Introduction

Rédigé par: Jean-Luc Colson			
Date première rédaction: Jan 2020			
SUIVI DES MODIFICATIONS A LA FICHE			
Série	Date	Page(s) modifiée(s)	Raison

Contents

Introduction au cours HTML et CSS

HTML et CSS : deux langages incontournables

Il existe aujourd'hui des dizaines et des dizaines de langages informatiques et de programmation différents : HTML, CSS, JavaScript, PHP, Python, Ruby on Rails, C, C#, C++, Java, etc. pour ne citer qu'eux.

Certains de ces langages ont des possibilités et des rôles similaires, ce qui signifie qu'ils vont être (dans une certaine mesure) interchangeables : on va pouvoir utiliser un langage ou l'autre pour effectuer une même opération selon notre sensibilité personnelle ou selon l'environnement dans lequel on se trouve.

D'autres langages, en revanche, vont être beaucoup plus exclusifs ou ne pas avoir de concurrent et on va donc obligatoirement devoir passer par eux pour effectuer certaines opérations. Cela va être le cas du HTML et du CSS.

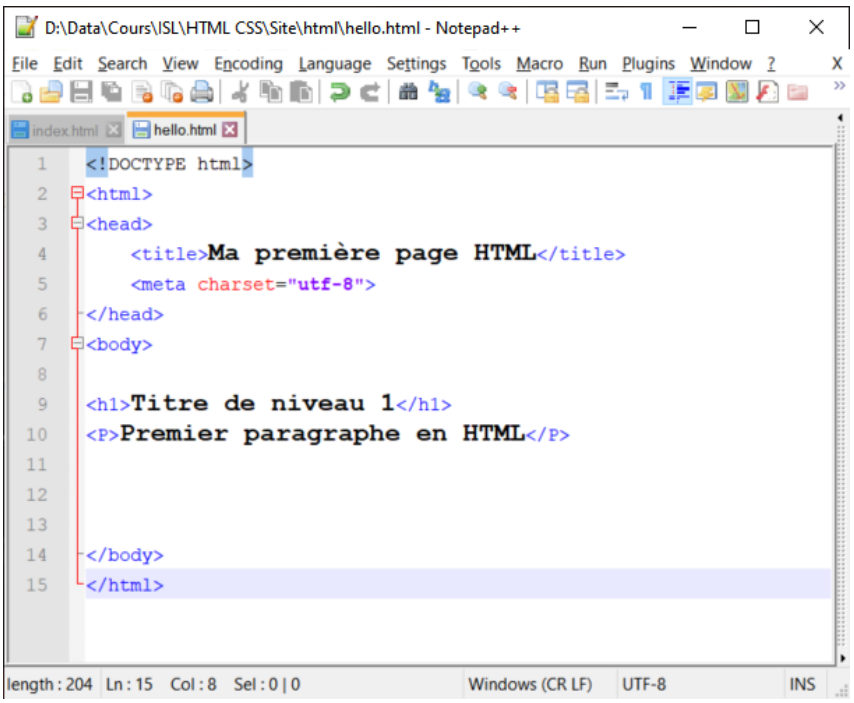
En effet, le HTML et le CSS sont deux véritables standards en informatique qui n'ont à l'heure actuelle aucun concurrent comme cela va pouvoir être le cas pour le langage PHP par exemple (pour lequel il existe des alternatives comme Ruby on Rails ou Python entre autres).

De plus, les langages HTML et CSS vont se trouver à la base de tout projet web car ils ont un rôle qui les rend incontournables : les navigateurs (Google Chrome, Safari, etc.) sont des programmes qui ont été construits pour pouvoir lire du code HTML au départ et qui ne peuvent aujourd'hui lire que du code HTML, CSS et JavaScript et nous allons donc nous appuyer sur ces langages (et sur le HTML en particulier) pour pouvoir afficher nos pages.

En bref : quel que soit votre projet web (blog, site e-commerce, application mobile, etc.), vous devrez forcément utiliser du HTML et du CSS.

Pour être un peu plus précis et pour anticiper un peu sur la suite de ce cours, le HTML est un langage de structure : il permet d'indiquer au navigateur que tel contenu est un titre, que tel autre est un simple texte, que cet objet est une image, que celui-ci est une liste, etc. Le navigateur, qui « comprend » le HTML, va se baser sur ces indications pour afficher les contenus.

Voici un premier exemple de code HTML :



Je n'ai pas l'intention de vous expliquer précisément ce que représente chaque élément dans ce code pour le moment car ceci n'aurait aucun intérêt. Pas d'inquiétude, nous aurons le temps de voir tout cela par la suite.

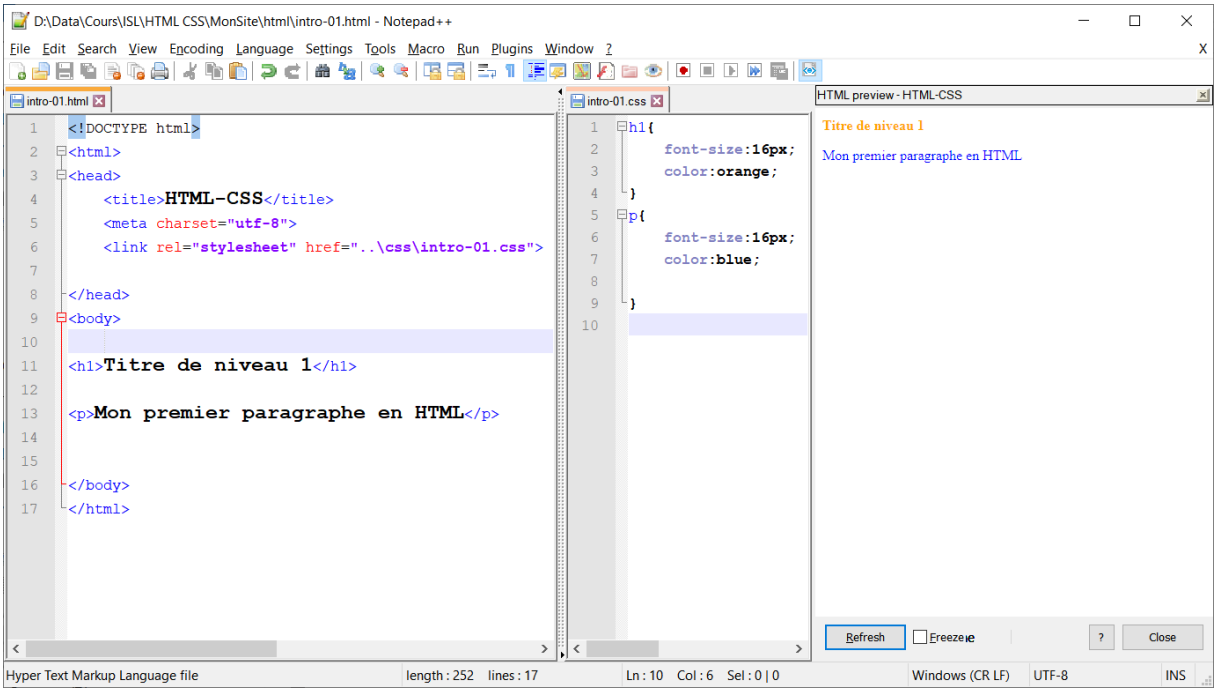
L'idée ici est simplement de commencer à vous familiariser avec la syntaxe du HTML et de voir comment le navigateur va traiter ce code. On va ici se concentrer sur deux lignes : la ligne contenant l'élément HTML h1 qui représente un titre de niveau 1 et celle contenant l'élément p qui représente un paragraphe.

Observons comment le navigateur va traiter ces contenus en ouvrant notre fichier dans Google Chrome par exemple :

Comme vous pouvez le voir, seul le contenu textuel placé entre <h1> et </h1> et entre et est affiché à l'utilisateur. On voit bien que le navigateur comprend que ces deux contenus ne sont pas équivalents puisqu'il les traite de manière différente en les affichant différemment.

Ajoutons maintenant un peu de CSS à la page (colonne du milieu). Le CSS est un langage de styles : il permet de modifier l'apparence ou le rendu visuel de nos contenus HTML et donc de nos pages web.

Ce code CSS indique au navigateur que les titres de niveau 1 devront avoir une taille de 16px (pixels) et être de couleur orange. On indique également que nos paragraphes doivent avoir une taille de 16px et une couleur bleue. Voici le résultat (colonne de droite):



Quelles alternatives à l'apprentissage des langages informatiques ?

De nombreuses personnes veulent lancer leur site internet sans forcément apprendre à coder. Est-ce une bonne idée ? Quels sont les risques ? Quelles alternatives existent pour créer un site web sans avoir à le coder ? Quels sont leurs points forts et points faibles ?

Il existe selon moi trois alternatives à l'apprentissage de la programmation qu'on va pouvoir considérer lorsqu'on a un projet web mais qu'on ne veut pas forcément devenir développeur :

- Le recours à un développeur / une agence externe ;
- L'utilisation d'un CMS ;
- L'utilisation d'un éditeur WYSIWIG.

Je vais tenter d'expliquer les points forts et faibles de chacune de ces alternatives et vous montrer en quoi la connaissance ou à minima la compréhension du fonctionnement des langages informatiques reste quasi indispensable quelle que soit l'option choisie.

Solution n°1 : faire appel à un développeur externe

La première solution, qui est certainement la plus évidente pour créer un site internet lorsqu'on ne veut pas le coder soi-même, est de faire appel à un développeur externe ou même à une agence de développement.

Cette première alternative présente à la fois des avantages et des inconvénients. Tout d'abord, parlons du prix. Ici, il va être très variable selon le site que vous allez vouloir créer bien sûr mais également selon le prestataire choisi.

En effet, si vous faites appel à une agence, le coût sera très souvent beaucoup plus élevé que si vous faites appel à un freelance (d'autant plus si vous passez par une plateforme d'annonces où ils sont en concurrence).

Sans rentrer dans le débat sur la qualité, il me semble quand même honnête de dire qu'aujourd'hui une personne seule ne peut pas tout (bien) faire : design, intégration, etc.

En plus de cela, deux autres questions d'importance se posent lorsque vous faites appel à un prestataire externe. La première est : comment bien expliquer ce que vous voulez créer comme site si vous ne savez pas comment le code fonctionne ?

Un client qui n'a aucune notion technique ne va pas pouvoir définir clairement ce qu'il veut. En effet, comprenez bien que le développement est une langue différente tout comme peut l'être la mécanique. Lorsque vous n'exprimez pas précisément votre besoin avec ses spécificités techniques, vous laissez le soin au développeur de les imaginer, ce qui entraine souvent des incompréhensions entre le client et le prestataire développeur.

Imaginez que vous faisiez appel à des ouvriers pour construire une maison mais sans passer par un architecte : vous n'allez pas pouvoir décrire précisément ce que vous voulez. Vous allez décrire votre besoin avec vos mots comme par exemple : « je veux une terrasse, 4 chambres, une cheminée », etc.

De votre point de vue vous « avez tout dit ». Cependant, du point de vue des ouvriers, l'essentiel est manquant : quelles matières utiliser ? quel type d'évacuation des eaux installer ? où faire passer les câbles électriques ? quel type de toiture choisir ?

Cela va donc naturellement mener à un client mécontent et qui va demander des modifications. Souvent, il va demander ce qui lui semble être de « petites modifications » mais les modifications demandées ne vont pouvoir être effectuées qu'en changeant en profondeur le site.

Si vous passez par une grosse agence, il est ici possible qu'il y ait une personne dont le rôle est de transcrire vos besoins en termes techniques. Cependant, ce genre d'agences est bien souvent hors budget pour le commun des mortels.

Enfin, reste la question de la maintenabilité du site : une fois le site livré, comment le faire évoluer ? Comment faire si des bugs apparaissent avec le temps et l'évolution des langages informatiques ? Ici, ne comptez pas trop sur votre prestataire de base pour vous servir d'aide en continu (à moins que vous ne le payiez à nouveau à chaque fois !).

Attention : je n'essaie pas de vous convaincre de ne pas passer par une agence ici. Ce que je veux vous montrer est que passer par un prestataire ne dispense pas de connaitre certaines bases en

informatique : si vous connaissez ces bases et si vous comprenez comment fonctionne un site web vous pourrez vous exprimer beaucoup plus clairement et exprimer votre besoin dans un langage qui sera compréhensible pas les développeurs. Vous aurez ainsi beaucoup plus de chances d'avoir un site conforme à vos attentes et vous économiserez sur la facture finale (car moins d'allers-retours et d'incertitude).

Solution n°2 : Utiliser un CMS

Un CMS (pour « Content Management System » ou « Système de Gestion de Contenu ») désigne un ensemble d'applications / de logiciels qui vont permettre de mettre en place et de gérer un site internet.

Un CMS permet donc de posséder un site internet sans se soucier de comment fonctionne le code. Vous pouvez ici imaginer les différentes commandes d'une voiture : lorsque vous achetez une voiture, vous n'avez pas besoin de savoir comment la voiture fonctionne en soi ou de qui se cache sous le capot ; vous n'avez qu'à utiliser les commandes à votre disposition : pédales, levier de vitesse, volant, etc.

Parmi les CMS les plus connus en France on peut notamment nommer WordPress qui permet aujourd'hui de créer plus ou moins n'importe quel type de site ou PrestaShop pour créer un site e-commerce.

Les CMS sont généralement conçus sur le même modèle d'architecture modulable : les utilisateurs vont installer le CMS de base et vont ensuite pouvoir ajouter différents modules ou plugins pour personnaliser leur site.

Utiliser un CMS semble donc parfait pour quelqu'un qui ne sait pas coder à priori. La réalité est toutefois différente. Ici, vous devez bien comprendre que les CMS sont souvent très complexes et que leurs plus grands avantages (la simplicité d'installation et la modularité) vont être également souvent leurs plus grandes faiblesses.

En effet, comme les CMS sont créés de façon à pouvoir être utilisés par tous, la plupart d'entre eux sont d'une complexité rare (du point de vue du code) et vont souvent posséder des tonnes de fonctionnalités qui ne vont pas vous être utiles afin de couvrir le plus de besoins possibles et ainsi satisfaire le plus d'utilisateurs.

De plus, une autre contrepartie de cette architecture commune et modulable est qu'il va être très compliqué d'effectuer des modifications sur la structure de votre site à posteriori sans tout casser à moins d'avoir de sérieuses connaissances techniques sur le CMS en question.

Enfin, il reste le problème de l'incompatibilité possible entre différents modules que vous pourriez installer après avoir créé votre site.

L'usage des CMS reste cependant un bon compromis pour une personne lançant un site sans prétention ou pour quelqu'un qui n'a pas un besoin très spécifique mais, pour maitriser complètement son CMS et pour faire évoluer son site en toute sérénité, il faut finalement être un excellent développeur et un utilisateur expérimenté du CMS en question.

Solution n°3 : la création avec un éditeur WYSIWIG

Les éditeurs WYSIWIG (« What You See Is What You Get » ou en français « Ce que vous voyez est ce que vous obtenez ») sont des éditeurs qui vont créer eux même le code à partir d'un modèle que vous allez créer.

Un éditeur WYSIWIG va se présenter de la façon suivante : une palette d'outils sur le côté et une page dans laquelle vous allez pouvoir utiliser ces outils. Vous allez ainsi par exemple pouvoir insérer un rectangle dans votre page puis placer du texte dedans puis changer la taille de ce texte et etc.

L'éditeur va ensuite se charger de transformer ce que vous avez créé (« ce que vous voyez ») en code.

Le gros souci avec ces éditeurs est qu'ils ne possèdent pas la même logique qu'un humain et qu'ils ne peuvent pas penser un projet ou une page dans sa globalité. Ainsi, le code créé va très souvent être de très mauvaise qualité et cela va impacter le référencement de votre site entre autres.

Je ne parlerai pas plus de ces éditeurs qui ne constituent pas une alternative valide à mes yeux par rapport aux autres solutions proposées.

En résumé : apprendre à coder, ce n'est pas que pour les développeurs !

Je pense vous avoir démontré l'intérêt de maitriser au moins les bases du développement dans la partie précédente si vous avez un projet lié au web : économie d'argent, plus d'efficacité, création d'un site conforme à vos attentes, etc.

En effet, apprendre à coder ou tout au moins connaitre les bases en développement c'est avant tout mettre un pied dans le monde des développeurs pour pouvoir les comprendre et pouvoir vous faire comprendre.

De plus, connaitre le rôle des différents langages et comprendre comment fonctionne votre site permet de pouvoir le gérer de manière beaucoup plus efficace. En effet, je suis et reste persuadé qu'on ne peut pas travailler sereinement lorsqu'on ne comprend pas son outil de travail ou la structure dans laquelle on travaille.

C'est évident et pourtant la plupart des gens essayent de se persuader et de persuader les autres du contraire. Pourquoi ? Car tout le monde utilise internet aujourd'hui et la majorité des gens ne veulent pas faire l'effort de comprendre comment tout cela fonctionne.

Ce n'est pas un problème lorsque vous êtes un simple utilisateur, mais ça le devient lorsque vous devez gérer un site internet ou même lorsque vous travaillez dans un domaine lié au digital.

Deviendriez-vous plombier sur un coup de tête ? Non, car vous n'y connaissez rien en plomberie. C'est exactement pareil sur le web.

L'immense majorité des échecs liés au web proviennent du fait que des personnes se lancent dans l'aventure sans la moindre connaissance de leur environnement.

N'oubliez pas qu'il est essentiel pour qu'un commerce fonctionne -et aujourd'hui plus que jamais- d'avoir une compréhension de son propre business, de son architecture et de son infrastructure.

Si vous faîtes l'effort d'acquérir ces connaissances, je vous garantis que vous avez d'ores-et-déjà battu 99% de vos concurrents.

Convaincu de l'utilité d'apprendre à coder ? Dans ce cas, passons à la suite ! Car je suis certain que vous êtes impatients de découvrir ce que signifient les initiales « HTML » et « CSS » !

Définitions et usages du HTML et du CSS

Le HTML : langage de balisage

Le HTML est un langage qui a été créé en 1991. Les sigles « HTML » sont l'abréviation de « HyperText Markup Language » ou « langage de balisage hypertexte » en français.

Le HTML est donc un langage de balisage, c'est-à-dire un langage qui va nous permettre de définir les différents contenus d'une page.

Pourquoi est-il si important de définir les différents contenus d'une page ? Pour répondre à cette question, il va avant tout falloir que vous compreniez ce qu'est un site internet et ce qu'il se passe lorsque vous essayez d'accéder à un site internet.

Ici, je vais essayer de rester le plus simple possible. Tout d'abord, qu'est-ce qu'un site internet ? Un site internet est un ensemble de fichiers de code et de medias (images, videos, etc.) liés entre eux et disponibles à tout moment via le web.

Pour que les différentes ressources constituant un site internet soient toujours accessibles, on les héberge généralement sur un serveur d'un hébergeur professionnel comme OVH par exemple.

Un serveur est un ordinateur plus ou moins comme les nôtres et qui stocke les différentes ressources d'un ou de plusieurs site internet. Les seules différences entre un serveur et nos ordinateurs est qu'un serveur est généralement beaucoup plus puissant que nos ordinateurs, qu'il est (normalement) toujours connecté à Internet et qu'il va posséder quelques logiciels supplémentaires lui permettant d'exécuter certains codes.

Notez ici qu'on appelle ces ordinateurs des « serveurs » car leur seul et unique rôle va être de « servir » (au sens de « fournir ») des pages dès que quelqu'un va le leur demander.

Comment accède-t-on à une page d'un site internet ? Pour répondre à cette deuxième question, je vais m'aider d'un exemple. Imaginons que l'on souhaite accéder à la page d'accueil de Wikipedia. Pour cela, on va taper www.wikipedia.org dans la barre de notre navigateur.

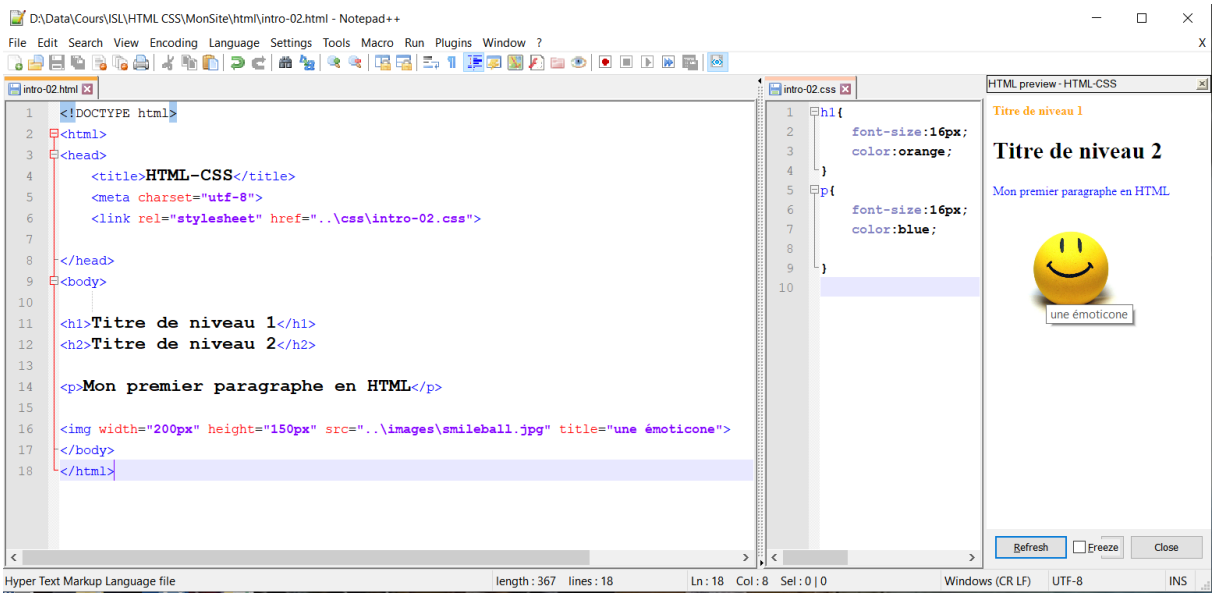
Lorsqu'on demande à accéder à une page d'un site internet, nous sommes ce qu'on appelle des « clients ».

Notre navigateur va contacter le serveur sur lequel est hébergée la page à laquelle on souhaite accéder et lui demander de renvoyer les différents éléments de la page : la page sous forme de code HTML et potentiellement les différents médias intégrés dans la page (ce qu'il se passe est en pratique bien évidemment plus complexe mais encore une fois je simplifie volontairement ici).

Le navigateur va donc recevoir ces différents éléments et les afficher. Cependant, comment fait-il pour savoir ce qu'il doit afficher ? Il va bien évidemment utiliser le code HTML. En effet, le navigateur comprend bien les différentes balises HTML (le HTML utilise ce qu'on appelle des « balises » pour définir les contenus) et va donc « comprendre » de quoi est constituée notre page et ce qu'il doit afficher.

Le rôle du HTML est donc crucial puisqu'il va être notre langage privilégié pour indiquer aux navigateurs ce quoi est constituée chaque page et ce qu'ils doivent afficher. Grâce au HTML, on va par exemple pourvoir indiquer que tel contenu est un texte qui n'est qu'un paragraphe, que tel autre contenu est un texte qui est un titre de niveau 1 dans notre page, que tel autre contenu est une liste, un lien, etc.

En plus de cela, le HTML va également nous permettre d'insérer différents médias (images, vidéos, etc.) dans nos pages web en indiquant au navigateur « à cette place-là dans ma page, je veux que s'affiche cette image ». Notez que dans ce cas précis, pour que le navigateur affiche la bonne image, on va lui fournir l'adresse de l'image dans le code HTML.



Le CSS : langage de styles

Le CSS a été créé en 1996, soit 5 ans après le HTML. Les sigles « CSS » sont l'abréviation de « Cascading StyleSheets » ou « feuilles de styles en cascade » en français.

Le CSS vient résoudre un problème bien différent du HTML : en effet, le HTML sert à définir les différents éléments d'une page, à leur donner du sens. Le CSS, lui, va servir à mettre en forme les différents contenus définis par le HTML en leur appliquant des styles.

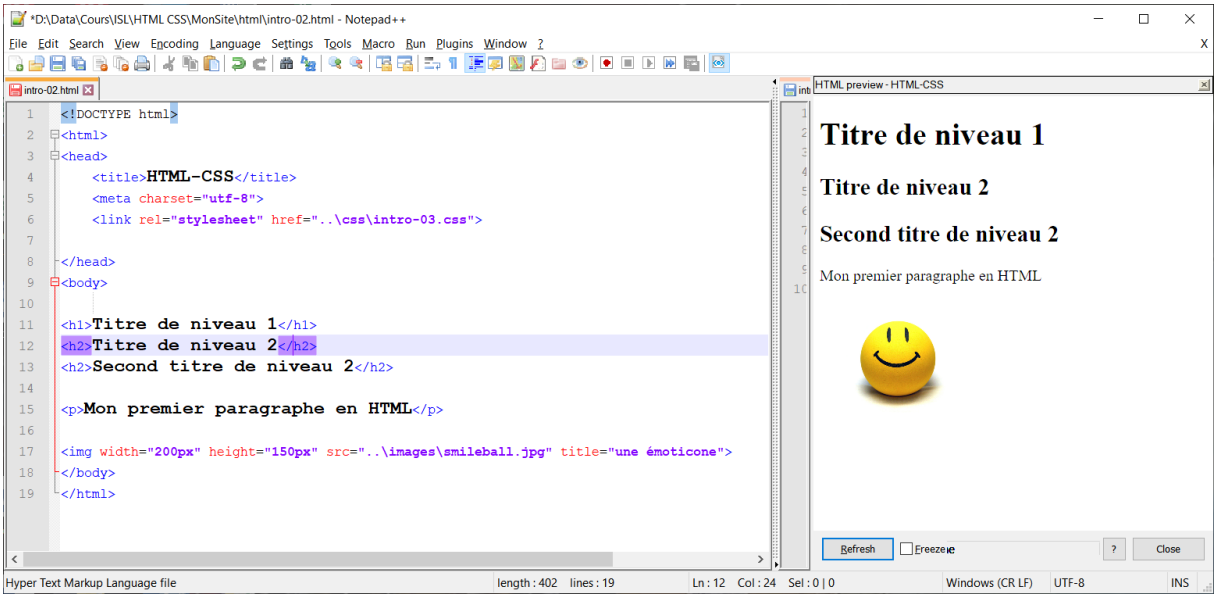
Le HTML va donc créer la structure des pages tandis que le CSS (colonne centrale dans l'exemple précédent) va nous permettre de modifier l'apparence des contenus de la page. On va ainsi par exemple pouvoir définir la taille, la couleur ou l'alignement de certains contenus HTML et notamment en l'occurrence de certains textes dans notre page.

Pour styliser le contenu lié à un élément HTML en CSS, nous allons pouvoir le cibler en nous basant sur l'élément HTML en question ou en utilisant d'autres procédés que nous verrons plus tard dans ce cours.

A retenir : n'utilisez pas le HTML pour mettre en forme vos contenus !

Voilà une chose que je vais vous répéter encore et encore au fil de ces premiers chapitres : vous ne devez jamais utiliser le HTML pour faire le travail du CSS.

En effet, si vous affichez la page en HTML créée ci-dessus dans votre navigateur sans y ajouter de CSS, vous pouvez remarquer que le contenu qui a été déclaré comme étant un titre s'affiche en grand et en gras, tandis que la taille du texte de notre paragraphe est plus petite et la police moins grasse.



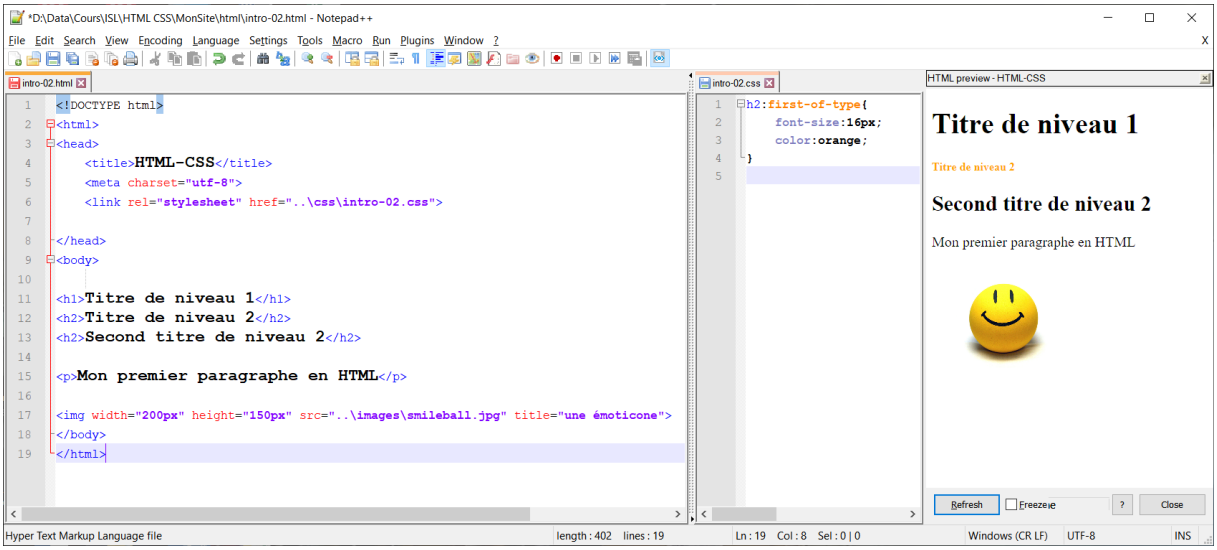
Certains débutants en déduisent donc « pour mettre un texte en grand et en gras, il suffit d'utiliser une balise représentant un titre en HTML ». Sortez-vous cela immédiatement de la tête !

Le HTML est un langage qui a été créé pour structurer des pages et pour donner du sens au contenu. Le principe même du HTML est d'indiquer aux navigateurs que tel contenu est un texte qui doit être considéré comme un titre tandis que tel autre contenu est un texte qui doit être considéré comme un simple paragraphe et etc.

Si votre navigateur affiche par défaut les contenus textuels que vous avez déclaré comme des titres en HTML en grand et en gras et à l'inverse les paragraphes en plus petit c'est justement parce que chaque navigateur possède sa propre feuille de styles (c'est-à-dire son propre code CSS) qui sera utilisé si aucun code CSS n'a été précisé de notre côté.

Je répète ici car cela me semble très important : chaque navigateur a une feuille de style dans laquelle sont définis des styles pour chaque contenu HTML qui seront appliqués par défaut si aucun autre code CSS n'est fourni. Cela permet d'apporter une mise en forme minimale au contenu dans les cas rares où les styles CSS ne seraient pas définis ou si notre code CSS ne peut pas être chargé.

En fournissant nos styles CSS pour les différents contenus de notre page, nous allons donc pouvoir définir l'apparence de ceux-ci puisque les styles que nous allons fournir vont être considérés comme prioritaires par rapport à ceux du navigateur.



N'essayez pas forcément de comprendre comment fonctionne le code ci-dessus, ce n'est pas le but ici. La seule chose que vous devez comprendre est qu'on applique ici des styles CSS à notre premier contenu HTML placé entre les balises <h2> et </h2> (qui servent à définir un titre de niveau 2) mais pas au deuxième.

Pour notre premier titre de niveau 2, on indique qu'on veut une taille de texte de 16px et une couleur de texte orange et c'est donc ce que va afficher le navigateur. Comme on n'indique aucun style pour le deuxième titre de niveau 2, le navigateur va charger ses styles par défaut.

Note : Chaque version de chaque navigateur possède sa propre feuille de style, et c'est la raison pour laquelle une même page peut être rendue différemment d'un navigateur à un autre ou même d'une version d'un navigateur à une autre version de ce même navigateur ! Cependant, aujourd'hui, un véritable effort d'harmonisation a été fait et la plupart des navigateurs utilisent plus ou moins les mêmes styles par défaut.

En bref, retenez que vous ne devez jamais utiliser le HTML pour modifier l'apparence d'un contenu car cela est le rôle du CSS. Si vous faussez le tout en déclarant par exemple des titres qui n'en sont pas, vous pervertissez le rôle du HTML et cela va impacter fortement la qualité de votre page web, sa validité et votre référencement global.

L'évolution de l'informatique et des langages Web

L'informatique, un domaine en constante et rapide évolution

L'informatique est un domaine qui évolue vite et dont les normes ne sont pas encore tout à fait fixées, à la différence de matières comme les mathématiques par exemple. En effet, il y a 20 ans de cela, le web était bien différent d'aujourd'hui et il sera certainement à nouveau très différent dans quelques années.

En effet, l'évolution des technologies et des composants physiques toujours plus performants soutiennent cette perpétuelle évolution.

Pour rappel, jusqu'au début des années 2000, les données étaient stockées sur des disquettes dont la capacité de stockage n'excédait pas les 1440 Ko c'est-à-dire le poids d'une image de qualité moyenne aujourd'hui et nous avions des modems de connexion avec une capacité de transmission des données de 64 Ko/s.

Avec le succès et la démocratisation des ordinateurs et d'internet un peu partout, cependant, les technologies n'ont cessé d'évoluer de plus en plus vite et les composants physiques sont devenus de plus en plus performants entre les années 2000 et 2010.

Or, avec plus de puissance dans nos ordinateurs et des connexions beaucoup plus rapides, de nouvelles opportunités se sont ouvertes dans le monde de l'informatique et du web en particulier.

Les langages informatiques, qui sont au centre du web, ont donc bien évidemment évolué en même temps que le champ des possibilités s'est ouvert, en s'enrichissant et en multipliant leurs fonctionnalités.

L'augmentation de la vitesse de connexion a entre autres permis de créer des sites web beaucoup plus complexes et d'intégrer de plus en plus de médias, comme de l'audio et de la vidéo ou des images de plus en plus lourdes, ce qui était inconcevable avant car ces médias auraient mis des jours à charger.

L'augmentation des performances des différents composants physiques a elle permis de réaliser des calculs toujours plus complexes côté serveur et de multiplier également les capacités de stockage de ceux-ci.

Langages de programmation : création et évolution

Pour bien comprendre comment sont créés et comment évoluent les différents langages de programmation, il faut avant tout savoir qu'il existe deux types de langages :

- Les langages dits « Open Source » : ce sont des langages qu'on va pouvoir utiliser sans licence et n'importe qui va (généralement) pouvoir participer à leur développement ;

- Les langages créés et gérés par des entreprises privées comme le langage Java qui appartenait à l'origine à la société Sun qui a ensuite été rachetée par la société Oracle.

Concernant les langages Open Source, comme je viens de le dire, n'importe qui peut en théorie contribuer à leur évolution en testant ou en fournissant des notes sur les sites officiels des groupes qui gèrent les différents langages.

En effet, bien qu'Open Source, des groupes de travail différents se sont constitués autour des différents langages. Deux groupes vont particulièrement nous intéresser dans le cadre du HTML et du CSS : le W3C ou « World Wide Web Consortium » et le WHATWG ou « Web Hypertext Application Technology Working Group ».

Pour vous donner un peu de contexte sans entrer dans la polémique, le W3C était à l'origine le groupe de référence chargé de définir et de veiller au développement des langages HTML et CSS.

Cependant, en 1998, le W3C décide d'abandonner l'évolution du HTML pour se concentrer sur un autre langage qui devait lui succéder : le XHTML.

Certaines personnes, mécontentes de l'inertie du W3C et du temps que prenait le développement de nouveaux standards et qui voyaient du potentiel dans le HTML ont alors décidé de continuer son évolution en formant un groupe nommé le WHATWG.

Face à l'échec du XHTML, en 2006, le W3C revient sur ses pas et exprime son intérêt pour travailler avec le WHATWG sur le HTML. Ce travail en commun a fonctionné jusqu'en 2011, date à laquelle l'écart entre les objectifs et les méthodes des deux groupes est devenu trop grand.

En effet, le WHATWG milite pour un « Living Standard » pour le HTML, c'est-à-dire des améliorations constantes et immédiatement intégrées tandis que le W3C préfère mettre à jour le HTML dès que beaucoup d'avancées ont été faites et après s'être assuré que chaque nouveau composant fonctionne parfaitement.

Ce qu'il faut bien comprendre ici est que le W3C est un groupe composé de plus de 300 grandes entreprises qui font du lobbysme et donc que ce groupe subit de grandes pressions. A l'inverse, le WHATWG est le résultat de l'effort commun de trois « entreprises du web » : Opera, Mozilla et Apple, ce qui le rend beaucoup plus flexible.

Aujourd'hui, on se retrouve donc dans une situation conflictuelle où le WHATWG avance beaucoup plus vite que le W3C et accuse le W3C de voler les différents standards créés par le WHATWG pour les porter à leur crédit.

Dans ce cours, je me baserai sur les recommandations du W3C puisque ce sont celles communément admises et puisqu'elles reprennent (c'est un fait) la majorité du travail du WHATWG, avec un peu de retard certes mais ce « retard » laisse le temps aux navigateurs d'intégrer les changements ce qui est finalement plutôt une bonne chose.

Le W3C définit trois états d'avancement pour ses différents documents qui vont chacun traiter d'un aspect d'un langage :

- Le statut de recommandation, ce qui veut dire que le document sert de standard et que le support pour les éléments ou pratiques qu'il définit doit être intégré par les navigateurs ;

- Le statut de candidat à la recommandation, ce qui signifie que les points abordés dans le document sont à priori fixés et attendent une dernière validation ;
- Le statut « travail en cours », ce qui signifie que le document possède encore des zones d'ombre à mettre au clair, etc.

Notez qu'il est bien évidemment également très important pour un développeur de se tenir au courant des nouveautés pour d'une part exploiter toutes les possibilités des langages et d'autre part savoir lorsque certains codes ou pratiques sont dépréciés, c'est-à-dire lorsque le support d'un certain élément de langage est abandonné pour pouvoir mettre à jour les parties du code d'un site les utilisant et éviter les bugs sur le site en question.

Concernant le langage CSS, c'est beaucoup plus simple puisque dans ce cas le W3C est le seul groupe qui s'occupe de son évolution et cela n'est contesté par personne.

Pour résumer, voici les groupes de travail faisant autorité sur les langages Open Source les plus populaires :

- HTML : W3C / WHATWG ;
- CSS : W3C ;
- JavaScript : ECMA ;
- PHP : PHP Group ;
- Python : Python Software Foundation.

Les versions actuelles du HTML et CSS

Jusqu'à récemment, les évolutions des langages HTML et CSS étaient « brutales » pour l'utilisateur final puisque le travail d'amélioration était effectué sur l'ensemble du langage avant d'être donné comme recommandation au public.

Ainsi, on est passé du HTML version 1 en 1991 au HTML version 2 en 1994 au HTML3 et etc. jusqu'au HTML5 en novembre 2014.

Concernant le CSS, nous sommes passés de la version 1 en 1996 au CSS2 en 1998 puis finalement au CSS2.1 en 2011.

La durée entre le CSS2 et le CSS2.1 s'explique par les nombreux allers-retours qui ont été faits entre les statuts de « candidat à la recommandation » et de « travail en cours » suite aux rejets consécutifs du W3C de valider la nouvelle version du langage comme recommandation.

Cela a permis de montrer les limites de ce système de travail sur l'ensemble du langage : il était même devenu quasiment impossible de procéder comme cela suite à la multiplication des fonctionnalités de chaque langage et le travail était mal organisé.

Sous la pression du WHATWG et pour pouvoir les concurrencer, le W3C a donc commencé à organiser le travail sur chaque langage par « modules ». Chaque module va concerner un aspect

précis d'un langage et son développement va pouvoir avancer plus ou moins indépendamment des autres modules.

Par exemple, pour le CSS, nous avons un module de travail sur la couleur, un autre sur la police d'écriture, un autre sur la mise en page, etc.

Cela nous a ainsi permit de commencer à travailler sur l'évolution de chaque fonctionnalité d'un langage de manière indépendante.

Dès qu'un module est suffisamment avancé, on l'envoie comme candidat au statut de recommandation et il peut alors être validé comme nouveau standard (ou pas).

Dans ce cours, j'utiliserai toujours les derniers modules reconnus comme « recommandation » par le W3C. Certains possèdent l'étiquette « HTML5 », d'autres « HTML Living Standard » ou « CSS3 » ou « CSS4 ». Parfois, j'aborderai également certaines fonctionnalités qui ne sont que candidates à la recommandation mais qui vont très probablement obtenir le statut de recommandation dans un futur proche. Je vous l'indiquerai dans ce cas.

Travail en local et en production

Que signifie travailler en « local » ? Travailler en « production » ?

Lorsqu'on travaille sur un projet en informatique il existe deux façons de travailler : nous allons pouvoir travailler soit en local, soit en production.

Travailler en local signifie travailler sur et avec des fichiers enregistrés sur notre propre ordinateur. Comme toutes les ressources sont situées sur notre ordinateur, nous seuls allons pouvoir voir les résultats des différents codes que l'on crée.

Travailler en production, au contraire, signifie travailler sur des fichiers qui sont stockées sur un serveur web, c'est-à-dire modifier des ressources auxquels les utilisateurs peuvent accéder via internet.

Quand travailler en local ou en production ?

Imaginons que nous devions intervenir sur un site internet déjà existant et visité par des utilisateurs tous les jours.

En intervenant directement sur les fichiers sur le serveur, c'est-à-dire en production, nos modifications vont être immédiatement visibles par les utilisateurs.

Cela peut avoir deux impacts évidents : tout d'abord, selon les modifications effectuées, certains utilisateurs peuvent voir leur expérience sur le site altérée : imaginez un site internet qui est modifié en même temps que vous le visitez ! De plus, si jamais nous faisons une erreur sur notre code, c'est tout le site internet qui risque d'être vulnérable ou même inaccessible.

Pour ces raisons, un bon développeur préfèrera toujours tant que possible effectuer les modifications ou développer les nouvelles fonctionnalités en local, afin de limiter l'impact sur le site live.

Du local à la pré-prod, de la pré-prod à la prod, de la prod au local

Comment travailler en local sur un site internet déjà « en ligne », c'est-à-dire accessible par des visiteurs partout dans le monde ?

C'est finalement très simple : nous allons effectuer une copie du site, c'est-à-dire une copie de tous les fichiers et média constituant le site sur nos ordinateurs puis travailler sur cette copie.

Une fois les modifications effectuées et testées en local, nous allons les déployer sur serveur à un moment de fréquentation minimum ou allons placer le site en maintenance durant le temps de l'implémentation des modifications selon le degré d'importance de celles-ci.

Cette façon de travailler est la plus sûre et la meilleure dans le cas de site personnels ou de petits sites. Cependant, dans le cas de sites déjà importants et sur lesquels de nombreuses personnes travaillent, il va être très compliqué de travailler comme cela puisque le site risque de changer entre

le moment de la copie et le moment où nos modifications sont terminées car d'autres développeurs auront eux-mêmes implémenté de nouvelles fonctionnalités.

Dans ce cas-là, l'entreprise aura tout intérêt à mettre en place ce qu'on appelle une pré-production ou « préprod ». L'idée ici va être de copier l'ensemble du site sur un autre serveur test qui ne sera accessible que pour certaines personnes (pour toutes les personnes de l'entreprise par exemple).

Cette solution, plus compliquée à mettre en place lorsqu'on est seul, permet de travailler sereinement à plusieurs sur plusieurs nouvelles fonctionnalités en parallèle et permet également de tester « en conditions réelles » l'impact des différentes modifications sur le site de base et sur les autres.

Pour ce cours, je pars du principe que vous n'avez pas encore de site et que vous travaillez seul. Nous travaillerons donc en local c'est-à-dire chacun avec des fichiers stockés sur nos propres ordinateurs

Choix et installation d'un éditeur de texte

Qu'est-ce qu'un éditeur de texte ?

Un éditeur de texte est un programme qui va nous permettre d'écrire des lignes de code et de simplifier l'écriture de ce code en embarquant une panoplie de fonctionnalités utiles comme l'auto-complétion de certaines commandes de code et etc.

Un éditeur de texte ne doit pas être confondu avec un outil de traitement de texte comme Word. L'éditeur de texte a véritablement vocation à créer des pages de code dans n'importe quel langage comme le HTML, le CSS, le JavaScript ou même encore le PHP en utilisant du texte brut tandis qu'un outil de traitement de texte comme Word permet de créer du texte « enrichi » c'est-à-dire du texte mis en forme (choix de l'alignement, de la police, du poids de celle-ci, etc.).

Aujourd'hui, chaque système d'exploitation reconnu (Windows, Mac, Linux) embarque son ou ses propres éditeurs de texte nativement, ce qui signifie que votre ordinateur dispose déjà certainement d'un éditeur de texte pré-installé vous permettant de créer des pages de code en HTML, CSS, etc.

Comment bien choisir son éditeur de texte ?

Il existe aujourd'hui des milliers d'éditeurs disponibles sur le web.

Certains fonctionnent exclusivement sous certains environnements (Windows, Mac Os, etc.) tandis que d'autres vont être cross-plateformes (fonctionner sous plusieurs environnements). Certains éditeurs de texte vont être gratuits tandis que d'autres vont être payants ou disponibles sous forme d'abonnement. Le plus connu des éditeurs de texte est certainement NotePad++, le fameux éditeur fonctionnant avec Windows.

Un éditeur va donc nous permettre de pouvoir écrire des pages de code et de les enregistrer au bon format (c'est-à-dire avec la bonne extension). Un bon éditeur de texte est selon moi un éditeur qui va proposer différentes options pour vous aider à coder. Parmi ces options, on peut notamment citer :

- La mise en couleur des différents éléments d'un langage informatique pour pouvoir les distinguer (un bon éditeur de texte fera apparaitre les éléments HTML d'une couleur différente des attributs, etc.) ;
- L'affichage d'indications lorsque l'on fait une faute de syntaxe dans le code avec une explication du problème en question ;
- L'auto-complétion des balises (un bon éditeur écrira automatiquement la balise fermante d'un élément HTML par exemple dès que vous aurez écrit la balise ouvrante) ;
- Des bibliothèques (notamment jQuery) intégrées ou téléchargeables et installables en 1 clic ;
- Une indentation automatique et pertinente.

Aux vues de ce que nous allons réaliser durant ce cours, et pour un débutant, la plupart des éditeurs vont se valoir et vont parfaitement convenir. Je vous conseille donc simplement d'en trouver un compatible avec votre système et qui a déjà fait ses preuves.

Personnellement, j'utiliserai pour ce cours Notepad++.

Une fois l'éditeur de votre choix installé, n'hésitez pas à l'ouvrir pour vous familiariser avec son affichage et découvrir les nombreuses options qu'il propose.

Un bon éditeur de texte devrait avoir un fond foncé lorsque vous créez une nouvelle page. Si ce n'est pas le cas, je vous conseille fortement de changer la couleur de fond (vous devrez certainement également changer la couleur de votre code afin d'y voir) pour coder avec un plus grand confort visuel. Rem : pour la rédaction du cours, j'ai laissé un fond blanc pour les exemples, car un fond noir aurait consommé beaucoup d'encre à l'impression.

Voici une liste d'autres éditeurs reconnus si jamais vous voulez davantage de choix :

- Microsoft Visual Code : Microsoft Visual Code est le dernier éditeur à la mode parmi les développeurs et ceci pour de bonnes raisons. Gratuit, il dispose de toutes les fonctionnalités qu'un développeur peut attendre et d'une très bonne ergonomie ;
- Atom : Atom est doté d'une excellente ergonomie qui facilite grandement la prise en main et l'approche du code pour les nouveaux développeurs. Cet éditeur de texte dispose de toutes les fonctions qu'on attend d'un bon éditeur : bibliothèques intégrées, auto-complétion des balises, etc.
- NotePad++ : Certainement l'éditeur de texte le plus connu de tous les temps, NotePad++ est également l'un des plus anciens. Il a passé le test du temps et a su s'adapter au fur et à mesure en ajoutant des fonctionnalités régulièrement comme l'auto-complétion des balises, le surlignage des erreurs de syntaxe dans le code etc. Le seul bémol selon moi reste son interface qui est à peaufiner.
- Brackets : Brackets est un éditeur très particulier puisqu'il est tourné uniquement vers les langages de développement front-end (c'est-à-dire HTML, CSS et JavaScript). Cependant, il dispose d'une excellente ergonomie (UI / UX) et d'un support extensif pour les langages supportés.
- Komodo

Logiciel éditeur de texte contre éditeur en ligne

Si vous vous intéressez à la programmation, vous connaissez peut-être les sites codepen.io ou jsbin.com ? Ces deux sites permettent d'écrire du code HTML, CSS ou JavaScript et de voir le résultat immédiatement.

En cela, ils servent le même rôle qu'un éditeur de texte HTML mais sont encore plus pratiques, notamment lorsque vous voulez tester rapidement un bout de code ou pour des démonstrations de cours. J'utiliserai d'ailleurs souvent codepen.io dans mes leçons pour vous fournir directement un code et pour que vous puissiez le voir immédiatement en action ou le modifier.

Cependant, retenez bien qu'ils sont aussi limités car il y a plusieurs choses que vous ne pourrez pas faire en termes de développement avec ces sites. Parmi celles-ci, on notera que vous ne pourrez par exemple pas exécuter de code PHP ou un quelconque code utilisant un langage dit server-side ou

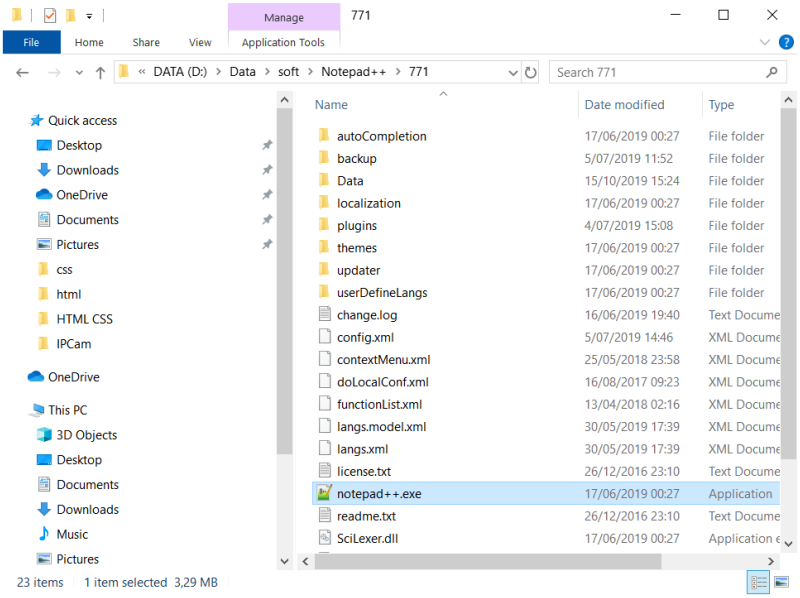
encore que vous ne pourrez pas à proprement parler créer plusieurs pages et les lier entre elles (comme c'est le cas lorsque l'on doit créer un site) ou du moins pas gratuitement.

Cette solution n'est donc pas satisfaisante si vous souhaitez véritablement vous lancer dans le développement et c'est la raison pour laquelle tous les développeurs utilisent un éditeur de texte.

Je vous conseille donc durant ce cours d'utiliser un maximum votre éditeur pour bien vous familiariser avec celui-ci et pour assimiler les différentes syntaxes des langages que l'on va étudier.

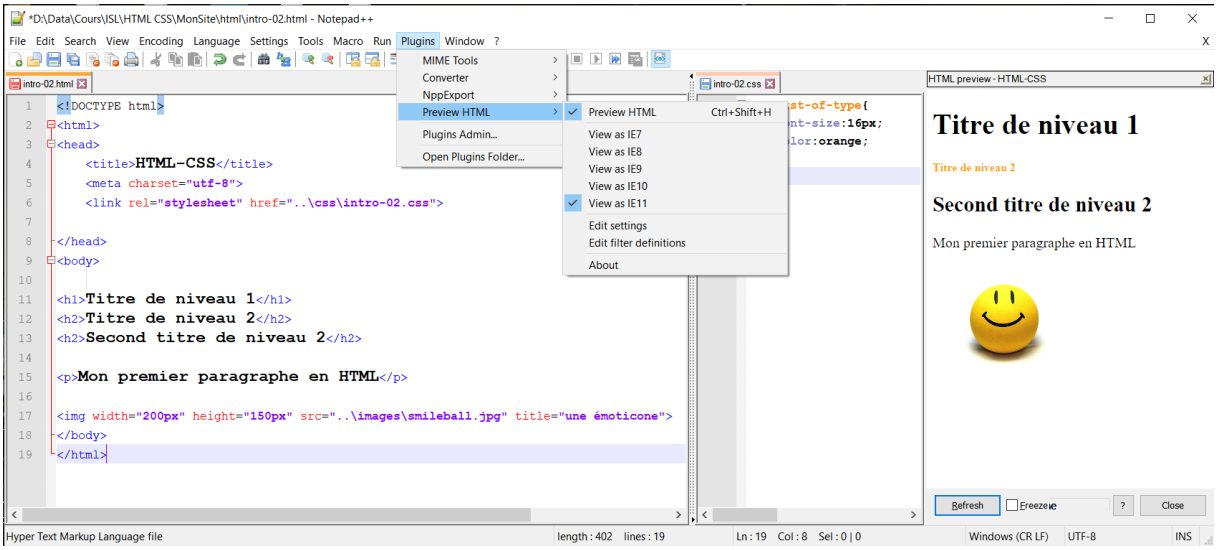
Utilisation de Notepad++

Notepad++ s'installe de deux manières, soit comme une application (setup), vous devez donc être administrateur de votre ordinateur ou comme un exécutable dans un répertoire. Dans ce dernier cas, il suffit d'extraire les fichiers dans un répertoire et de double-cliquer le fichier notepad++.exe.

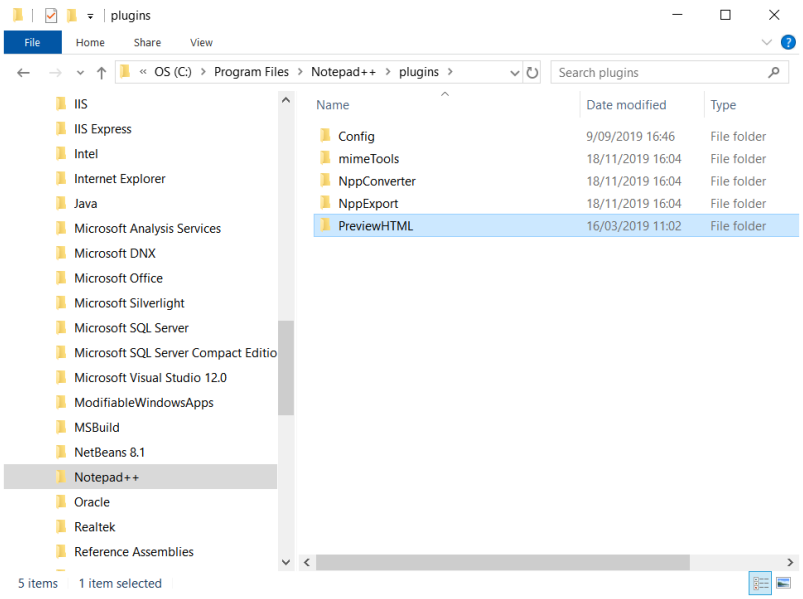


Notepad++ fonctionne, comme un éditeur pour plusieurs langages de programmation et avec un système de plugins qu'il convient de gérer.

Dans le cadre de ce cours, le seul plugin que j'ai installé est « Preview HTML », qui vous donne un aperçu direct « type navigateur » (colonne de droite).



Un menu « plugins/Plugins admin » vous permet de gérer les plugins installés. Les plugins sont activables directement ou téléchargeables sur le site de Notepad et s'installent sous forme de fichiers .dll dans le répertoire plugins du programme. Comme j'ai installé le plugin « PreviewHTML, un répertoire PreviewHTML a été créé.



Il sera intéressant pour les exercices de customiser l'éditeur sous la forme 3 colonnes (HTML, CSS, Preview HTML).