Les sélecteurs complexes en CSS

Rédigé par: Jean-Luc Colson

Date première rédaction: Janvier 2020

SUIVI DES MODIFICATIONS A LA FICHE

Série	Date	Page(s) modifiée(s)	Raison

Contents

Les sélecteurs CSS d'attributs 4
Sélectionner un élément selon qu'il possède un attribut 4
Sélectionner un élément selon qu'il possède un attribut avec une valeur exactement 4
Sélectionner un élément selon qu'il possède un attribut **contenant** une sous-valeur distincte (séparée du reste par un espace) 5
Sélectionner un élément selon qu'il possède un attribut commençant par une certaine sous-valeur 6
Sélectionner un élément selon qu'il possède un attribut se finissant par une certaine sous-valeur 7
Sélectionner un élément selon qu'il possède un attribut possédant une valeur parmi d'autres 8
Rendre ses sélecteurs d'attributs insensibles à la casse 9
Tableau récapitulatif des différents sélecteurs d'attributs 10
Les pseudo-classes CSS 12
Définition et syntaxe des pseudo-classes 12
Les pseudo-classes :hover, :visited, :active et :link 12
Les pseudo-classes :first-child, :first-of-type, :last-child et :last-of-type 13
Les pseudo-classes :nth-child() et :nth-of-type() 15
Liste complète des pseudo-classes et définition 16
Les pseudo-éléments CSS 20
Insérer du contenu avant ou après le contenu d'un élément HTML 20
Sélectionner la première lettre d'un élément en CSS 21
Sélectionner la première ligne d'un élément en CSS 22
Cibler la partie d'un élément déjà sélectionnée par l'utilisateur 22

Les sélecteurs CSS d'attributs

L'une des grandes forces du CSS réside dans le fait qu'on va pouvoir cibler très précisément tel ou tel élément HTML grâce à la grande variété de ces sélecteurs.

Cette partie est dédiée à l'étude de trois types de sélecteurs que nous n'avons pas encore étudiés : les sélecteurs d'attributs, les pseudo-classes et les pseudo-éléments.

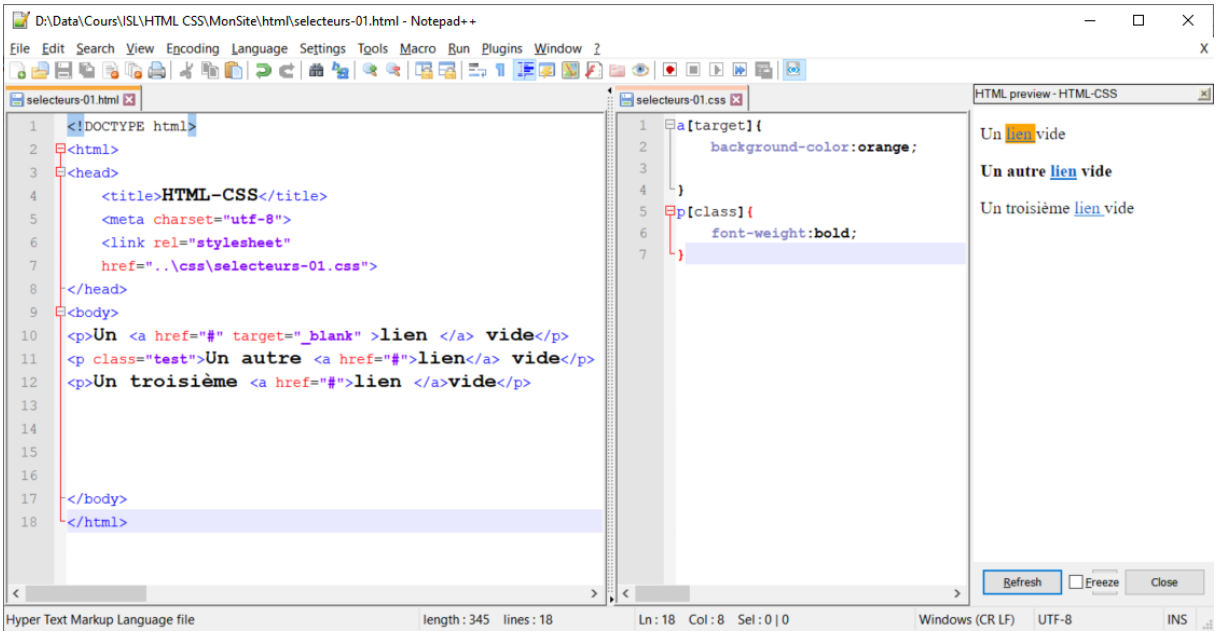
Dans cette leçon, nous allons déjà apprendre à manipuler les sélecteurs d'attributs qui vont nous permettre de sélectionner un élément HTML selon qu'il possède un attribut avec une certaine valeur ou pas.

Sélectionner un élément selon qu'il possède un attribut

Pour commencer, nous allons en CSS pouvoir cibler un élément HTML selon qu'il possède un certain attribut comme un attribut href, src, target, class, etc.

Pour faire cela, nous allons utiliser la syntaxe suivante : E[foo] où « E » représente un nom d'élément et « foo » représente un nom d'attribut.

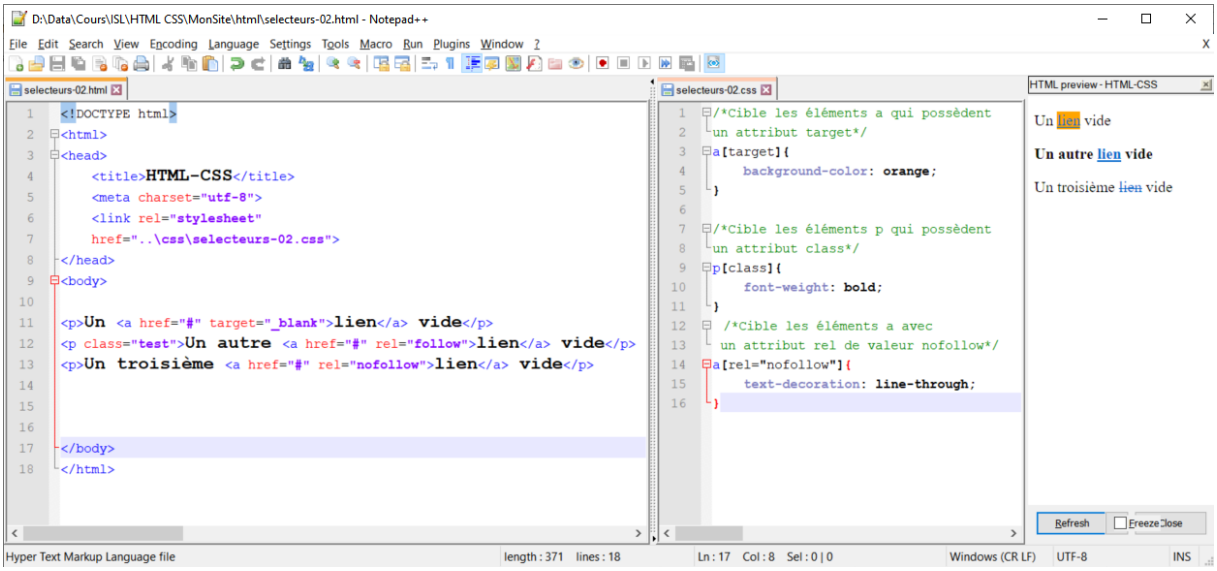
Par exemple, on va pouvoir cibler tous les éléments a qui possèdent un attribut target en écrivant a[target] ou encore tous les éléments p qui possèdent un attribut class avec p[class].



Sélectionner un élément selon qu'il possède un attribut avec une valeur exactement

Nous allons également pouvoir sélectionner de façon plus précise les éléments HTML possédant un attribut auquel on a attribué une valeur en particulier en CSS. Par exemple, nous allons pouvoir

sélectionner tous les éléments a possédant un attribut rel avec une valeur exactement égale à nofollow en utilisant la syntaxe a[rel="nofollow"].



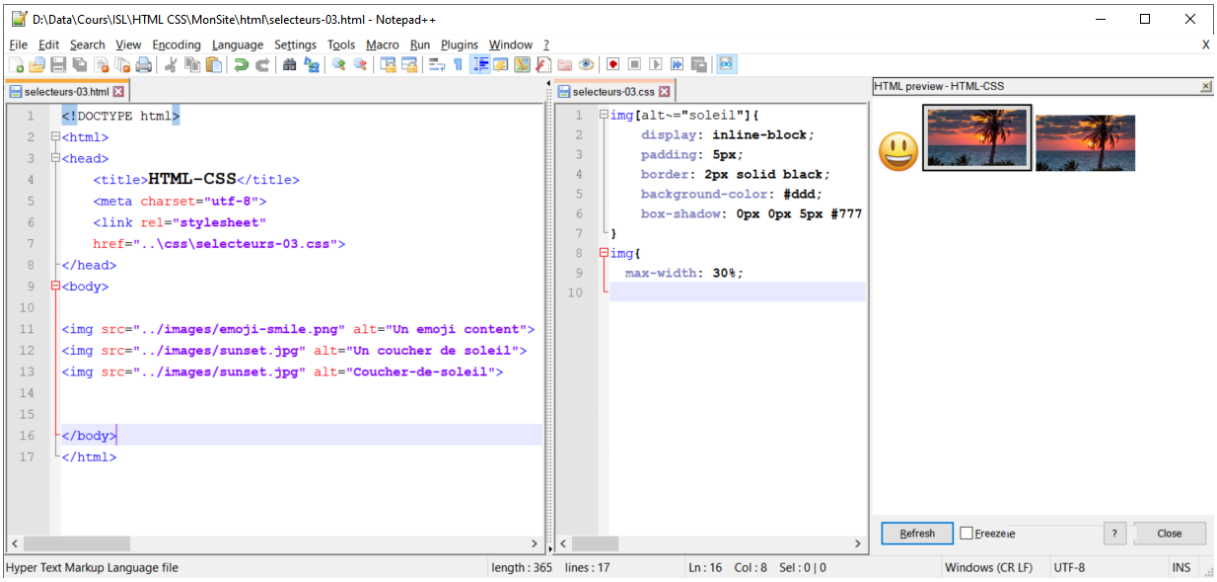
Note : l'attribut rel sert à indiquer la relation entre le document de départ et le document lié. Ici, la valeur nofollow permet d'indiquer aux robots des moteurs de recherche qu'il n'est pas nécessaire de suivre le lien. Cet attribut est souvent utilisé à des fins d'optimisation du référencement.

Sélectionner un élément selon qu'il possède un attribut contenant une sous-valeur distincte (séparée du reste par un espace)

Nous allons encore pouvoir sélectionner en CSS un élément HTML possédant un attribut qui contient une certaine sous-valeur distincte des autres, c'est-à-dire séparée des autres par une espace.

Ici, je parle de « sous-valeur » pour désigner une partie de la valeur d'un attribut, c'est-à-dire pour désigner un ou plusieurs caractères inclus dans la valeur passée à l'attribut.

Par exemple, on va pouvoir cibler tous les éléments img d'une page dont l'attribut alt contient le texte « soleil » grâce à la syntaxe suivante : img[alt~="soleil"].

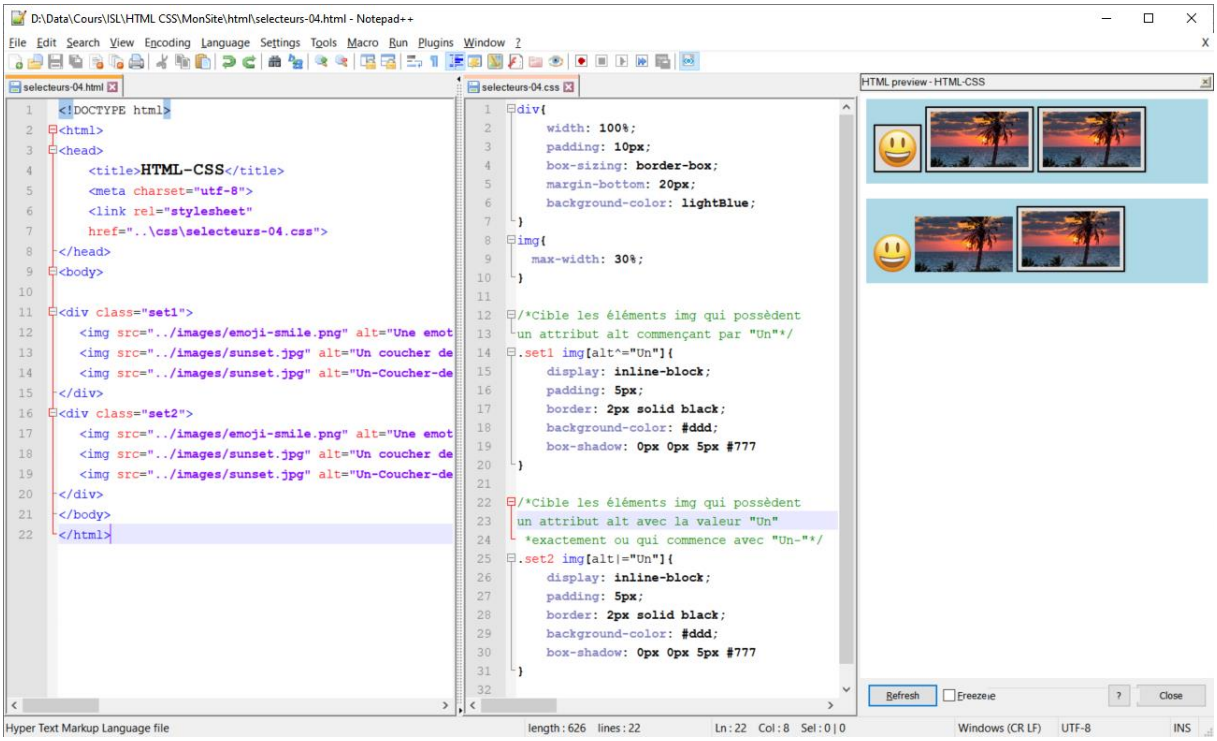


Sélectionner un élément selon qu'il possède un attribut commençant par une certaine sous-valeur

Pour sélectionner un élément E selon qu'il possède un attribut foo commençant par la sous valeur val, nous utiliserons la syntaxe suivante : E[foo^=val].

En utilisant ce sélecteur, on n'impose pas plus de contrainte sur la sous-valeur val. Celle-ci peut donc faire partie d'un mot plus grand par exemple.

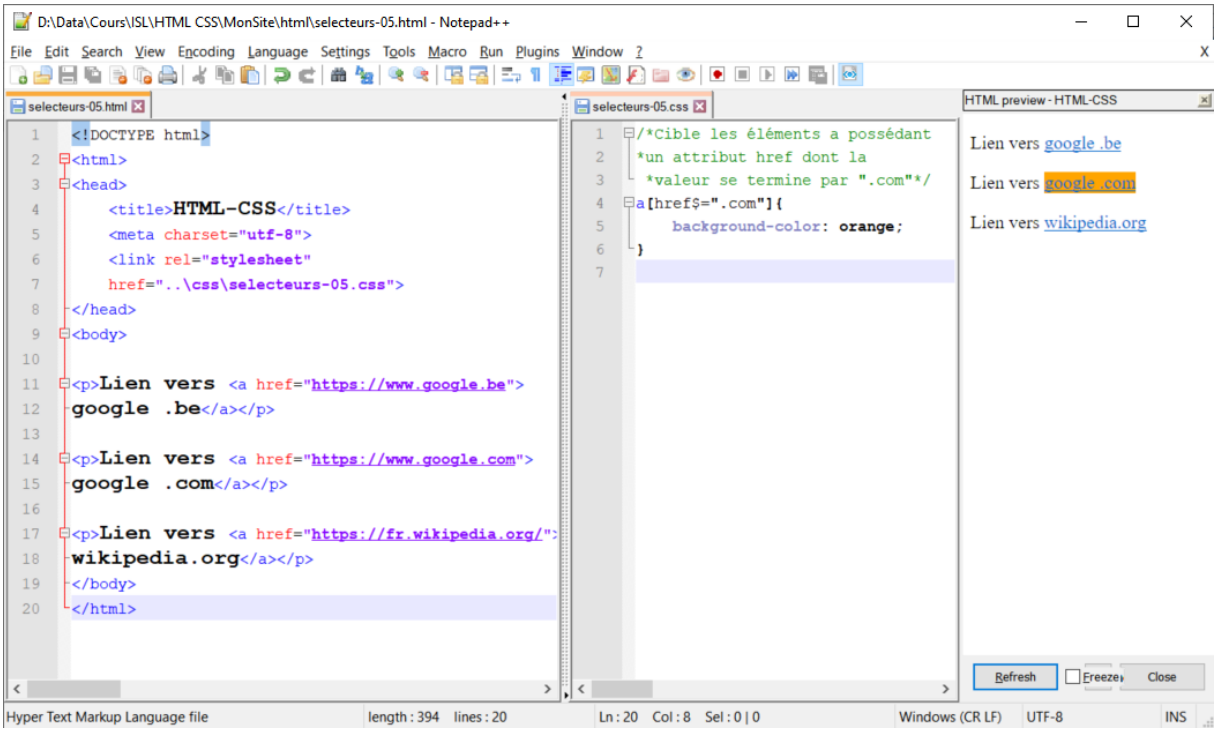
On va également pouvoir utiliser le sélecteur E[foo|=val] dans le cas où l'on souhaite cibler un élément possédant un attribut foo dont la valeur est exactement val ou commence par val suivi d'un tiret - (hyphen) en anglais).



Sélectionner un élément selon qu'il possède un attribut se finissant par une certaine sous-valeur

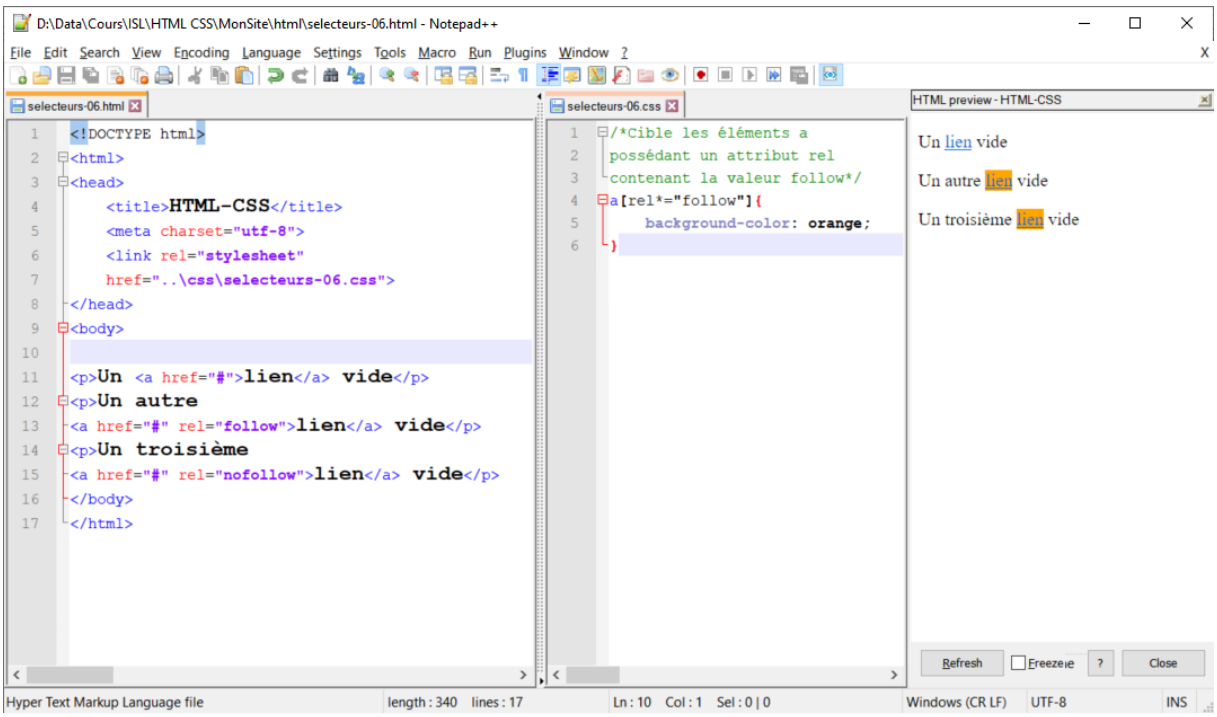
Nous allons encore pouvoir en CSS cibler des éléments HTML possédant un attribut dont la valeur se termine par une certaine valeur, sans plus de restriction que cela.

Nous allons ainsi par exemple pouvoir cibler tous nos éléments a possédant un attribut href dont la valeur se termine par « .com » avec la syntaxe suivante : a[href$=".com"].



Sélectionner un élément selon qu'il possède un attribut possédant une valeur parmi d'autres

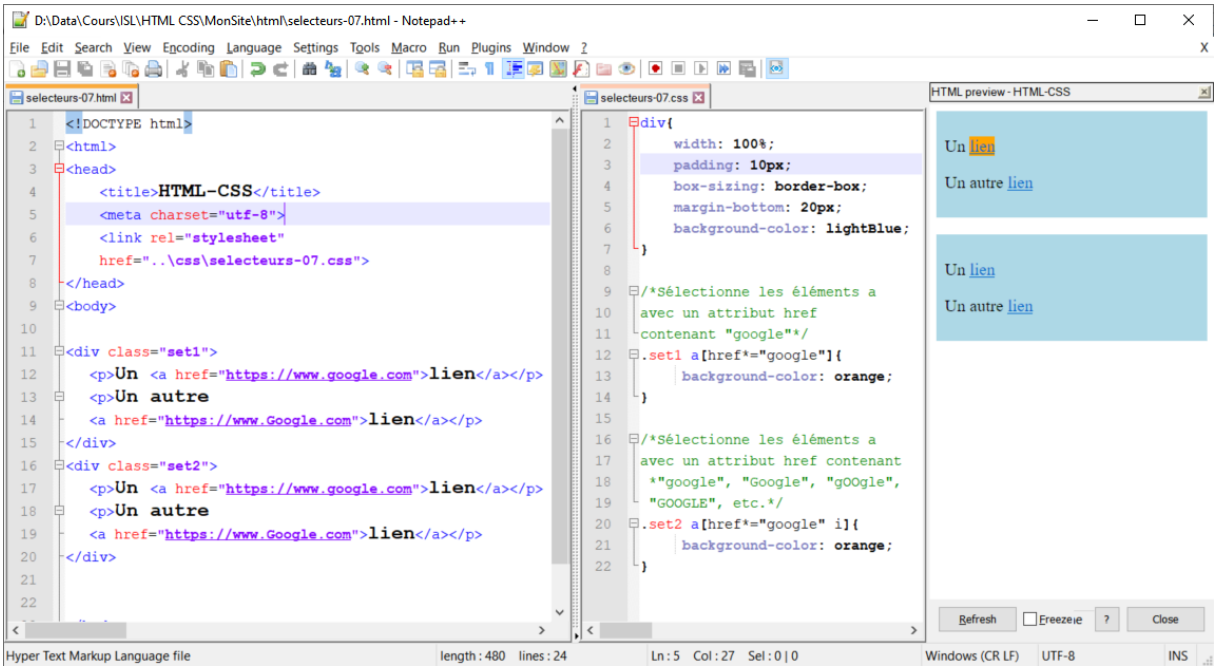
Finalement, on va pouvoir cibler en CSS un élément HTML qui possède un attribut contenant lui-même une certaine sous-valeur sans aucune restriction en utilisant la syntaxe E[foo*="val"] (où E représente un élément, foo représente un attribut et val une sous-valeur.



Rendre ses sélecteurs d'attributs insensibles à la casse

Par défaut, les sélecteurs d'attributs vont être sensibles à la casse c'est-à-dire faire la différence entre un caractère en majuscule et en minuscule. Ainsi, les valeurs « un », « UN », « Un » et « uN » vont être considérées comme différentes.

On va pouvoir changer ce comportement et rendre nos sélecteurs insensibles à la casse en ajoutant la lettre i (I minuscule) à la fin de nos sélecteurs d'attributs.



Attention cependant : cette notation est récente est n'est à l'heure actuelle par encore une recommandation et n'est pas supportée par les navigateurs Edge ni d'Internet Explorer.

Tableau récapitulatif des différents sélecteurs d'attributs

Vous pourrez retrouver ci-dessous les différents sélecteurs d'attributs avec leur définition et la version CSS où ils ont été passés en recommandation.

Sélecteur	**Description**	**Recommandation CSS**
E[foo]	Sélectionne tout élément E possédant un attribut foo	CSS2
E[foo="bar"]	Sélectionne tout élément E possédant un attribut foo dont la valeur est exactement « bar »	CSS2
E[foo~="bar"]	Sélectionne tout élément E possédant un attribut foo dont la valeur contient distinctement « bar » (c'est-à-dire dont la valeur contient le mot « bar » séparé du reste par des espaces)	CSS2
E[foo|="en"]	Sélectionne tout élément E possédant un attribut foo dont la valeur commence par « en » séparé du reste par un tiret (ou hyphen en anglais)	CSS2
E[foo^="bar"]	Sélectionne tout élément E possédant un attribut foo dont la valeur commence exactement par « bar »	CSS3
E[foo$="bar"]	Sélectionne tout élément E possédant un attribut foo dont la valeur se termine exactement par « bar »	CSS3
E[foo*="bar"]	Sélectionne tout élément E possédant un attribut foo dont la valeur contient la valeur « bar »	CSS3

Sélecteur	Description	Recommandation CSS
E[foo{~|^$*}="bar" i]	L'ajout d'un i à la fin des sélecteurs précédents rend la sélection « case insensitive » c'est-à-dire « insensible à la casse ». La valeur recherchée ne tiendra pas compte de la casse, c'est-à-dire de l'utilisation de majuscules et de minuscules.	4

Les pseudo-classes CSS

Les pseudo-classes vont nous permettre d'appliquer des styles à des éléments HTML uniquement lorsque ceux-ci sont dans un certain état (cliqués, activés, etc.).

Il existe de nombreuses pseudo-classes en CSS. Dans cette leçon, nous n'allons étudier en détail que les plus utilisées car toutes les pseudo-classes s'utilisent de manière similaire (la syntaxe sera toujours la même).

Ainsi, il vous suffira de comprendre comment appliquer une pseudo-classe pour savoir toutes les utiliser, à condition bien évidemment de savoir à quoi correspond chaque pseudo-classe.

Pour cela, je vais également vous fournir un tableau récapitulatif de toutes les pseudo-classes actuellement disponibles afin que vous puissiez piocher dedans selon vos besoins.

Définition et syntaxe des pseudo-classes

Les pseudo-classes vont nous permettre de cibler des éléments HTML en fonction de leur état, ou plus exactement d'appliquer des règles CSS à des éléments HTML uniquement dans un certain contexte (lorsqu'ils sont dans un certain état).

Ainsi, nous allons pouvoir modifier les styles d'un lien HTML selon que le curseur de la souris d'un visiteur soit dessus (état hover) ou qu'il ait été déjà cliqué (état visited).

Pour utiliser une pseudo-classe en CSS, il faudra commencer avec un sélecteur « classique » (simple ou complexe) qui sera suivi du caractère : puis du nom de la pseudo-classe ou de l'état à appliquer.

Par exemple, on va pouvoir changer la couleur de fond des éléments div d'une page en orange lors du survol du curseur de la souris uniquement en écrivant en CSS div:hover{background-color : orange ;}.

Les pseudo-classes :hover, :visited, :active et :link

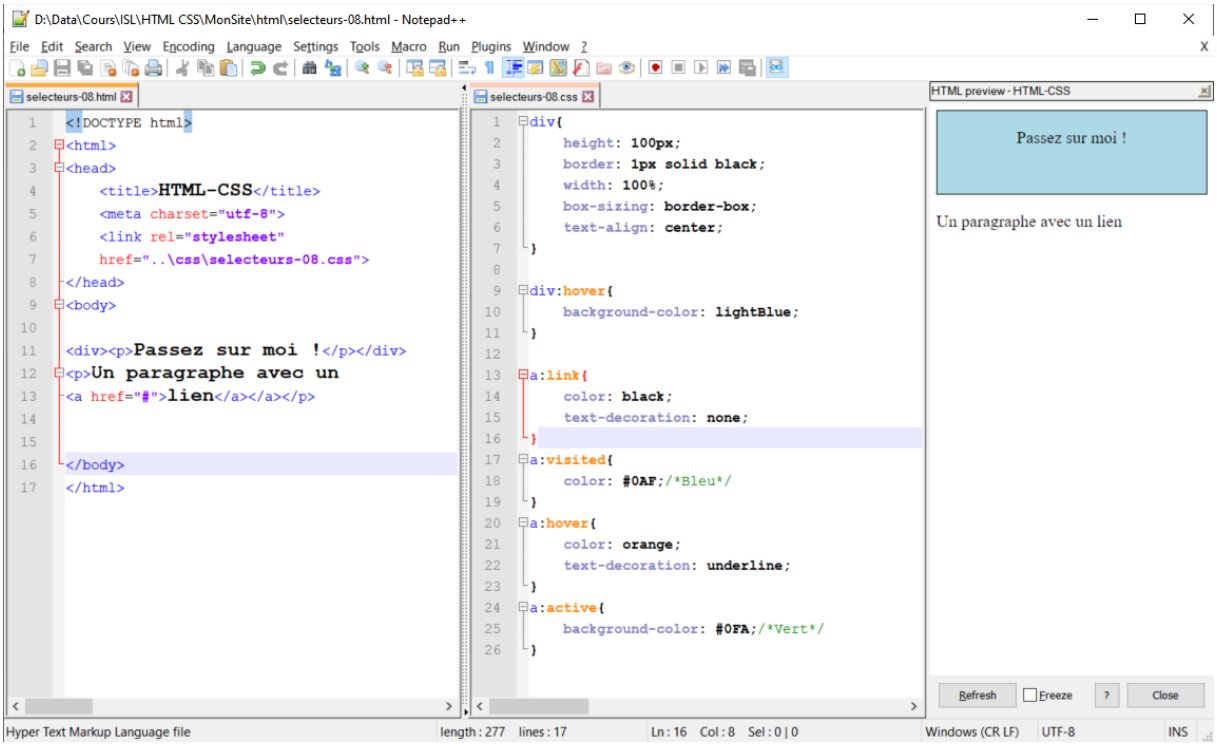
La pseudo-classe :hover va nous permettre d'appliquer des styles à un élément HTML uniquement lorsque celui-ci est survolé par la souris d'un utilisateur. Cette pseudo-classe peut être utilisée avec la majorité des éléments HTML mais on l'utilisera généralement pour appliquer des styles différents à des liens lorsqu'un utilisateur passe sa souris dessus.

La pseudo-classe CSS :visited va nous permettre d'appliquer des styles à un élément déjà visité, c'est-à-dire déjà cliqué. En pratique, cette pseudo-classe va une nouvelle fois surtout être utilisée avec des éléments de lien. Ainsi, on va pouvoir changer l'apparence d'un lien après que celui-ci ait été cliqué, lorsque l'utilisateur revient sur notre page de départ.

La pseudo-classe CSS :active va elle nous permettre de modifier les styles d'un élément HTML lors d'un état très particulier qui est le moment où l'on clique sur l'élément. Pour bien visualiser cet état, je vous conseille de rester cliqué sur l'élément en question le temps de voir les changements de style.

Une nouvelle fois, en pratique, cette pseudo-classe va généralement être utilisée pour modifier l'apparence des liens au moment du clic.

La pseudo-classe :link va elle nous permettre au contraire de cibler tous les liens non visités et de leur appliquer des styles particuliers en CSS.




```
<!DOCTYPE html>
<html>
<head>
    <title>HTML-CSS</title>
    <meta charset="utf-8">
    <link rel="stylesheet"
    href="..\css\selecteurs-08.css">
</head>
<body>

<div><p>Passez sur moi !</p></div>
<p>Un paragraphe avec un
<a href="#">lien</a></a></p>


</body>
</html>
```

```
div{
    height: 100px;
    border: 1px solid black;
    width: 100%;
    box-sizing: border-box;
    text-align: center;
}

div:hover{
    background-color: lightBlue;
}

a:link{
    color: black;
    text-decoration: none;
}
a:visited{
    color: #0AF;/*Bleu*/
}
a:hover{
    color: orange;
    text-decoration: underline;
}
a:active{
    background-color: #0FA;/*Vert*/
}
```

Notez qu'en cas d'application de plusieurs des pseudo-classes ci-dessus à un élément on respectera l'ordre d'écriture suivant en CSS : d'abord les déclarations liées à :link puis :visited puis :hover et enfin :active.

Cela va permettre d'avoir le comportement le plus logique en cas de conflit : les styles liés à :active s'appliqueront uniquement lors du clic sur l'élément, ceux liés à :hover s'appliqueront lors du survol sans clic et etc.

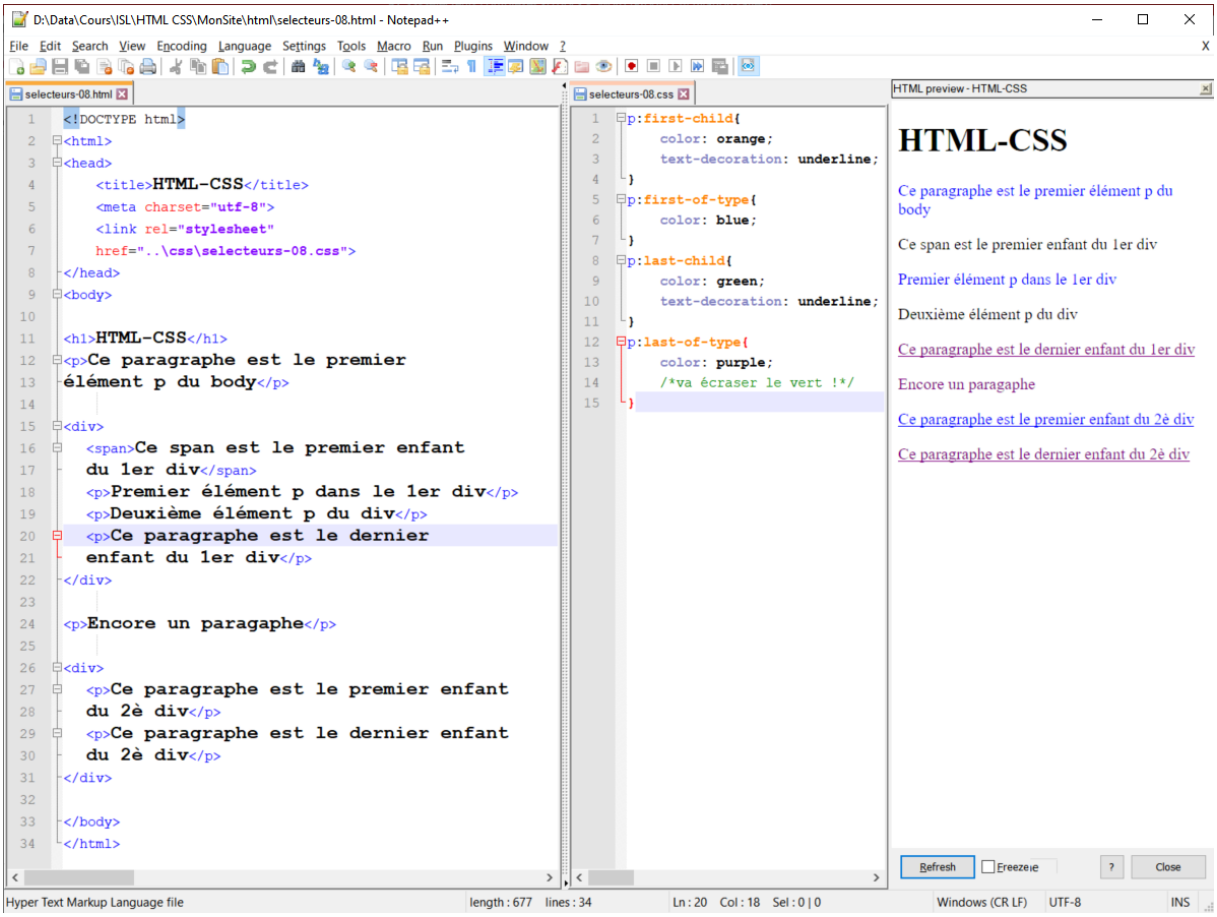
Les pseudo-classes :first-child, :first-of-type, :last-child et :last-of-type

Les pseudo-classes :first-child et :first-of-type vont nous permettre de sélectionner respectivement un élément qui va être le premier enfant de son parent et un élément qui va être le premier élément de ce type de son parent.

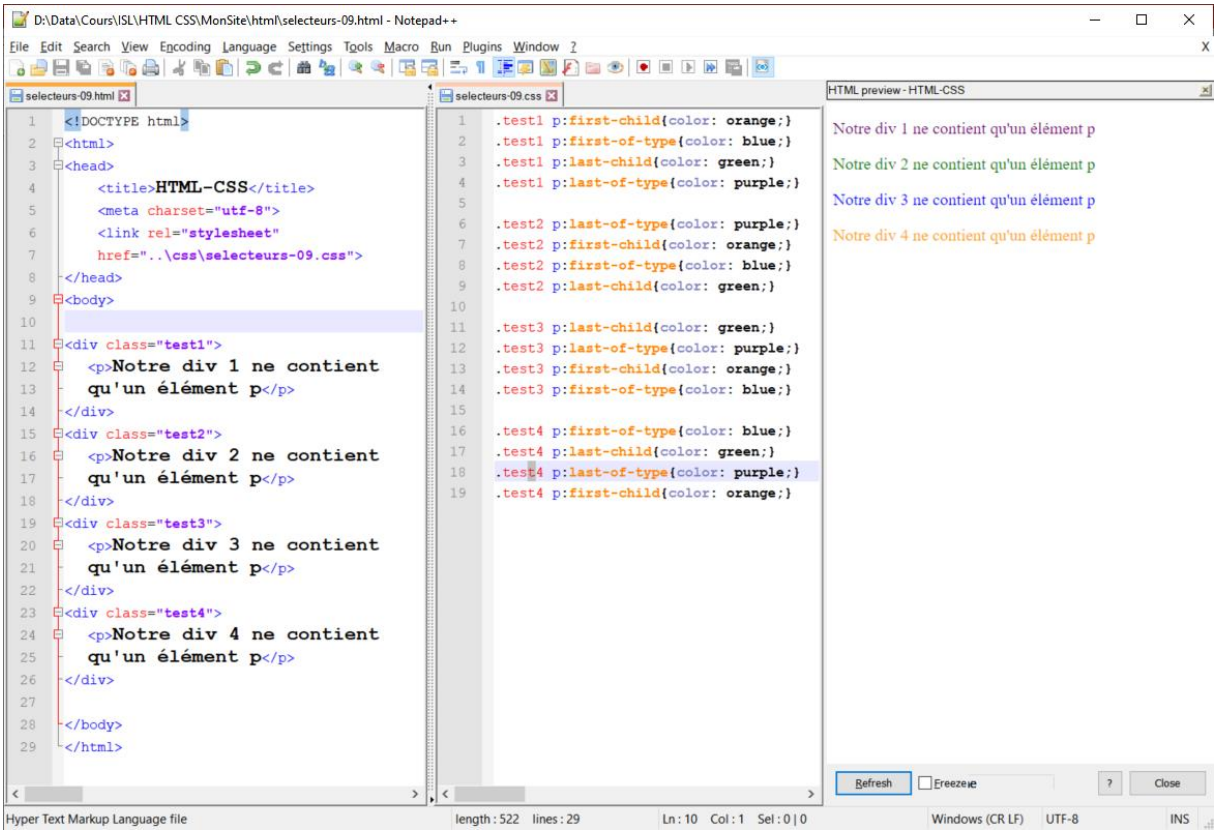
Par exemple, le sélecteur p:first-child va sélectionner tous les éléments p d'une page qui sont les premiers enfants de leur parent tandis que le sélecteur CSS p:first-of-type va nous permettre de sélectionner tous les éléments p qui sont le premier élément p enfant de leur parent.

A l'inverse, les pseudo-classes CSS :last-child et :last-of-type vont nous permettre de sélectionner respectivement un élément qui va être le dernier enfant de son parent et un élément qui va être le dernier élément de ce type de son parent.

Ainsi, le sélecteur CSS last-child va sélectionner tous les éléments p d'une page qui sont les derniers enfants de leur parent tandis qu'on va pouvoir sélectionner tous les éléments p qui sont le dernier élément p enfant de leur parent grâce au sélecteur CSS last-of-type.



Notez que dans si plusieurs pseudo-classes ciblent un même élément, alors les styles appliqués en cas de conflit à l'élément seront une nouvelle fois ceux de la dernière pseudo-classe déclarée en CSS. Regardez plutôt le code ci-dessous pour vous en convaincre :

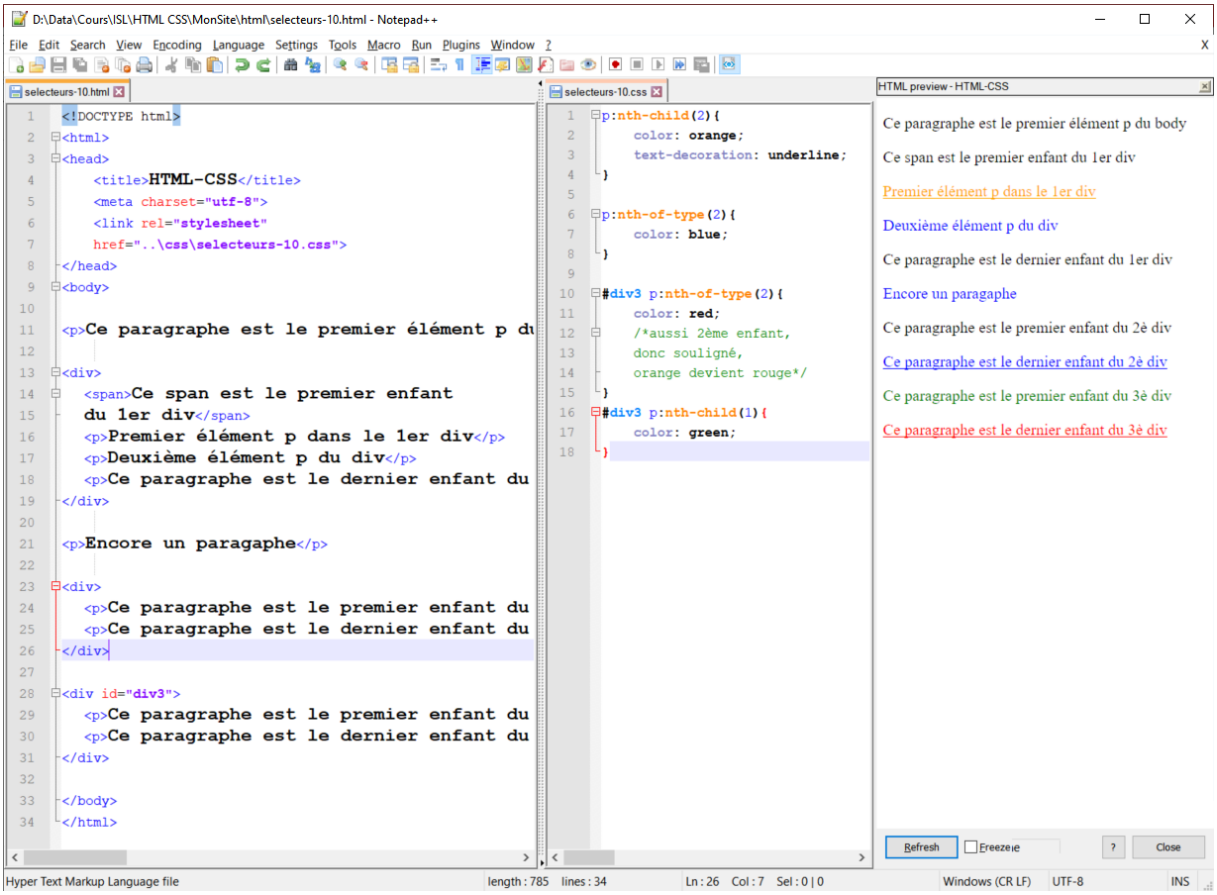


Ici, chacun de nos éléments div ne contient qu'un paragraphe qui est donc à la fois le premier enfant du div, premier élément de ce type, dernier enfant et dernier élément de ce type. On se contente d'inverser l'ordre des déclarations en CSS à chaque fois.

Les pseudo-classes :nth-child() et :nth-of-type()

Pour pouvoir cibler n'importe quel élément HTML en fonction de sa place dans le document, nous allons également pouvoir utiliser les sélecteurs CSS :nth-child(n) et nth-of-type(n) qui vont nous permettre de sélectionner un élément HTML qui est le énième enfant de son parent ou un élément qui est le énième enfant de ce type de son parent.

Par exemple, le sélecteur CSS p:nth-child(2) va nous permettre de sélectionner tous les éléments p qui sont les deuxièmes enfants de leur parent tandis que le sélecteur p:nth-of-type(2) sélectionne tous les paragraphes qui sont les deuxièmes enfants de type p de leur parent.



Là encore, si plusieurs pseudo-classes ciblent le même élément (que ce soit :first-child, :nth-child(), :nth-of-type() ou :last-child, etc. alors en cas de conflit les styles de la dernière pseudo-classe déclarée s'appliqueront.

Liste complète des pseudo-classes et définition

Vous pourrez trouver ci-dessous toutes les pseudo-classes avec la description de leur comportement et la version CSS dans laquelle elles sont passées comme recommandation pour référence.

Les pseudo-classes dont la version est « 3/4 » sont celles dont la définition initiale a été posée avec le CSS3 mais dont la définition va certainement être modifiée avec la prochaine version du CSS.

De plus, notez que certaines pseudo-classes ont été créées pour ne fonctionner qu'avec certains éléments en particulier et notamment avec des éléments de formulaire que nous étudierons plus tard.

Sélecteur	Description	Version CSS
E:link	Sélectionne tout élément E représentant l'ancre d'un lien non visité jusqu'à présent	CSS1
E:visited	Sélectionne tout élément E représentant l'ancre d'un lien déjà visité	CSS1
E:active	Sélectionne un élément E au moment où il est cliqué	CSS1
E:hover	Sélectionne un élément E lorsque le curseur de la souris passe dessus	CSS2
E:focus	Sélectionne un élément E qui a le focus (dans lequel le curseur de la souris est placé)	CSS2
E:first-child	Sélectionne tout élément E étant le premier enfant de son parent	CSS2
E:lang(fr)	Sélectionne tout élément E dont l'attribut langage possède la valeur « fr »	CSS2
E:target	Sélectionne un élément E contenant une ancre qui vient d'être cliquée à partir d'un lien ancre	CSS3
E:enabled	Sélectionne tout élément E avec lequel l'utilisateur peut interagir et qui est activé	CSS3
E:disabled	Sélectionne tout élément E avec lequel l'utilisateur peut interagir et qui est desactivé	CSS3
E:root	Sélectionne un élément E racine du document	CSS3
E:empty	Sélectionne tout élément E qui ne possède pas d'enfant (ni de noeud de type texte)	CSS3

Sélecteur	Description	Version CSS
E:nth-child(n)	Sélectionne tout élément E étant le n-ième enfant de son parent	CSS3
E:nth-last-child(n)	Sélectionne tout élément E étant le n-ième enfant de son parent en comptant les enfants à partir du dernier	CSS3
E:checked	Sélectionne tout élément E de type input coché au sens large (checked ou selected)	CSS3
E:last-child	Sélectionne tout élément E étant le dernier enfant de son parent	CSS3
E:only-child	Sélectionne tout élément E qui est le seul enfant de son parent	CSS3
E:nth-of-type(n)	Sélectionne tout élément E étant le n-ième enfant d'un certain type par rapport à son parent	CSS3
E:nth-last-of-type(n)	Sélectionne tout élément E étant le n-ième enfant d'un certain type par rapport à son parent en comptant à partir de la fin	CSS3
E:first-of-type	Sélectionne tout élément E premier enfant de son type par rapport à son parent	CSS3
E:last-of-type	Sélectionne tout élément E dernier enfant de son type par rapport à son parent	CSS3
E:only-of-type	Sélectionne tout élément E seul enfant de son type par rapport à son parent	CSS3
E:read-write	Sélectionne tout élément E de type input avec lequel l'utilisateur peut interagir (comme un champ dans lequel il peut écrire par exemple)	3-UI/4
E:read-only	Sélectionne tout élément E de type input avec lequel l'utilisateur ne peut pas interagir (éléments possédant un attribut disabled par exemple)	3-UI/4

Sélecteur	Description	Version CSS
E:placeholder-shown	Sélectionne tout élément E qui affiche actuellement la valeur de son attribut placeholder	3-UI/4
E:default	Sélectionne un élément E dans une liste ou un groupe qui est l'élément défini par défaut	3-UI/4
E:valid	Sélectionne tout élément E de type input dont la valeur est évaluée comme valide (dont la valeur possède une forme correspondant à ce qui est attendu)	3-UI/4
E:invalid	Sélectionne tout élément E de type input dont la valeur est évaluée comme invalide (valeur ne correspondant pas à ce qui est attendu)	3-UI/4
E:in-range	Sélectionne tout élément E de type input dont la valeur fournie se situe dans une fourchette de valeurs prédéfinies	3-UI/4
E:out-of-range	Sélectionne tout élément E de type input dont la valeur fournie se situe en dehors d'une certaine fourchette de valeurs prédéfinies	3-UI/4
E:required	Sélectionne tout élément E de type input dont la valeur doit être renseignée (élément possédant un attribut required)	3-UI/4
E:optional	Sélectionne tout élément E de type input dont la valeur ne doit pas obligatoirement être renseignée	3-UI/4
E:not(E1, .c1)	Sélectionne tout élément E qui n'est pas de type E1 et qui ne possède pas d'attribut class= »c1"	3/4

Notez qu'il existe encore d'autres pseudo-classes très particulières comme :first (pour gérer l'impression de la première page d'un document) ou qui ne sont pas encore au statut de recommandation mais qui sont pour le moment en développement comme dir(), scope, drop, indeterminate, etc.

Les pseudo-éléments CSS

Les pseudo-éléments CSS vont nous permettre de ne cibler qu'une partie d'un élément HTML pour lui appliquer des styles. Il convient de ne pas les confondre avec les pseudo-classes qui elles servent à cibler un élément selon son état.

Les pseudo-éléments ont une syntaxe particulière puisqu'il faudra utiliser un double deux-points :: entre le sélecteur élément et le mot clef désignant la partie de l'élément qui va être ciblé.

Il existe aujourd'hui 4 pseudo-éléments qui sont des recommandations du W3C :

- ::first-letter ;
- ::first-line ;
- ::before ;
- ::after.

Il existe également un cinquième pseudo-élément ::selection qu'il convient de connaitre car celui-ci devait faire partie des recommandations du CSS3 puis a finalement été abandonné et est de nouveau candidat aujourd'hui.

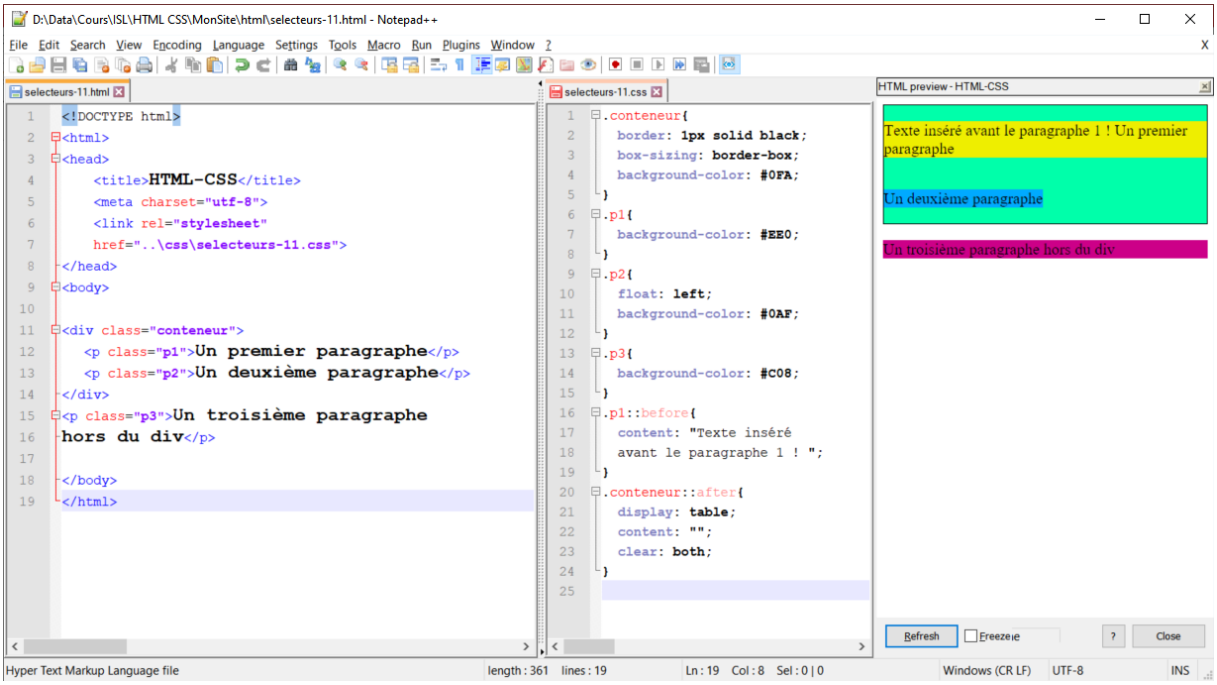
Insérer du contenu avant ou après le contenu d'un élément HTML

Les pseudo-éléments ::before et ::after vont nous permettre respectivement de cibler l'emplacement avant et après le contenu d'un élément HTML.

Dans l'immense majorité des cas nous allons utiliser ces pseudo-éléments de concert avec la propriété CSS content pour ajouter du texte avant ou après le contenu de l'élément.

Le pseudo-élément ::after va également nous permettre de réaliser le clearfix CSS pour qu'un élément parent contienne ses enfants flottants. En effet, en appliquant ::after à un élément contenant des éléments flottants, on va créer un pseudo-élément qui sera le dernier enfant du parent.

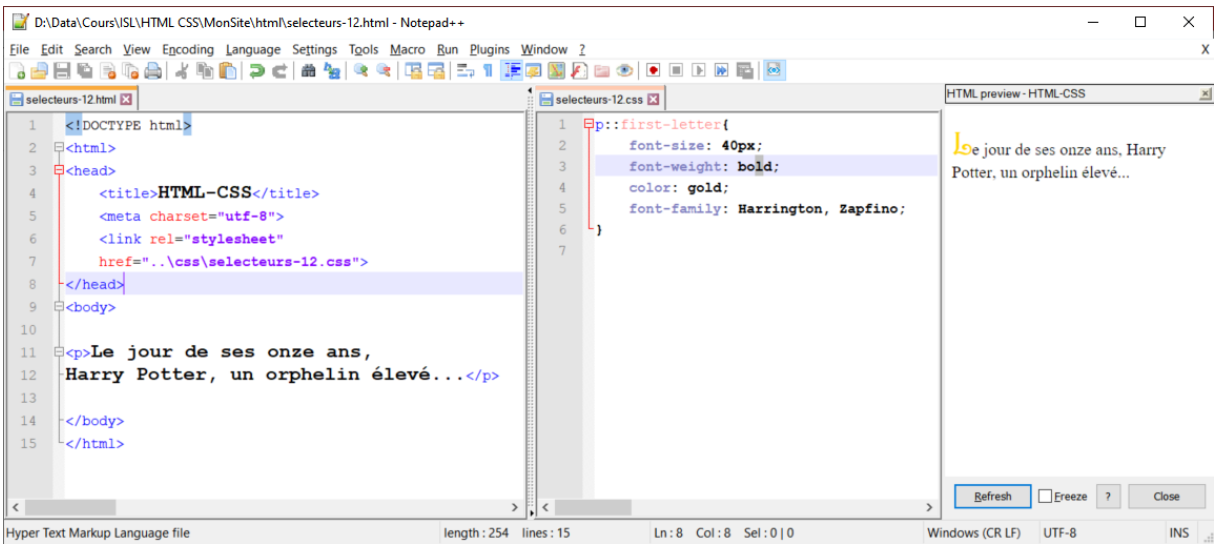
Il suffit ensuite d'appliquer un clear à ce pseudo élément tout en lui attribuant une boite de contenu avec content et en lui définissant un type d'affichage avec display.



Sélectionner la première lettre d'un élément en CSS

Le pseudo-élément ::first-letter va nous permettre de sélectionner uniquement la première lettre du contenu d'éléments HTML.

Nous allons ensuite pouvoir appliquer cette lettre toutes sortes de mise en forme. Par exemple, pour sélectionner la première lettre de tous les paragraphes d'une page, nous utiliserons le sélecteur CSS p::first-letter.

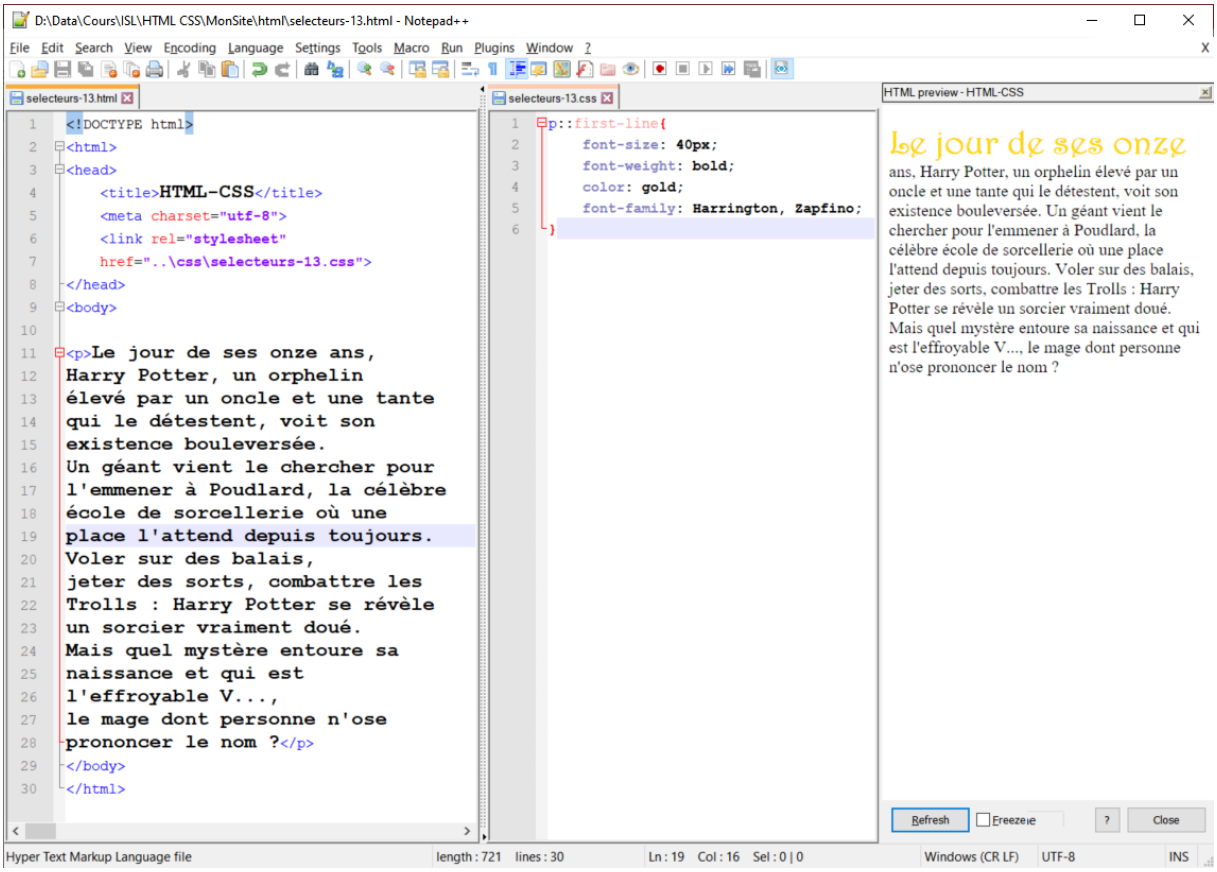


Sélectionner la première ligne d'un élément en CSS

De manière analogue à ::first-letter, nous allons pouvoir sélectionner uniquement la première ligne d'un élément HTML grâce au pseudo-élément ::first-line pour lui appliquer des styles.

Attention, la mise en forme va ici être dépendante de l'écran de chacun de vos visiteurs puisque la « première ligne » correspond ici à la première ligne dans le rendu visuel de la page de vos visiteurs.

Ainsi, pour sélectionner la première ligne de tous les paragraphes de notre page nous utiliserons le sélecteur CSS p::first-line.



Cibler la partie d'un élément déjà sélectionnée par l'utilisateur

Finalement, nous allons pouvoir appliquer des styles particuliers à certaines parties de nos éléments HTML en fonction de ce qui est déjà présélectionné par l'utilisateur grâce au pseudo-élément ::selection.

Les styles n'apparaitront que sur la partie de l'élément sélectionnée par un utilisateur. Pour sélectionner et appliquer des mises en forme particulières à la partie sélectionnée de tous les paragraphes d'une page, nous utiliserons donc le sélecteur CSS p::selection.

Attention toutefois à la compatibilité et à l'utilisation de ce sélecteur : celui-ci devait être intégré à la sortie du CSS3 mais a finalement été abandonné. Il fera normalement parti des recommandations pour le CSS4.

Je vous en parle ici car le fait que ce pseudo-élément devait faire partie des recommandations du CSS3 fait que la plupart des navigateurs le supportent déjà bien.