Insertion de médias en HTML

Rédigé par: Jean-Luc Colson

Date première rédaction: Janvier 2020

SUIVI DES MODIFICATIONS A LA FICHE

Série	Date	Page(s) modifiée(s)	Raison

Contents

Insérer des images dans des pages avec l'élément HTML img

Nous allons pouvoir insérer des images au sein de nos fichiers HTML en utilisant l'élément HTML img représenté par une balise orpheline <img>.

Dans cette leçon, nous allons nous intéresser aux différents formats d'image (jpeg, png, etc.) et allons voir comment insérer une ou plusieurs images dans nos pages en HTML.

Nous discuterons également de la notion d'accessibilité du web à tous ainsi que de la mise en forme des images.

Un point sur les différents formats d'image

Comme vous le savez certainement, vous pouvez enregistrer vos images sous différents formats. Les formats les plus utilisés sont :

- Le JPG ou JPEG ;
- Le PNG ;
- Le GIF ;
- Le BITMAP.

Chaque format possède ses propres spécificités et il faut donc faire bien attention au choix du format lorsqu'on enregistre une image.

Le format JPEG (pour Joint Photographic Expert Group), ou plus communément JPG est un format qui permet généralement de compresser le poids d'une image par dix, tout en conservant une très bonne qualité. Généralement, nous choisirons ce format pour enregistrer des photos.

Le PNG (Portable Network Graphic) est un format qui a été créé à l'origine pour remplacer le format GIF. Le grand intérêt de ce format est qu'il gère la transparence. Celui-ci a un très bon taux de compression tout en conservant une bonne qualité d'image. Nous utiliserons généralement ce format pour enregistrer nos images qui ne sont pas des photographies.

Le GIF (Graphic Interchange Format) est un vieux format d'images que je ne recommande plus d'utiliser aujourd'hui, sauf dans le cas d'images animées.

Finalement, le BITMAP (ou BMP) est un format qui possède une très bonne prise en charge par tous les navigateurs et éditeurs. Cependant, la compression des images est assez mauvaise, ce qui donne au final des images très lourdes et donc longues à charger. Pour cette raison, je vous déconseille également d'utiliser le BITMAP pour enregistrer vos images.

Insérer des images en HTML

L'insertion d'images en HTML va se faire au moyen de l'élément HTML img. Cet élément est représenté par une balise orpheline.

Au sein de l'élément img, nous allons obligatoirement devoir préciser deux attributs : les attributs src et alt.

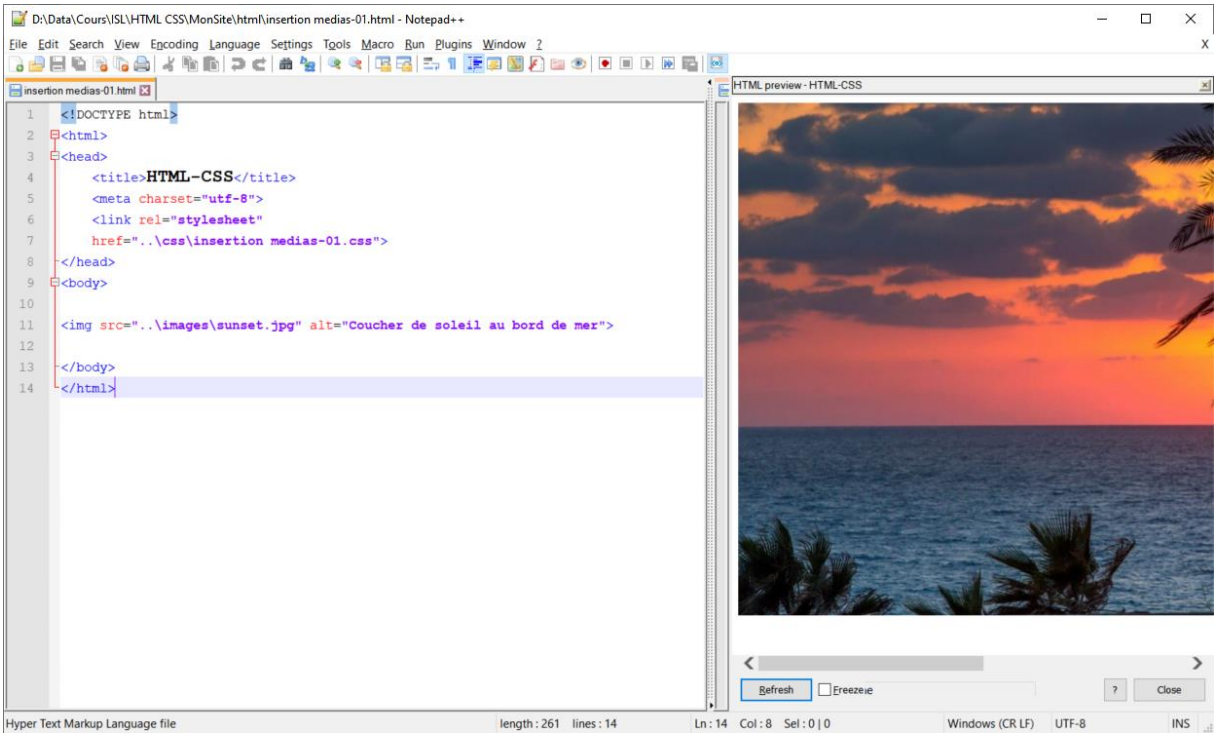
L'attribut src (pour source) va prendre comme valeur l'adresse de l'image (adresse relative ou absolue) tandis que l'attribut alt (pour alternative) va contenir un texte alternatif décrivant l'image. Ce texte va être affiché si l'image ne peut pas l'être pour une raison ou pour une autre, et est également très utile pour les non-voyants ou les mal voyants qui vont pouvoir « lire » notre image grâce à leurs lecteurs particuliers.

J'insiste ici sur l'importance de l'attribut alt : le web a été créé avec l'idée d'accessibilité à tous et il est donc de notre devoir de tout faire pour rendre chacune de nos pages lisibles pour tous et particulièrement pour les gens souffrant de déficiences.

Voyons maintenant comment fonctionne l'insertion d'images en pratique. On va commencer par enregistrer une image dans le même dossier que notre page HTML. Pour ma part, mon image s'appelle « sunset.png » et possède une largeur et une hauteur de 150 pixels.

Une nouvelle fois, choisissez un nom d'image sans espace ni caractère spécial (pas d'accent notamment). Cela évitera des problèmes potentiels.

L'attribut src va fonctionner de la même manière que l'attribut href pour les liens. Ainsi, si vous enregistrez votre image dans un dossier différent de votre page HTML, pensez bien à en tenir compte dans la valeur donnée à src.



Comme vous pouvez le voir, notre image s'affiche bien à sa taille d'origine. Dans l'absolu, on essaiera de redimensionner l'image à priori (avant téléchargement sur serveur) pour éviter de consommer des ressources serveurs inutilement. En effet, une image plus grosse de base va demander plus de ressources et de temps à charger car celle-ci va être plus lourde.

Pour redimensionner une image à posteriori, nous avons deux façons de faire : soit en utilisant le CSS, soit en ajoutant des attributs width et height dans notre élément img en HTML.

Bien évidemment, nous préférerons toujours pour des raisons sémantiques et de maintenabilité du code dans la mesure du possible effectuer ces opérations en CSS. Cependant, dans certaines situations particulières, nous serons obligés de le faire en HTML et il est donc bon de savoir le faire.

Insérer des images provenant d'autres sites en HTML

Comme je vous l'ai précisé, on peut aussi préciser une adresse absolue comme valeur pour notre attribut src.

Par exemple, si je souhaite afficher une image provenant d'un autre site, j'utiliserai évidemment une adresse absolue, c'est-à-dire que j'indiquerai l'URL complète de la ressource image que je souhaite afficher en valeur de mon attribut src.

Vous pouvez essayer avec n'importe quelle image sur le web, cela fonctionnera. Faites cependant attention à bien récupérer l'URL complète de l'image ainsi qu'aux droits d'auteur ! Notez également qu'il est généralement déconseillé d'utiliser des images provenant d'autres sites car si ces sites les suppriment, elles ne s'afficheront plus non plus sur le vôtre.

Redimensionner une image en HTML

On va pouvoir ajouter des attributs facultatifs pour modifier l'affichage de la plupart des éléments HTML. Cependant, répétons-le, ce n'est jamais la façon recommandée de faire les choses pour des raisons de séparation de rôle des langages (raisons de sémantique) et de maintenabilité du code.

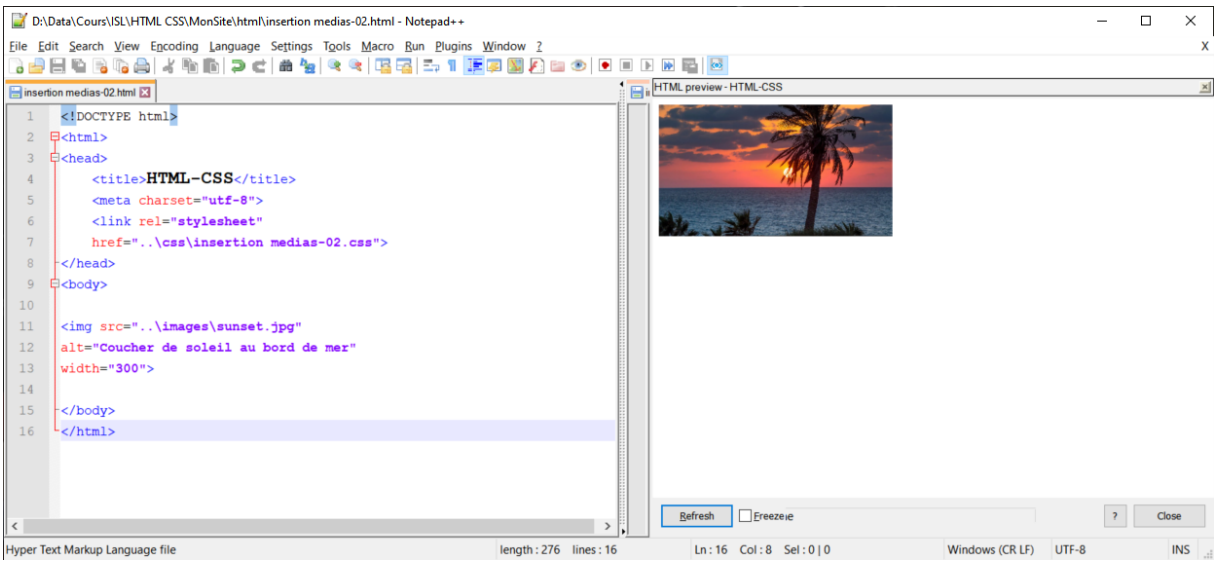
Le HTML est un langage de balisage qui ne doit normalement servir qu'à indiquer l'identité des différents contenus d'une page. Pour mettre en forme ces contenus, il faut utiliser dans la mesure du possible le CSS qui est là pour ça.

Cependant, dans certaines situations, nous n'allons pas avoir accès au CSS ou il va être beaucoup plus compliqué d'appliquer des styles à nos images en CSS qu'en HTML.

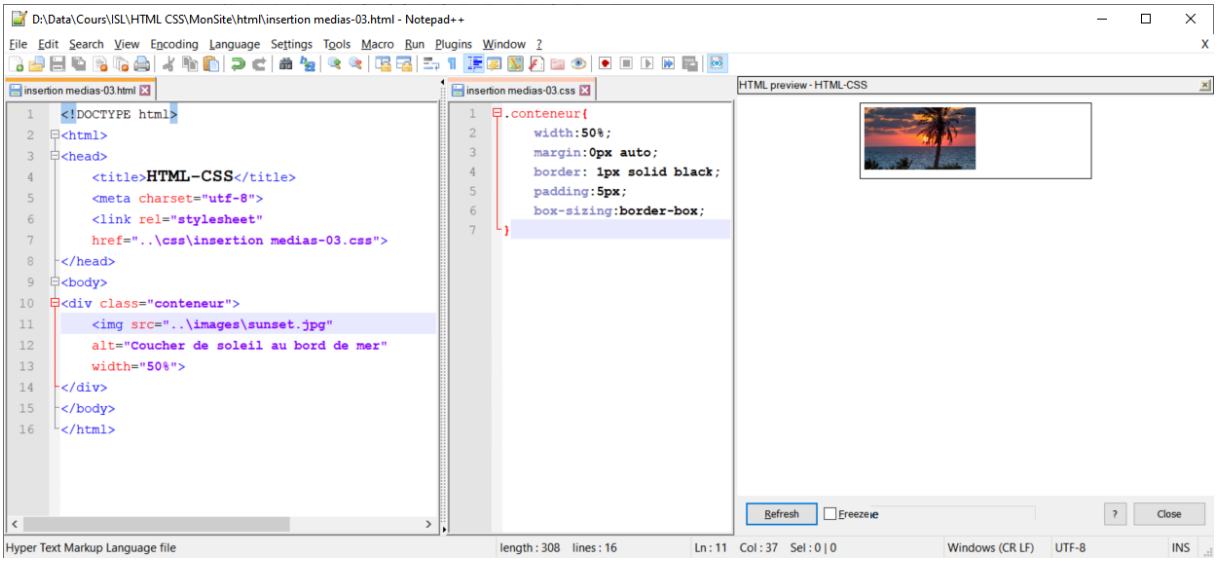
Dans ces cas-là, nous allons déjà pouvoir utiliser les attributs width (« largeur ») et height (« hauteur ») au sein de notre élément img pour modifier la largeur et la hauteur de l'image. Ces attributs peuvent prendre des valeurs absolues (en px généralement) ou relatives (en % dans la majorité des cas) ou des mots clefs comme auto.

Généralement, on ne modifiera que l'une des deux dimensions de l'image (largeur ou hauteur, mais plutôt la largeur car modifier la hauteur peut rapidement poser des problèmes d'ergonomie) afin que l'image se redimensionne d'elle-même et conserve ses proportions.

On pourra éventuellement préciser la valeur auto pour la deuxième dimension si on a peur qu'une taille ait déjà été définie quelque part dans le code.



Notez que dans le cas où on indique des valeurs en pourcentage, le pourcentage donné représente l'espace que l'image doit prendre *par rapport à la taille de son conteneur* (son élément parent le plus proche, c'est-à-dire l'élément dans lequel il est directement inclus dans le code) et n'est pas un pourcentage de la taille de base de l'image !



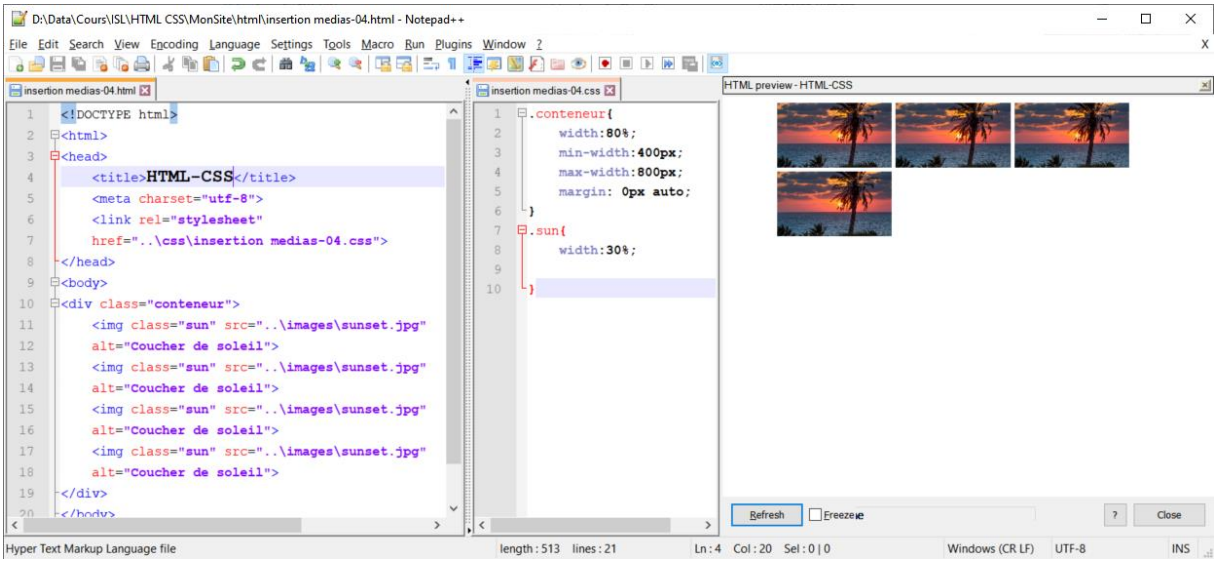
Mettre en forme nos images en CSS

Nous utiliserons de préférence le CSS pour mettre en forme nos images, que ce soit pour les redimensionner ou pour les afficher d'une façon ou d'une autre ou à un endroit ou à un autre dans nos pages.

Redimensionner une image en CSS

Pour redimensionner une image en CSS, nous allons utiliser les propriétés width (pour la largeur) et height (pour la hauteur).

Ici, nous ne préciserons généralement que la valeur de l'une de ces deux propriétés. En procédant comme cela, l'autre dimension sera calculée automatiquement afin de ne pas casser le rapport largeur/hauteur de base de l'image.

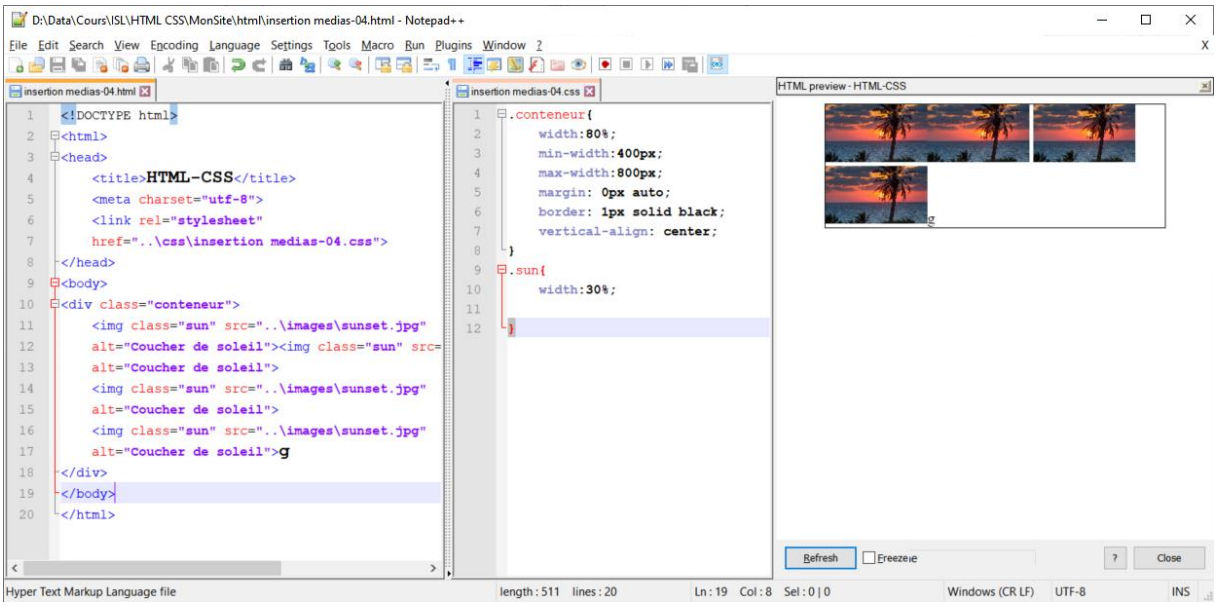


Ici, je me contente d'insérer 4 fois la même image, ce qui est tout à fait autorisé puis je redimensionne mes images en CSS avec la propriété width, tout simplement.

Notez que des espaces se créent automatiquement entre chacune des images, à droite et en dessous. Cela est dû au fait que nos images sont des éléments de type inline.

L'espace à droite provient de mon code : les éléments inline vont conserver une espace s'il y en a au moins une dans le code. C'est le cas dans mon exemple puisque je suis allé à la ligne dans mon code à chaque nouvelle déclaration d'un élément img. Pour annuler cela, on peut par exemple faire flotter nos images ou les déclarer à la suite dans le code.

L'espace en bas provient du fait que les images sont alignées par défaut sur la baseline, c'est-à-dire sur une ligne imaginaire sur laquelle sont alignés les différents caractères de texte. Les navigateurs ajoutent toujours par défaut une espace sous la baseline pour laisser de la place pour les caractères qui passent dessous comme les « g », « p » ou « y » par exemple. Pour annuler cela, on peut utiliser la propriété vertical-align et définir un autre type d'alignement vertical.

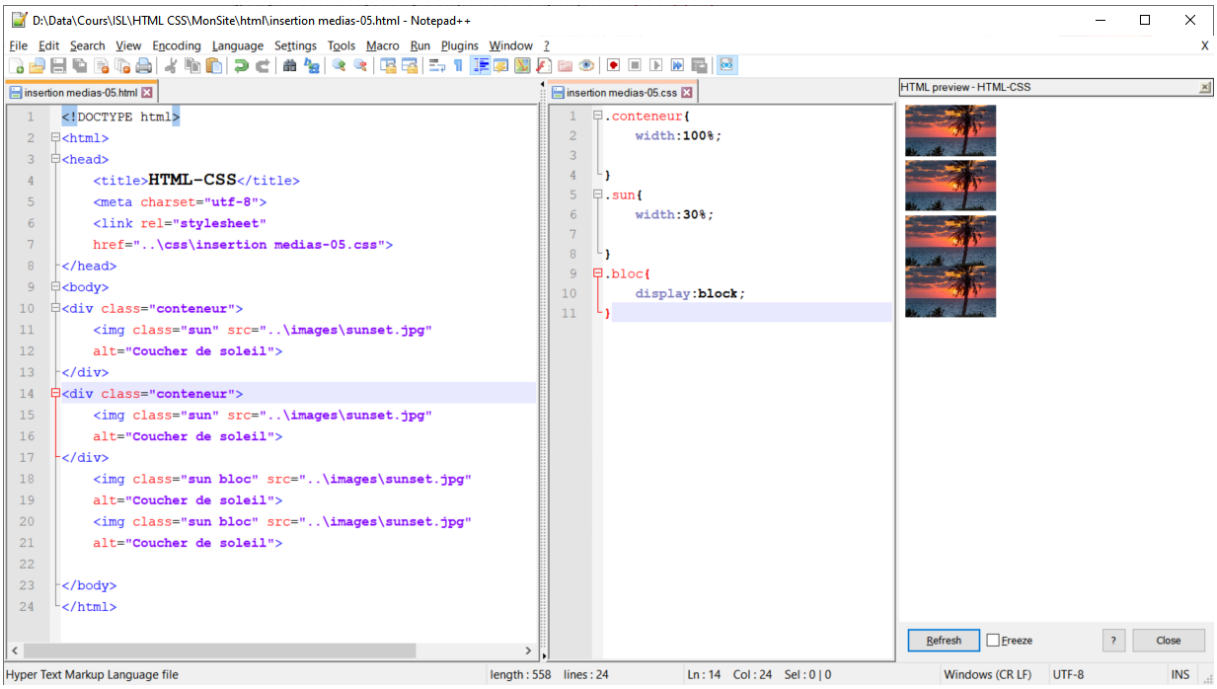


Définir le type d'affichage et le positionnement des images en CSS

La plupart des navigateurs appliquent par défaut un display : inline aux images même si certains d'entre eux préfèrent appliquer un display : inline-block.

Ces deux types d'affichage font que nos images vont venir se positionner en ligne et se placer à côté d'autres contenus si elles en ont la place.

Nous avons différents moyens pour annuler ce comportement et faire que nos images s'affichent sur leur propre ligne. Les deux solutions les plus évidentes vont être soit d'enfermer chaque image à l'intérieur d'un élément de type block comme un élément p ou div par exemple ou d'appliquer directement un display : block en CSS à nos images.



Poids des images et performances

Au début de cette leçon, j'ai beaucoup parlé de « poids des images ». Toute image possède un poids généralement exprimé de kilo-octets. Plus une image est lourde, plus elle va demander de ressources de la part du serveur et du navigateur pour être chargée.

En effet, lors de la mise en ligne de votre site web, vous allez louer un espace serveur chez un hébergeur afin d'y envoyer (« héberger ») toute l'architecture de votre site.

Le problème est que vous disposez d'un espace serveur et d'un débit limités. Il vous faudra donc déjà faire attention à ne pas le saturer avec des images inutilement lourdes.

Ensuite, selon la qualité de la connexion de vos utilisateurs et du navigateur utilisé, certaines images, si elles sont trop lourdes, risquent de mettre longtemps à s'afficher complètement.

Cela aura un effet négatif sur votre site puisque des visiteurs vont le quitter plutôt que de patienter. Cette problématique est véritablement à considérer aujourd'hui avec l'essor de l'Internet mobile car les mobiles disposent de moins de puissance que les ordinateurs et également d'une connexion plus lente.

Pour conserver des poids d'images minimaux tout en vous assurant des images d'une qualité convenable, on pensera déjà à les enregistrer au bon format et également à les recadrer à la bonne taille avant de les envoyer sur votre serveur. Ainsi, elles consommeront moins de ressources lors de leur affichage.

Media et sémantique : les éléments figure et figcaption

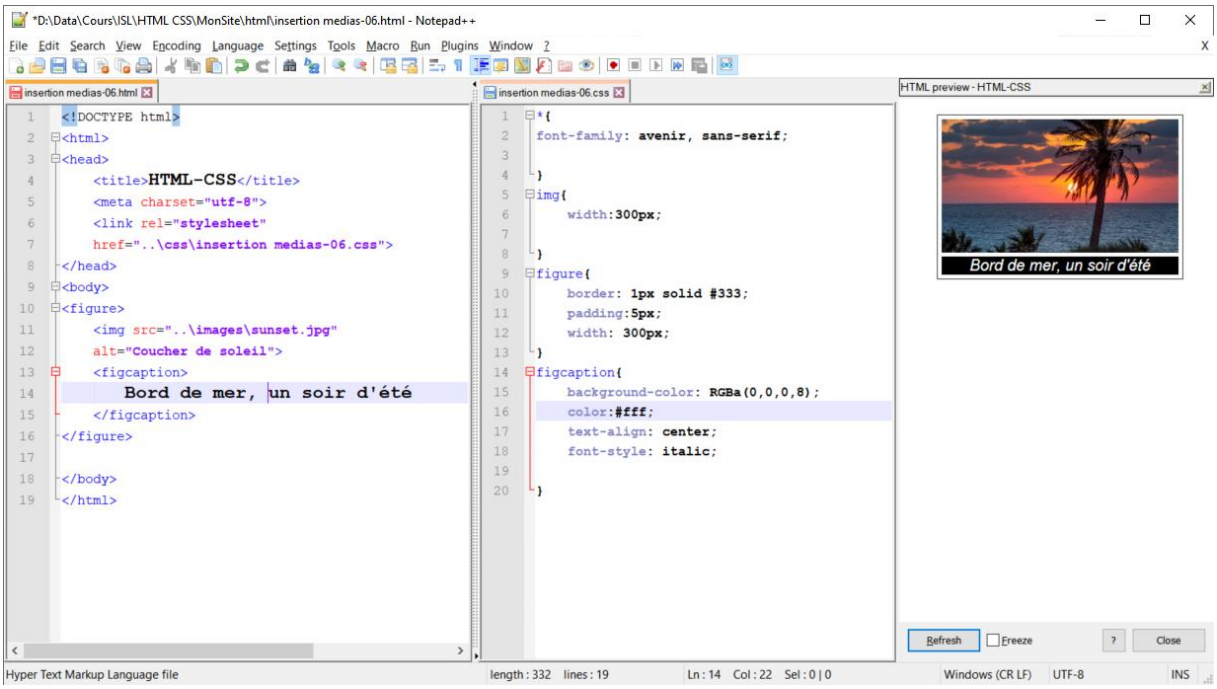
Les éléments figure et figcaption vont nous aider à marquer sémantiquement du contenu média comme des images, de l'audio ou des vidéos.

En effet, il est très difficile pour un navigateur de savoir « de quoi parle » ou « ce que représente » votre image, votre fichier audio ou vidéo sans plus de détail.
On va utiliser l'élément HTML figure pour indiquer qu'une image, une piste audio ou vidéo n'est pas strictement décorative, mais sert à la compréhension générale de notre page web. On n'utilisera donc pas cet élément si nos contenus ne sont là que pour habiller la page.

Notez qu'un élément figure peut englober plusieurs images ou d'autres types de contenus média (audio, vidéo, etc.).

On va ensuite utiliser l'élément figcaption à l'intérieur de figure pour accoler une légende au contenu de notre élément figure. Cette légende va être utile pour les personnes souffrant de déficiences et qui ne peuvent pas comprendre nos médias ainsi que pour les moteurs de recherche et navigateurs.

Par défaut, la légende va s'afficher sous les médias. Nous allons bien évidemment pouvoir mettre en forme les éléments figure et figcaption en CSS si on le souhaite.



Insérer de la musique avec l'élément HTML audio

Nous allons pouvoir intégrer un contenu audio dans un document avec l'élément audio. Dans cette nouvelle leçon, nous allons apprendre à l'utiliser et découvrir les subtilités des formats de codec audio.

Les formats audio et leur support

Il existe différents formats de fichiers audio de la même façon qu'on a pu voir précédemment qu'il existait différents formats d'image.

Jusqu'à très récemment, toutefois, l'insertion de fichiers audio était beaucoup plus complexe que l'insertion d'images pour la raison qu'aucun format de fichier audio n'était universellement reconnu : chaque navigateur possédait sa liste de formats audio qu'il était capable de lire.

La raison ici était une question de brevets déposés sur des éléments servant à constituer les différents formats audio ou sur les formats audio en soi, comme le brevet sur le MP3 effectif jusqu'en 2017. Ainsi, les navigateurs étaient soit obligés de posséder des licences soit de payer des royalties pour pouvoir utiliser tel ou tel format audio.

La situation s'est récemment beaucoup améliorée sur ce point-là et aujourd'hui certains formats audio sont bien reconnus par la plupart des navigateurs.

Cependant, comme certains problèmes peuvent toujours subsister, je vais vous montrer dans ce cours la méthode « historique » d'insertion d'audio qui consiste à contourner le problème en proposant plusieurs fichiers audio source de différents formats afin que chaque navigateur puisse choisir celui qui lui convient.

Les formats audio les plus courants généralement utilisés vont être les suivants :

- Le format audio Vorbis du conteneur WebM audio/webm ;
- Le format audio Vorbis du conteneur Ogg audio/ogg ;
- Le format audio MP3 audio/mpeg ;
- Le format audio PCM du conteneur WAVE audio/wave.

Pour information, voici les supports de ces différents formats par les navigateurs les plus utilisés dans leur version la plus récente :

Navigateur	**Vorbis**	**MP3**	**PCM (WAVE)**
Chrome	Supporté	Supporté	Supporté

Navigateur	Vorbis	MP3	PCM (WAVE)
Safari	Non supporté	Supporté	Supporté
Firefox	Supporté	Supporté	Supporté
Edge	Supporté	Supporté	Supporté
Opera	Supporté	Supporté	Supporté
Safari (iOS)	Non supporté	Supporté	Supporté
Android	Supporté	Supporté	Supporté
Chrome (Android)	Supporté	Supporté	Supporté

Insérer de l'audio dans un fichier HTML

Pour pouvoir insérer de l'audio dans nos pages, nous allons déjà devoir nous munir d'une piste audio qui devra être enregistrée sous différents formats.

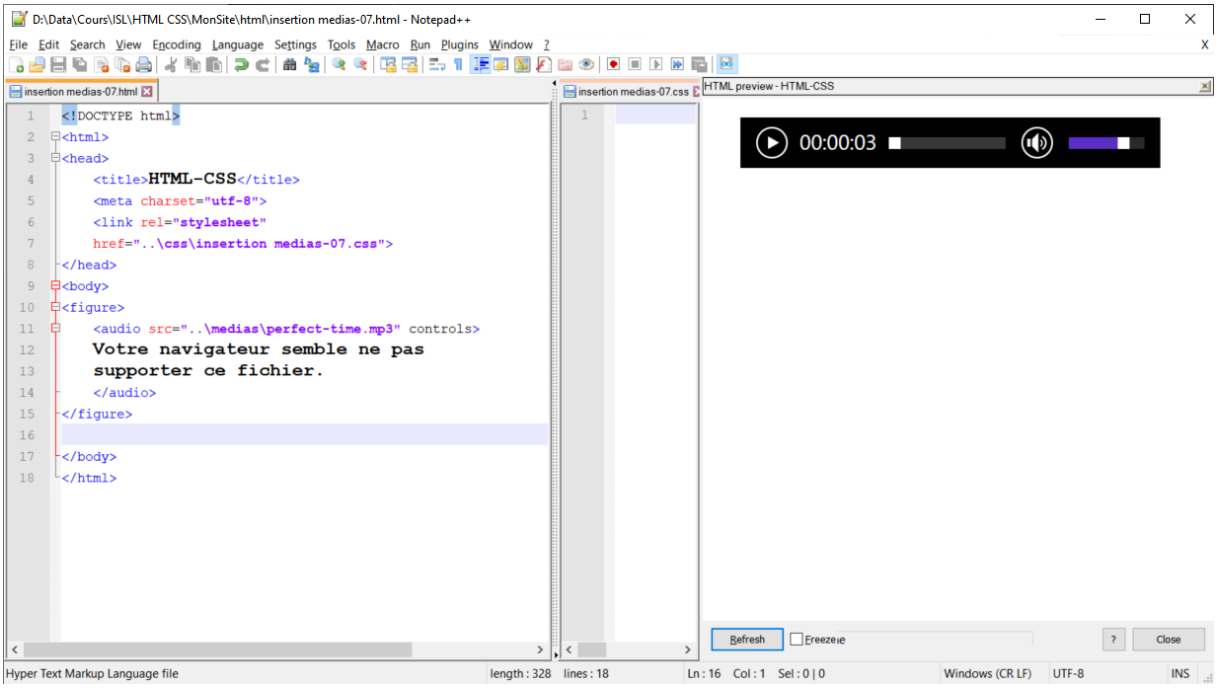
Pour ma part, je vais utiliser un fichier audio qui s'appelle « perfect-time » et qui a été enregistré sous trois formats différents : mp3, ogg et wav.

Vous pouvez trouver ces fichiers dans le répertoire medias du cours. Je vous invite à placer ces fichiers dans le même dossier que vos fichiers de code.

La façon la plus simple d'insérer un fichier audio dans un fichier va être d'utiliser un élément audio avec deux attributs src et controls.

L'attribut src va indiquer le chemin vers la ressource à intégrer tandis que l'attribut controls va permettre de faire apparaitre les contrôles fournis par le navigateur (notamment un bouton de lecture / arrêt, un bouton pour choisir le niveau sonore, etc.).

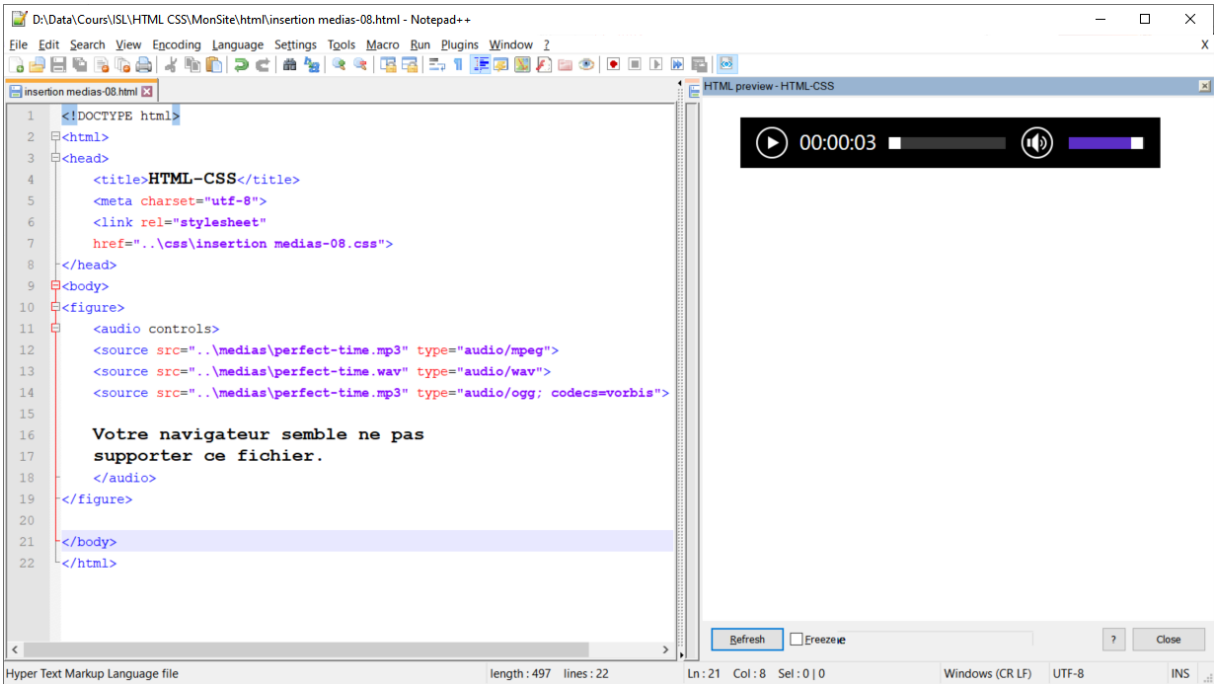
Nous allons également renseigner un texte dans l'élément audio qui pourra être affiché dans certaines anciennes versions de navigateurs dans le cas où le navigateur ne pourrait pas lire le fichier audio fourni.



Le problème en utilisant l'élément audio de cette manière est qu'on ne va pas pouvoir fournir de format audio alternatif au cas où le navigateur ne puisse pas lire le format fourni.

Nous utiliserons donc généralement plutôt l'élément audio de concert avec des éléments source qui vont nous permettre d'intégrer différents fichiers parmi lesquels le navigateur fera son choix.

Dans l'élément source, nous allons préciser l'emplacement du fichier audio dans un attribut src et allons également facultativement pouvoir ajouter un attribut type qui va nous permettre d'indiquer rapidement au navigateur le type de codec audio utilisé dans notre fichier pour que le navigateur sache immédiatement s'il peut le lire ou pas sans même avoir à le tester. Cela optimisera les performances de notre page.



Ici, le navigateur va s'arrêter dès qu'il va rencontrer un format audio qu'il sait lire et ignorer les autres formats fournis en dessous. Cette technique permet de pouvoir lire un fichier audio sur (quasiment) tous les navigateurs.

Les attributs de l'élément audio

En plus de l'attribut controls qui est obligatoire pour des raisons évidentes d'ergonomie et d'accessibilité, nous allons pouvoir indiquer d'autres attributs facultatifs à notre élément audio qui vont nous permettre de mieux contrôler comment le fichier audio doit être lu.

Ces attributs sont les suivants :

preload

L'attribut preload permet d'indiquer au navigateur si le fichier doit être préchargé ou pas. On va pouvoir lui passer l'une des valeurs suivantes :

- none : le fichier audio ne sera pas préchargé ;
- metadata : seules les méta-données seront préchargées ;
- auto : le fichier sera préchargé.

autoplay

L'attribut autoplay va nous permettre de lancer automatiquement la lecture du fichier audio dès qu'il sera chargé. Il suffit de le renseigner (même sans valeur explicite) pour que le fichier audio se lance

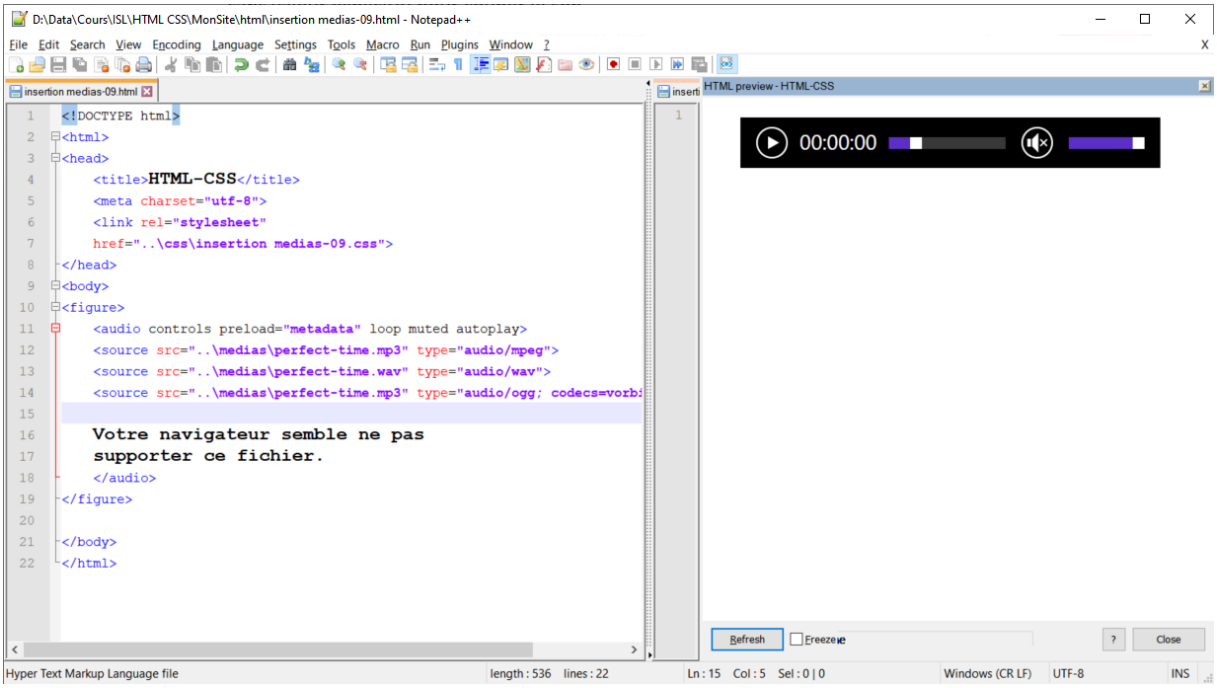
automatiquement. Notez que certains navigateurs (dont Chrome) peuvent bloquer cet attribut car celui-ci est jugé mauvais pour vos visiteurs.

loop

L'attribut loop permet de lire le fichier audio en boucle. Il suffit donc de le renseigner (même sans valeur explicite) pour que le fichier soit lu en boucle.

muted

L'attribut muted sert à définir si le son doit être initialement coupé. Il suffit de le renseigner (même sans valeur explicite) pour couper le son.



Insérer des vidéos avec l'élément HTML video

Nous allons pouvoir intégrer un contenu vidéo dans un document avec l'élément video. Cet élément fonctionne de manière analogue à l'élément audio vu précédemment.

Les formats vidéo et leur support

Un fichier vidéo est relativement plus complexe qu'un simple fichier audio puisqu'une vidéo est va être composée à la fois d'une piste audio et d'une piste vidéo. Ces deux pistes vont être de formats différents et vont être regroupées dans un conteneur qui va se charger de les faire fonctionner ensemble.

En termes de support par les navigateurs, nous allons donc nous heurter aux mêmes problèmes qu'on a déjà vu avec l'intégration d'audio et notamment aux problèmes de brevet cette fois-ci non seulement sur les codecs des pistes audios mais également sur ceux des pistes vidéo et des conteneurs.

Comme pour l'élément audio, nous indiquerons donc généralement plusieurs formats de conteneurs dans notre élément video afin de s'assurer que la vidéo puisse être lue sur la plupart des navigateurs.

Les conteneurs les plus utilisés aujourd'hui sont les suivants :

- WebM : contient de l'audio Ogg Vorbis avec de la vidéo VP8/VP9 ;
- MP4 : contient de l'audio AAC ou MP3 en audio avec de la vidéo H.264.

Notez par ailleurs que le MP4-HEVC utilisant le format vidéo H.265, annoncé pour la première fois en 2013, est censé succéder au MP4 utilisant de la vidéo H.264.

Le format AV1, annoncé en 2018 est lui censé succéder au WebM et concurrencer le HEVC. Ces deux formats n'ont cependant que peu de support par les navigateurs aujourd'hui : le HEVC n'est supporté que par Safari tandis que l'AV1 n'est supporté que par les dernières versions de Chrome (desktop).

Pour information, voici les supports de ces différents formats par les navigateurs les plus utilisés dans leur version la plus récente :

Navigateur	**MP4/H.264**	**WebM**
Chrome	Supporté	Supporté
Safari	Supporté	Non supporté
Firefox	Supporté	Supporté
Edge	Supporté	Supporté partiellement
Opera	Supporté	Supporté
Safari (iOS)	Supporté	Non supporté
Android	Supporté	Supporté
Chrome (Android)	Supporté	Supporté

Insérer de la vidéo dans un fichier HTML

Pour pouvoir insérer de la vidéo dans nos pages, nous allons déjà devoir nous munir d'un fichier vidéo disponible sous différents formats.

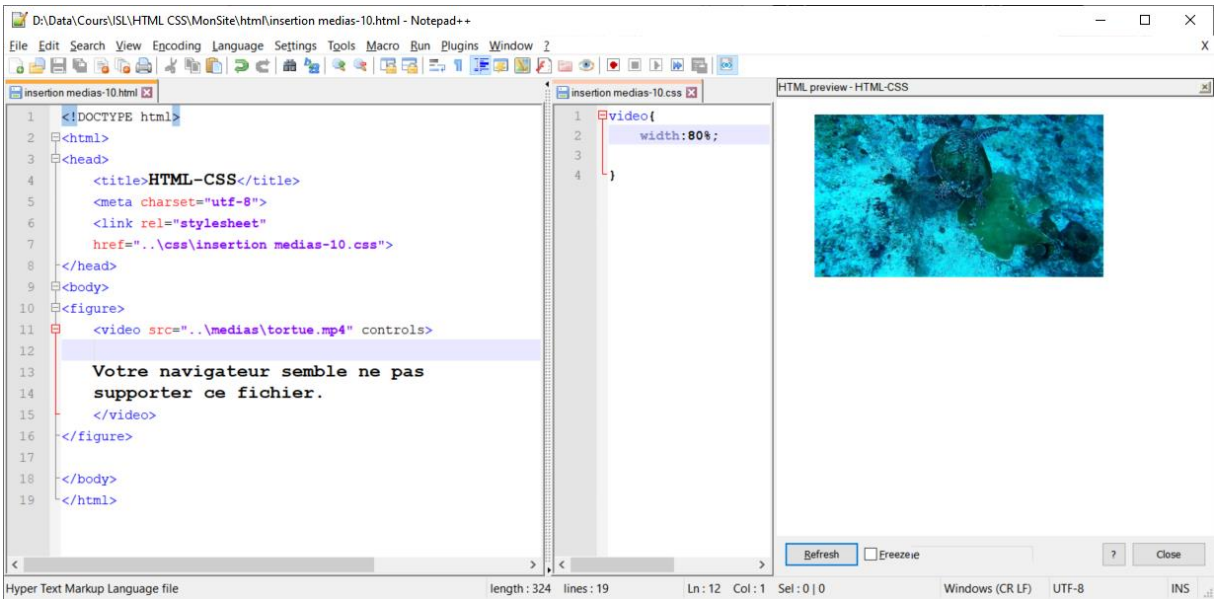
Ici, je vais utiliser un fichier vidéo qui s'appelle « tortue » qui est originellement un fichier MP4 et que j'ai converti également en WebM.

Les fichiers se trouvent dans le répertoire médias du support à ce cours. Je vous invite à placer ces fichiers dans le même dossier que vos fichiers de code.

La façon la plus simple d'insérer un fichier vidéo dans une page HTML va être d'utiliser un élément video avec deux attributs src et controls, exactement de la même façon que pour l'élément audio

L'attribut src va indiquer le chemin vers la ressource à intégrer tandis que l'attribut controls va permettre de faire apparaitre les contrôles fournis par le navigateur (notamment un bouton de lecture / arrêt, un bouton pour choisir le niveau sonore, etc.).

Nous allons également renseigner un texte dans l'élément video qui pourra être affiché dans certaines anciennes versions de navigateurs dans le cas où le navigateur ne pourrait pas lire le fichier vidéo fourni.

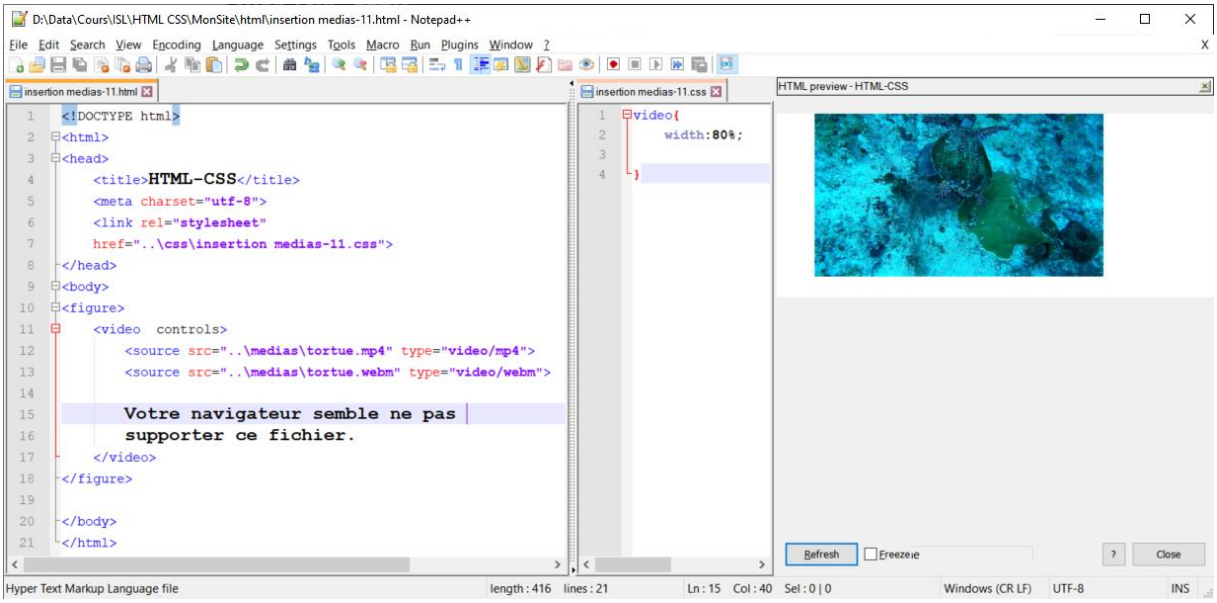


En pratique, toutefois, cette façon de faire n'est pas optimale puisqu'elle ne nous permet pas de proposer plusieurs formats de fichiers différents pour les navigateurs.

Nous utiliserons donc généralement plutôt l'élément video de concert avec des éléments source qui vont nous permettre d'intégrer différents formats de fichier parmi lesquels le navigateur fera son choix.

Dans l'élément source, nous allons devoir inclure un attribut src qui va nous permettre d'indiquer l'emplacement du fichier vidéo.

Nous allons également pouvoir, de manière facultative, ajouter un attribut type qui va nous permettre d'indiquer rapidement au navigateur le type de codecs utilisé dans notre fichier pour que le navigateur sache immédiatement s'il peut le lire ou pas sans même avoir à le tester. Cela optimisera les performances de notre page.



Ici, le navigateur va s'arrêter dès qu'il va rencontrer un format qu'il sait lire et ignorer les autres formats fournis en dessous.

Les attributs de l'élément video

L'élément video possède un attribut strictement obligatoire qui est l'attribut controls. Cet attribut permet d'afficher les contrôles de base pour l'utilisateur comme la lecture, la pause, le choix du volume, etc. Les différentes options de contrôle vont être fournies par le navigateur.

En plus de cet attribut controls, on va pouvoir passer d'autres attributs facultatifs à l'élément video afin de contrôler comment le fichier doit être lu et notamment :

preload

L'attribut preload permet d'indiquer au navigateur si le fichier doit être préchargé ou pas. On va pouvoir lui passer l'une des valeurs suivantes :

- none : le fichier vidéo ne sera pas préchargé ;
- metadata : seules les méta-données seront préchargées ;
- auto : le fichier sera préchargé.

autoplay

L'attribut autoplay va nous permettre de lancer automatiquement la lecture du fichier vidéo dès qu'il sera chargé. Il suffit de le renseigner (même sans valeur explicite) pour que le fichier vidéo se lance automatiquement. Notez que certains navigateurs (dont Chrome) peuvent bloquer cet attribut car celui-ci est jugé mauvais pour vos visiteurs.

loop

L'attribut loop permet de lire le fichier vidéo en boucle. Il suffit donc de le renseigner (même sans valeur explicite) pour que le fichier soit lu en boucle.

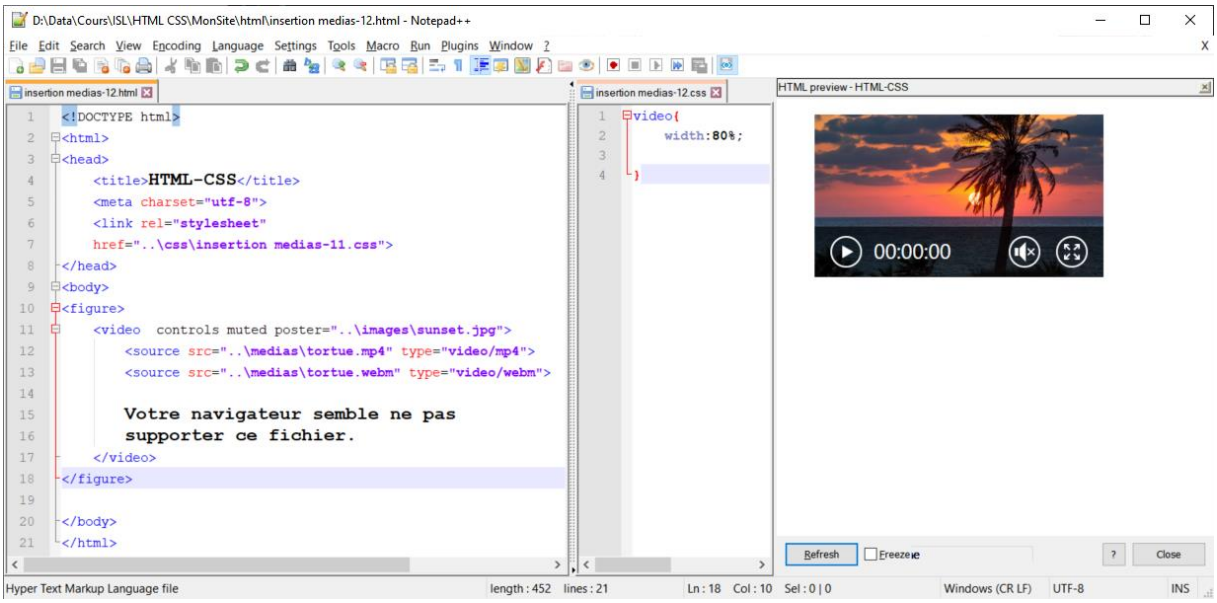
muted

L'attribut muted sert à définir si le son doit être initialement coupé. Il suffit de le renseigner (même sans valeur explicite) pour couper le son.

poster

L'attribut poster va nous permettre de renseigner l'adresse d'une image qui devra être utilisée comme image d'illustration de fond de la vidéo avant que celle-ci ne soit chargée et lancée.

En plus de ces attributs, notez que nous allons pouvoir gérer la taille de nos vidéos grâce aux attributs HTML width et height ou idéalement avec les propriétés CSS du même nom. Pour des raisons évidentes de ratio, nous ne préciserons toujours que l'une de ces deux propriétés (généralement width) afin que la deuxième dimension soit calculée automatiquement.



Ajouter des sous-titres à une vidéo

L'importance de l'ajout de sous-titres

Il est toujours très important lorsqu'on code en HTML de penser en termes d'accessibilité. L'idée ici est que votre code devrait toujours être accessible à tous, c'est-à-dire que vos pages devraient toujours pouvoir être comprises d'une façon ou d'une autre par chacun de vos visiteurs.

C'est là tout l'intérêt d'ajouter des descriptions sur nos images avec l'attribut alt par exemple. C'est la même logique qui se cache derrière l'ajout de sous-titres sur nos vidéos.

Pour ajouter des sous-titres à nos vidéos, nous allons déjà devoir les écrire dans un fichier séparé en respectant une certaine syntaxe. Le fichier sera enregistré sous le format WebVTT pour « Web Video Text Tracks ». Nous allons ensuite pouvoir intégrer ces sous titres dans notre vidéo en utilisant l'élément track.

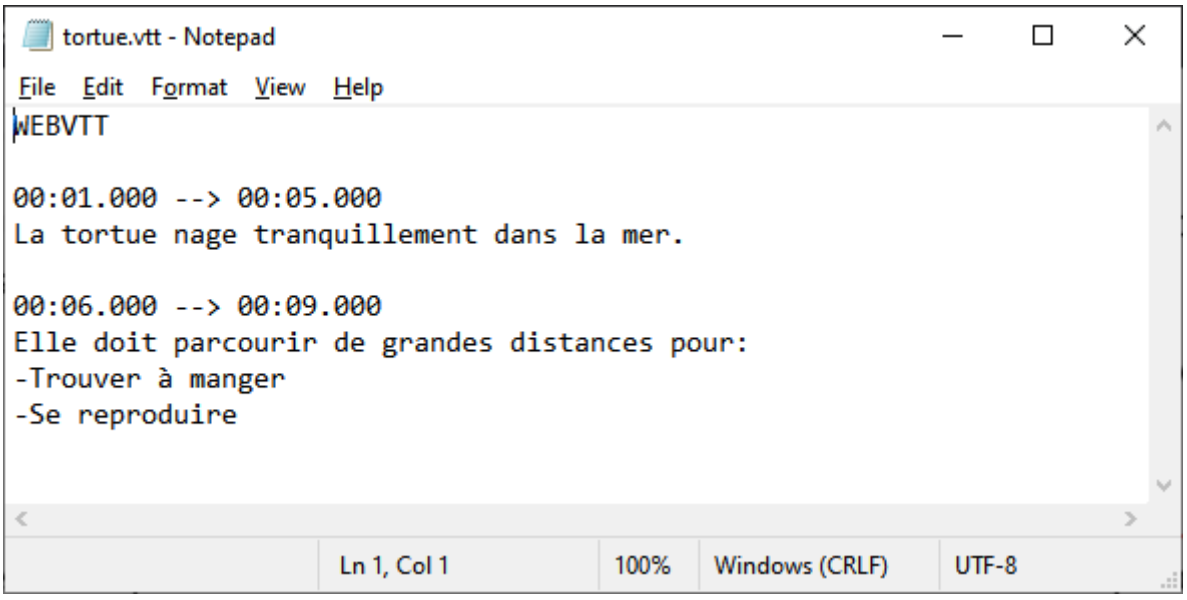
L'écriture des sous-titres au format WebVTT

Nous allons écrire les sous-titres pour notre vidéo dans un fichier séparé au format WebVTT. Pour faire cela, nous pouvons utiliser notre éditeur ou un programme de bloc note et enregistrer le fichier avec l'extension .vtt.

Un fichier WebVTT doit respecter une certaine syntaxe pour pouvoir être lu convenablement :

- Tout fichier WebVTT doit commencer par le mot WEBVTT, qui peut être éventuellement suivi sur la même ligne par un texte d'en-tête ;
- La ligne sous la mention WEBVTT doit être vide ;
- Ensuite, nous allons placer des indications de temps sous la forme 00:01.000 --> 00:05.000 suivies à chaque fois sur la ligne du dessous du sous-titre à afficher durant la période décrite.
- Il est préférable de le sauvegarder en utf-8 ;

Voici à quoi va pouvoir ressembler notre fichier :



L'intégration des sous-titres avec l'élément track et ses attributs

Une fois que nous avons écrit nos sous-titres et qu'on les a enregistrés au bon format, il va falloir les intégrer dans nos vidéos.

Pour cela, nous allons utiliser un élément track que nous allons placer à l'intérieur de notre élément video.

Cet élément track va devoir être obligatoirement accompagné d'un attribut src qui va nous permettre d'indiquer l'adresse du fichier contenant les sous titres ainsi que d'un attribut srclang qui va nous permettre de préciser la langue utilisée (fr pour français, en pour anglais, es pour espagnol, etc.).

Nous allons également pouvoir lui passer les attributs facultatifs suivants :

default

Indique qu'on souhaite utiliser ce fichier de sous-titres par défaut (utile dans le cas où on dispose de plusieurs fichiers de sous-titrage pour différentes langues) ;

kind

Cet attribut nous permet de préciser la nature du texte ajouté à la vidéo. Sa valeur par défaut est subtitles (« sous-titres ») mais si une mauvaise valeur est renseignée c'est la valeur metadata qui sera utilisée. Les valeurs possibles sont :

- subtitles : Le texte passé correspond à des sous-titres. Les sous-titres sont pertinents pour des utilisateurs qui ne peuvent pas comprendre le contenu (cas des utilisateurs qui regardent des vidéos dans une autre langue que la leur) et ont également pour rôle de donner des informations contextuelles ;
- captions : Le texte est une traduction de l'audio de la vidéo. Le type captions est pertinent pour des malentendants ou lorsque le son est désactivé par exemple ;
- descriptions : Le texte est une description du contenu vidéo. Ce type est adapté pour les personnes malvoyantes ;
- chapters : Ce type représente les titres des chapitres lorsque l'utilisateur navigue au sein du media ;
- metadata : Le texte sera utilisé par des scripts mais sera invisible pour l'utilisateur.

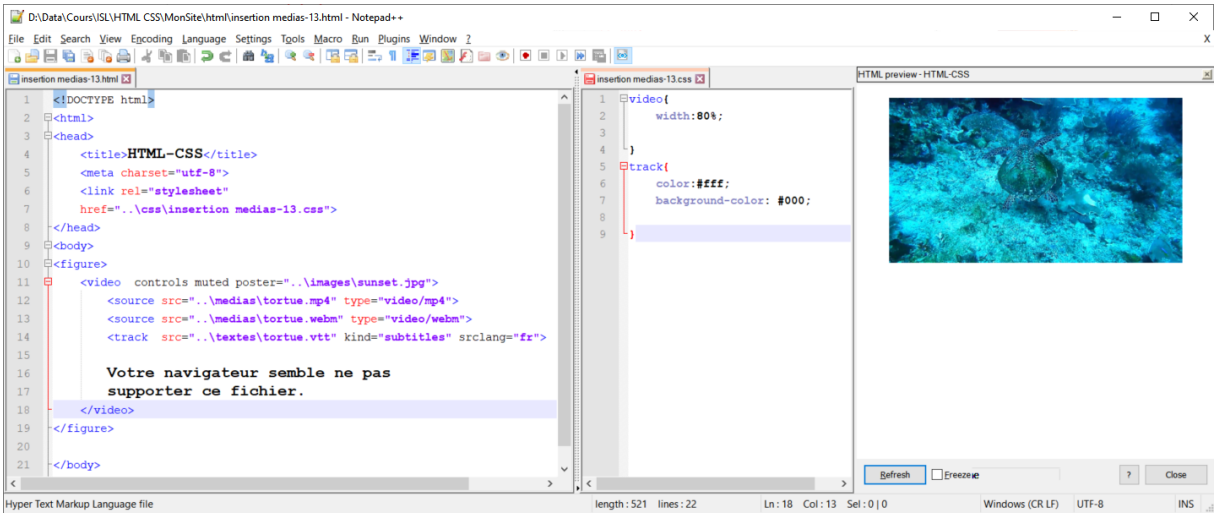
label

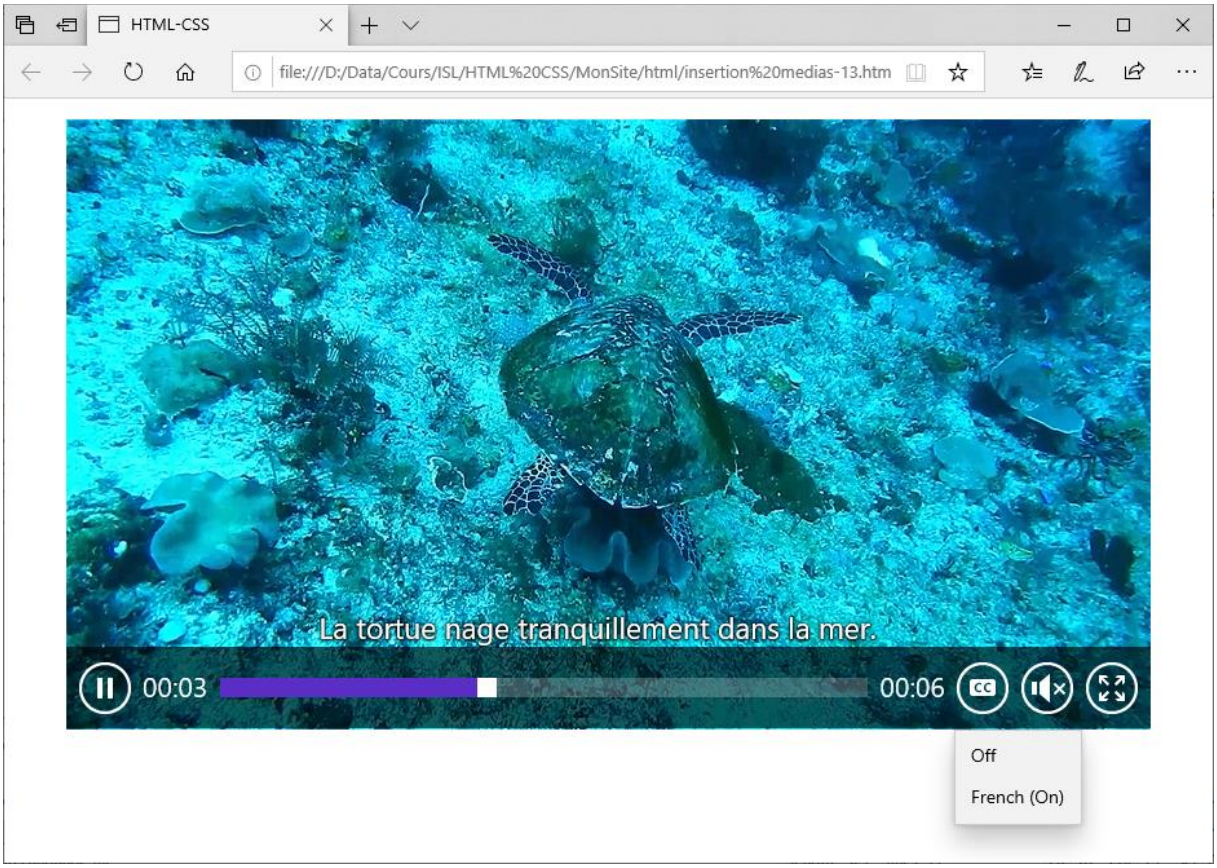
Permet d'indiquer la langue de chaque sous-titres comme « Français », « English » ou « Español. La liste de langues sera visible pour l'utilisateur et lui permettra de choisir le sous-titrage de son choix ;

Notez que dans le cas où nous avons plusieurs langues de sous-titrage à proposer, il faudra enregistrer chaque sous-titres dans un fichier .vtt différent et utiliser un nouvel élément track pour chacun d'entre eux.

Nous pouvons également proposer plusieurs fichiers de texte de type (kind) différents dans un même langage.

Il ne nous reste plus qu'à définir les styles de notre texte en ciblant l'élément track en CSS.





Notez ici que certains navigateurs ne vont pas afficher les sous-titres pour des fichiers en local (vs en production ou sur serveur). Cela va être le cas pour Chrome notamment.

L'élément HTML iframe

Nous savons désormais comment ajouter des images, de l'audio ou des vidéos dans nos pages en intégrant nos propres médias. Il nous reste donc une chose à voir : *comment intégrer une autre page web dans un fichier HTML*.

L'élément iframe, qui est l'objet de cette leçon, va nous permettre de faire cela et d'intégrer, entre autres, des vidéos YouTube directement dans nos pages.

L'élément iframe et ses attributs

L'élément iframe va nous permettre d'intégrer un fichier web entier dans un autre. Nous allons notamment utiliser cet élément pour intégrer des vidéos YouTube dans nos pages web ou des cartes Google Maps.

Pourquoi intégrer des contenus provenant d'autres sites web plutôt qu'héberger nos propres contenus ? Il y a deux raisons principales.

La première concerne l'économie des ressources : en hébergeant une vidéo sur YouTube, par exemple, et en l'intégrant après sur notre site on ne va pas utiliser notre espace serveur ni notre bande passante mais ceux de YouTube, ce qui peut se transformer en des économies substantielles de notre côté.

La deuxième raison est purement technique : il est souvent plus facile et sûr d'utiliser les services créés par d'autres entités reconnues plutôt que de créer nous-mêmes ces choses. Cela est le cas pour les cartes Google Maps par exemple : il est hors de question de se procurer une carte du monde détaillée et de recréer l'outil de notre côté pour afficher une carte quand on peut utiliser celles de Google.

De même pour l'intégration de vidéos YouTube : en passant par YouTube, on n'a pas à se soucier de tous les problèmes de compatibilité des différents codecs avec les différentes versions des navigateurs puisque YouTube se charge de faire ce travail de son côté lorsqu'on télécharge une vidéo sur la plateforme.

Les attributs généralement précisés lors de l'utilisation de l'élément iframe sont les suivants :

src

Cet attribut va nous permettre d'indiquer l'adresse du document à intégrer

width **et** height

Ces attributs permettent de définir les dimensions de notre élément iframe. On pourra les utiliser dans le cas où nous ne pouvons pas modifier les dimensions de l'iframe en CSS.

allow

Cet attribut permet de définir une politique de fonctionnalité pour notre iframe. Le terme « politique de fonctionnalité » désigne le fait de choisir quelles fonctionnalités de l'iframe activer ou désactiver. Cet attribut est utilisé pour renforcer la sécurité du code.

sandbox

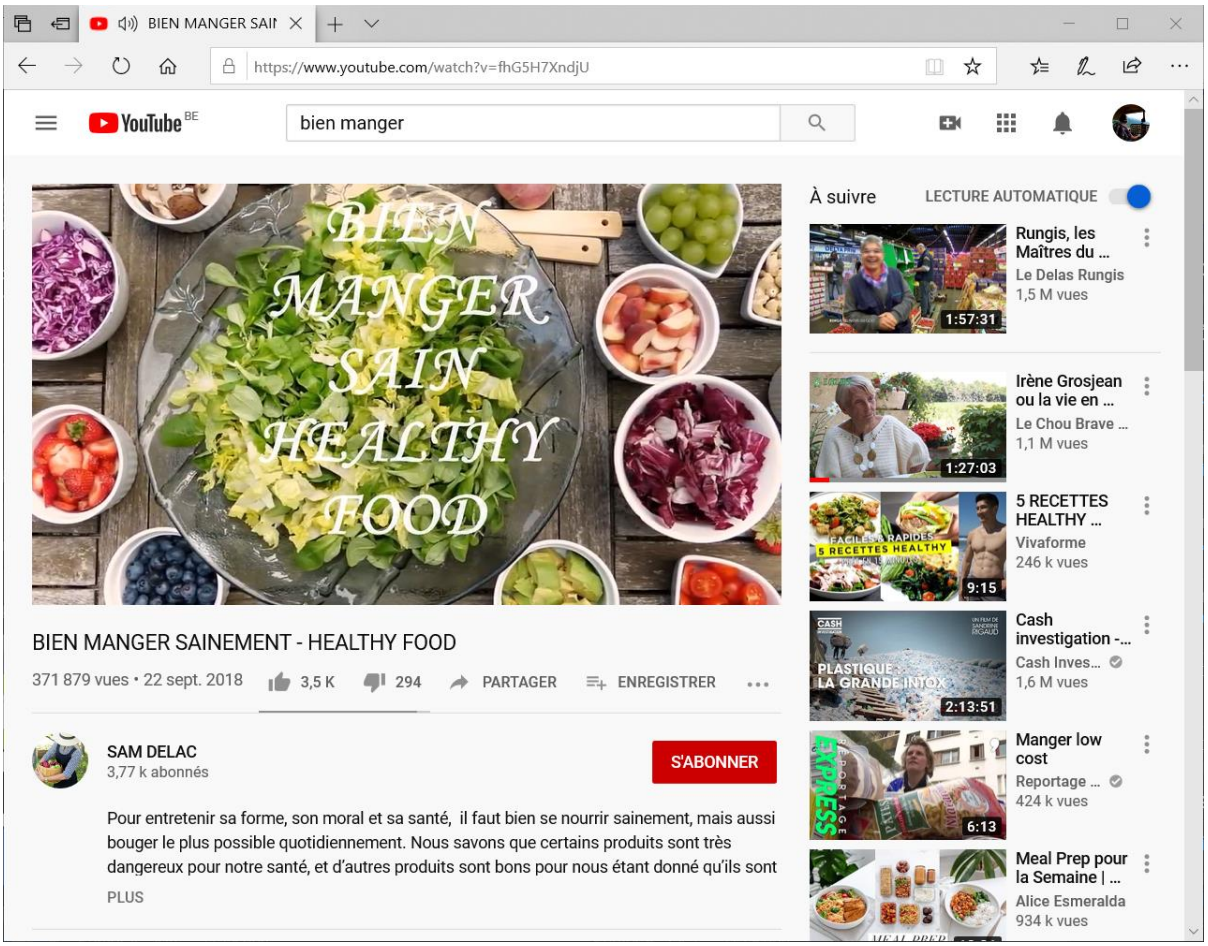
Cet attribut est relativement récent. Il nous permet de limiter les permissions de notre iframe c'est-à-dire de limiter ses possibilités. Cet attribut est utilisé pour renforcer la sécurité de notre code.

Intégrer une vidéo YouTube avec l'élément iframe

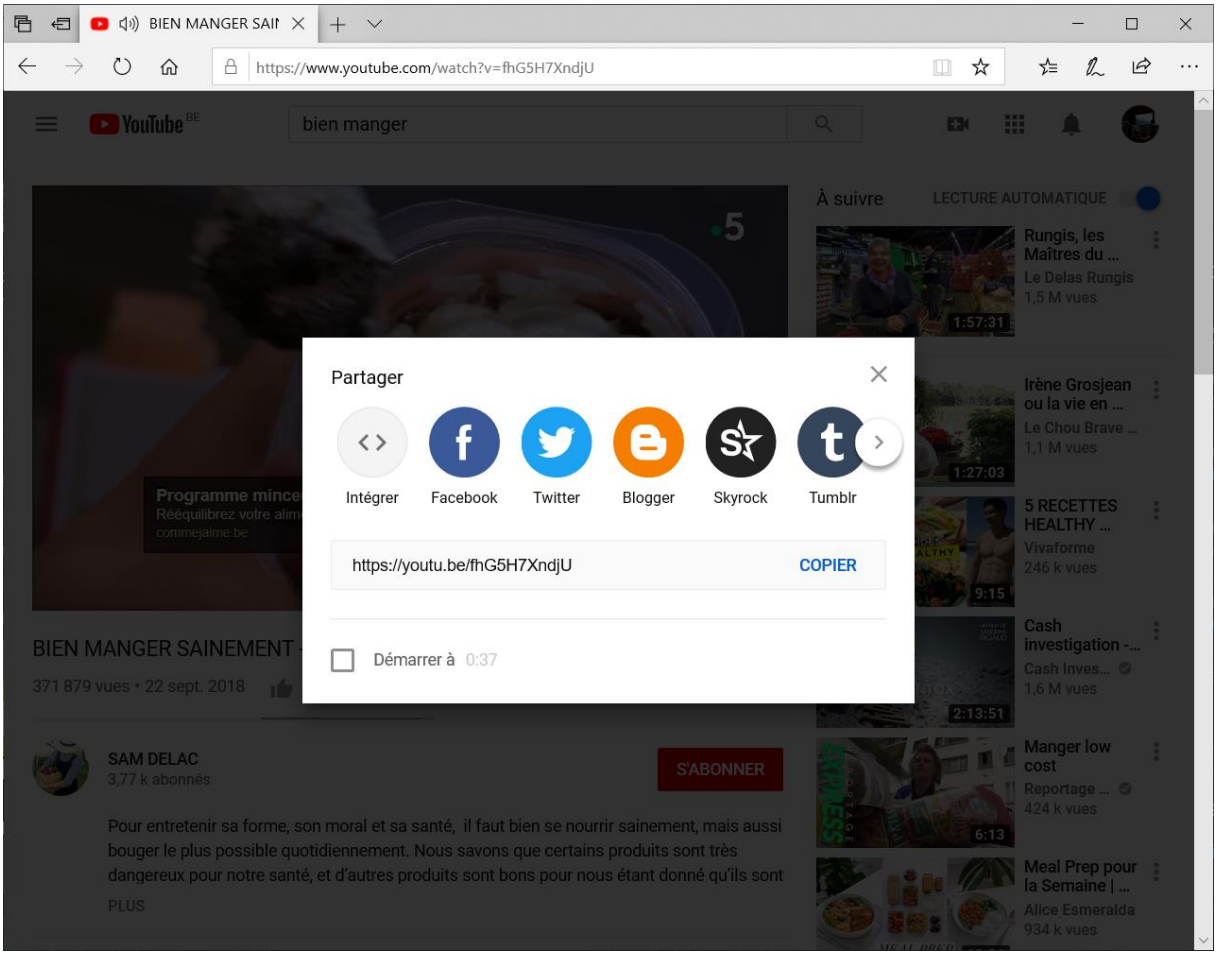
Voyons immédiatement comment utiliser l'élément iframe en pratique en intégrant une vidéo YouTube dans notre page.

Pour cela, utilisons une vidéo au hasard de YouTube..

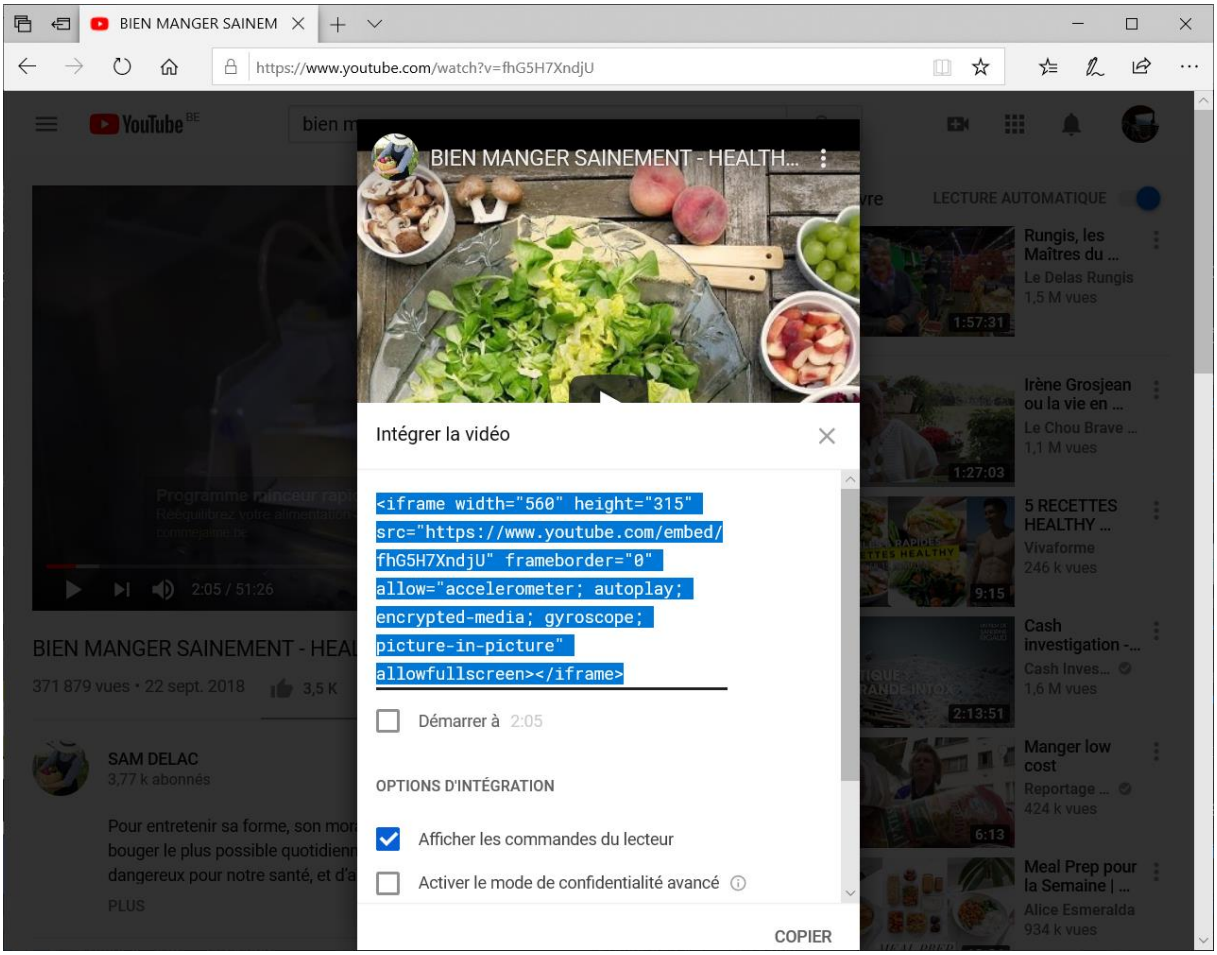
Ici, nous allons cliquer sur le bouton « partager » sous la vidéo…



…puis sur « intégrer » (« embed » en anglais)…



… et nous allons finalement copier le code d'intégration qui est bien un élément iframe



Nous n'avons ensuite plus qu'à coller ce code dans notre page HTML pour intégrer notre vidéo !

D:\Data\Cours\ISL\HTML CSS\MonSite\html\insertion medias-14.html - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

insertion medias-14.html

```
<!DOCTYPE html>
<html>
<head>
    <title>HTML-CSS</title>
    <meta charset="utf-8">
    <link rel="stylesheet"
    href="..\css\insertion medias-14.css">
</head>
<body>
<figure>
    <iframe width="560" height="315"
        src="https://www.youtube.com/embed/fhG5H7XndjU"
        frameborder="0" allow="accelerometer; autoplay;
        encrypted-media; gyroscope; picture-in-picture"
        allowfullscreen>
    </iframe>
</figure>

</body>
</html>
```

insertion medias-14.css

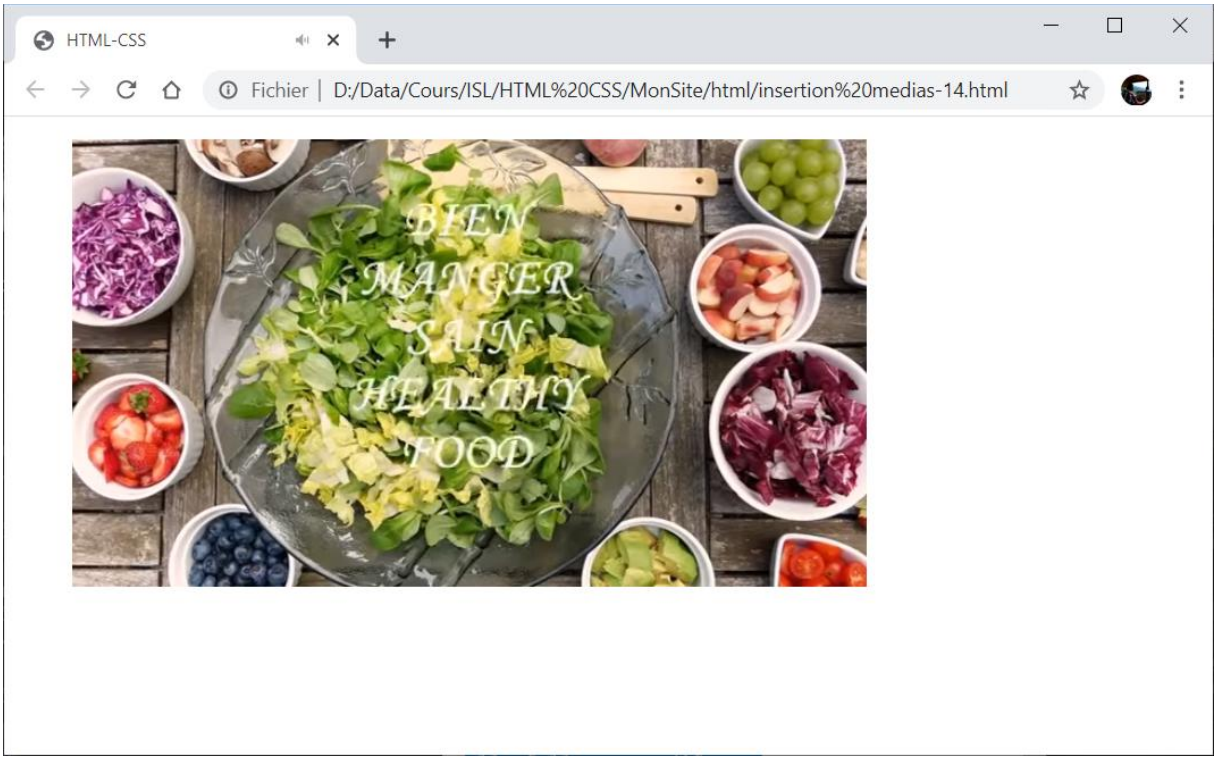
```
video{
    width:80%;

}
```

HTML previ...

Close

Hyper Text Markup Language fi length : 431 lines : 20 Ln : 8 Col : 8 Sel : 0 | 0 Windows (CR LF) UTF-8 INS



iframes et APIs : le cas de l'intégration de cartes Google Maps

L'intégration de cartes Google Maps va être plus complexe ou tout au moins plus longue que celles de vidéos YouTube.

Cela est dû au fait que l'API Google Maps est plus restrictive que celle de YouTube. Une API « Application Programming Interface » ou « Interface de Programmation Applicative » en français est une interface généralement complexe qui va permettre à différentes applications de communiquer entre elles et de s'échanger des données.

L'intégration d'iframes va souvent de pair avec l'utilisation d'API. Pour nous, vous pouvez considérer qu'une API nous permet d'accéder à une application sans avoir à se soucier des mécanismes d'arrière-plan permettant de la faire fonctionner.

D'un autre côté, les API servent également à limiter les intégrations « sauvages » en demandant à l'utilisateur de s'enregistrer et de générer une clef pour pouvoir utiliser une application en question sur son site.

Cela va être le cas avec l'intégration de cartes Google Maps : nous allons devoir passer une clef de sécurité (API Key ou API Token) en attribut de notre iframe pour pouvoir s'identifier auprès de Google et pouvoir utiliser le service de cartes.

Pour obtenir une clef d'API, il suffit généralement de s'inscrire sur le site proposant l'API. Dans le cas de Google Maps, cependant, cela n'est pas suffisant puisque l'utilisation de l'API Google Maps est payante depuis cette année. Il faudra donc également être prêt à payer pour intégrer des cartes Google Maps.

Pour cette raison, je ne vous présenterai pas comment intégrer des cartes Google Maps dans ce cours mais cet exemple me semblait intéressant car il m'a permis d'introduire le concept d'API et également de vous expliquer les différentes contraintes auxquelles nous pouvons être soumis en tant que développeurs.

iframe, sécurité et performance

Il convient de faire preuve de prudence lors de l'utilisation d'un élément iframe comme pour tout autre élément faisant appel à des données externes. En effet, ce type d'élément peut potentiellement se transformer en point d'entrée pour des personnes mal intentionnées (des « hackeurs ») qui peuvent exploiter certaines failles pour dérober des informations sensibles à nos utilisateurs ou pour rendre notre site non opérationnel.

Les problématiques de sécurité sont très complexes et ne sont pas à la portée de développeurs débutants et c'est la raison pour laquelle je n'en parlerai pas en détail dans ce cours. En effet, pour en comprendre les tenants et les aboutissants, il faut comprendre comment fonctionnent les sites Internet, les réseaux, ainsi que connaitre les langages comme le JavaScript et comprendre l'interaction entre les différents langages.

Je resterai donc très superficiel ici et vais simplement vous conseiller d'ajouter des attributs allow et sandbox dans vos éléments iframe qui vont permettre de très largement limiter les menaces entrantes.

Chacun de ces deux attributs peut prendre de nombreuses valeurs. En les utilisant sans valeur, nous désactivons la plupart des fonctionnalités liées aux iframes. A chaque fois qu'on ajoute une valeur dans ces attributs, on indique qu'on souhaite réactiver la fonctionnalité liée à la valeur en question.

Pour information, on va pouvoir passer les valeurs suivantes (et donc activer les fonctionnalités correspondantes) à allow :

- Accelerometer ;
- Ambient light sensor ;
- Autoplay ;
- Camera ;
- Encrypted media ;
- Fullscreen ;
- Geolocation ;
- Gyroscope ;
- Lazyload ;
- Microphone ;
- Midi ;
- PaymentRequest ;
- Picture-in-picture ;
- Speaker ;
- USB ;
- VR / XR.

De même, l'attribut sandbox va nous permettre d'activer les fonctionnalités suivantes :

- allow-scripts : l'iframe peut exécuter des scripts ;
- allow-forms : l'envoi de formulaire dans l'iframe est permis ;
- allow-popups : l'iframe peut ouvrir des popup (fenêtres contextuelles) ;
- allow-modals : l'iframe peut ouvrir des fenêtres modales ;
- allow-orientation-lock ;
- allow-pointer-lock ;
- allow-popups-to-escape-sandbox ;
- allow-presentation ;

- allow-same-origin ;
- allow-top-navigation ;
- allow-top-navigation-by-user-activation.

Un autre souci lié à l'utilisation d'iframe est l'intégration du contenu de notre site sur d'autres sites. Nous pouvons appeler ça des menaces « sortantes » pour les distinguer des précédentes.

Les intentions liées à ce comportement sont généralement malveillantes : vol de contenu, tentative d'imitation de notre site ou épuisement de nos ressources.

En effet, vous devez bien comprendre que lorsque quelqu'un intègre nos contenus sur son site avec un élément iframe, l'affichage du contenu de l'iframe va puiser dans nos ressources serveur puisque le contenu va être demandé et chargé à partir de notre serveur.

Pour se prémunir contre cela, nous allons pouvoir interdire aux autres sites d'intégrer nos contenus via des éléments iframe. Pour faire cela, nous allons devoir définir une politique de sécurité de contenu ou CSP (Content Security Policy) adaptée.

A ce niveau, le sujet commence à devenir vraiment complexe puisqu'il va falloir configurer nos en-tête HTTP à partir de notre serveur. Je n'entrerai pas plus dans le détail ici car le sujet mériterait un cours à lui seul et serai bien trop complexe pour le moment.

Cependant, retenez tous ces termes pour pouvoir vous renseigner plus avant si un jour vous avez besoin d'utiliser ces objets.

Les autres éléments d'intégration

Pour être tout à fait exhaustif, je dois ici préciser que l'élément iframe n'est pas le seul élément d'intégration de ressources externes.

En effet, d'autres éléments comme embed et object existent et servent à intégrer des documents comment des PDF, des graphiques vectoriels (SVG) ou même des Flash.

Cependant, ces éléments sont aujourd'hui à mon sens complètement désuets et vous ne devriez jamais avoir à les utiliser. Nous ne les étudierons donc pas dans ce cours.