Les bases du HTML

Rédigé par: Jean-Luc Colson			
Date première rédaction: Octobre 2019			
SUIVI DES MODIFICATIONS A LA FICHE			
Série	Date	Page(s) modifiée(s)	Raison

Contents

Eléments, balises et attributs HTML

Les éléments HTML

Le langage HTML tout entier repose sur l'utilisation d'éléments. Si vous comprenez bien ce qu'est un élément, vous comprenez le HTML.

Les éléments HTML vont nous permettre de créer la structure d'une page HTML, de définir chaque contenu de notre page et également de passer certaines informations utiles au navigateur pour afficher la page (comme le type d'encodage utilisé par exemple pour que celui-ci affiche bien nos accents français).

Dans une page, nous allons utiliser les éléments en HTML pour marquer du contenu, c'est-à-dire pour lui donner du sens aux yeux des navigateurs et des moteurs de recherche. Selon l'élément utilisé, un navigateur va reconnaître le contenu comme étant de telle ou telle nature.

Ainsi, on va utiliser des éléments pour définir un paragraphe ou un titre par exemple, ou encore pour insérer une image ou une vidéo dans un document.

L'élément p, par exemple, sert à définir un paragraphe, tandis que l'élément a va nous permettre de créer un lien, etc.

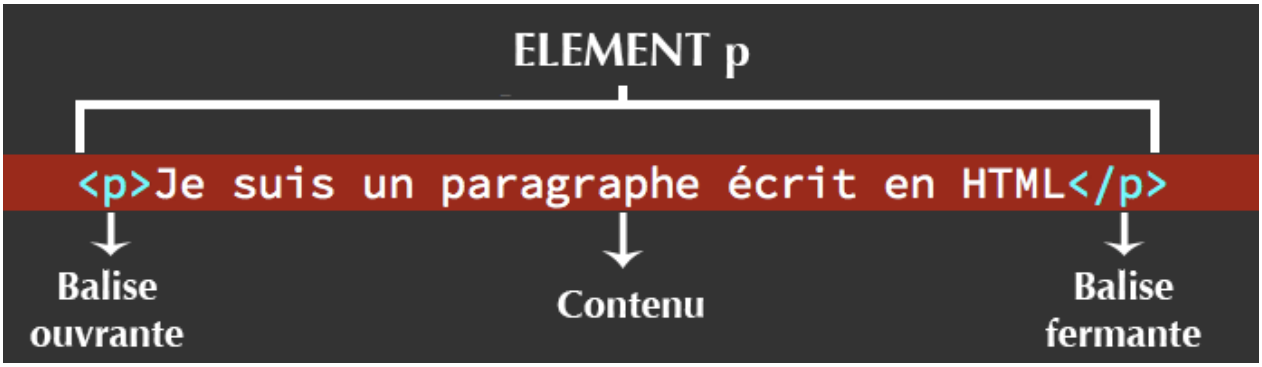
Aujourd'hui, il existe plus de 120 éléments HTML différents aux rôles très variés et qui font la richesse de ce langage. Nous allons étudier et nous servir d'une grande partie d'entre eux dans ce cours.

Les balises HTML

Un élément HTML peut être soit constitué d'une paire de balises (ouvrante et fermante) et d'un contenu, soit d'une balise unique qu'on dit alors « orpheline ».

L'élément p (qui sert à définir un paragraphe) est par exemple constitué d'une balise ouvrante, d'une balise fermante et d'un contenu textuel entre les balises. L'idée ici est que le texte contenu entre les deux balises va être le texte considéré par le navigateur comme étant un paragraphe.

Voici comment on va écrire cela :

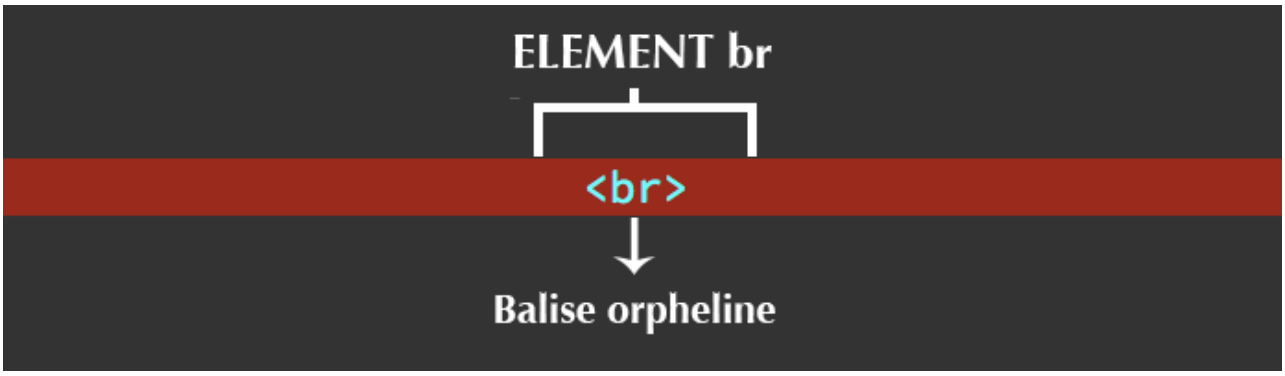


Comme vous pouvez le constater, la balise ouvrante de l'élément est constituée d'un chevron ouvrant <, du nom de l'élément en question et d'un chevron fermant >.

Notez bien ici la différence entre la balise ouvrante et la balise fermante de notre élément p : la balise fermante contient un slash avant le nom de l'élément.

Vous pouvez déjà retenir cette syntaxe qui sera toujours la même en HTML.

Certains éléments en HTML ne vont être constitués que d'une balise qu'on appelle alors orpheline. Cela va être le cas pour certains éléments qui ne possèdent pas de contenu textuel comme l'élément br par exemple qui sert simplement à créer un retour à la ligne en HTML et qui va s'écrire comme ceci :



Notez ici qu'il est possible que vous rencontriez un jour une syntaxe un peu différente pour les balises orphelines utilisant un slash après le nom de l'élément comme ceci : <br/>.

Cette syntaxe est une syntaxe qui était acceptée il y a quelques années mais qui est aujourd'hui dépréciée en HTML. Elle provient en fait d'un autre langage qui est le XML. Il est déconseillé de l'utiliser aujourd'hui en HTML puisqu'elle risque de ne plus être reconnue par les navigateurs dans le futur.

Par ailleurs, notez que certains développeurs ont tendance, par abus de langage, à confondre les termes « élément » et « balise » en utilisant le mot « balise » pour désigner un élément. Vous verrez peut-être cela écrit dans d'autres cours.

C'est toutefois un abus de langage et je pense qu'il est préférable pour une bonne compréhension d'appeler chaque objet d'un langage par son vrai nom. Je vous conseille donc de retenir plutôt ce que j'ai expliqué plus haut.

Les attributs HTML

Finalement, les éléments vont également pouvoir contenir des attributs qu'on va alors placer au sein de la balise ouvrante de ceux-ci. Pour certains éléments, les attributs vont être facultatifs tandis que pour d'autres ils vont être obligatoires pour garantir le bon fonctionnement du code HTML.

Les attributs vont en effet venir compléter les éléments en les définissant plus précisément ou en apportant des informations supplémentaires sur le comportement d'un élément.

Un attribut contient toujours une valeur, qu'on peut cependant parfois omettre dans le cas des attributs ne possédant qu'une seule valeur (car la valeur est alors considérée comme évidente).

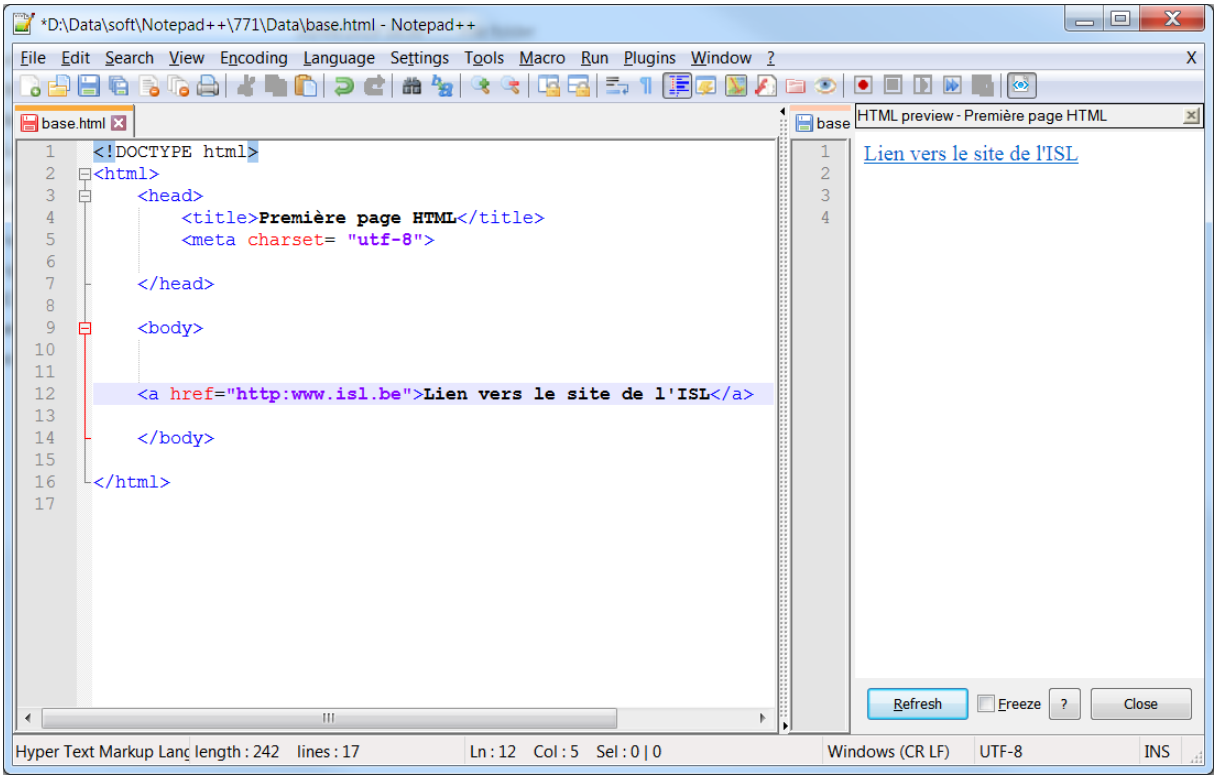
Prenons ici l'exemple de l'élément a qui est l'abréviation de « anchor » ou « ancre » en français. Cet élément va principalement nous servir à créer des liens vers d'autres pages.

Pour le faire fonctionner correctement, nous allons devoir lui ajouter un attribut href pour « hypertexte référence » ou « référence hypertexte » en français.

En effet, c'est l'attribut href qui va nous permettre de préciser la cible du lien, c'est-à-dire la page de destination du lien en lui passant l'adresse de la page en question en valeur.

 Lien vers le site de l'institut

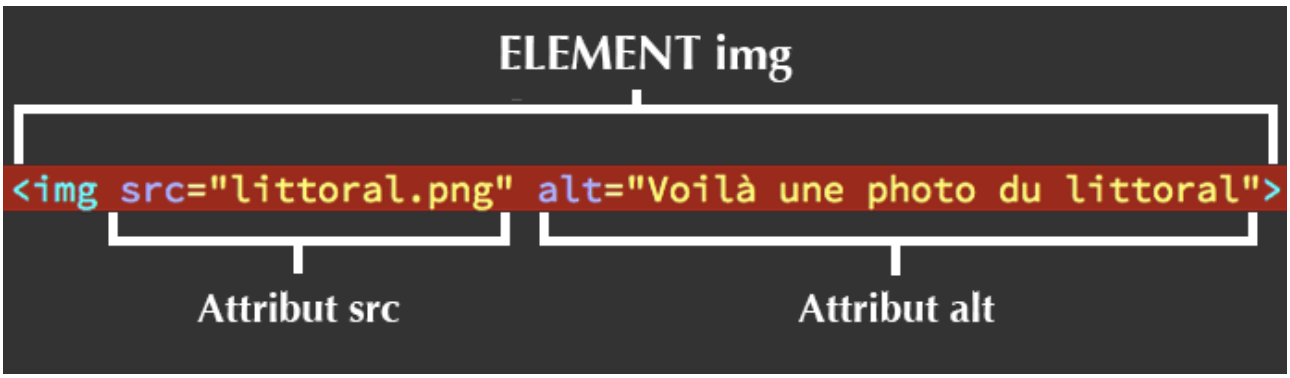
Le lien s'affichant comme ceci dans le navigateur : Lien vers le site de l'ISL



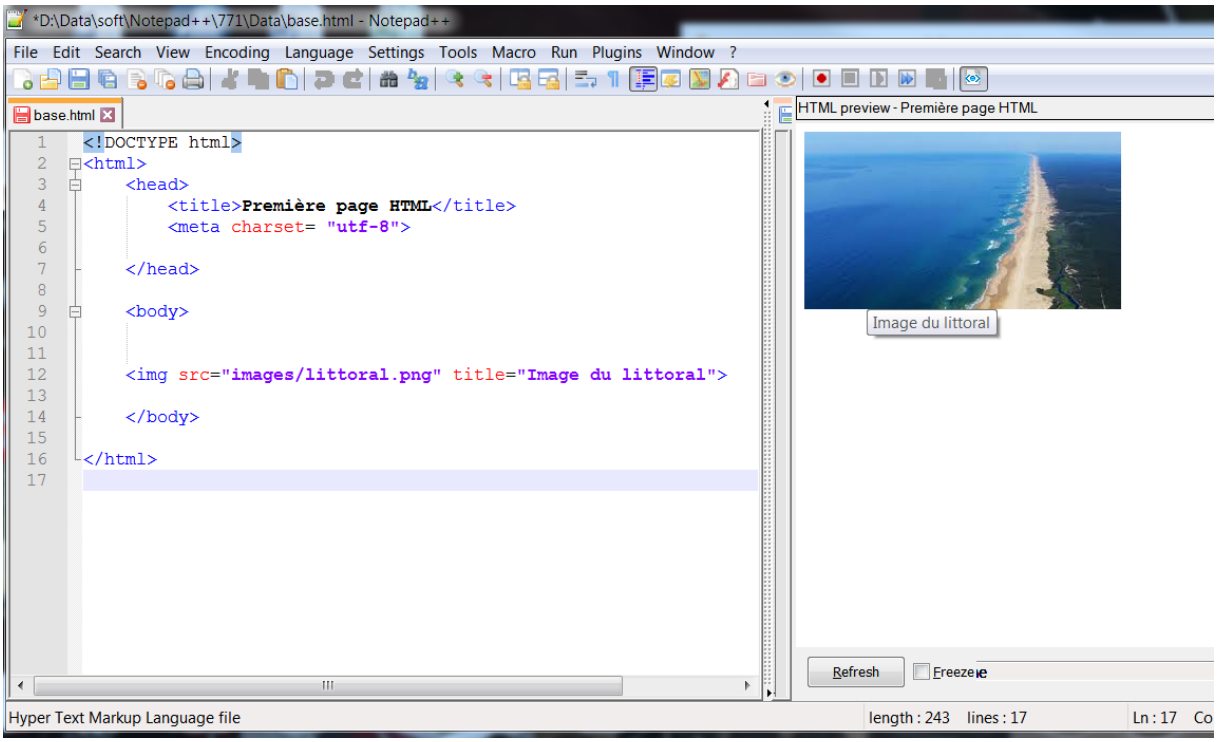
Un autre exemple : l'élément img, servant à insérer une image dans une page HTML, va lui nécessiter deux attributs qui sont les attributs src et alt.

L'attribut src (pour « source ») va prendre comme valeur l'adresse de l'image tandis que l'attribut alt (pour « alternative ») va nous permettre de renseigner une description textuelle de l'image qui sera affichée dans les cas où l'image ne serait pas disponible pour une raison ou une autre : image introuvable, impossible à charger, etc.

L'attribut alt va également se révéler indispensable pour rendre notre site accessible aux non-voyants ou aux malvoyants et pour leur fournir une bonne expérience de navigation puisqu'ils sont généralement équipés de lecteurs spéciaux qui vont pouvoir lire la valeur de l'attribut alt et donc leur permettre de se faire une représentation du contenu de l'image.



Notez au passage que l'élément img n'est constitué que d'une seule balise orpheline, tout comme l'élément br vu précédemment. On place dans ce cas les attributs au sein de la balise orpheline.



Structure minimale d'une page HTML valide

Le W3C, en définissant des standards de développement, a véritablement simplifié la tâche aux développeurs.

En effet aujourd'hui, tous les navigateurs sérieux suivent les standards proposés par le W3C, ce qui n'était pas forcément le cas dans le passé.

Cela nous permet donc d'être plus ou moins certains qu'un même code va produire le même résultat quel que soit le navigateur utilisé par les visiteurs qui consultent la page.

Cependant, cela n'est possible que parce que les développeurs eux-mêmes suivent certains schémas de construction de leur page qui sont anticipables et attendus par les navigateurs.

Le schéma de base d'une page HTML va donc toujours être le même. C'est ce que nous appellerons la « structure minimale d'une page HTML valide ».

Cette nouvelle leçon va être consacrée à la création d'une première page HTML dite « valide », c'est-à-dire qui respecte les standards, et va nous permettre de comprendre le rôle de chaque élément de cette structure minimale.

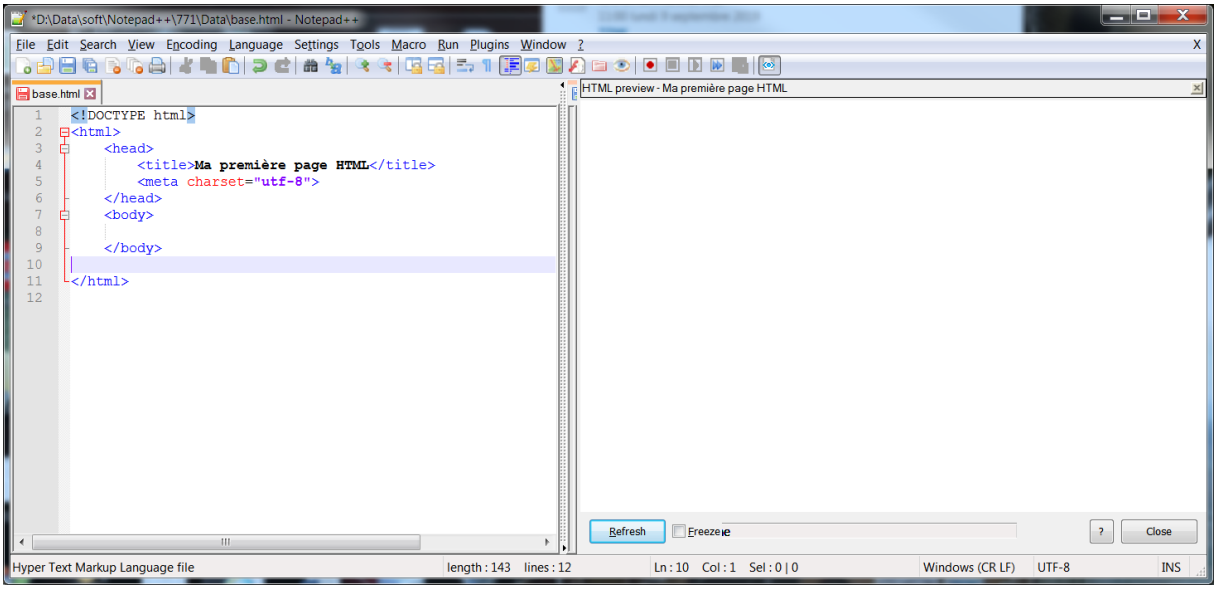
La structure minimale d'une page HTML

Pour qu'une page HTML soit déclarée valide, elle doit obligatoirement comporter certains éléments et suivre un schéma précis.

Vos pages HTML devraient toujours être valides pour les raisons évoquées ci-dessus. En effet, une page non valide ne sera pas comprise par le navigateur qui va alors potentiellement mal l'afficher voire dans certains cas ne pas l'afficher du tout.

De plus, une page non valide sera également mal analysée par les moteurs de recherche et ces mêmes moteurs de recherche risquent donc de ne pas la valoriser, c'est-à-dire de ne pas la proposer aux utilisateurs recherchant une information que votre page contient. En d'autres termes, posséder des pages non valides risque d'impacter négativement le référencement de ces pages et de votre site en général dans les moteurs de recherche.

Voici ci-dessous le code minimum pour créer une page HTML valide. Nous allons dans la suite de ce chapitre expliquer le rôle de chaque élément et attribut.



Le DOCTYPE

Tout d'abord, nous devrons toujours démarrer une page HTML en précisant le doctype de notre document, <!DOCTYPE html>. Comme son nom l'indique, le doctype sert à indiquer le type du document.

Faites bien attention à l'écriture du doctype : vous pouvez remarquer que la balise représentant le doctype commence par un point d'exclamation. Ceci est un cas unique.

Dans la balise de l'élément doctype, on va préciser le langage utilisé, à savoir le HTML dans notre cas.

Il est possible que vous trouviez encore sur certains sites une déclaration du doctype possédant une syntaxe différente et plus complexe comme celle-ci :

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">.

Cette écriture correspond à une ancienne syntaxe qui était utilisée jusqu'à la version 4 du HTML.

L'élément HTML

Après avoir renseigné le type du document, nous devons utiliser un élément html. Cet élément est composé d'une paire de balises ouvrante <html> et fermante </html>.

L'élément html va représenter notre page en soi. On va insérer tout le contenu de notre page (et donc les autres éléments) à l'intérieur de celui-ci.

Les éléments head et body

A l'intérieur de l'élément html, nous devons à nouveau indiquer obligatoirement deux éléments qui sont les éléments head et body et qui vont avoir des rôles très différents.

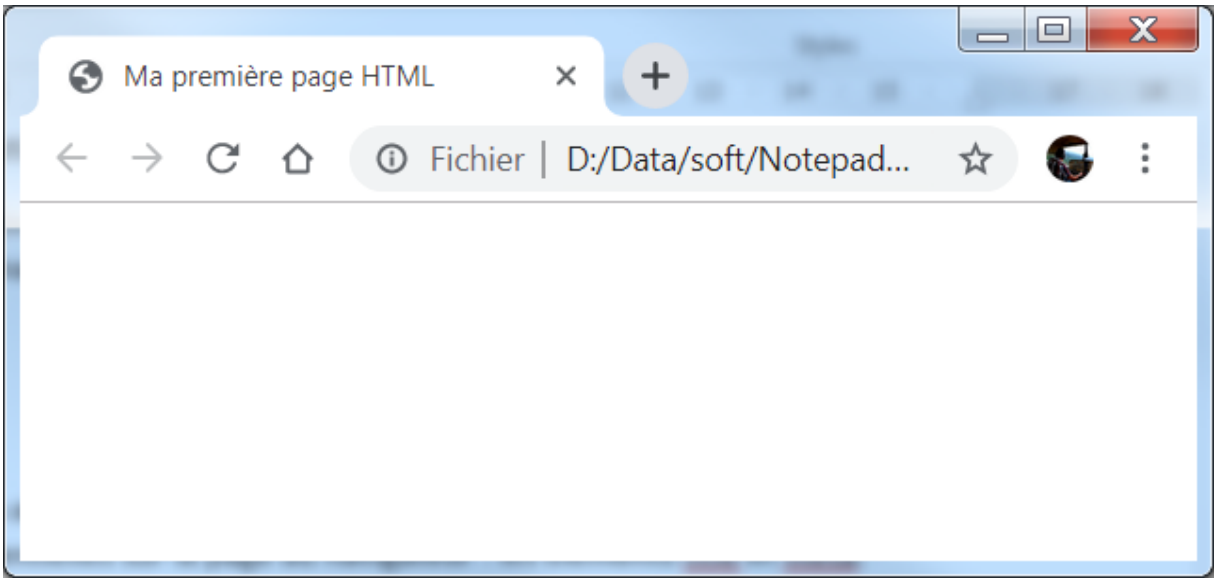
L'élément head est un élément d'en-tête. Il va contenir des éléments qui vont servir à fournir des informations sur la page au navigateur, comme le titre de la page ou encore le type d'encodage utilisé pour que celui-ci puisse afficher les caractères de texte correctement.

L'élément body va lui contenir tous les éléments définissant les contenus « visibles » de la page, c'est-à-dire les contenus à destination de l'utilisateur et notamment les différents textes présents dans la page, les images, etc.

Les éléments title et meta

Au sein de l'élément head, nous allons devoir à minima indiquer deux éléments qui vont permettre de donner des informations essentielles sur la page au navigateur : les éléments title et meta.

L'élément title va nous permettre d'indiquer le titre de la page en soi, qui ne doit pas être confondu avec les différents textes définis comme des titres dans la page. Ce titre de page est le texte visible sur le haut des onglets de votre navigateur par exemple :



L'élément meta sert lui à transmettre des meta informations sur la page au navigateur. Cet élément possède de nombreux attributs différents. Le type d'informations transmis va dépendre de l'attribut que l'on va préciser.

Ici, ce qui nous intéresse va être de préciser le type d'encodage utilisé dans nos pages. Cela va permettre aux navigateurs d'afficher correctement nos différents textes avec les accents, les cédilles, etc.

Pour faire cela, nous allons utiliser l'attribut charset pour « characters set » c'est-à-dire « ensemble de caractères » de l'élément meta et lui fournir la valeur utf-8.

La valeur utf-8 est de loin la valeur la plus utilisée sur le web et est la valeur de référence pour tous les alphabets latins. Cela va permettre à chacun de nos caractères de s'afficher correctement dans le navigateur. Le format Unicode Text Format utf-8 est un format dynamique de codage des caractères sur 1 à 4bytes.

Note : Lorsque vous travaillez en local vous devrez, avec certains éditeurs, renseigner également l'encodage de la page dans l'éditeur afin que le contenu de celle-ci s'affiche bien et notamment si vous utilisez un outil de prévisualisation de page fourni par votre éditeur.

Voilà tout pour la structure minimale d'une page HTML valide, qui représente toujours la première étape de création d'une vraie page web. Pour le moment, notre page ne possède pas de contenu visible. Nous allons en ajouter progressivement au cours des prochaines leçons.

Un mot sur l'imbrication des balises et des éléments

Un autre concept qu'il vous faut comprendre absolument pour coder en HTML est la façon dont les éléments HTML s'imbriquent les uns dans les autres.

Vous l'avez probablement remarqué : ci-dessus, nous avons placé des éléments HTML entre les balises ouvrantes et fermantes d'autres éléments (par exemple, les éléments title et meta ont été placés à l'intérieur de l'élément head).

On appelle cela l'imbrication. L'imbrication est l'une des fonctionnalités du HTML qui fait toute sa force (nous découvrirons réellement pourquoi plus tard, avec l'étude du CSS).

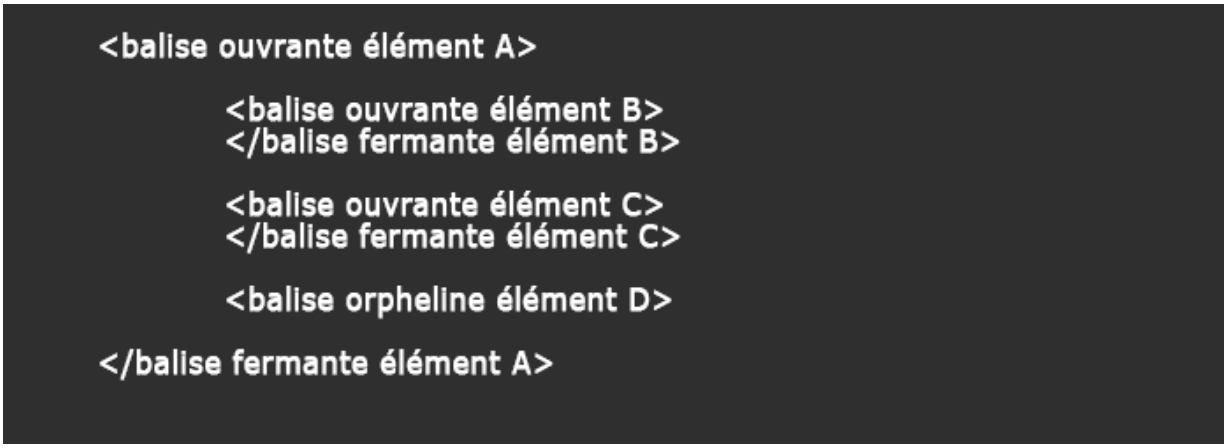
Cependant, comme toujours, on ne peut pas imbriquer des éléments HTML n'importe comment et il faut suivre des règles précises.

Ainsi, nous n'avons absolument pas le droit de « croiser » les balises des éléments ou, pour le dire plus clairement : le premier élément déclaré doit toujours être le dernier refermé, tandis que le dernier ouvert doit toujours être le premier fermé.

Par exemple, vous voyez que notre premier élément déclaré est l'élément html, qui contient les éléments head et body. L'élément html devra donc être le dernier élément refermé.

Faites bien attention à distinguer les éléments contenus dans d'autres et ceux qui sont au même niveau hiérarchique. Par exemple, ici, title et meta sont deux éléments « enfants » de l'élément head (car ils sont contenus à l'intérieur de celui-ci), mais sont deux « frères » : aucun des deux ne contient l'autre.

Le schéma ci-dessous présente toutes les situations valides d'imbrication d'éléments :



Pour savoir quand un élément doit être placé dans un autre, il faut penser en termes de « dépendance » : les informations transmises par l'élément dépendent-elle d'un autre élément ou sont-elles totalement indépendantes ?

Par exemple, l'élément <body></body> doit se trouver DANS l'élément <html></html>.

Ce concept peut être complexe à se représenter lorsqu'on débute à peine, mais ne vous inquiétez pas en pratiquant un peu tout cela va très vite devenir évident.

Enregistrement et affichage d'une page HTML

L'enregistrement de notre fichier

Maintenant que nous avons établi les bases, nous allons pouvoir commencer à coder ensemble.

Pour cela, commencez par ouvrir votre éditeur de texte puis créez un nouveau fichier vide. Par défaut, votre fichier devrait être un fichier au format .txt, c'est-à-dire un simple fichier texte.

La plupart des éditeurs de texte proposent aujourd'hui des fonctionnalités nous aidant à coder plus rapidement et à créer un meilleur code.

Parmi ces fonctionnalités, on peut notamment citer l'auto-complétion des balises HTML qui correspond à l'écriture automatique de la balise fermante d'un élément dès la déclaration de la balise ouvrante (pour un élément constitué d'une paire de balises) ou encore la vérification automatique de la validité du code écrit.

Pour que ces fonctionnalités s'activent, il va cependant falloir que notre éditeur de texte sache quel langage de code est utilisé dans notre page.

Pour le lui indiquer, nous avons deux façons de procéder : on peut soit déjà enregistrer notre fichier avec la bonne extension c'est-à-dire ici .html, soit préciser le langage utilisé dans l'un des onglets de l'éditeur (si vous cherchez bien, vous devriez voir quelque part une liste de langages informatiques proposés).

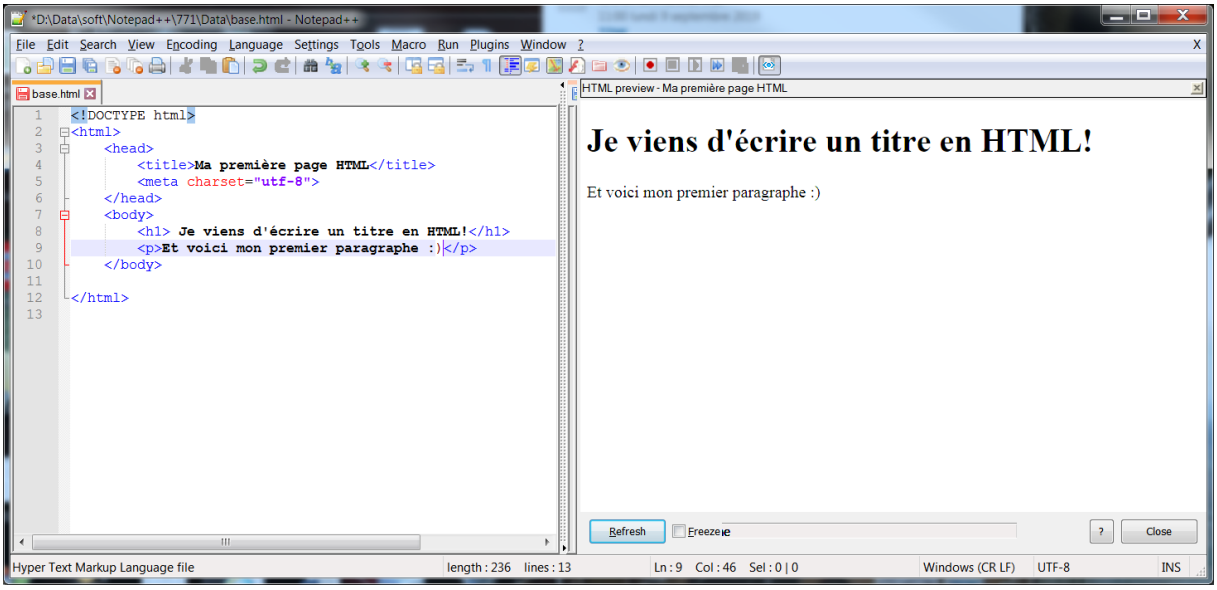
Ici, nous allons utiliser la méthode la plus simple qui va être d'enregistrer notre fichier. Pour cela, allez dans l'onglet « fichier » puis « enregistrer sous » ou utilisez le raccourci CMD+SHIFT+S (pour Mac) ou CTRL+SHIFT+S / CTRL+S (pour Windows).

Pour la suite de ce cours, je vous propose de créer un dossier qu'on va appeler « cours » sur votre bureau ou dans le dossier/répertoire prévu pour le cours. Nous allons enregistrer nos différents fichiers dans ce dossier.

Il y a quelques règles à respecter lorsqu'on enregistre un fichier de code : le nom du fichier ne doit pas contenir d'espace ni de caractères spéciaux (pas de caractères accentués ou de ponctuation ni de sigles) et doit commencer par une lettre. Ici, on peut enregistrer notre fichier sous le nom cours.html par exemple pour faire très simple.

Dès que votre fichier est enregistré au bon format, nous allons retourner dans notre éditeur et écrire la structure minimale d'une page HTML. Notre éditeur sait maintenant qu'on écrit en HTML et va donc certainement utiliser une palette de couleurs, l'auto-complétion, etc. pour nous aider à coder.

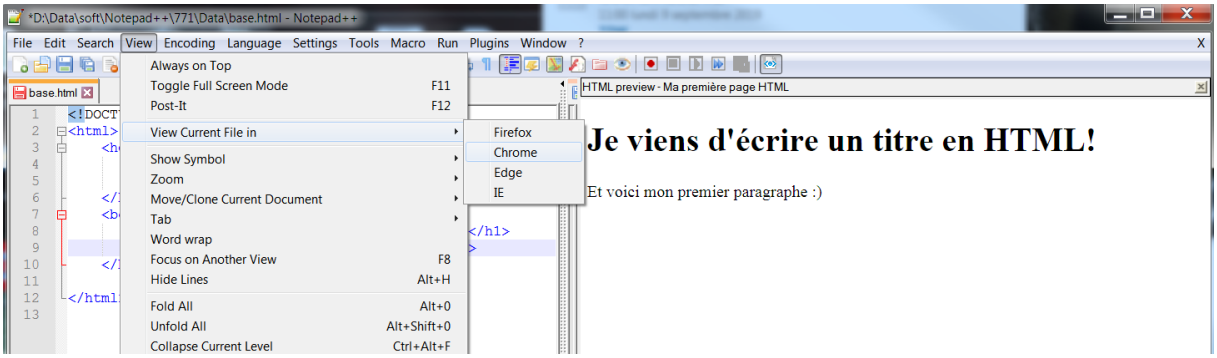
Ensuite, nous allons ajouter un grand titre dans notre page et un paragraphe en utilisant respectivement les éléments h1 et p.



Dès que votre page est prête, nous allons l'enregistrer à nouveau pour ensuite l'afficher dans notre navigateur.

Afficher le résultat d'un code HTML dans le navigateur

Pour afficher notre fichier dans le navigateur, nous allons aller le chercher dans notre dossier puis effectuer un clic droit dessus et choisir « ouvrir avec… » + le navigateur de votre choix. Notepad++ permet cependant de lancer le navigateur de votre choix à partir de son menu.



L'indentation et les commentaires en HTML

Avant d'aller plus loin dans notre apprentissage du HTML et du CSS, il me semble intéressant de déjà vous parler de certaines pratiques de développeurs qui sont appelées des « bonnes pratiques » afin que vous puissiez rapidement les assimiler et qu'elles deviennent rapidement des automatismes.

Les bonnes pratiques sont là pour nous aider à coder plus proprement et plus lisiblement, donc pour nous faciliter la vie, pas le contraire !

Nous allons dans cette partie parler d'indentation et de commentaires HTML.

Retours à la ligne et indentation

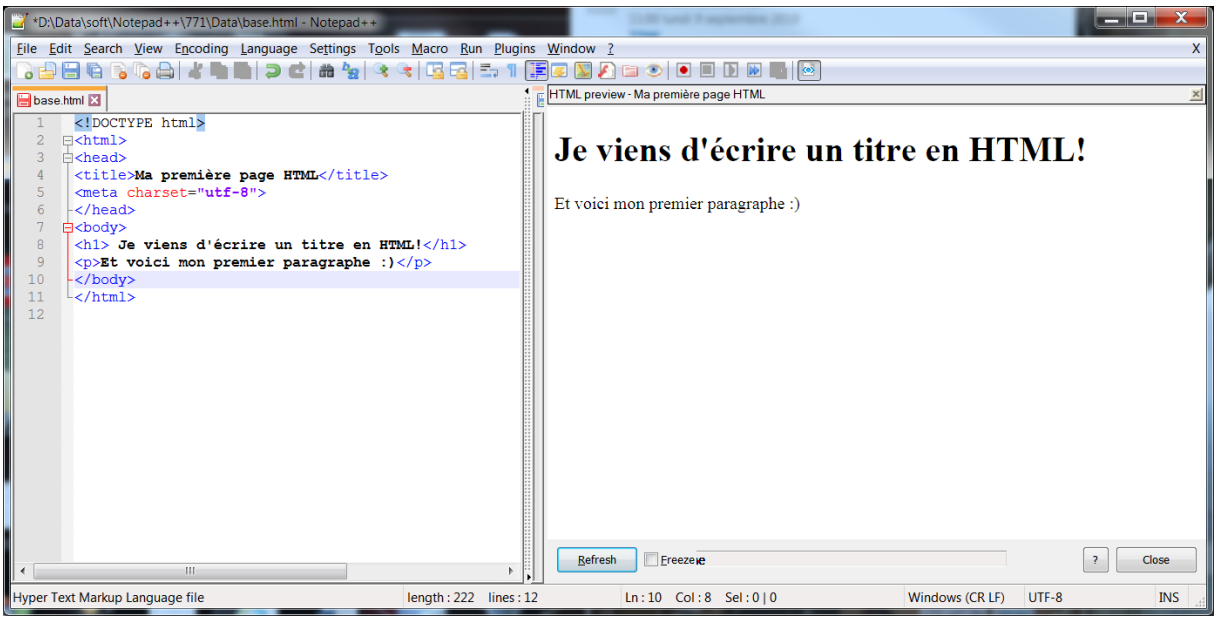
Indenter correspond à créer des retraits en début de ligne dans votre éditeur de façon cohérente et logique.

Vous l'avez certainement remarqué dans les codes précédents : j'accentue plus au moins le retrait à gauche de chacune de mes lignes de code.

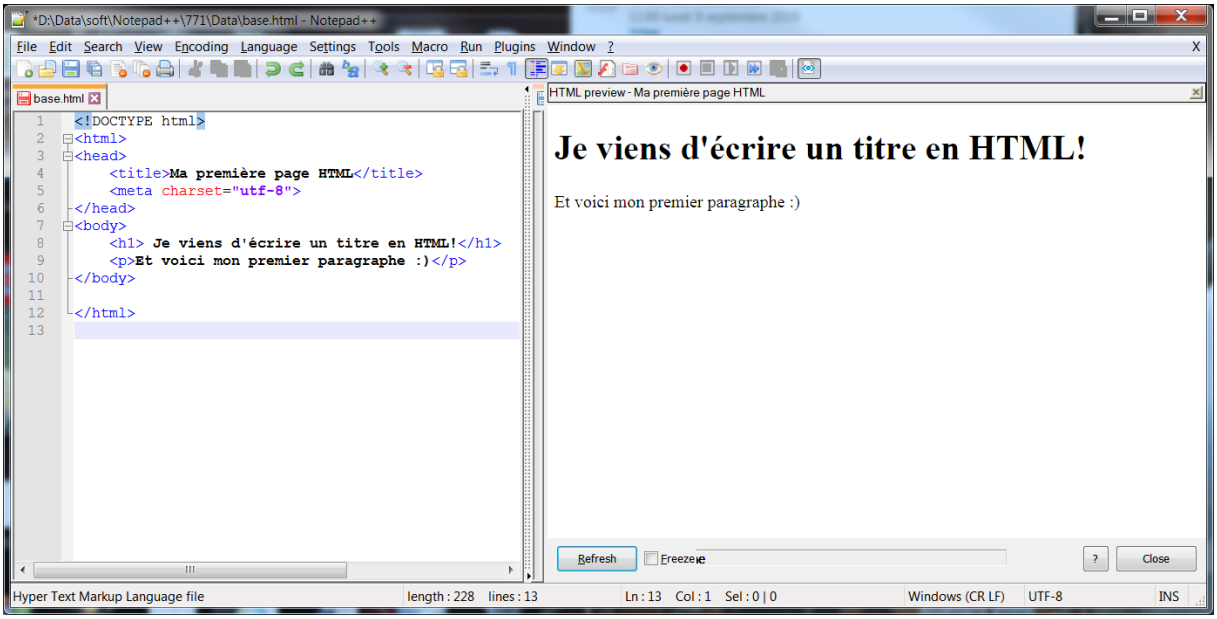
Indenter va nous permettre d'avoir un code plus propre et plus lisible, donc plus compréhensible. Indenter permet également de plus facilement détecter les erreurs potentielles dans un code.

Le but de l'indentation est de nous permettre de discerner plus facilement les différents éléments ou parties de code. Regardez plutôt les deux exemples suivants.

Sans indentation :



Avec indentation :



J'espère que vous serez d'accord pour dire que le second code est plus clair que le premier.

Concernant l'indentation, il n'y a pas de règle absolue, notamment par rapport à la taille du retrait de chaque ligne et différents éditeurs peuvent d'ailleurs posséder différentes règles par défaut sur ce sujet.

Pour ma part, je me contente de retourner à la ligne et d'effectuer une nouvelle tabulation à chaque fois que j'ouvre un nouvel élément dans un autre, ce qui correspond généralement à un retrait de 4 espaces.

Par exemple, ci-dessus, vous pouvez voir que j'ai utilisé une tabulation avant d'écrire la balise ouvrante de mon élément head qui est contenu dans mon élément html.

J'ai ensuite effectué une nouvelle tabulation à l'intérieur de l'élément head pour écrire mon élément title.

En revanche, je n'ai pas créé de nouvelle tabulation pour l'élément meta étant donné que cet élément n'est pas contenu dans title (ce n'est pas un élément enfant de l'élément title) mais est au même niveau hiérarchique que lui (on parle d'élément frère).

Vous pouvez observer la même chose avec les éléments head et body par rapport à l'élément html : j'ai décalé head et body avec le même écart par rapport à html car ces deux éléments sont frères (aucun des deux ne contient / n'est contenu dans l'autre).

Au final, indenter permet de créer une hiérarchie dans le code et l'un des buts principaux de l'indentation en HTML va être de comprendre très rapidement quels éléments sont imbriqués dans quels autres dans notre page pour avoir une meilleure vue d'ensemble de notre code et faciliter sa compréhension.

Note : Les retours à la ligne et l'indentation créés dans l'éditeur n'affectent pas le résultat final dans le navigateur. Nous aurons l'occasion de reparler de cela un peu plus tard dans ce cours.

Définition et utilité des commentaires en HTML

Les commentaires vont être des lignes de texte que l'on va écrire au milieu de notre code, afin de donner des indications sur ce que fait le code en question.

Les commentaires ne seront pas affichés par le navigateur lorsque celui-ci va afficher la page : ils ne vont servir qu'aux développeurs créant ou lisant le code.

Les commentaires vont pouvoir être très utiles dans trois situations :

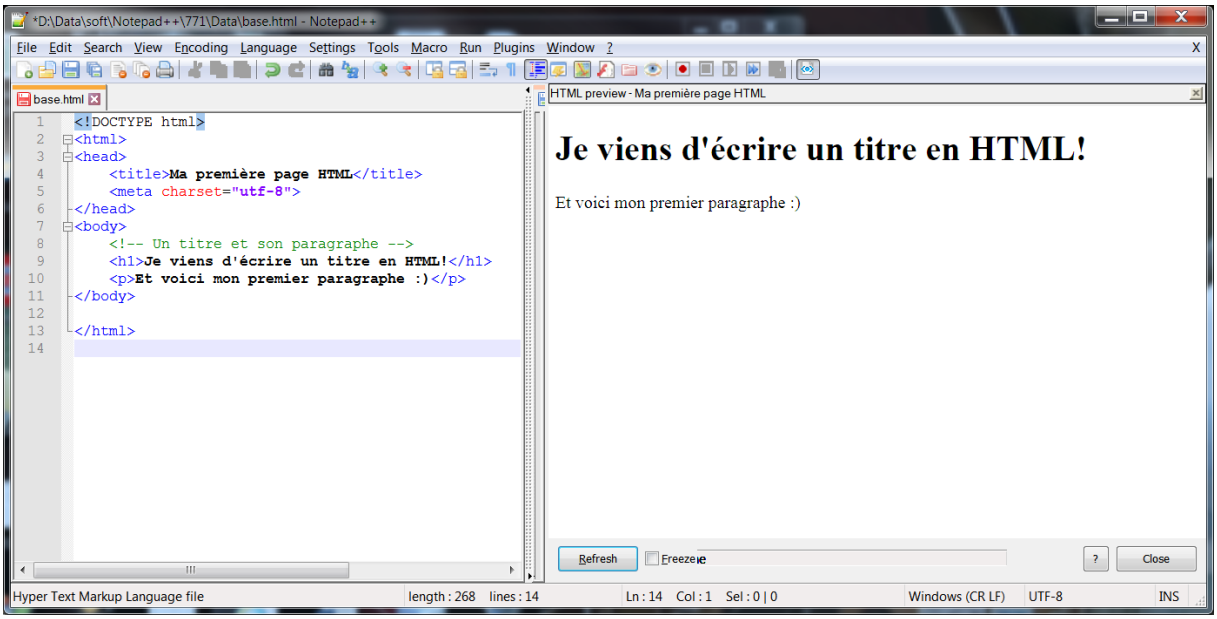
1. Dans le cas d'un gros/long projet, afin de bien se rappeler soi-même pourquoi nous avons écrit tel ou tel code, ou encore pour se repérer dans le code ;
2. Si l'on souhaite distribuer son code, ou si l'on travaille à plusieurs, cela fait beaucoup plus professionnel et permet aux autres développeurs de comprendre beaucoup plus rapidement et facilement le code distribué ;
3. Pour « neutraliser » une certaine partie d'un code sans toutefois le supprimer. Il suffit en effet de placer toute la partie du code en question en commentaire afin que celui-ci soit ignoré par le navigateur.

Syntaxe et écriture des commentaires en HTML

Les commentaires peuvent être mono-ligne (c'est-à-dire écrits sur une seule ligne dans l'éditeur) ou multi-lignes (écrits sur plusieurs lignes dans l'éditeur).

A la différence de nombreux autres langages informatiques utilisant également des commentaires, la syntaxe utilisée pour créer un commentaire mono-ligne va être la même que celle pour créer un commentaire multi-lignes en HTML.

Voici comment s'écrit un commentaire en HTML :



Les commentaires en HTML vont prendre la forme d'une balise orpheline très particulière, avec un chevron ouvrant suivi d'un point d'exclamation suivi de deux tirets au début, du commentaire en soi puis à nouveau de deux tirets et d'un chevron fermant (sans point d'exclamation cette fois, attention !).

<!-- Un titre et son paragraphe -->

Même si vous ne voyez pas forcément l'intérêt de commenter dès maintenant, je vous garantis que c'est souvent ce qui sépare un bon développeur d'un développeur moyen. Faites donc l'effort d'intégrer cette bonne pratique dès à présent, cela vous sera bénéfique dans le futur.

Mise en garde : ne placez pas d'informations sensibles en commentaire !

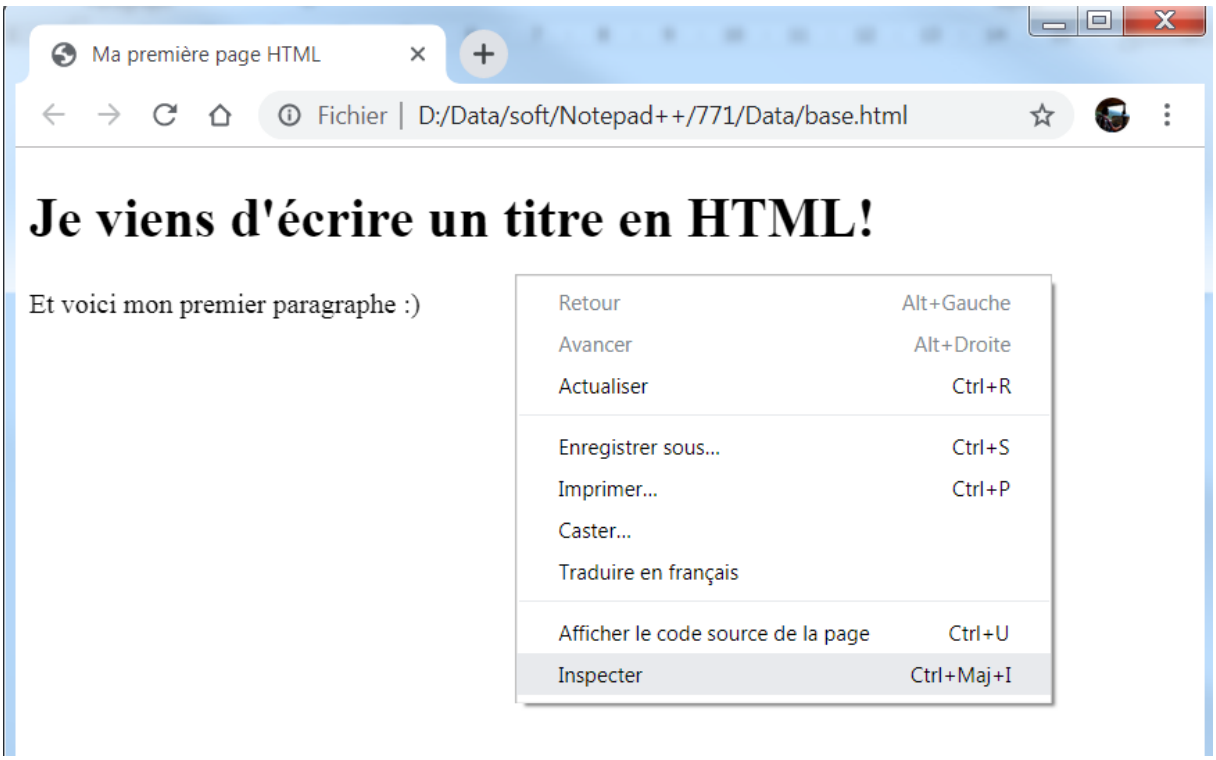
Vous vous rappelez lorsque je vous ai dit que vos commentaires n'étaient pas affichés aux yeux de vos visiteurs ?

C'est vrai, mais faîtes bien attention, car cela ne veut pas dire qu'il est impossible pour un visiteur ou un robot web de voir ce que vous avez écrit en commentaire.

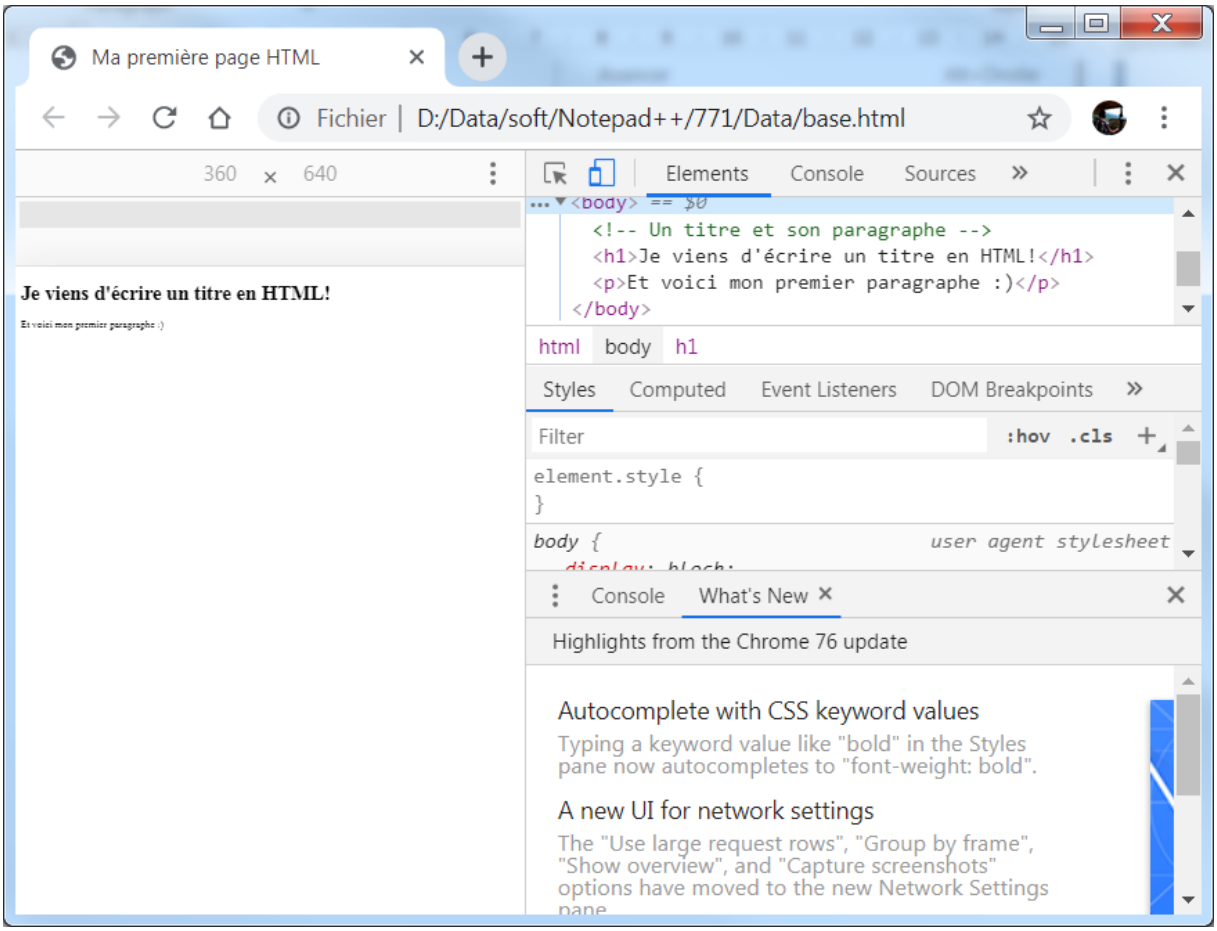
En effet, tous vos visiteurs peuvent voir à tout moment, s'ils le souhaitent, non seulement le contenu de vos commentaires mais aussi l'intégralité de votre code HTML.

Pour cela, il suffit simplement d'activer les outils pour développeurs dont dispose chaque navigateur puis de faire un clic droit sur la page et « d'inspecter l'élément ».

Cela est valable pour n'importe quel site. Une fois les outils pour développeurs activés, il suffit d'un clic droit…



…et le contenu HTML de la page est affiché dans une console !



Faîtes donc bien attention à ne jamais écrire d'informations sensibles au sein de vos commentaires, comme des codes ou mots de passe par exemple.

Les titres et les paragraphes en HTML

Dans cette leçon et dans les suivantes nous allons étudier quelques-uns des éléments HTML les plus utiles et les plus utilisés. Cela va également nous permettre de bien nous familiariser avec la syntaxe du HTML grâce à des exemples d'application.

Dans cette leçon, nous allons en particulier nous concentrer sur la définition de titres et de paragraphes en HTML.

Pourquoi différencier titres et paragraphes en HTML ?

Une nouvelle fois, rappelez-vous que le HTML est un langage de balisage.

Le seul et unique rôle du HTML est de nous permettre de créer un document structuré en différenciant d'un point de vue sémantique les différents éléments d'une page.

Utiliser les bons éléments HTML pour définir précisément les différents contenus de nos pages va permettre aux navigateurs (et aux moteurs de recherche) de comprendre de quoi et comment est composée notre page et donc de l'afficher et de la référencer au mieux.

Or, les titres font partie des éléments auxquels les moteurs de recherche, entre autres, vont apporter une grande importance pour comprendre le sujet de notre page. En effet, Google par exemple va se servir des contenus définis comme titres pour comprendre de quoi notre page traite en y accordant plus d'importance qu'aux contenus définis comme des paragraphes.

Définition de titres en HTML

Il existe six niveaux hiérarchiques de titres ("heading" en anglais) définis par les éléments h1, h2, h3, h4, h5 et h6 qui vont nous permettre d'organiser le contenu dans nos pages.

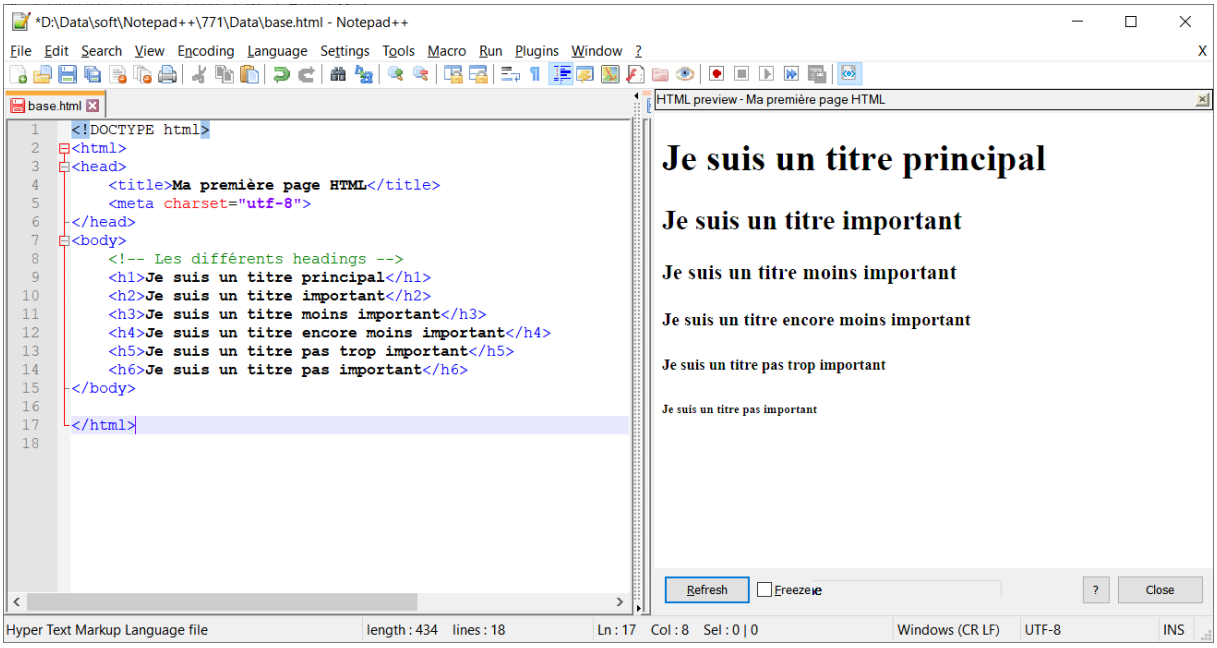
Bon à savoir : « h » signifie « heading », soit l'équivalent du mot « titre » en français. Les éléments en HTML portent souvent l'initiale de ce qu'ils représentent, en anglais.

L'élément h1 représente un titre principal dans notre page ou dans une section de page et, à ce titre, nous n'allons pouvoir utiliser qu'un seul élément h1 par page (ou par section de page, nous reviendrons sur ce concept plus tard).

Ici, je tiens d'ores-et-déjà à attirer votre attention sur un point : si vous déclarez plusieurs titres h1 dans votre page, ceux-ci s'afficheront bien : il n'y aura aucun blocage au niveau de l'éditeur ou du navigateur. Cependant, rappelez-vous une nouvelle fois que le HTML est un langage de sémantique et que sa bonne utilisation repose donc sur des règles et un ensemble de bonnes pratiques.

Dans le cas présent, utiliser plusieurs h1 serait aberrant car le h1 est censé représenter le titre principal de la page.

En revanche, nous allons pouvoir utiliser autant de titres de niveau h2, h3 etc. que l'on souhaite dans notre page. Théoriquement, si nos pages sont bien construites, nous ne devrions que rarement dépasser le niveau de titre h3.



Comme vous pouvez le voir, le navigateur comprend bien qu'il s'agit de titres d'importances diverses et les traite donc différemment par défaut, en leur attribuant une mise en forme différente (très grand et très gras pour un titre de niveau h1, puis de plus en plus petit jusqu'à h6).

Encore une fois, n'utilisez pas un élément de type h* pour écrire un texte en grand et en gras ! Utilisez le pour définir un titre dans votre page. Nous nous chargerons de la mise en forme du contenu plus tard, grâce au CSS.

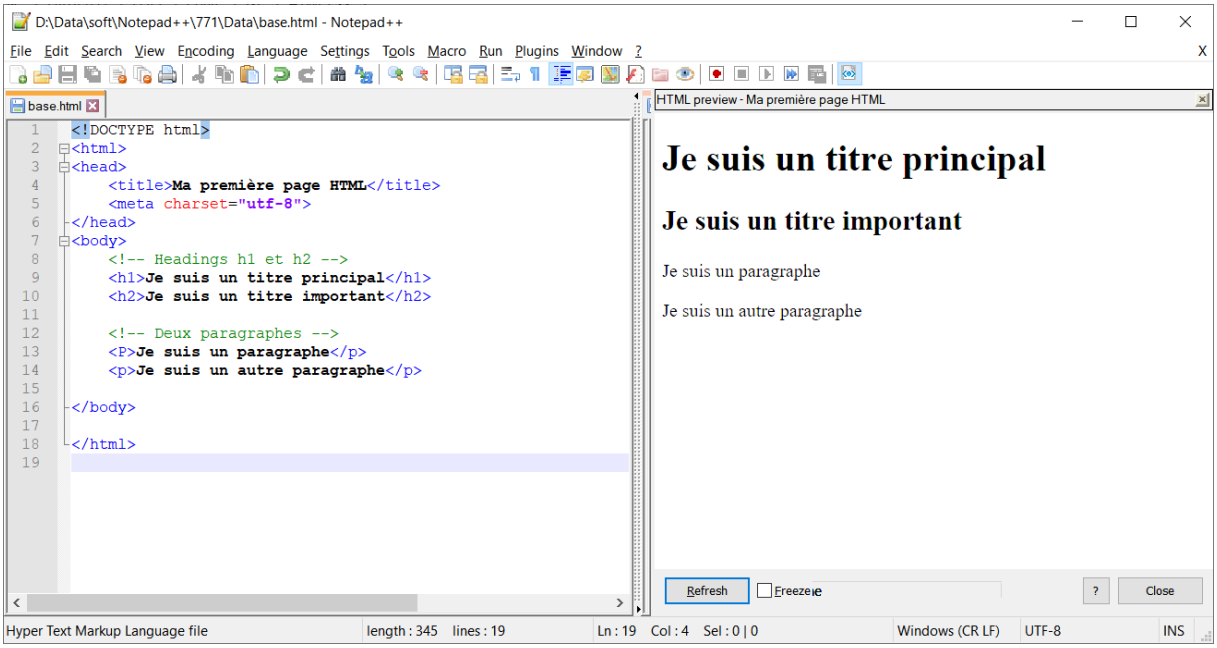
Note : Il convient de ne pas confondre les éléments h* et l'élément title : l'élément title sert à donner un titre A notre page tandis que les éléments h* servent à définir des titres DANS notre page, c'est-à-dire à hiérarchiser et à organiser le contenu de notre page.

Définition de paragraphes en HTML

Pour créer des paragraphes en HTML, nous allons utiliser l'élément p.

On peut créer autant de paragraphes que l'on souhaite dans une page. A chaque nouveau paragraphe, il faut utiliser un nouvel élément p.

Pour chaque nouveau paragraphe, un retour à la ligne va être créé automatiquement et affiché par votre navigateur (exactement comme c'était le cas avec les titres).

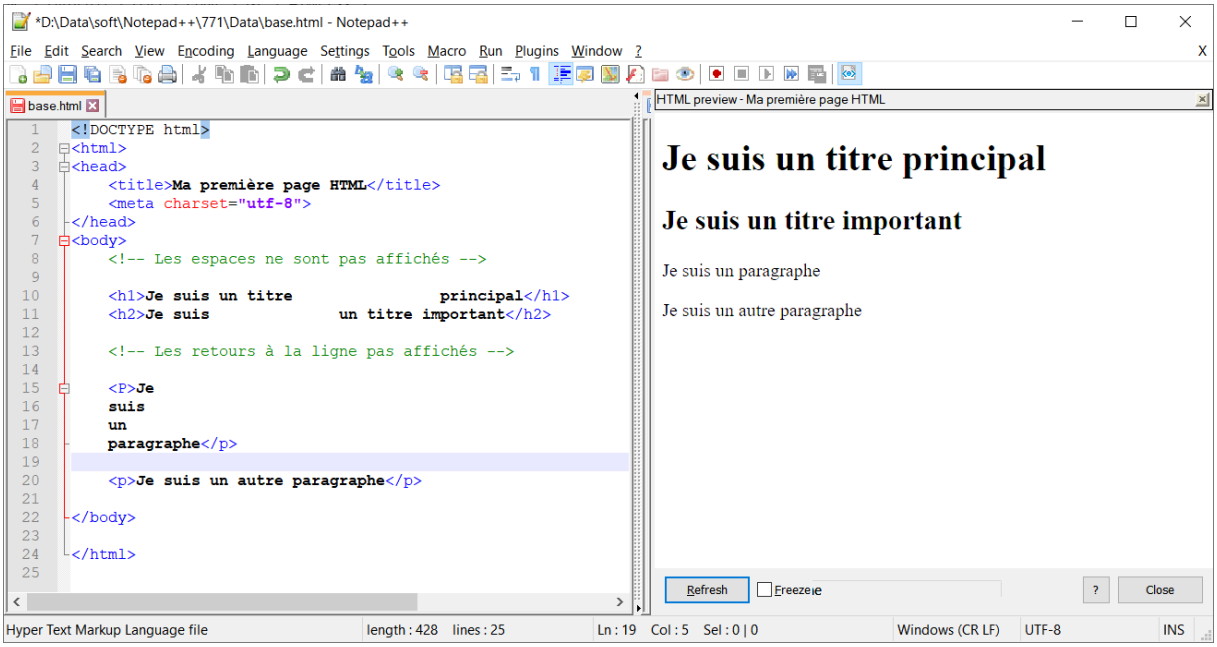

D:\Data\soft\Notepad++\771\Data\base.html - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
base.html
<!DOCTYPE html>
<html>
<head>
<title>Ma première page HTML</title>
<meta charset="utf-8">
</head>
<body>
<!-- Headings h1 et h2 -->
<h1>Je suis un titre principal</h1>
<h2>Je suis un titre important</h2>
<!-- Deux paragraphes -->
<P>Je suis un paragraphe
Je suis un autre paragraphe
</body>
</html>
HTML preview - Ma première page HTML
Je suis un titre principal
Je suis un titre important
Je suis un paragraphe
Je suis un autre paragraphe
Refresh
Freeze
Close
Hyper Text Markup Language file
length : 345 lines : 19
Ln : 19 Col : 4 Sel : 0 | 0
Windows (CR LF)
UTF-8
INS

Les espaces et retours à la ligne en HTM

Dans cette nouvelle leçon, nous allons voir comment déclarer des espaces et des retours à la ligne dans notre code HTML qui vont être conservés dans le rendu de la page fait par le navigateur.

Les espaces et les retours à la ligne dans le code ne sont pas rendus

Les plus curieux d'entre vous auront, je suppose, déjà fait le test : vous pouvez ajouter autant d'espaces que vous le voulez au sein d'un paragraphe ou d'un titre ou effectuer des retours à la ligne dans votre code, ceux-ci ne seront jamais affichés visuellement dans votre navigateur.



En effet, pour effectuer des retours à la ligne ou marquer des espaces en HTML, nous allons à nouveau devoir utiliser des éléments.

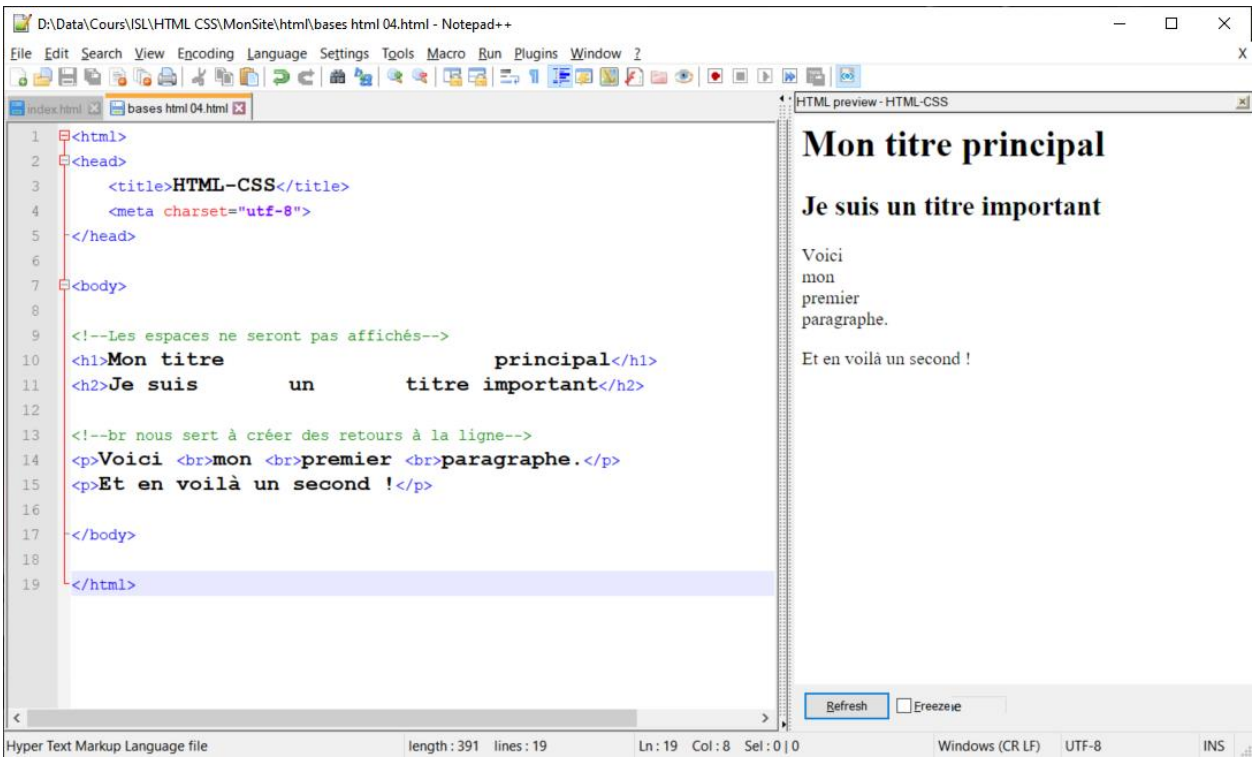
Pensez bien à nouveau que le navigateur va simplement interpréter votre code pour afficher un résultat et pour cela il va se baser sur les éléments HTML fournis.

Les retours à la ligne en HTML

Pour effectuer un retour à la ligne en HTML nous allons devoir utiliser l'élément br qui est représenté par une unique balise orpheline.

Le nom de l'élément br est l'abréviation de « break », l'équivalent de « casser » en anglais (dans le sens de « casser une ligne »).

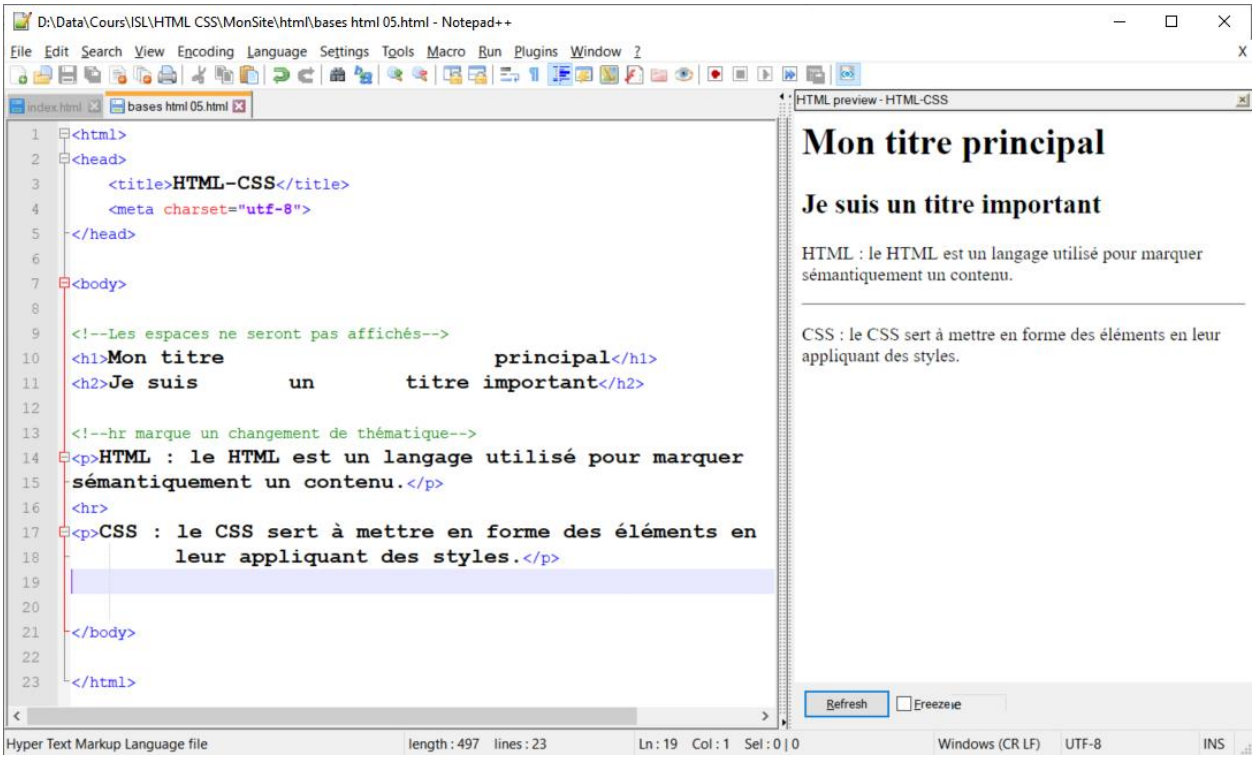
On peut utiliser autant d'éléments br que l'on souhaite au sein d'un titre ou d'un paragraphe par exemple.



Les changements de thématique en HTML

Parfois, au sein d'un article par exemple, vous aborderez des thématiques différentes que vous voudrez séparer.

Dans ce cas-là, il peut être intéressant d'utiliser l'élément hr plutôt que br. L'élément hr a justement été créé pour définir un retour à la ligne avec changement de thématique. Tout comme br, cet élément est représenté par une balise orpheline.



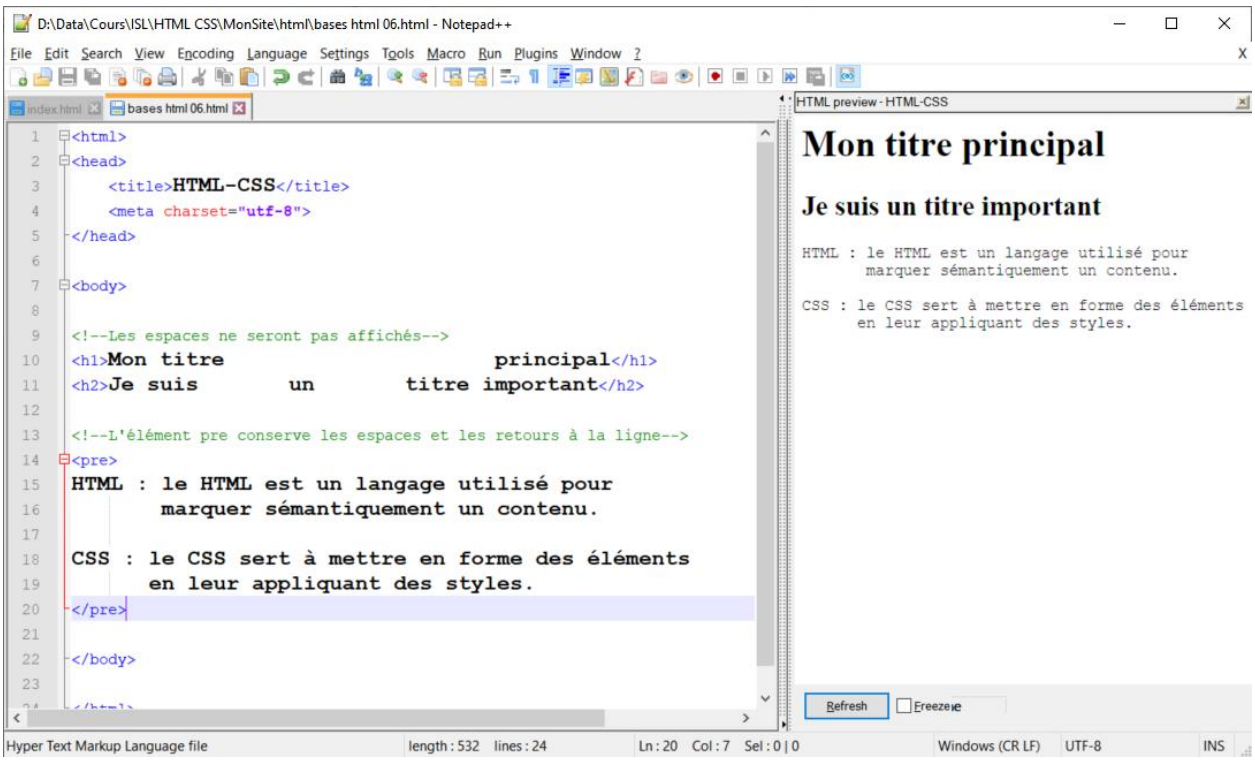
La gestion des espaces en HTML

Il n'existe pas à proprement parler d'élément permettant de définir les espaces en HTML. Cependant, des solutions existent pour ajouter des espaces au sein de nos textes qui vont être conservés dans le rendu visuel. On va ainsi pouvoir :

- Utiliser l'élément HTML de préformatage pre ;
- Utiliser des entités HTML ;
- Utiliser les marges CSS (dont nous parlerons dans le chapitre qui leur est consacré).

L'élément HTML pre

L'élément pre sert à préformater un texte. Cela signifie que tout le contenu qui se trouve à l'intérieur de cet élément va conserver la mise en forme que nous allons lui donner lors du rendu fait par le navigateur.



Attention ici : le contenu va s'afficher exactement de la même façon que dans votre éditeur par rapport à la page complète de code. C'est la raison pour laquelle j'ai enlevé l'indentation ici.

Pour des raisons de sémantique, on essaiera tant que possible de se passer de cet élément HTML qui ne possède pas de sens en soi mais qui sert justement à conserver des mises en forme… Ce qui est plutôt le rôle du CSS normalement. On préfèrera tant que possible utiliser le CSS pour ce genre d'opérations de mise en forme.

Les entités HTML

Une entité HTML est une suite de caractère qui est utilisée pour afficher un caractère réservé ou un caractère invisible (comme un espace) en HTML.

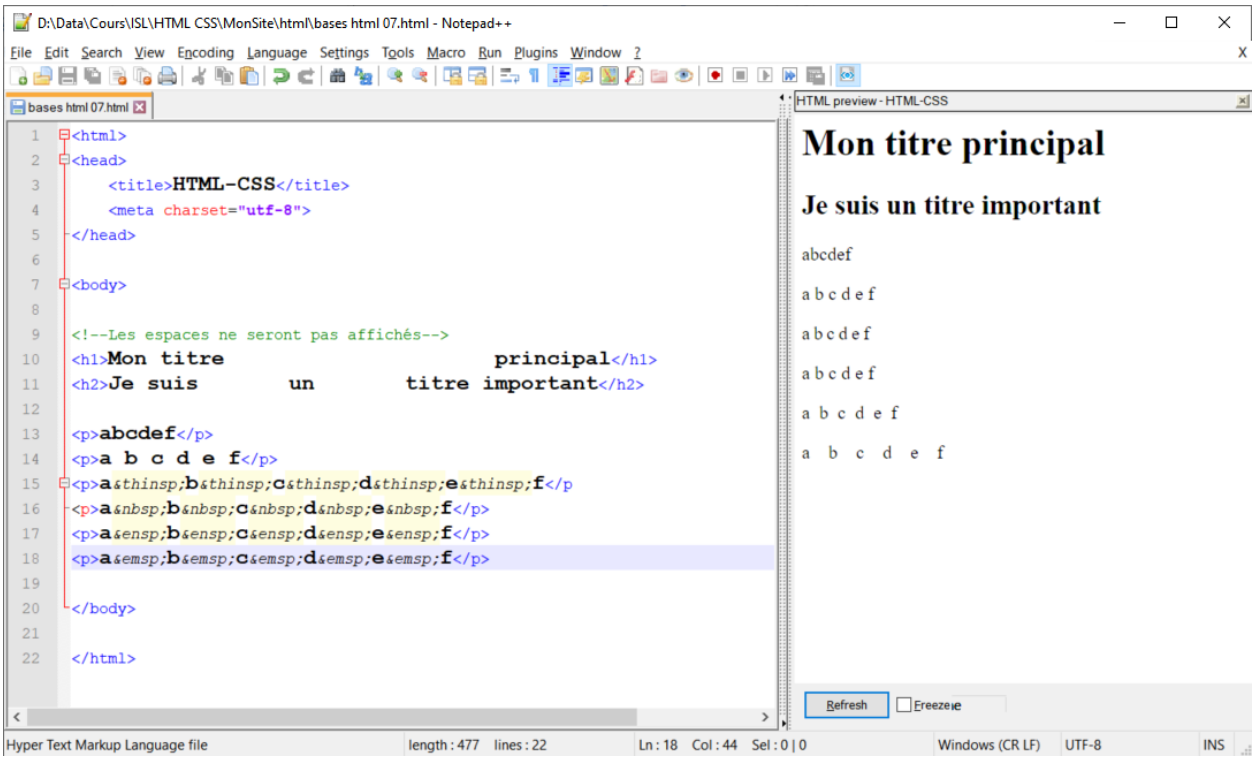
Qu'est-ce qu'un caractère réservé ? C'est un caractère qui possède déjà une signification particulière en HTML. Par exemple, imaginions que l'on souhaite afficher le caractère < dans un texte.

On ne va pas pouvoir mentionner ce caractère tel quel dans notre éditeur car le navigateur va interpréter cela comme l'ouverture d'une balise d'un élément. Il va donc falloir indiquer au navigateur qu'on souhaite afficher le caractère < en tant que tel et non pas ouvrir une balise.

Pour cela, il va falloir échapper le sens de ce caractère, et c'est ce à quoi vont nous servir les entités HTML. Nous aurons l'occasion de reparler de ces entités HTML plus tard dans cette partie. Pour le moment, concentrons-nous sur celles qui nous intéressent à savoir celles qui vont nous permettre d'ajouter des espaces.

- L'entité HTML   (« non breaking space ») va nous permettre d'ajouter une espace simple dit espace « insécable » ;
- L'entité HTML   (« en space ») va nous permettre de créer une espace double ;

- L'entité HTML   (« em space ») va nous permettre de créer une espace quadruple ;
- L'entité HTML   (« thin space ») va nous permettre de créer un espace très fin (demi-espace).



Comme vous pouvez le voir, les espaces sont bien créés. Ici, je n'ai utilisé qu'une entité entre chaque caractère mais rien ne vous empêche d'en utiliser plusieurs d'affilée.

Notez cependant que l'utilisation des entités HTML à cette fin devrait toujours être une solution de dépannage et qu'on préférera généralement laisser toutes les questions de mise en page au CSS tant que possible.

Définir le niveau d'importance des contenus en HTML

Dans cette leçon, nous allons voir comment indiquer aux navigateurs et surtout aux moteurs de recherche quelles parties de texte sont plus importantes que d'autres et doivent être considérées en priorité.

La problématique des niveaux d'importance des textes

Pour comprendre la problématique des niveaux d'importance des textes, il faut avant tout que vous ayez des notions en optimisation du référencement (SEO) ou du moins que vous compreniez les enjeux liés au SEO.

Lorsqu'on crée un site, en général, on veut que ce site soit visité. Pour qu'il soit visité, l'un des meilleurs moyens est que les pages de notre site ressortent parmi les premiers résultats lorsqu'un utilisateur fait une recherche sur un moteur de recherche (comme Google) sur un sujet abordé dans nos pages.

Les techniques d'optimisation que l'on va pouvoir mettre en place pour faire en sorte que nos pages soient les mieux classées possibles dans Google (et dans les autres moteurs de recherche) par rapport à certains mots clefs sont regroupées sous le terme « SEO ».

Le contenu d'une page est aujourd'hui le critère SEO le plus important pour Google puisque le but de Google est de fournir la réponse la plus pertinente pour chaque requête de ses utilisateurs.

En ce sens, il va donc falloir porter une attention toute particulière à la manière dont on va rédiger chaque page si on veut que celles-ci soient optimisées. Ainsi, il va avant tout falloir cibler un mot clef principal pour chaque page et écrire à chaque fois du contenu pertinent par rapport à ce mot clef.

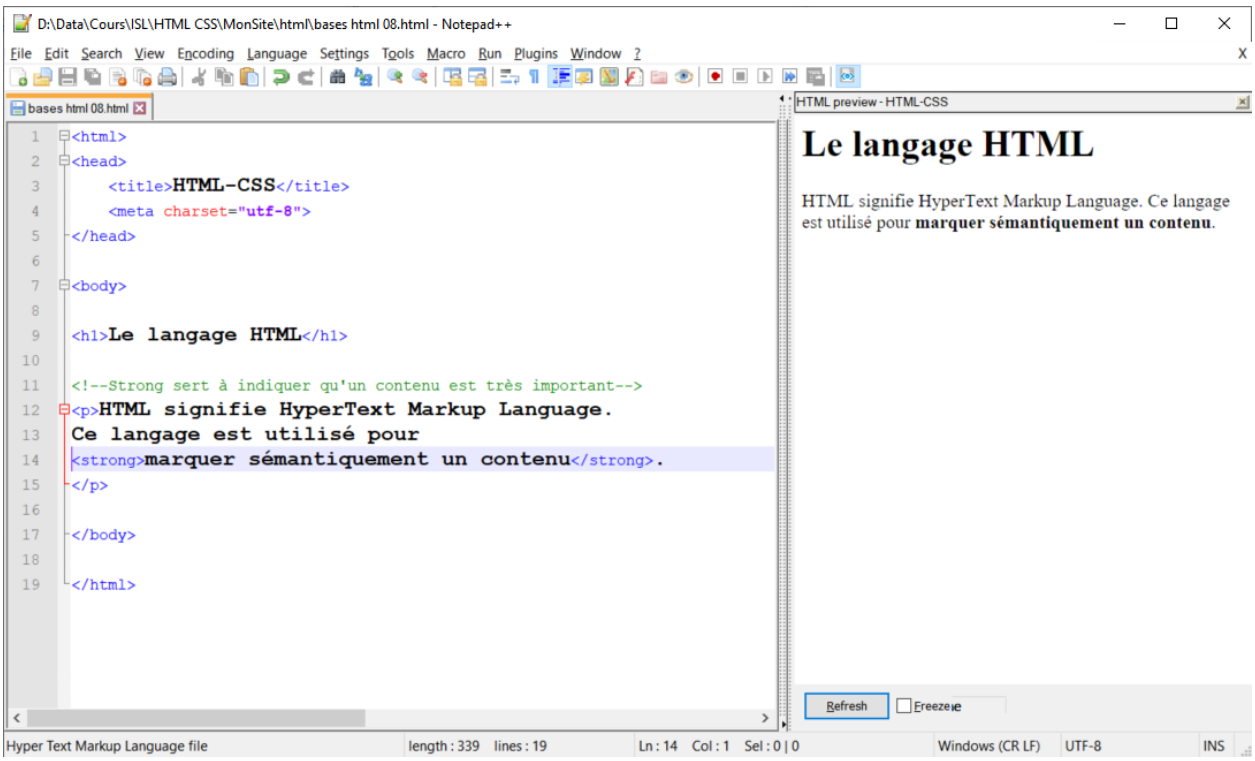
En écrivant nos textes, souvent, certaines parties de contenu vont être plus importantes que d'autres et on va donc vouloir que les moteurs de recherche les remarquent en priorité et les traitent avec plus d'importance que le reste de nos textes.

On va pouvoir faire cela grâce à différents éléments HTML qui vont nous servir à indiquer des niveaux d'importance relatifs pour certaines parties de notre contenu.

Indiquer qu'un texte est très important avec l'élément strong

L'élément HTML strong va être utilisé pour signifier qu'un contenu est très important et doit être considéré comme tel par les moteurs de recherche (et les navigateurs).

En résultat, le navigateur affichera par défaut le contenu à l'intérieur de l'élément strong en gras. Encore une fois, n'utilisez pas strong pour mettre un texte en gras ! Utilisez strong pour marquer un texte qui vous semble très important. Nous utiliserons le CSS pour gérer le poids d'un texte.

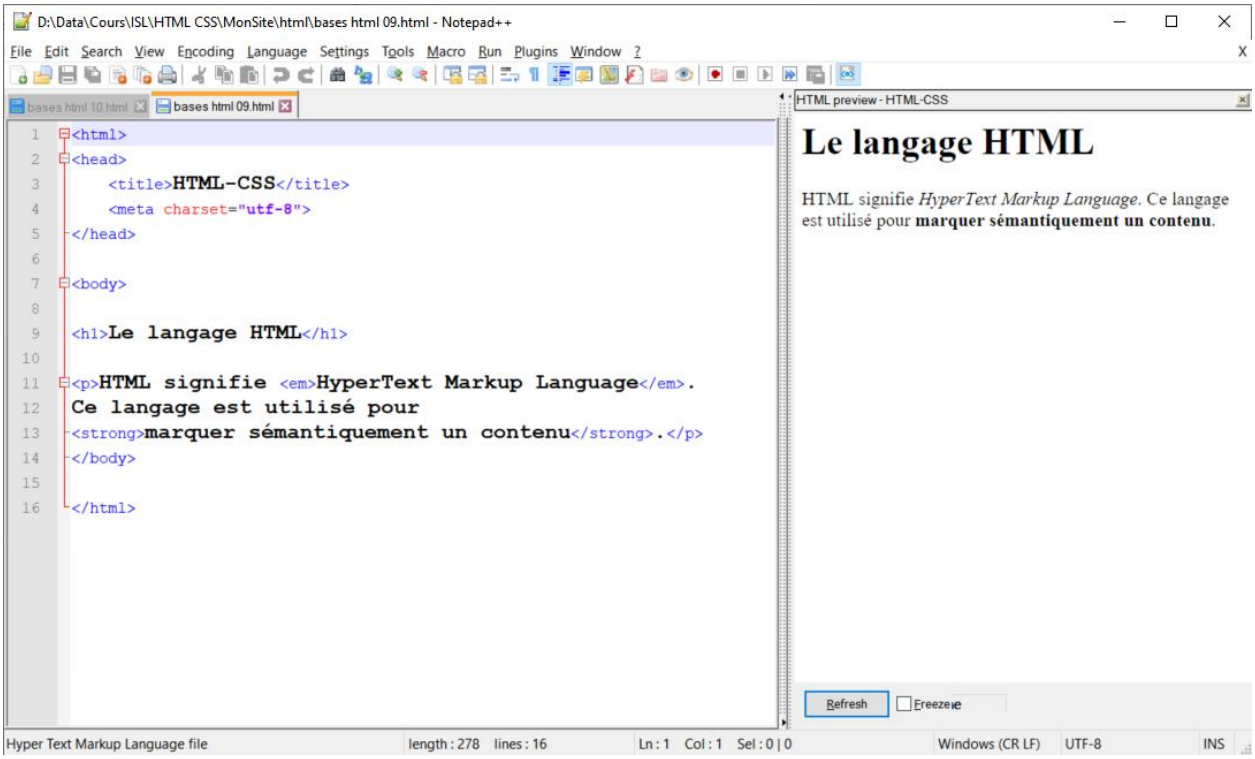


Mettre un texte en emphase avec l'élément em

L'élément HTML em (pour emphasis ; « emphase » ou « accentuation » en français) va être utilisé pour mettre des termes en emphase.

On va pouvoir utiliser cet élément pour souligner un contraste par exemple ou une définition. Concrètement, les moteurs de recherche vont accorder une importance moins grande aux textes dans les éléments em qu'à ceux dans strong mais une importance plus grande à ces textes qu'à des simples paragraphes.

Le résultat visuel par défaut de l'emphase est la mise en italique du texte contenu dans l'élément.



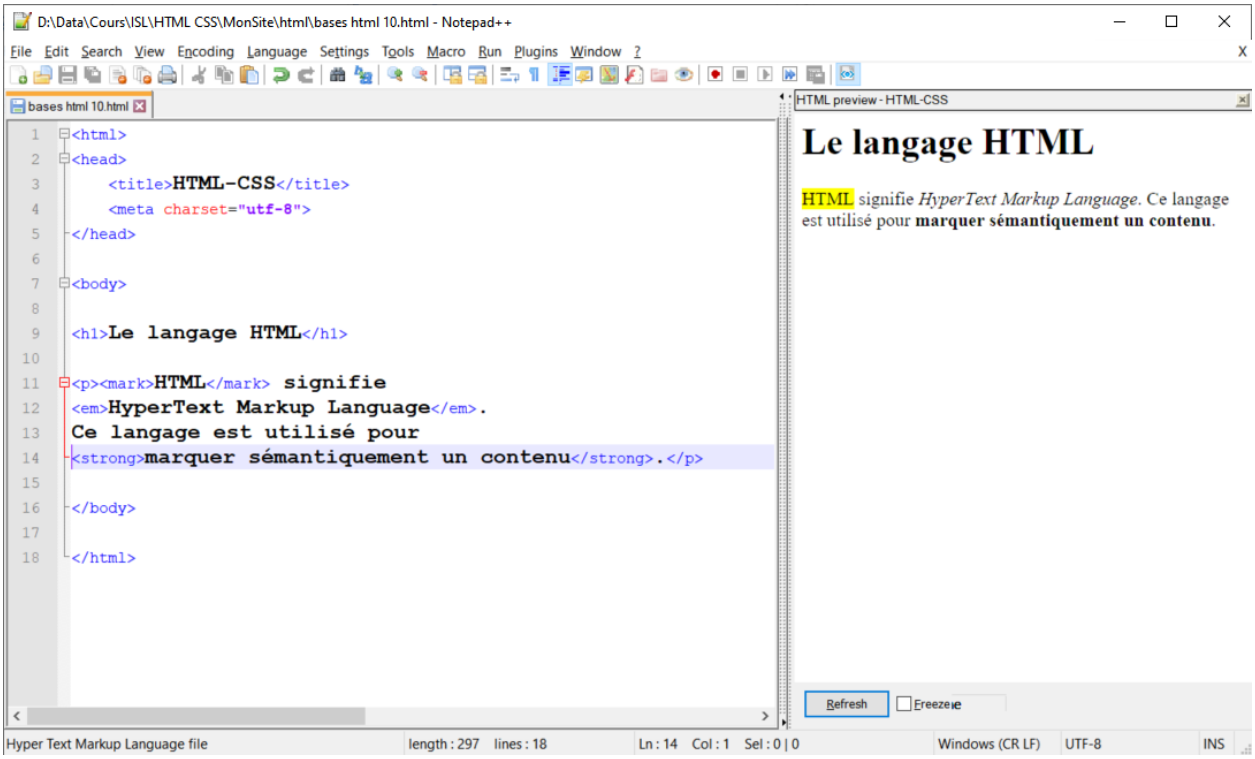
Mettre un contenu pertinent en relief avec l'élément mark

L'élément mark va être utilisé pour mettre en relief certains textes qui vont être pertinents dans un certain contexte. Le texte mis en relief n'est pas forcément important en soi par rapport au sujet de l'article mais va être pertinent pour un certain utilisateur dans un certain contexte.

Il est difficile d'illustrer l'intérêt de cet élément pour le moment car celui-ci va souvent être utilisé de manière dynamique et avec des langages dynamiques donc comme du JavaScript.

Imaginons par exemple que votre site possède un champ de recherche. Lorsque l'utilisateur recherche un terme dans une page, ce terme en particulier va donc être important pour lui et on va donc vouloir le mettre en relief par rapport aux autres.

L'élément de marquage à utiliser dans cette situation-là est l'élément mark dont le rôle est encore une fois d'indiquer la pertinence d'un certain texte en fonction d'un contexte.



De manière pratique, toutefois, l'élément mark est relativement peu utilisé.

Créer des listes en HTML

Dans cette nouvelle leçon, nous allons voir à quoi servent les listes, découvrir les différents types de listes en HTML et apprendre à en créer.

Qu'est-ce qu'une liste HTML ? A quoi servent les listes ?

Le HTML est un langage de sémantique : son rôle est de donner du sens aux différents contenus d'une page afin que ceux-ci soient correctement reconnus et afficher et que les navigateurs et les moteurs de recherche « comprennent » nos pages.

Les listes HTML répondent tout à fait à cet objectif puisqu'elles permettent d'ordonner du contenu. Ce contenu peut être hiérarchisé ou non.

Une liste en HTML est composée de différents éléments de listes. Visuellement une liste va ressembler à ceci :

- Premier élément de ma liste,
- Deuxième élément de ma liste,
- Troisième élément de ma liste.

Les listes vont ainsi nous permettre de lister plusieurs éléments en les groupant sous un dénominateur commun qu'est la liste en soi. Les navigateurs et les moteurs de recherche vont donc comprendre qu'il y a une relation entre les différents éléments de liste.

Les listes vont donc déjà être très utiles pour apporter de la clarté et de l'ordre à nos documents. En plus de cela, nous allons également utiliser des listes HTML pour créer des menus de navigation (nous verrons comment plus tard dans ce cours).

Il existe deux grands types de listes en HTML : les listes ordonnées et les listes non-ordonnées. Il existe également un troisième type de liste un peu particulier et moins utilisé : les listes de définitions.

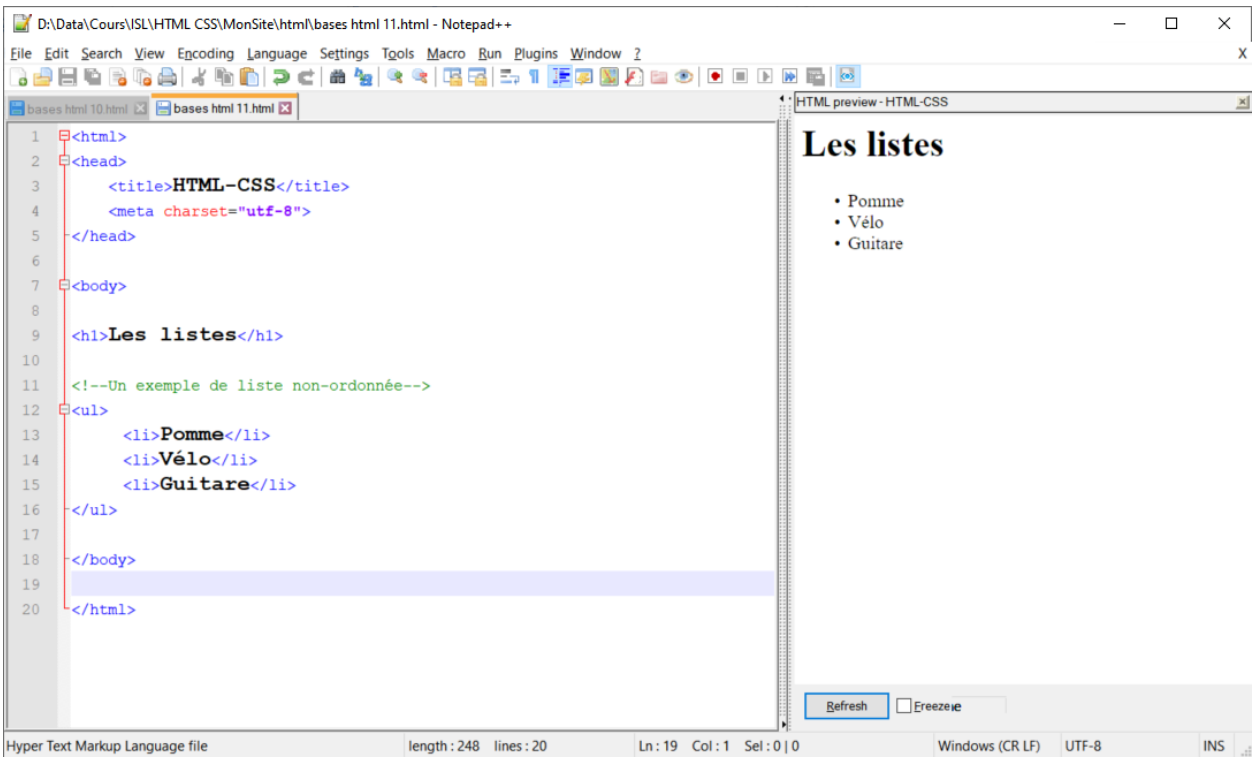
Les listes non ordonnées

Les listes non-ordonnées vont être utiles pour lister des éléments sans hiérarchie ni ordre logique.

Par exemple, si je souhaite lister les mots « pomme », « vélo » et « guitare », sans plus de contexte, j'utiliserai une liste non-ordonnée.

En effet, on ne peut pas dégager de notion d'ordre, de hiérarchie ou de subordination entre ces trois termes (du moins pas sans un contexte précis).

Pour créer une liste non-ordonnée, nous allons avoir besoin d'un élément ul (pour « unordered list », ou « liste non-ordonnée » en français) qui va représenter la liste en soi ainsi que d'un élément li (« list item » ou « élément de liste ») pour chaque nouvel élément de liste.



Comme vous pouvez le remarquer, on va placer les éléments li à l'intérieur de l'élément de liste ul (à partir de ce moment, bien indenter son code commence à être important pour ne pas se perdre). C'est tout à fait logique puisque les éléments de liste « appartiennent » à une liste en particulier.

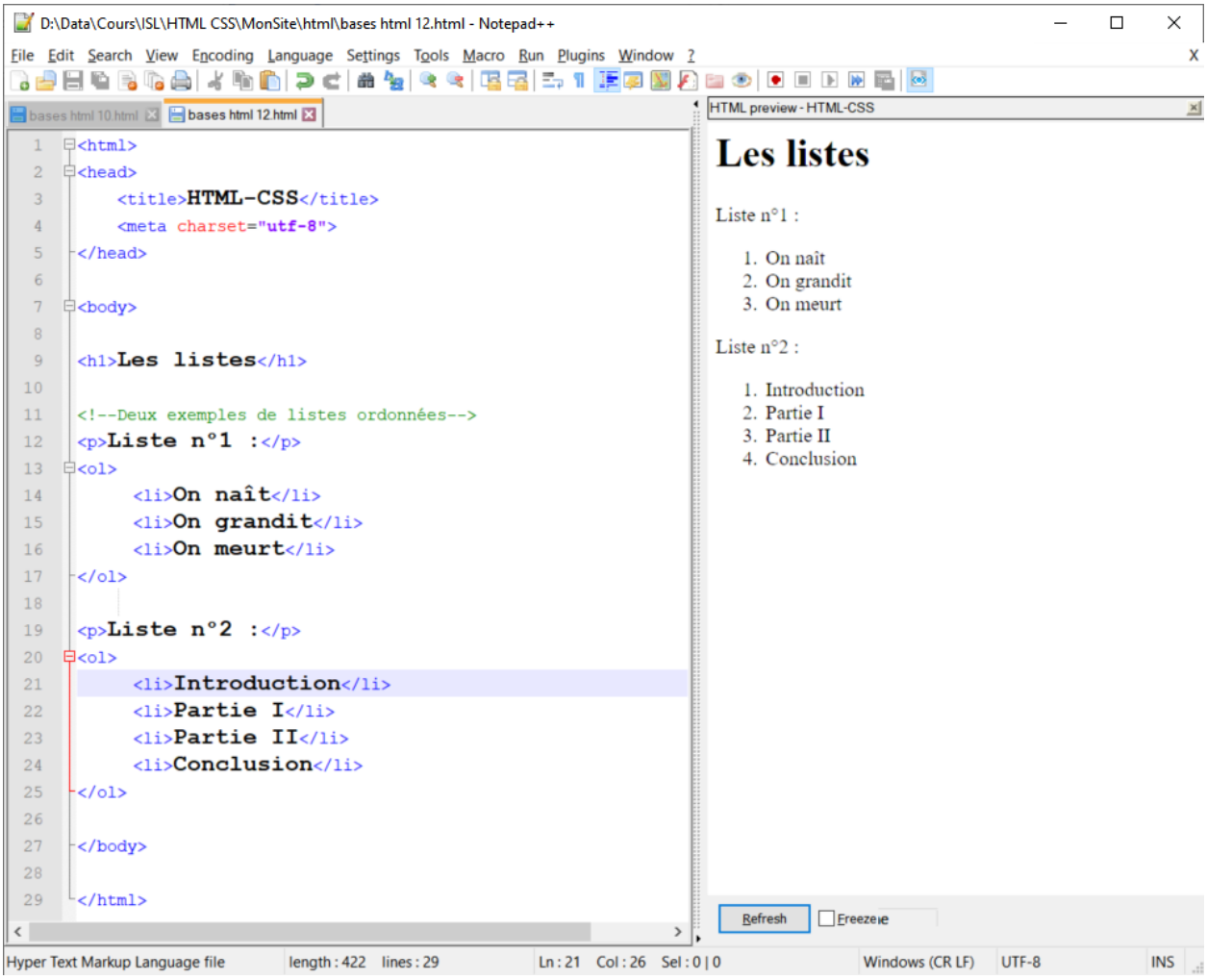
Visuellement, des puces (les points noirs) apparaissent automatiquement devant chaque élément d'une liste non-ordonnée par défaut. Nous allons pouvoir changer ce comportement et personnaliser l'apparence de nos listes en CSS grâce notamment à la propriété list-style-type que nous étudierions plus tard dans ce cours.

Les listes ordonnées

Au contraire des listes non-ordonnées, nous allons utiliser les listes ordonnées lorsqu'il y aura une notion d'ordre ou de progression logique ou encore de hiérarchie entre les éléments de notre liste.

Par exemple, si l'on souhaite lister les étapes naturelles de la vie, ou lorsque l'on crée un sommaire, on utilisera généralement des listes ordonnées.

Pour créer une liste ordonnée, nous allons cette fois-ci utiliser l'élément ol (pour « ordered list » ou « liste ordonnée ») pour définir la liste en soi et à nouveau des éléments li pour chaque élément de la liste.



Comme vous le voyez, ce sont cette fois-ci des numéros qui sont affichés devant chaque élément de la liste par défaut.

Encore une fois, nous allons pouvoir changer ce comportement et afficher différents styles de puces avec la propriété CSS list-style-type.

Les attributs HTML des listes ordonnées

En plus des propriétés CSS, on va pouvoir utiliser certains attributs HTML avec nos listes ordonnées pour modifier leur présentation.

Ici, vous devez absolument comprendre que ces attributs ne devraient pas exister à priori puisque le HTML ne devrait pas se soucier de la présentation mais devrait laisser l'apparence au CSS.

Cependant, les langages sont en constante évolution ce qui signifie que leur rôle n'est pas encore tout à fait fixé et que certains langages ne sont pas capables de faire certaines choses aujourd'hui. Ce sont les raisons principales pour l'existence de ces attributs que je vais vous présenter par souci d'exhaustivité.

Retenez simplement que vous devriez toujours privilégier l'utilisation du langage CSS pour la mise en forme des contenus HTML tant que vous le pouvez.

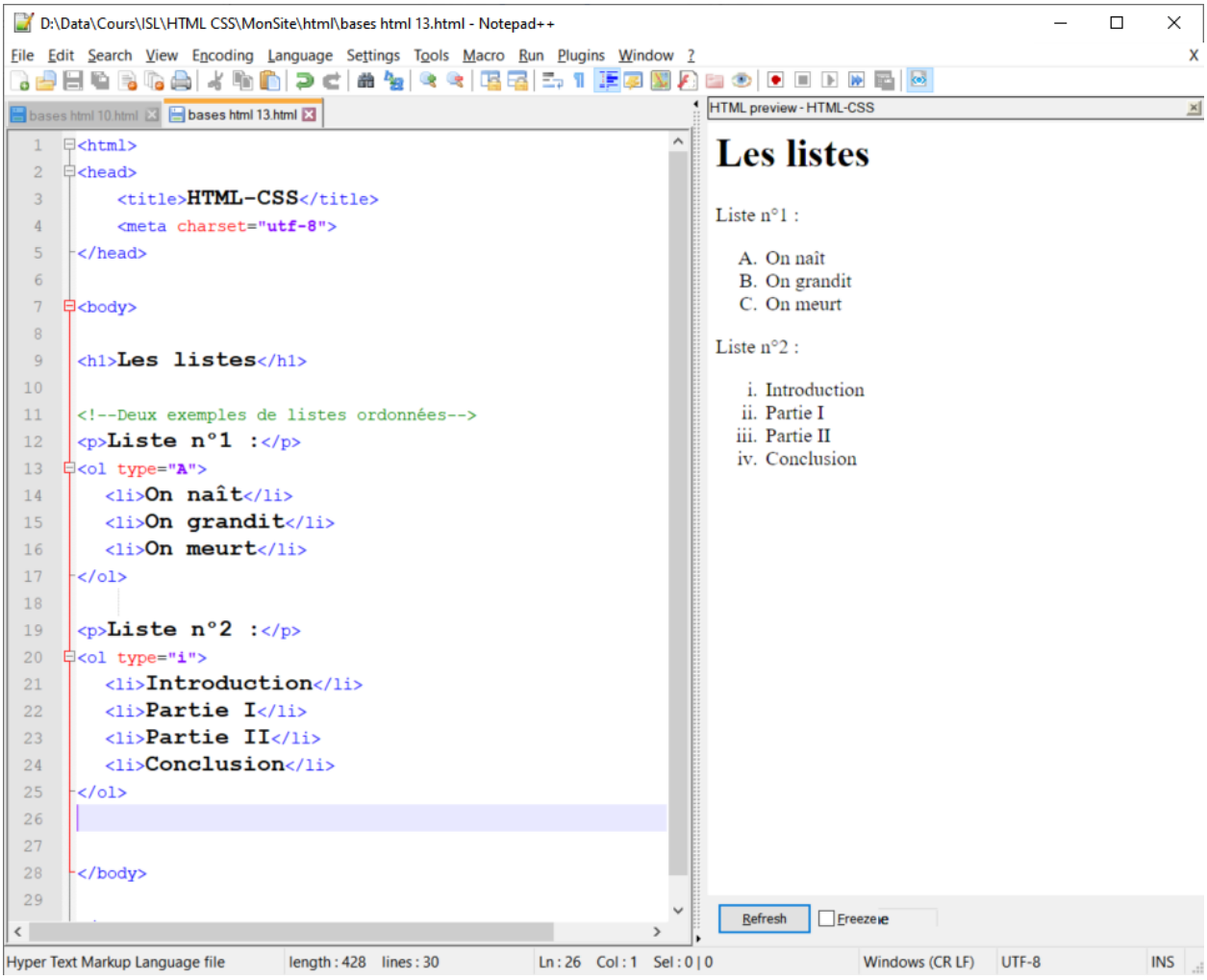
Commençons avec l'attribut type qui va nous permettre de changer l'apparence des puces d'une liste ordonnée.

Cet attribut existait autrefois également pour les listes non ordonnées mais a été déprécié pour celles-ci. Il est possible qu'il soit également déprécié dans le futur pour les listes ordonnées puisque le HTML ne devrait pas se charger de la mise en forme.

Pour cette raison, je vous déconseille de l'utiliser et vous conseille de préférer l'utilisation du CSS. Cependant, je vous présente tout de même les valeurs qu'on va pouvoir lui fournir par souci d'exhaustivité :

- « 1 » : valeur par défaut. Des chiffres apparaitront devant chaque élément de la liste ;
- « I » : Des chiffres romains majuscules apparaitront devant chaque élément de la liste ;
- « i » : Des chiffres romains minuscules apparaitront devant chaque élément de la liste ;
- « A » : Des lettres majuscules apparaitront devant chaque élément de la liste ;
- « a » : Des lettres minuscules apparaitront devant chaque élément de la liste.

Voici également un exemple d'utilisation de cet attribut :



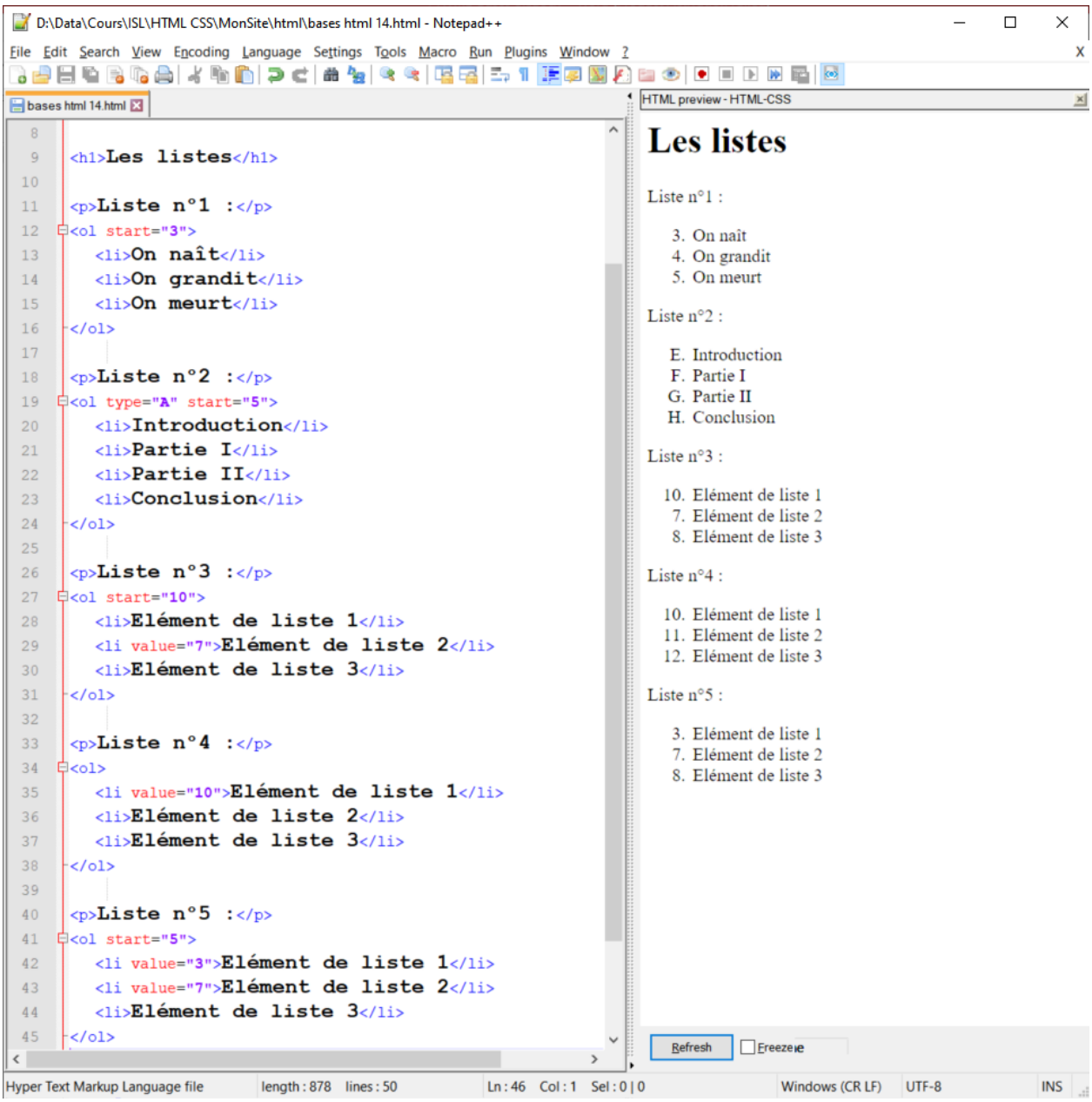
Notez que par défaut le premier élément d'une liste ordonnée va avoir comme puce le chiffre « 1 » ou la première lettre de l'alphabet « a ». On va pouvoir changer ce comportement et faire démarrer notre liste ordonnée à partir d'un point choisi grâce aux attributs HTML start ou value.

L'attribut start va nous permettre de choisir un point de départ pour notre liste ordonnée. On va donc le placer dans la balise ouvrante de l'élément représentant la liste ol.

L'attribut value va lui en revanche nous permettre de choisir la valeur de chaque puce d'éléments de liste. On va pouvoir ajouter un attribut value pour chaque élément li. Dans le cas où certains éléments li ne possèderaient pas d'attribut value, la valeur de leur puce va s'incrémenter normalement par rapport à la valeur de la puce de l'élément précédent (c'est-à-dire ajouter 1 par rapport à la puce précédente).

Notez que dans le cas où un attribut start est précisé pour la liste et un attribut value est précisé pour le premier élément de liste, l'attribut value, plus précis, va avoir la priorité et imposer sa valeur.

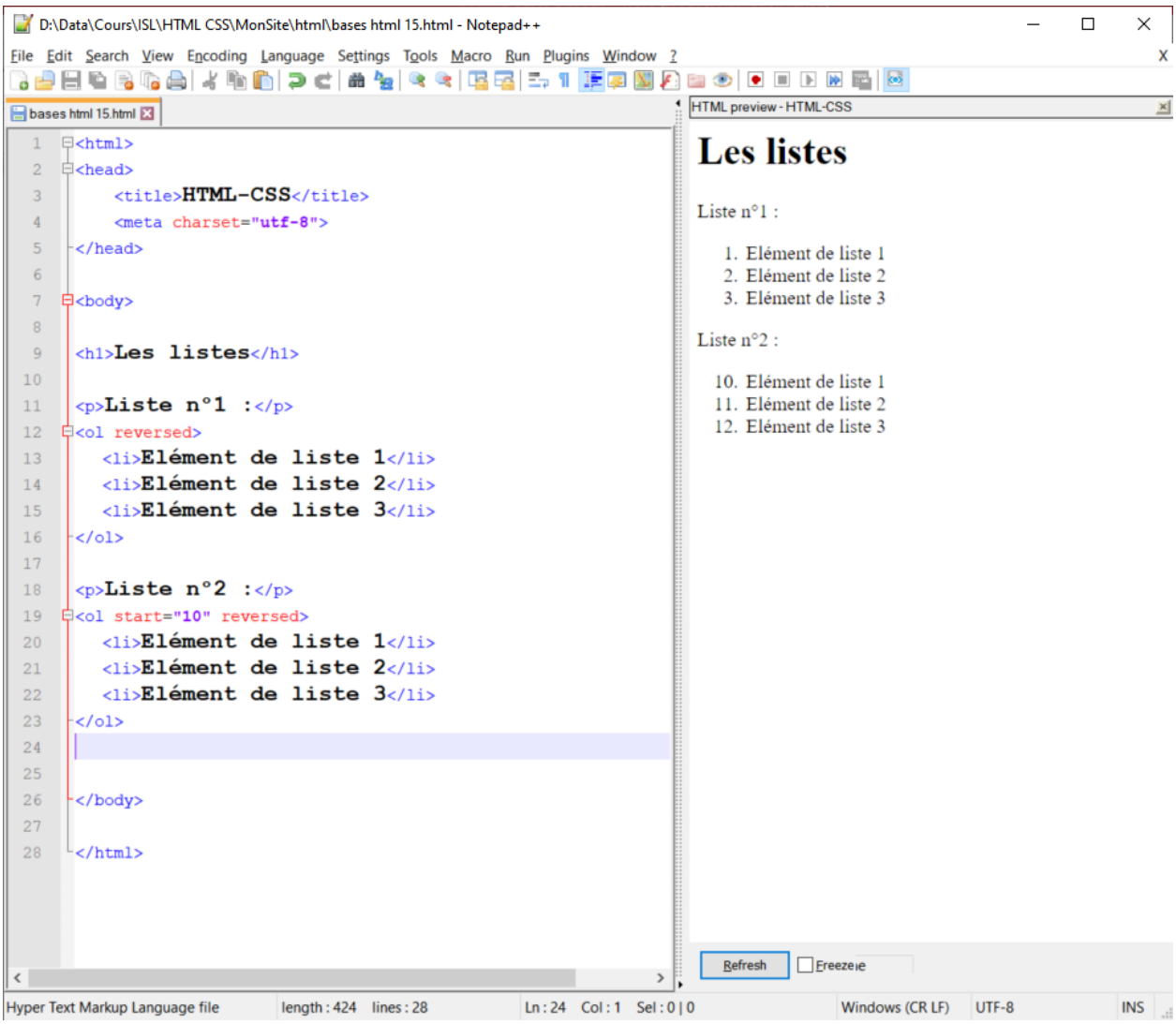
Voici quelques exemples d'utilisation de ces attributs :




```
<h1>Les listes</h1>

<p>Liste n°1 :</p>
<ol start="3">
   <li>On naît</li>
   <li>On grandit</li>
   <li>On meurt</li>
</ol>

<p>Liste n°2 :</p>
<ol type="A" start="5">
   <li>Introduction</li>
   <li>Partie I</li>
   <li>Partie II</li>
   <li>Conclusion</li>
</ol>

<p>Liste n°3 :</p>
<ol start="10">
   <li>Elément de liste 1</li>
   <li value="7">Elément de liste 2</li>
   <li>Elément de liste 3</li>
</ol>

<p>Liste n°4 :</p>
<ol>
   <li value="10">Elément de liste 1</li>
   <li>Elément de liste 2</li>
   <li>Elément de liste 3</li>
</ol>

<p>Liste n°5 :</p>
<ol start="5">
   <li value="3">Elément de liste 1</li>
   <li value="7">Elément de liste 2</li>
   <li>Elément de liste 3</li>
</ol>
```

Finalement, nous allons pouvoir inverser le compte des puces des éléments de liste ordonnées grâce à l'attribut reversed. Le premier élément de la liste aura alors la puce avec la valeur la plus élevée, puis on enlèvera un par nouvel élément jusqu'à arriver à « 1 » ou « a » pour le dernier élément de liste par défaut.

L'attribut reversed ne possède qu'une valeur qui est reversed (identique au nom de l'attribut, comme pour tous les attributs qui ne possèdent qu'une valeur en HTML). Comme reversed ne possède qu'une valeur, la valeur est dite évidente et peut être omise.

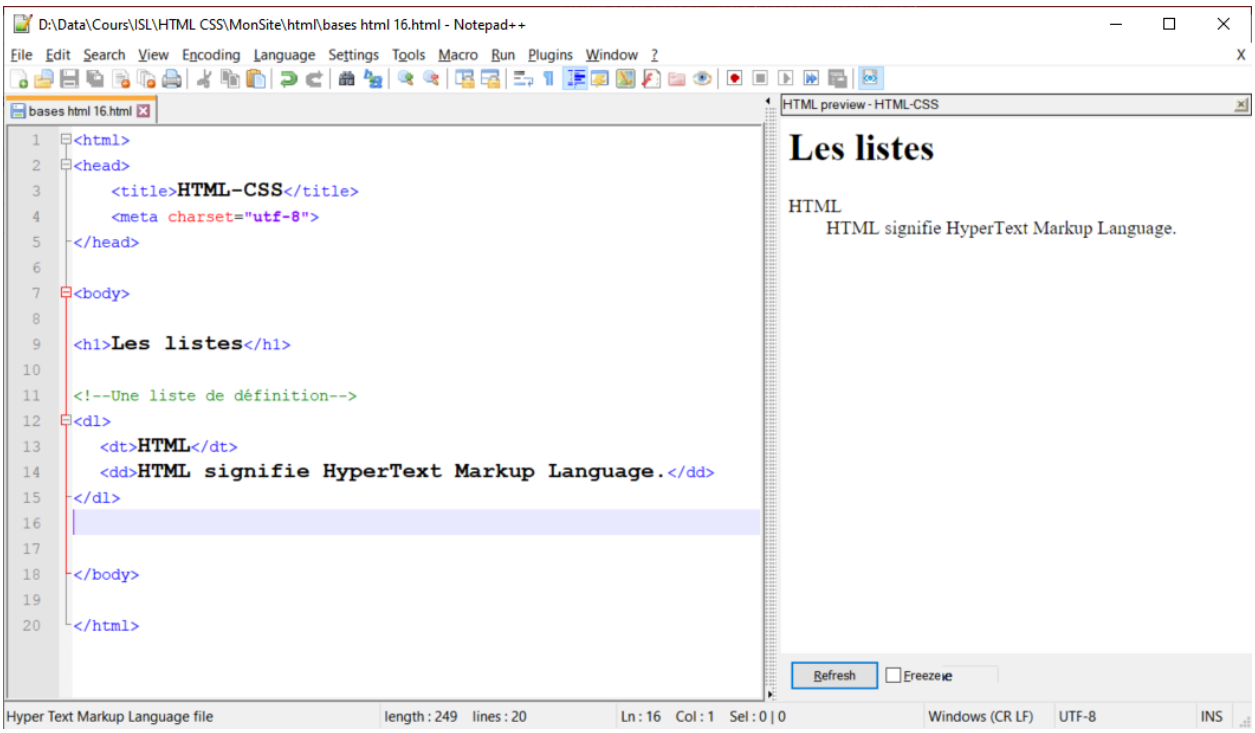


Les listes de définitions

Les listes de définitions, encore appelées « listes de descriptions » vont nous permettre de lister des termes et d'ajouter des définitions ou descriptions pour chacun de ces termes.

Pour créer une liste de définitions, nous allons cette fois-ci utiliser l'élément dl signifiant « description list » ou « liste de description / définition » en français pour définir la liste en soi, puis des éléments dt (description term) pour chaque élément à décrire et enfin l'élément dd pour la définition / description en soi.

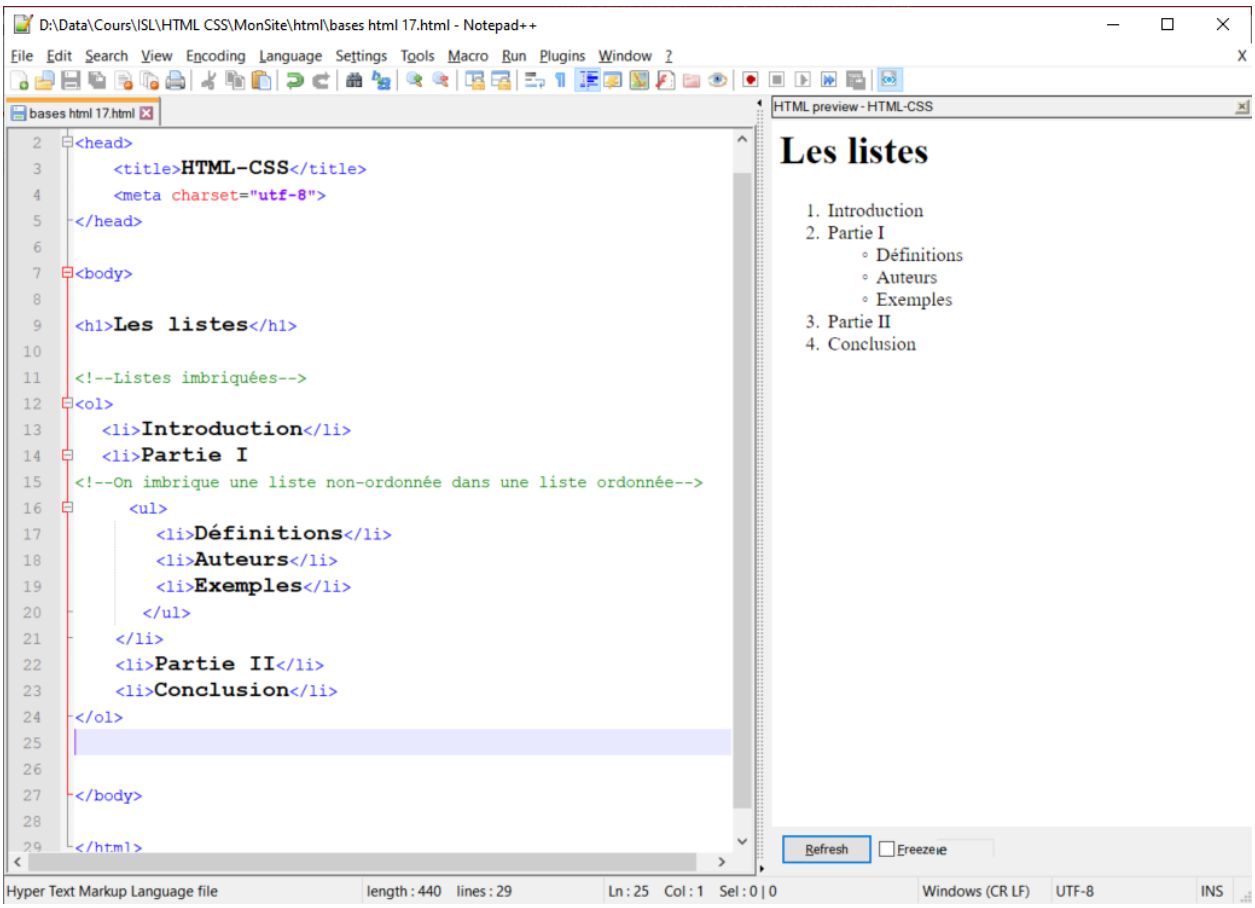
Pensez bien lorsque vous créez une liste de définitions à toujours placer le terme à définir avant sa définition, c'est-à-dire l'élément dt avant l'élément dd. Cela est normalement assez intuitif.



L'imbrication de listes

Finalement, sachez qu'il est tout-à-fait possible d'imbriquer une liste dans une autre en suivant quelques règles simples.

Pour imbriquer une liste dans une autre, il suffit de définir une nouvelle liste à l'intérieur de l'un des éléments d'une autre liste, juste avant la balise fermante de cet élément.




```
<head>
    <title>HTML-CSS</title>
    <meta charset="utf-8">
</head>

<body>

<h1>Les listes</h1>

<!--Listes imbriquées-->
<ol>
   <li>Introduction</li>
   <li>Partie I
<!--On imbrique une liste non-ordonnée dans une liste ordonnée-->
     <ul>
        <li>Définitions</li>
        <li>Auteurs</li>
        <li>Exemples</li>
      </ul>
    </li>
    <li>Partie II</li>
    <li>Conclusion</li>
</ol>


</body>

</html>
```

Comme vous pouvez le voir, il devient ici très important de bien indenter son code afin de ne pas se perdre au milieu de nos listes !

Notez que l'on peut imbriquer autant de listes que l'on souhaite les unes dans les autres. Cependant, pour des raisons évidentes de lisibilité, il est conseillé de ne pas créer plus de niveaux de listes que ce qui est strictement nécessaire pour servir vos besoins.

Créer des liens en HTML

Les liens hypertextes sont l'une des fondations du HTML que est, rappelons-le, un langage de marquage hypertexte justement.

Dans cette nouvelle leçon, nous allons découvrir et expliquer à quoi correspond un lien en HTML et allons apprendre à créer différents « types » de liens en HTML, que ce soit des liens ramenant à un autre endroit d'une même page (liens ancres), des liens menant vers d'autres pages d'un même site (liens internes) ou des liens envoyant l'utilisateur vers un autre site (liens externes).

Définition d'un lien HTML

Les liens en HTML vont nous servir à créer des ponts entre différentes pages d'un même site ou de sites différents. Le principe d'un lien est le suivant : en cliquant sur une ancre (qui est la partie visible par l'utilisateur d'un lien et qui peut être un texte comme une image), nos utilisateurs vont être redirigés vers une page cible.

Il existe deux types principaux de liens hypertextes en HTML :

- Les liens internes qui vont servir à naviguer d'une page à l'autre dans un même site ;
- Les liens externes qui vont envoyer les utilisateurs vers une page d'un autre site.

Ces deux types de liens vont être l'objet d'enjeux et d'implications différents en termes d'optimisation du référencement (SEO) et vont pouvoir être créés de façons différentes en HTML.

Nous allons également pouvoir utiliser les liens en HTML pour naviguer au sein d'une même page et renvoyer à un endroit précis de celle-ci. Cela est utile pour fournir des accès ou des repères rapides à nos utilisateurs dans le cas d'un page très longue.

Nous appelons ce type de liens faits au sein d'une même page des liens « ancres » tout simplement (même si ce terme peut porter à confusion car le terme « ancre » est aussi utilisé pour définir la partie visible et cliquable d'un lien).

Dans chacun de ces cas, nous allons devoir utiliser l'élément HTML a accompagné de son attribut href pour créer un lien en HTML.

Notez que les liens sont également un aspect fondamental d'une stratégie d'optimisation de référencement (SEO), que ce soit dans le cas de liens internes (pour effectuer ce qu'on appelle un maillage interne et donner plus de valeur à certaines de nos pages aux yeux de Google) ou de liens externes qui sont un des critères principaux de classement pour Google. Il est donc très important de bien les comprendre et les maitriser !

Création de liens : l'élément a et son attribut href

Quel que soit le type de liens que l'on souhaite créer en HTML (liens internes, des liens externes, ou des liens vers un autre endroit d'une même page), nous utiliserons toujours l'élément a qui est l'abréviation

de « anchor » ou « ancre » en français accompagné de son attribut href pour « hypertext reference » ou « référence hypertexte » en français.

L'élément HTML a est composé d'une paire de balises (balises ouvrante et fermante) et d'un contenu entre les balises que l'on va appeler "ancre". Ce contenu peut être un texte, une image, etc. et sera la partie visible et cliquable du lien pour les utilisateurs.

L'attribut href va nous servir à indiquer la cible du lien, c'est-à-dire l'endroit où l'utilisateur doit être envoyé après avoir cliqué sur le lien. Nous allons indiquer cette cible en valeur de l'attribut href.

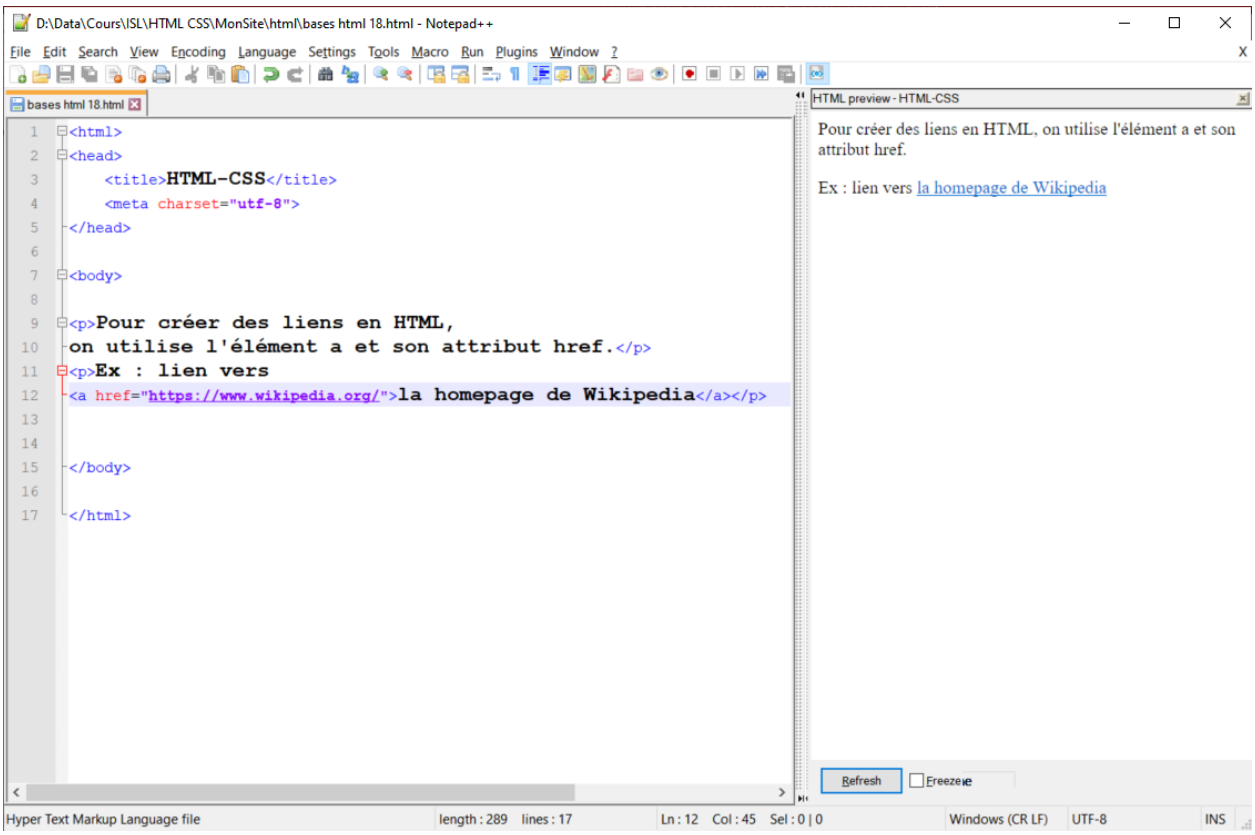
D'un point de vue du code, la seule chose qui va changer pour créer un lien ancre plutôt qu'interne ou interne plutôt qu'externe va être la façon dont on va construire la valeur qu'on va passer à l'attribut href.

Créer des liens externes en HTML

Un lien externe est un lien qui part d'une page d'un site et ramène les utilisateurs vers une autre page d'un autre site.

Pour créer un lien vers une page d'un autre site en HTML (ou lien externe), il va falloir indiquer l'adresse complète de la page (c'est-à-dire son URL) en valeur de l'attribut href de notre lien.

Imaginons par exemple que nous voulions créer un lien vers la page d'accueil de Wikipédia. L'URL de cette page est la suivante : https://www.wikipedia.org/.



Comme vous le voyez, on utilise bien un élément a pour créer un lien externe en HTML. On place notre attribut href au sein de la balise ouvrante de cet élément. Ici, on précise en valeur l'adresse (URL) de la page d'accueil de Wikipédia.

Entre les deux balises, nous plaçons la partie visible et cliquable (l'ancre) de notre lien. Ici, cela correspond au texte « la home de Wikipédia ». Les utilisateurs vont pouvoir cliquer sur ce texte pour être renvoyés vers la page d'accueil de Wikipedia.

Note : Ici, nous avons choisi de placer un texte comme ancre de notre lien, mais rien ne nous empêche de placer une image à la place afin de créer une image cliquable.

Vous pouvez remarquer que le navigateur applique automatiquement des styles à la partie cliquable de nos liens :

- le texte (notre ancre) est de couleur différente (bleu avant de cliquer puis volet une fois le lien visité) ;
- le texte est souligné ;
- le curseur de notre souris change de forme lorsqu'on passe sur le lien.

Nous allons bien évidemment pouvoir changer ces comportements en appliquant nos propres styles CSS à nos éléments a. Nous verrons comment faire cela plus tard dans ce cours.

Bon à savoir : Lorsque l'attribut href prend une URL complète en valeur, on parle de valeur absolue (car celle-ci est fixe et ne dépend de rien). Les liens externes utilisent toujours des valeurs absolues. On parle

de valeur absolue en opposition aux valeurs dites relatives, qui sont dans ce contexte des valeurs qui vont indiquer l'emplacement de la page cible relativement à la page source du lien.

Créer des liens internes en HTML

Le deuxième grand type de liens que l'on va pouvoir créer en HTML correspond aux liens internes, c'est-à-dire à des liens renvoyant vers d'autres pages d'un même site.

Explication du fonctionnement des liens internes

Nous allons avoir plusieurs solutions pour créer des liens internes. Tout d'abord, nous allons tout simplement pouvoir faire exactement comme pour les liens externes et indiquer l'adresse complète (adresse absolue) de la page cible du lien en valeur de l'attribut href

Cependant, si cette première manière de faire semble tout à fait fonctionner et être la plus simple à priori, ce n'est pas du tout la plus optimisée d'un point de vue de l'utilisation des ressources et elle peut même s'avérer problématique selon comment votre site est construit (notamment si vos URLs sont construites dynamiquement).

Pour ces raisons, nous préciserons généralement plutôt une valeur de type relatif en valeur de l'attribut href de nos liens internes en HTML.

On dit que la valeur est relative car on va devoir indiquer l'adresse de la page de destination relativement à l'adresse de la page de départ (c'est-à-dire celle à partir de laquelle on fait notre lien).

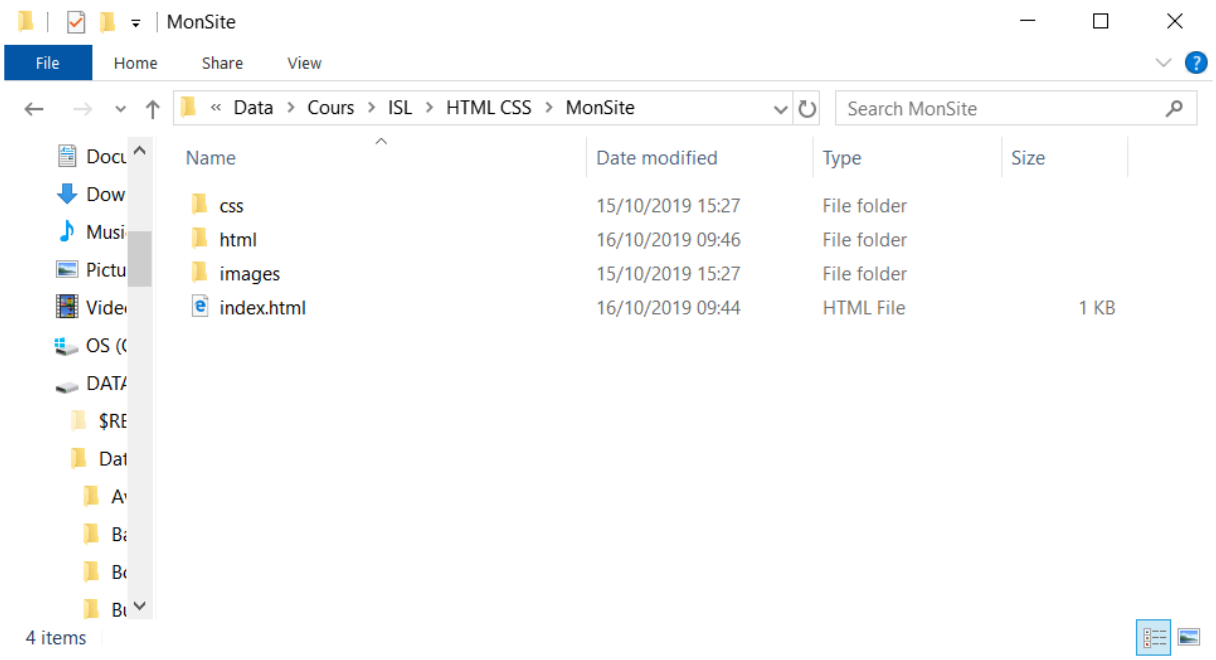
Comment savoir quelle valeur relative utiliser ? Pour bien comprendre comment fonctionnent les valeurs relatives, vous devez avant tout savoir ce qu'est un site web.

Un site web n'est qu'un ensemble de fichiers et de ressources (pages de code de différents types et fichiers médias comme des images, etc.) liés entre eux.

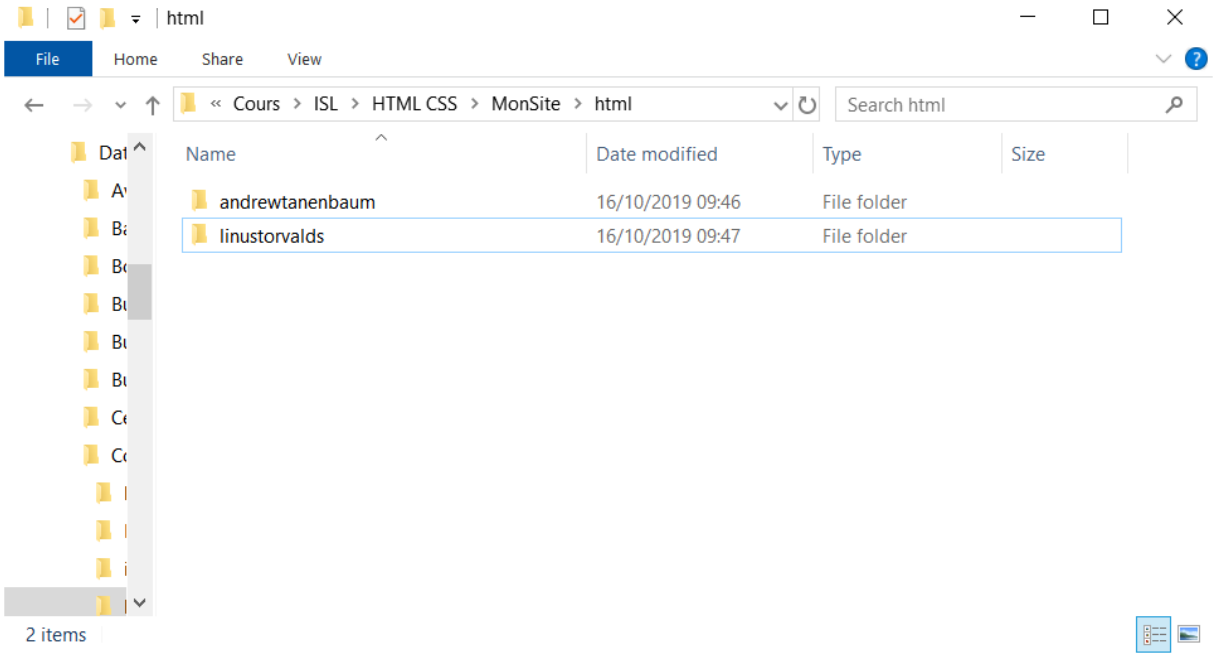
Toutes ces ressources vont être hébergées (stockées) sur un ordinateur très puissant et constamment connecté à Internet qu'on appelle serveur, dans un dossier principal qu'on va appeler la racine d'un site.

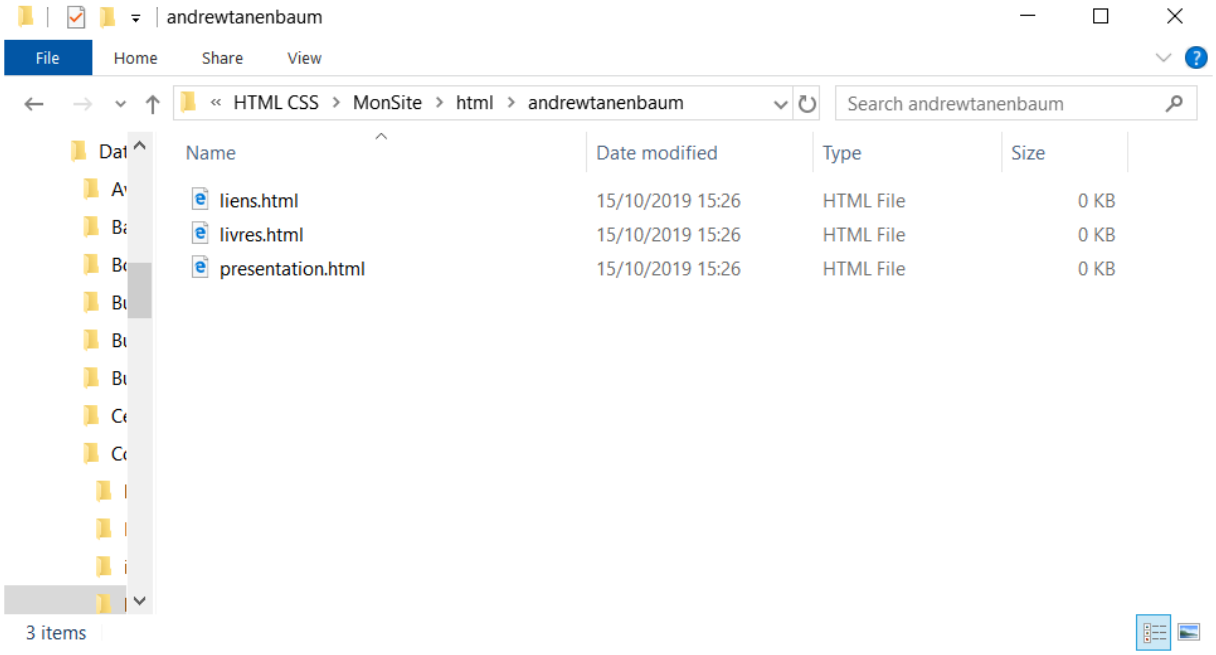
Pour pouvoir stocker nos différentes ressources sur un serveur et faire en sorte que notre site soit tout le temps accessible via Internet, nous passons généralement par un hébergeur qui va nous louer un serveur ou une partie d'un de ses serveurs.

Dans un dossier racine, il va très souvent y avoir d'autres dossiers (des sous-dossiers donc), des fichiers, des images, etc.



Vous pouvez voir différents fichiers et ressources et des dossiers. En cliquant sur le dossier « html» par exemple, on va trouver de nouveaux fichiers/dossiers :





Pour lier ces différents fichiers et ressources entre eux (et donc offrir entre autres à nos utilisateurs la possibilité de naviguer de l'un à l'autre), nous allons utiliser des valeurs relatives, c'est-à-dire que nous allons préciser l'emplacement de la ressource que l'on souhaite utiliser (ou vers laquelle on souhaite renvoyer) relativement à l'emplacement de la page / ressource qui souhaite l'utiliser (la page de départ).

C'est comme cela que nous allons procéder dans le cas précis de la création de liens internes : nous allons préciser en valeur de l'attribut href l'emplacement de la page cible (page de destination) par rapport à l'emplacement sur le serveur de la page source.

Trois cas vont alors se présenter à nous :

1. Le cas où les deux pages (source et destination) se trouvent dans un même dossier ;
2. Le cas où la page de destination se trouve dans un sous dossier par rapport à la page source ;
3. Le cas où la page de destination se trouve dans un dossier parent par rapport à la page source.

Pour chacun de ces cas, nous allons construire la valeur de notre attribut href de manière différente :

- Si les deux pages se situent dans le même dossier, alors on pourra se contenter de préciser le nom de la page cible/de destination (avec son extension) en valeur de l'attribut href ;
- Si la page cible se situe dans un sous dossier par rapport à la page à partir de laquelle on fait un lien, alors on précisera le nom du sous dossier suivi d'un slash suivi du nom de la page cible en valeur de l'attribut href ;
- Si la page cible se situe dans un dossier parent par rapport à la page de départ, alors il faudra indiquer deux points suivi d'un slash../ suivi du nom de la page de destination en valeur de l'attribut href.

Exemples pratiques de création de liens internes

Pour illustrer cela, je vous propose de créer un premier dossier sur le bureau de votre ordinateur. Nous imaginerons que ce dossier correspond à la racine d'un site Internet et nous pouvons par exemple l'appeler racine.

Dans ce dossier, vous allez créer une première page HTML que vous pourrez appeler index.html et les dossiers html, css, images par exemple.

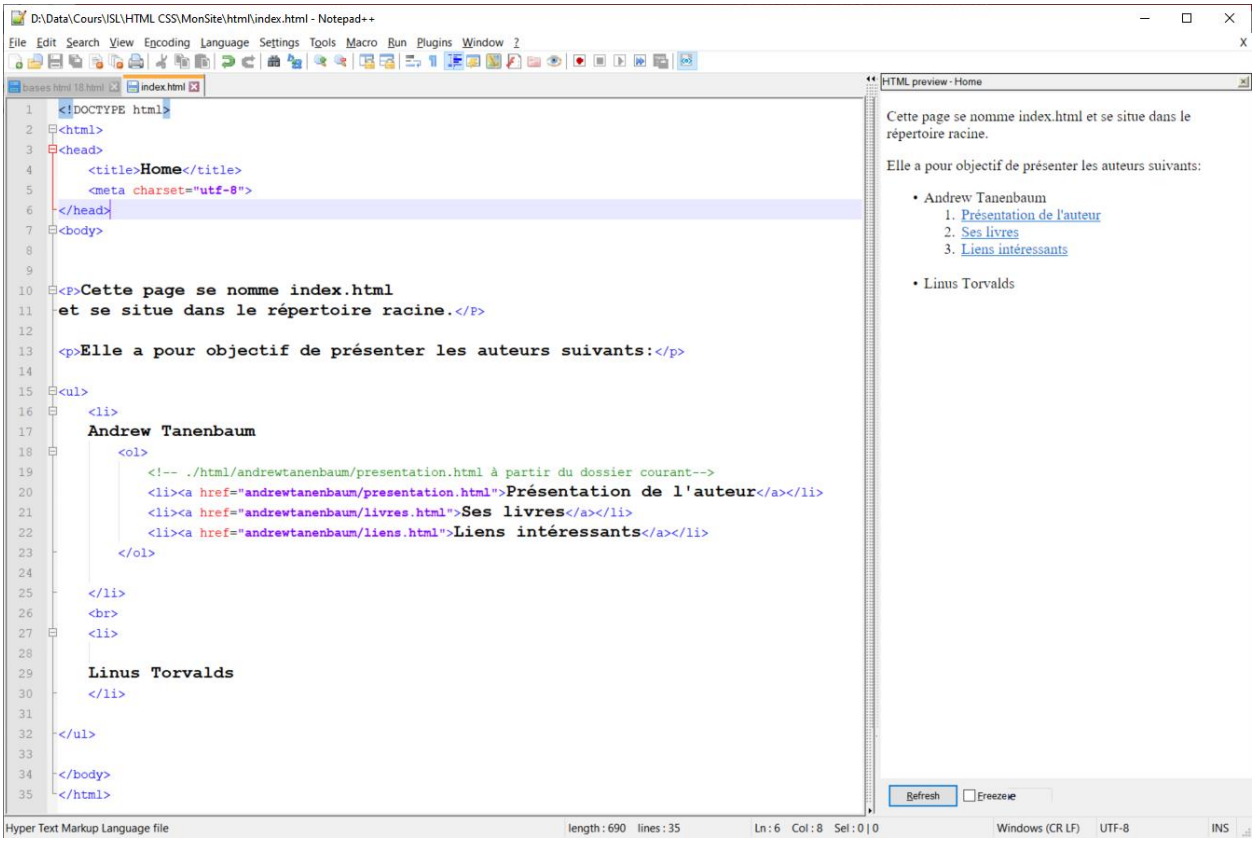
Dans le sous dossier html, je vous propose de créer un dossier AndrewTanenbaum contenant les pages presentation.html, liens.html et livres.html.

L'idée ici va être de faire des liens entre ces différentes pages. Encore une fois, vous pouvez imaginer que tout ce qui se trouve au sein de votre dossier racine est l'équivalent de la structure d'un site Internet (même si dans notre cas nous travaillons bien évidemment en local).

A partir de notre page index.html, on veut créer trois liens :

- Un vers la page presentation.html située dans le sous dossier AndrewTanenbaum du dossier html;
- Un vers la page livres.html située dans le sous dossier AndrewTanenbaum du dossier html;
- Un vers la page liens.html située dans le sous dossier AndrewTanenbaum du dossier html;

Voici comment on va s'y prendre :



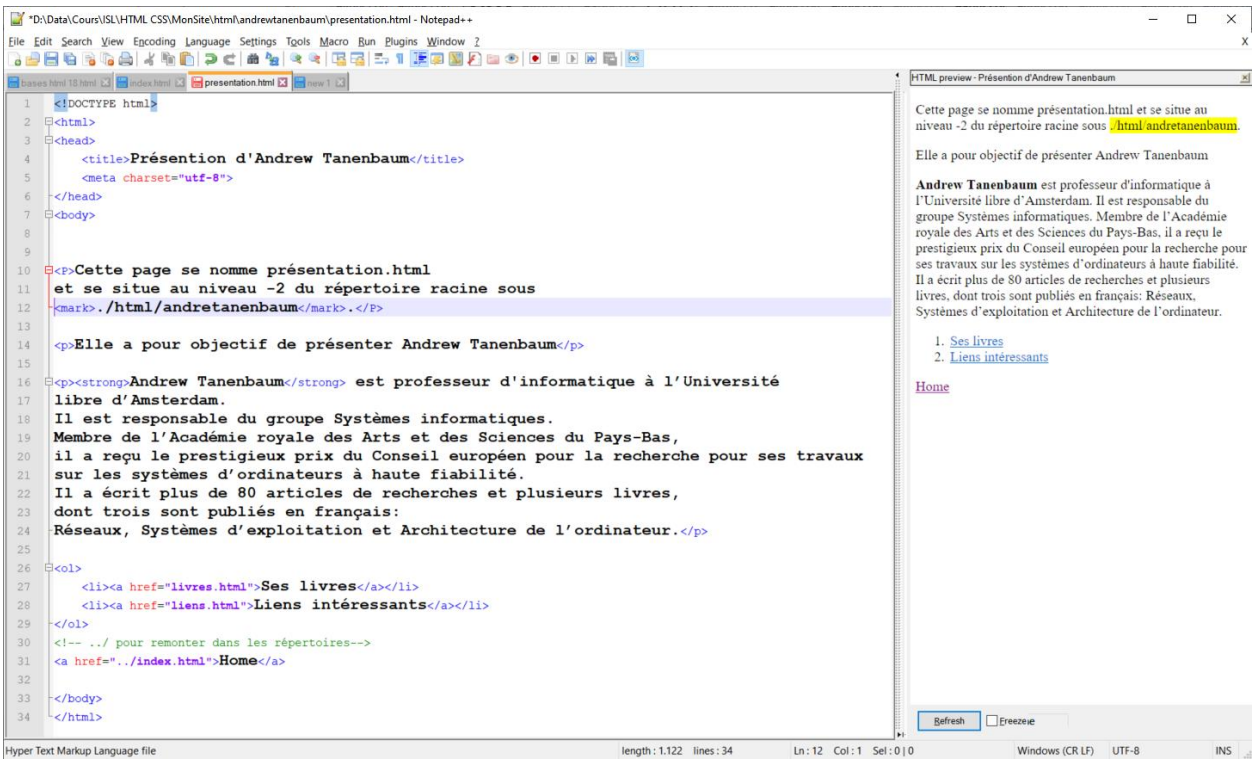
Ici, on voit bien que pour créer un lien vers une page située dans un sous dossier, il faut indiquer le nom du sous dossier suivi d'un slash suivi du nom de la page en valeur de l'attribut href.

Vous pouvez également remarquer que si la page cible du lien se situe dans un sous-sous dossier, alors il faudra préciser le nom du sous dossier suivi d'un slash suivi du nom du sous dossier puis à nouveau d'un slash et finalement le nom de la page de destination en valeur de l'attribut href.

Bon à savoir : Il faudra indiquer le nom de tous les sous dossiers traversés pour atteindre la page de destination à partir de la page de départ en valeur de l'attribut href.

On va également créer trois liens à partir de notre page presentation.html :

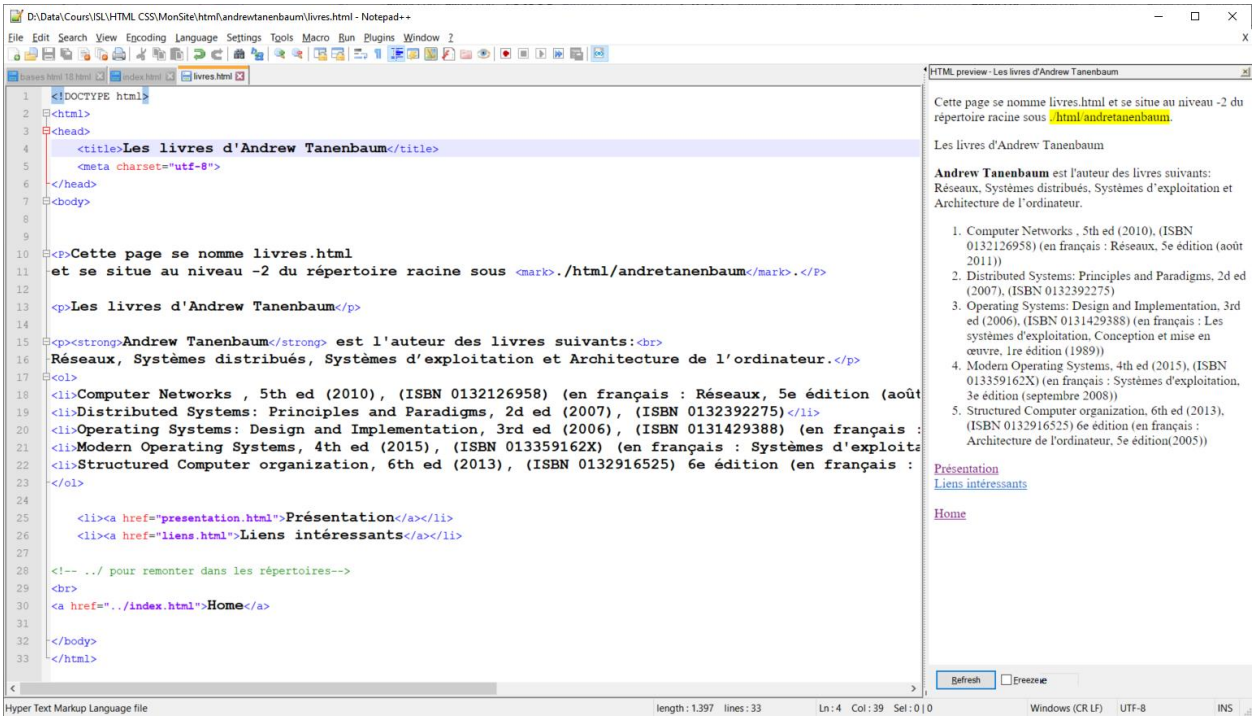
- Un vers la page index.html située dans le dossier racine ;
- Un vers la page livres.html située dans le même dossier andrewtanenbaum ;
- Un vers la page liens.html située dans le dossier andrewtanenbaum ;



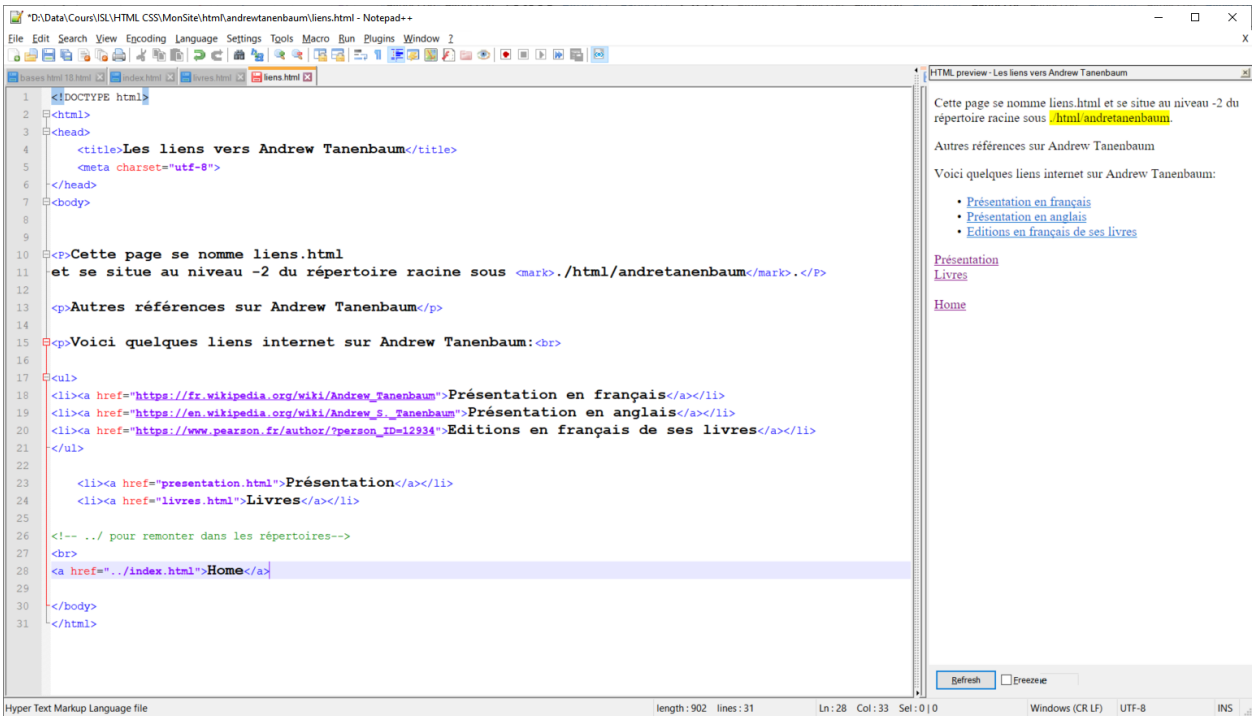
Ici, pas de surprise :

- On utilise la notation « ../index.html » pour créer un lien vers la page « index.html » située dans un dossier html ;
- La page « livres.html » est située dans le même dossier que la page « presentation.html », on se contente donc de préciser son nom en valeur de l'attribut href ;
- La page « liens.html » est située dans le même dossier que la page « presentation.html », on se contente donc de préciser son nom en valeur de l'attribut href ;
- Finalement, on fera la même chose pour les pages liens.html et livres.html

Voici la page livres.html (par exemple):



Et la page liens.html (par exemple)



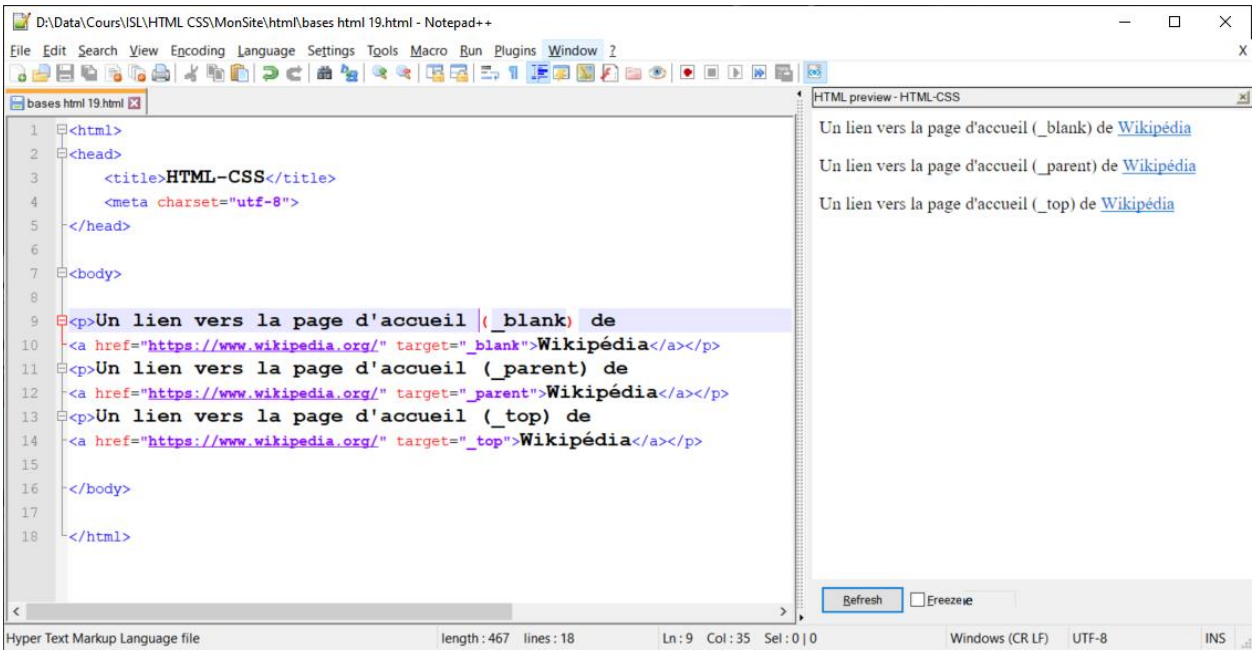
Ouvrir un lien dans un nouvel onglet grâce à l'attribut target

Jusqu'à présent, lorsque nous cliquions sur les liens créés en HTML, nous étions immédiatement redirigés vers la nouvelle page et notre page cible s'ouvrait dans le même emplacement (le même cadre ou « frame ») que le lien de départ.

Souvent, si vous avez un site, vous voudrez généralement que le lien s'ouvre dans un nouvel onglet afin que le visiteur ne perde pas la page de votre site. On va pouvoir faire cela avec l'attribut target.

L'attribut target va nous permettre de choisir où doit s'ouvrir notre page de destination. En pratique, nous utiliserons très souvent la valeur _blank qui spécifie que la nouvelle page doit s'ouvrir dans un nouvel onglet.

Voici un exemple pratique avec un lien externe menant vers la page d'accueil de Wikipédia et s'ouvrant dans un nouvel onglet :



Pour information, voici les différentes valeurs que peut prendre l'attribut HTML target afin de modifier le comportement de nos liens HTML :

Valeur de target	Comportement
_self	Valeur par défaut : la page cible s'ouvre dans le même emplacement (cadre ou « frame ») que là où l'utilisateur a cliqué
_blank	La page cible s'ouvre dans un nouvel onglet ou dans une nouvelle fenêtre

_parent	La page cible s'ouvre dans la cadre (frame) de niveau immédiatement supérieur par rapport à l'emplacement du lien
_top	La page cible s'ouvre dans la fenêtre hôte (par dessus le frameset)
Nom du cadre (frame)	Ouverture de la page cible dans le cadre portant le nom cité (en valeur de l'attribut name)

Créer des liens vers une autre partie d'une même page en HTML

Dans certains cas, lorsque vous construisez une page très longue, il va pouvoir être intéressant de proposer un sommaire avec des liens cliquables qui vont transporter l'utilisateur directement à un endroit précis de la page.

Nous allons également pouvoir créer ce type de liens (parfois appelé liens « ancre ») avec notre élément HTML a ainsi qu'avec un attribut id qui va nous servir à identifier l'endroit de la page où l'utilisateur doit être renvoyé.

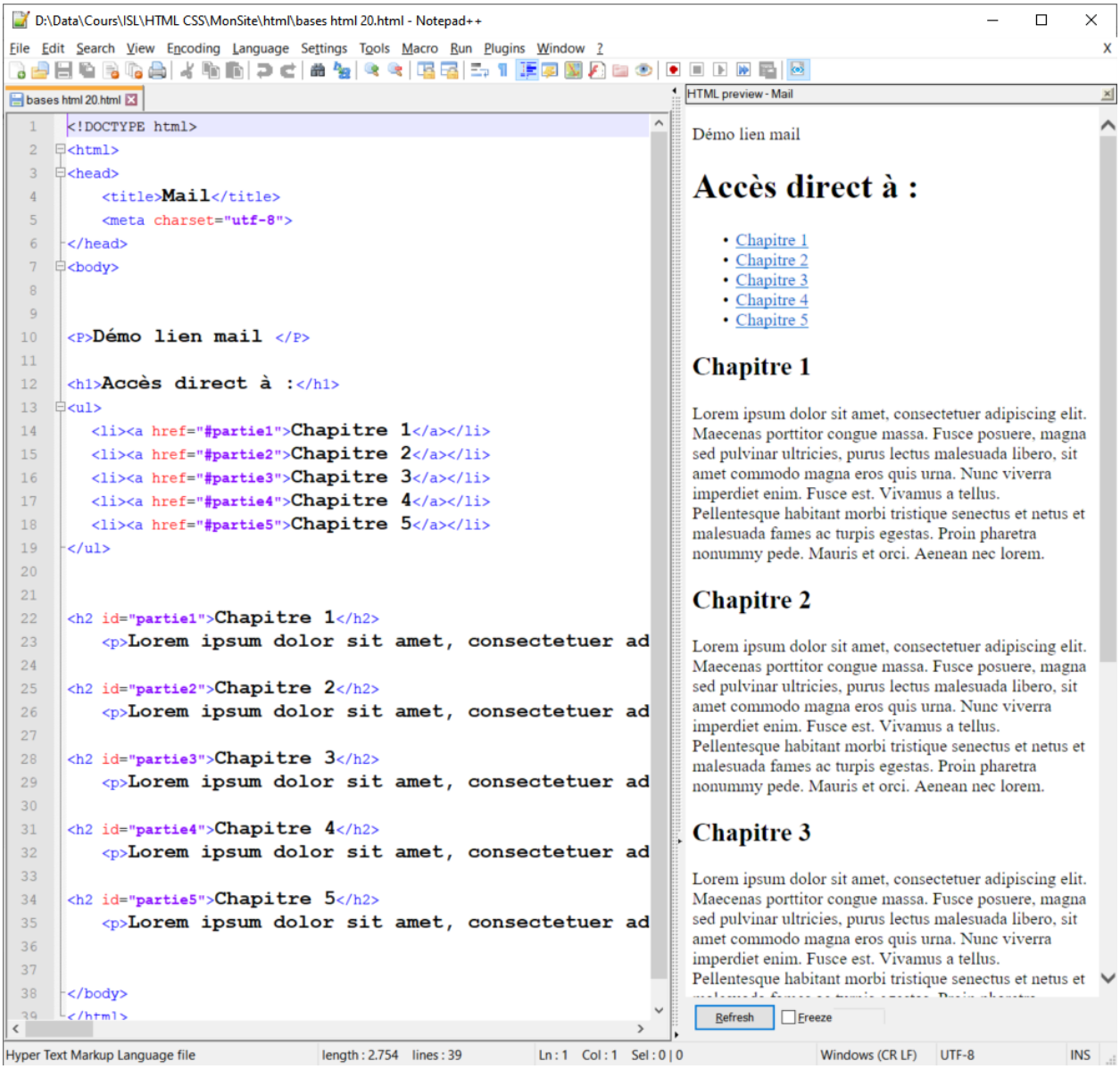
L'attribut id sert, comme son nom l'indique, à identifier un élément HTML en particulier dans une page pour ensuite pouvoir le cibler précisément (pour pouvoir lui appliquer des styles CSS entre autres).

Nous étudierons le fonctionnement de l'attribut id en détail plus tard dans ce cours. Pour le moment, retenez simplement que chaque id dans une page doit posséder une valeur unique puisqu'encore une fois un id sert à identifier précisément un élément en particulier.

Pour créer un lien de type « ancre », nous allons devoir procéder en deux étapes : tout d'abord, nous allons attribuer un attribut id aux différents éléments de notre page au niveau desquels on souhaite renvoyer nos visiteurs.

Ensuite, nous allons créer nos liens de renvoi en soi. Pour cela, nous allons passer à nos différents attributs href les valeurs correspondantes aux noms de mes id précédés d'un dièse (symbole #).

Voyons immédiatement comment créer ce type de lien en pratique. Pour cela, il va nous falloir une page très longue, et j'utilise la fonction Word =lorem(1,1) pour générer du texte pour mes paragraphes.



Dans l'exemple ci-dessus, j'ai créé une liste de type sommaire au début de ma page car celle-ci est très longue et j'ai accolé un id à chacun de mes titres h2.

Chaque élément de mon sommaire va correspondre à une partie vers laquelle je souhaite envoyer mes visiteurs. Je vais donc placer un élément de lien HTML a dans chacun des éléments de liste.

Ici, je vous invite à regarder attentivement les valeurs données à mes différents attributs href : vous remarquez que les différentes valeurs correspondent aux noms de mes id précédés d'un dièse (symbole #), symbole qui sert justement à cibler un id en CSS.

Lorsqu'on clique sur un élément du sommaire, nous sommes renvoyés directement à la partie de la page correspondante.

Utiliser une image en ancre de lien HTML

Nous n'avons pas encore étudié les images et je ne veux pas vous donner trop d'informations d'un coup afin que vous puissiez vous concentrer sur le sujet actuellement traité.

Cependant, sachez qu'il va être tout à fait possible d'utiliser une image à la place d'un texte comme ancre ou contenu cliquable pour un lien. Nous verrons comment faire cela dans le chapitre réservé à l'insertion d'images en HTML.

Envoi de mail et téléchargement de fichiers avec l'élément HTML a

L'élément a en HTML ne va pas uniquement être très utile pour créer des liens entre différentes pages ou de ramener à différents endroits d'une même page mais va également nous permettre de lier des ressources à nos pages.

Dans ce nouveau chapitre, nous allons étudier deux autres utilisations courantes de cet élément :

- Utiliser l'élément a pour permettre aux utilisateurs de nous envoyer un mail ;
- Utiliser l'élément a pour permettre aux utilisateurs de télécharger un fichier.

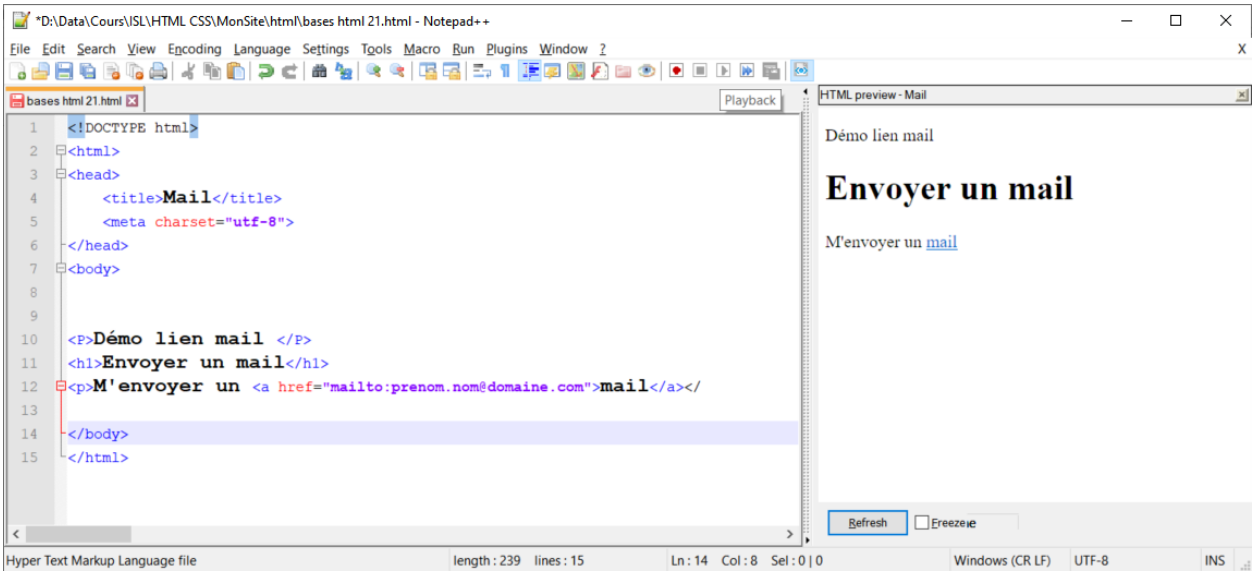
Utiliser l'élément a pour permettre l'envoi d'un mail

On peut utiliser l'élément a pour transmettre notre adresse mail à nos utilisateurs et leur permettre de nous envoyer simplement un mail.

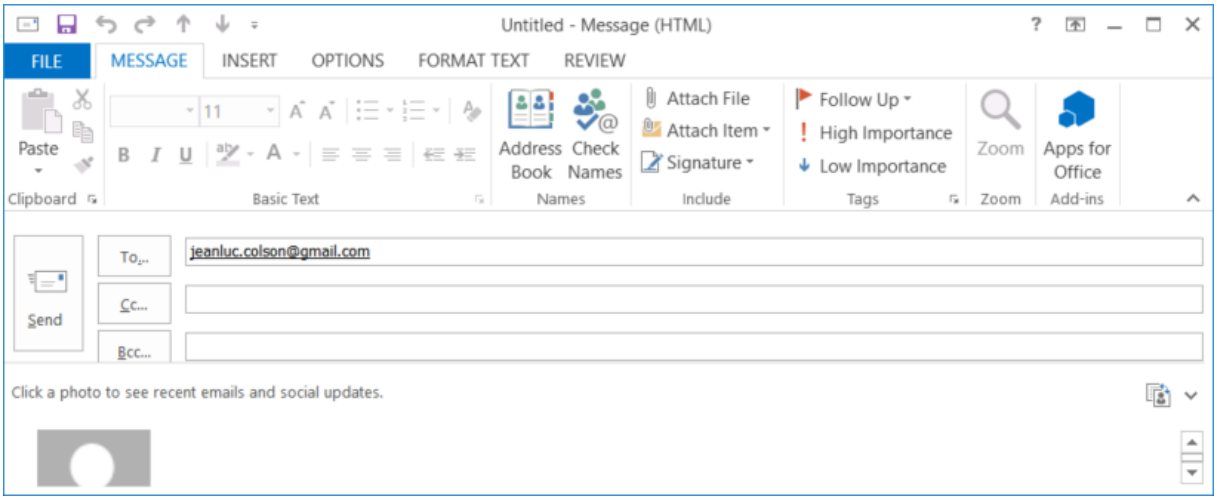
Pour permettre l'envoi d'un mail en HTML, on va placer indiquer en valeur de l'attribut href de notre élément a la valeur mailto : suivie de notre adresse email.

Lorsqu'un visiteur va cliquer sur notre lien, sa messagerie par défaut va automatiquement s'ouvrir. Par exemple, si vous avez un Mac, ce sera certainement l'application « Mail » qui va s'ouvrir. De plus, le champ destinataire sera automatiquement rempli avec notre adresse email.

Note : si vous travaillez sous Windows, il est possible que rien ne se passe si vous n'avez configuré aucune messagerie par défaut. Au sein d'une entreprise, ce sera vraisemblablement Outlook.



Dès qu'un visiteur clique sur le texte de notre lien, sa messagerie par défaut s'ouvre s'il en a configuré une :



Utiliser l'élément a pour permettre le téléchargement d'un fichier

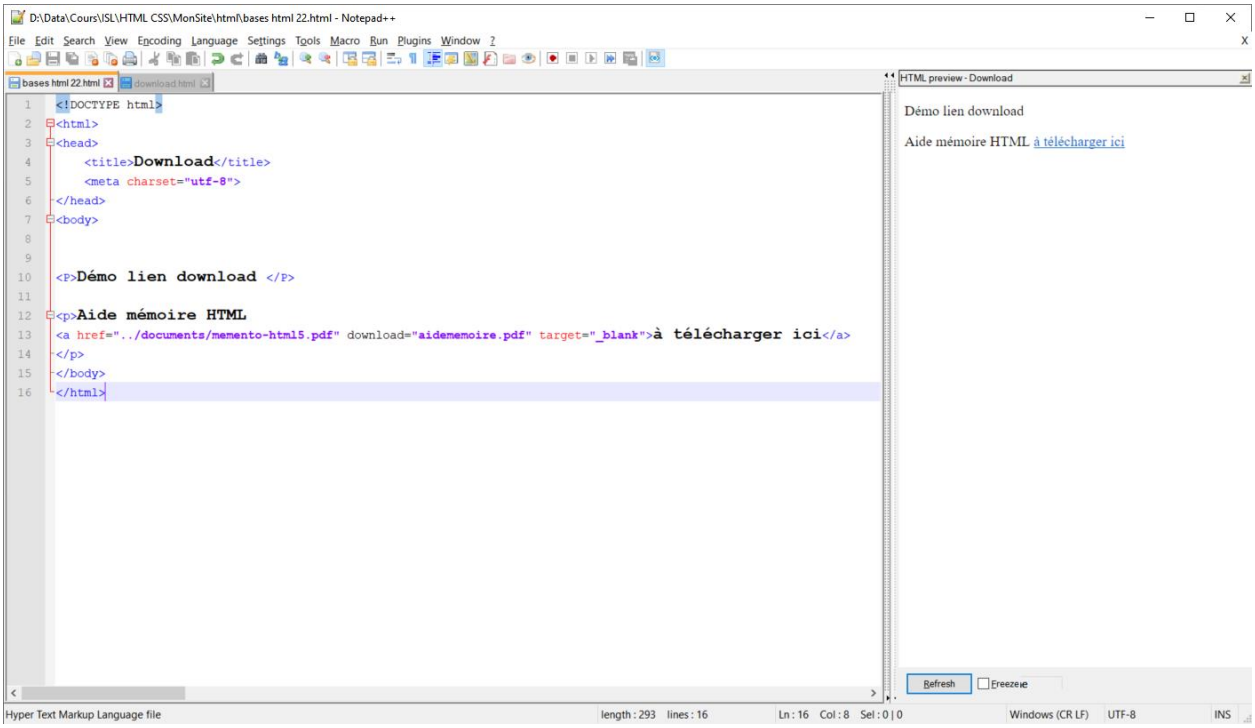
Vous pouvez encore utiliser l'élément a pour permettre à vos visiteurs de télécharger certains types de fichiers, comme des fichiers PDF ou Word par exemple.

Pour la plupart des fichiers, il va simplement suffire d'indiquer leur adresse (relative ou leur URL complète) en valeur de l'attribut href. Lorsqu'un utilisateur va cliquer sur le lien, le fichier lié va s'ouvrir dans le navigateur et l'utilisateur n'aura alors plus qu'à faire un clic droit sur le fichier ou utiliser les options de son navigateur pour l'enregistrer.

Cette première solution fonctionne mais demande à ce que l'utilisateur fasse lui-même la démarche de télécharger le fichier. On va également pouvoir « forcer » le téléchargement d'un fichier en ajoutant un attribut download dans l'élément a tout en indiquant l'adresse du fichier en question en valeur de l'attribut href.

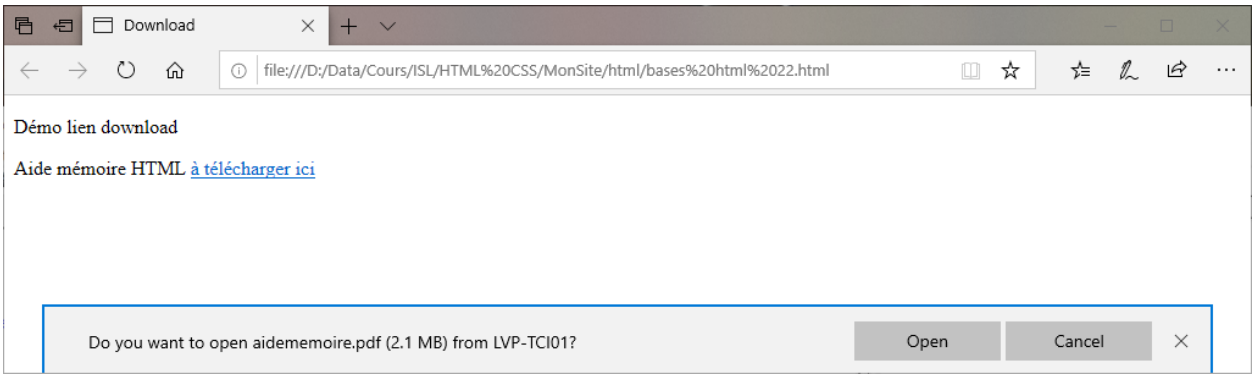
L'attribut download peut prendre en valeur (facultative) le nom sous lequel on souhaite que les utilisateurs téléchargent le fichier.

Dès que l'utilisateur va cliquer sur le lien, l'attribut download va faire que le téléchargement du fichier lié va se lancer automatiquement. Attention cependant, cet attribut n'a pendant très longtemps pas été supporté par certains navigateurs majeurs dont Safari.



Note : Vous n'allez pas pouvoir faire télécharger n'importe quel type de fichiers à vos visiteurs. En effet, les navigateurs récents vont bloquer le téléchargement de certains fichiers dont les extensions auront été jugées comme « potentiellement dangereuses ». De plus, certaines autres extensions de fichier vont être directement interprétées par le navigateur de vos visiteurs. Cela va être le cas si vous essayez de faire télécharger la source d'un fichier .html à vos visiteurs par exemple : le navigateur va ouvrir le fichier et interpréter son code comme pour n'importe quelle autre page et n'afficher donc que le résultat visuel lié au code HTML du fichier. Ici, une astuce simple consiste à compresser les fichiers que vous voulez faire télécharger en .zip par exemple.

Voici le comportement du navigateur Edge lorsque l'on clique sur le lien, il vous propose de télécharger le fichier .pdf sous le nom aidemémoire.pdf.



Compatibilité, support et validation du code HTML et CSS

Nous allons terminer cette première partie relative aux notions de base en HTML avec un mot sur la problématique complexe de la compatibilité du code entre les différents navigateurs ainsi que sur l'importance d'avoir toujours un code valide et optimisé.

La comptabilité intra-navigateur, inter-navigateurs et relatives aux différents appareils du code

Le problème de compatibilité et de consistance du code n'est pas nouveau : il y a quelques années seulement de cela, le web était beaucoup plus décousu qu'aujourd'hui et les règles n'étaient pas encore fixées.

Jusqu'au début des années 2000, Microsoft n'avait pas de concurrent sérieux et utilisait de sa puissance économique afin que son navigateur Internet Explorer (aujourd'hui appelé Edge) soit installé par défaut sur toutes les machines.

Cela permettait à Microsoft de s'offrir des libertés et notamment celle de développer de nouveaux standards de code ou d'implémenter des codes d'une façon différentes des autres.

Ensuite, dans les années 2000, de nouveaux acteurs sérieux ont fait apparition et ont commencé à se livrer une guerre économique. A cette époque, il était courant que différents navigateurs implémentent de manière totalement différente certains codes et utilisent leurs propres normes.

Ainsi, certains éléments ou attributs HTML par exemple n'étaient pas supportés par certains navigateurs et on devait donc créer des codes différents afin que chaque navigateur affiche le résultat voulu.

Depuis quelques années cependant, et grâce à l'impulsion du W3C qui n'a cessé de pousser des standards de développement, on assiste à une homogénéisation et une uniformisation de la prise en charge des codes sur la plupart des navigateurs sérieux. Cela est une excellente nouvelle pour nous, développeurs.

Cependant, tout n'est pas encore parfait comme on a pu le voir avec la prise en charge de l'attribut download par exemple dans les leçons précédentes.

Aujourd'hui, cependant, il nous faut faire face à de nouveaux défis tout aussi importants qui sont des défis liés aux différents appareils utilisés par nos visiteurs et à l'ergonomie.

En effet, les langages informatiques ont évolué au cours de ces dernières années pour proposer toujours davantage de fonctionnalités pour répondre notamment aux technologies émergentes et aux nouveaux besoins des utilisateurs.

L'apparition de l'internet mobile et sur tablette a notamment considérablement complexifié les questions liées à l'ergonomie des sites web puisqu'il était hors de question d'afficher autant d'information ou d'avoir des pages aussi lourdes sur mobile que sur un ordinateur de bureau classique par exemple.

Il faudra donc toujours garder ces questions en tête lorsque nous développerons nos propres projets et bien réfléchir en amont pour produire le résultat le plus conforme à nos attentes possibles.

Les entités HTML

Le HTML possède des caractères réservés. Par exemple, vous ne pouvez pas écrire les signes « < » et « > » tels quels dans vos pages web, tout simplement car le navigateur pensera que vous venez d'ouvrir une balise d'élément.

Pour remédier à ce problème, nous allons devoir « échapper » ces caractères réservés en utilisant ce qu'on appelle des entités HTML. Les entités vont être des suites de caractères représentant un caractère réservé en HTML.

Voici quelques-unes des entités les plus courantes et leur signification :

Nom de l'entité	**Résultat visuel**
&lt;	< (chevron ouvrant)
&gt;	> (chevron fermant)
&amp;	& (esperluette)
	(espace insécable)

Nous connaissons déjà certaines de ces entités comme l'entité   (pour « non-breaking space ») qui sert à créer une espace insécable en HTML.

Voyons immédiatement un exemple d'utilisation de ces entités :

Tester la validité de son code

Vous devez toujours vous efforcer d'écrire un code valide. Cela évitera des bugs potentiels et votre site sera au final mieux référencé sur les moteurs de recherche.

Pour vérifier la validité d'un code HTML ou CSS, le w3c (World Wide Web Consortium), c'est-à-dire l'organisme qui gère l'évolution des langages lus par les navigateurs comme le HTML et le CSS entre autres, a mis à disposition des validateurs de code qui sont gratuits.

Vous pouvez trouver les validateurs HTML et CSS aux adresses suivantes :

- HTML (http://validator.w3.org) : ici
- CSS (http://jigsaw.w3.org/css-validator) : ici