est un format multimédia ouvert qui a été lancé par Google (après rachat de la société On2 Technologies). L'utilisation est en libre et gratuite.

🌐 Pour en savoir plus : [WebM sur Wikipedia](http://fr.wikipedia.org/wiki/WebM) » <http://fr.wikipedia.org/wiki/WebM> .

Comme on peut le constater, il y a une certaine disparité dans le support des codecs, chacun défendant ses intérêts pour le meilleur ou pour le pire (commerciaux ou libres).

Navigateurs	H.264/MP4	OGG Theora	WebM
	Firefox 34+	Firefox 3.5+	Firefox 4+
	Opera 25+	Opera 10.5+	Opera 10.6+
	Internet Explorer 9+	non	si les codecs sont installés
	Google Chrome 4+	Google Chrome 4+	Google Chrome 6+
	Safari 5+	non	non

🌐 Pour plus de détails sur les supports selon les navigateurs :

- Support de h264 sur Canluse » <http://caniuse.com/#feat=mpeg4>
- Support de WebM sur Canluse » <http://caniuse.com/#feat=webm>
- Support de Ogg Theora sur Canluse » <http://caniuse.com/#feat=ogv>

Génération et conversion de fichiers

Plusieurs logiciels permettent de générer les fichiers MP4, WebM et OGV. Le plus simple et pratique à mon sens est [Miro Video Converter](#) » <http://www.mirovideoconverter.com/> (qui en plus à la qualité non négligeable d'être libre et gratuit). Il gère tous les formats (MP4, WebM et Ogg). [Wtheora](#) » <http://opensource.grisambre.net/wtheora/> (basé sur [FFMpeg](#) » <http://ffmpeg.org/>) vous permettra de générer des fichiers Ogg tout en ayant la possibilité de paramétrer finement la qualité de ces derniers.

Un exemple

Voici un premier exemple, notez que je mets la version mp4 en premier afin d'être sûr que les iPhone, iPad et autres iPod pourront la lire (ils lisent uniquement la version mp4, et il faut que cette dernière soit proposée en premier pour que cela fonctionne).

```
<video width="400" height="222" controls="controls">
  <source src="animations/Cavernae_Terragen2.mp4" type="video/mp4" />
  <source src="animations/Cavernae_Terragen2.webm" type="video/webm" />
  <source src="animations/Cavernae_Terragen2.theora.ogv" type="video/ogg" />
  Vous n'avez pas de navigateur moderne, donc pas de balise video HTML5 !
</video>
```

L'idée de cet exemple est de permettre au plus grand nombre de navigateurs **modernes** de pouvoir visualiser la vidéo, je propose donc les trois formats (MP4, WebM, et Ogg). Par contre, mon alternative pour les navigateurs ne supportant pas la balise `video` est un peu sèche dans l'exemple ci-dessus. :)

Comme l'idée est de proposer la vidéo au plus grand nombre, je vais insérer en version alternative une version flash de cette même vidéo, ainsi, Internet Explorer pourra quand même la lire. Pour cet exemple, j'ai choisi d'insérer le player YouTube de cette vidéo.

```
<video width="400" height="222" controls="controls">
<source src="animations/Cavernae_Terragen2.mp4" type="video/mp4" />
<source src="animations/Cavernae_Terragen2.webm" type="video/webm" />
<source src="animations/Cavernae_Terragen2.theora.ogv" type="video/ogg" />
  <object type="application/x-shockwave-flash" width="400" height="222"
data="http://www.youtube.com/v/HVMoJqg41Bw">
    <param name="movie" value="http://www.youtube.com/v/HVMoJqg41Bw" />
    <param name="wmode" value="transparent" />
    Vous n'avez pas de navigateur moderne, ni Flash installé...
  </object>
</video>
```

⚠ La balise `object` est parfaitement valide... toutefois, Internet Explorer 6 et inférieurs ne comprendront pas ce beau code standard et valide. Pour arriver à garder un code valide, tout en satisfaisant IE6, il faudra passer par la balise `embed` (qui n'est absolument pas standard dans la spécification HTML4/XHTML1.0 par exemple). En utilisant les commentaires conditionnels pour cacher ce vilain bout de code non standard (sauf à IE6 et inférieurs), on obtient ceci :

```
<video width="400" height="222" controls="controls">
<source src="animations/Cavernae_Terragen2.mp4" type="video/mp4" />
<source src="animations/Cavernae_Terragen2.webm" type="video/webm" />
<source src="animations/Cavernae_Terragen2.theora.ogv" type="video/ogg" />
  <object type="application/x-shockwave-flash" width="400" height="222"
```

```
data="http://www.youtube.com/v/HVMoJgg41Bw">
  <param name="movie" value="http://www.youtube.com/v/HVMoJgg41Bw" />
  <param name="wmode" value="transparent" />

  <!--[if lte IE 6 ]>
    <embed src="http://www.youtube.com/v/HVMoJgg41Bw" type="application/x-shockwave-
flash" allowscriptaccess="always" allowfullscreen="true" width="400" height="222">
    </embed>
  <![endif]-->
  Vous n'avez pas de navigateur moderne, ni Flash installé... suivez les liens ci-
dessous pour télécharger les vidéos.
</object>
</video>
```

Une bonne habitude à prendre est de proposer un lien de téléchargement de vos fichiers vidéos (ainsi les internautes pourront le lire localement), et tant qu'à faire de préciser les tailles desdits fichiers.

🔗 Si vous voulez voir le résultat de l'exemple final avec les liens permettant le téléchargement des divers fichiers :
[Exemple d'intégration de la balise video de HTML5 » /xmedia/tuto/html5/video/exemples.html](#) .

Au final, on a avec l'exemple donné ci-dessus un système qui permet de lire une vidéo sur iPhone, iPad, Google Chrome, Safari, Firefox 3.5 et supérieurs, Opera 10.50 et supérieurs, et Internet Explorer (6, 7 et 8) via l'alternative Flash. Autrement dit, tous les principaux navigateurs peuvent lire cette vidéo, les plus modernes pouvant en prime la lire nativement via la balise `video`, cette balise pouvant par ailleurs être stylée via une CSS... comme n'importe quel élément HTML et contrôlée grâce à du JavaScript. L'inconvénient reste bien sûr de devoir prévoir les différents formats adaptés.

Conclusion

Comme vous avez pu le voir, l'utilisation de la balise `video` est très simple. Générer les fichiers dans les divers formats est très rapide et aisé, la solution est respectueuse des standards (oui, cela passe aux validateurs !) et plutôt élégante. Maintenant que j'ai pris l'habitude de m'en servir, bien que ce soit récent, je me surprends même à me dire "mais comment faisait-on avant l'arrivée de cette balise..." :)