Le modèle des boîtes 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

Le modèle des boîtes

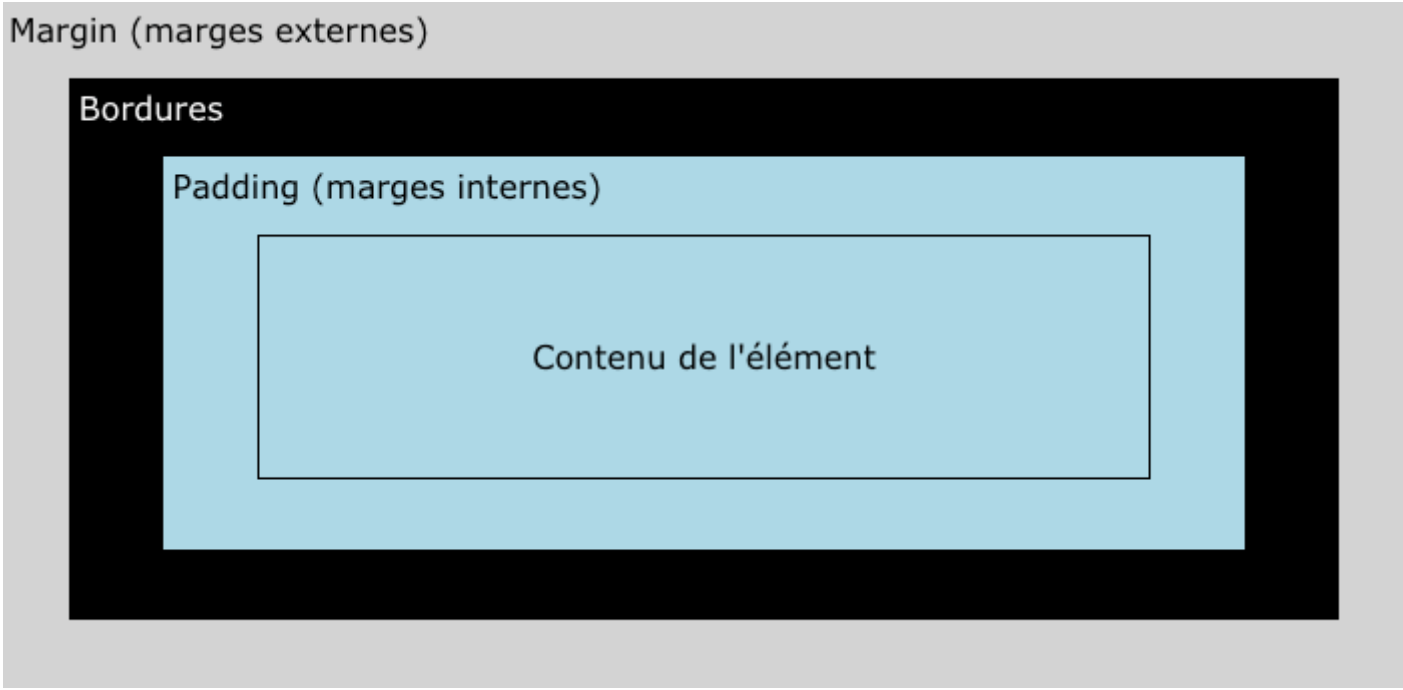
Dans cette nouvelle leçon, nous allons présenter le fameux « modèle des boites » CSS. Comprendre le concept de « boites » va être essentiel construire des designs de pages efficaces puisque cela va nous permettre d'appréhender la façon dont vont être calculées les dimensions de chaque élément.

Le concept de « boites » et le modèle des boites

L'idée centrale du modèle des boites est que tout élément HTML peut être représenté par un empilement de différentes boites rectangulaires :

- La première boite, centrale, va être composée du contenu de l'élément en soi ;
- La deuxième boite va être composée de la première boite ainsi que des marges internes de l'élément ;
- La troisième boîte va être composée de la deuxième boite et des bordures de l'élément ;
- La quatrième boite va être composée de la troisième boite et des marges externes de l'élément.

Voici la représentation d'un élément selon le modèle des boites :



Les propriétés CSS liées aux différentes boites

Le CSS va déjà nous fournir différentes propriétés qui vont nous permettre de spécifier la taille des différents éléments composants les différentes boites :

- Les propriétés width et height vont nous permettre de définir la largeur et la hauteur de la boite « contenu » ;
- La propriété padding va nous permettre de définir la taille des marges internes ;
- La propriété border va nous permettre de définir des bordures pour notre élément ;
- La propriété margin va nous permettre de définir la taille des marges externes.

Nous allons dans cette partie commencer avec l'étude de ces différentes propriétés qui sont fondamentales. Nous parlerons également de la propriété box-sizing qui va nous permettre de changer la façon dont la largeur et la hauteur d'un élément vont être calculées et donc de modifier le modèle des boites par défaut.

Le modèle des boites et le positionnement

Comprendre comment est calculée la taille de chaque élément et de quoi chaque élément est composé est essentiel pour créer des mises en page efficaces.

Pour choisir le type d'affichage et de positionnement des éléments HTML, le CSS va nous fournir des propriétés très puissantes qui vont nous permettre de modifier le flux normal de la page, c'est-à-dire de modifier l'ordre d'affichage des éléments ou la place réservée par défaut à chacun d'entre eux.

Ici, nous allons nous intéresser aux propriétés suivantes :

- La propriété display qui va nous permettre de définir un type d'affichage pour un élément ;
- La propriété position qui va nous permettre de positionner nos éléments de différentes façons dans une page ;
- La propriété float qui va nous permettre de faire « flotter » des éléments HTML dans la page.

Largeur (width) et hauteur (height) de la boite de contenu des éléments HTML

Tout contenu d'un élément HTML va prendre un certain espace dans une page, c'est-à-dire posséder une largeur et une hauteur. Cet espace pris, c'est-à-dire la largeur et la hauteur de ce contenu vont être représentées respectivement par les propriétés CSS width (largeur) et height (hauteur).

Dans cette leçon, nous allons étudier ces deux propriétés et apprendre à les manipuler.

L'impact du type d'affichage d'un élément sur ses dimensions

Pour comprendre comment fonctionnent les propriétés width et height et comment les manipuler il est avant tout nécessaire d'avoir une vue claire sur les types d'affichage principaux des éléments : l'affichage block (sous forme de bloc) et inline (en ligne).

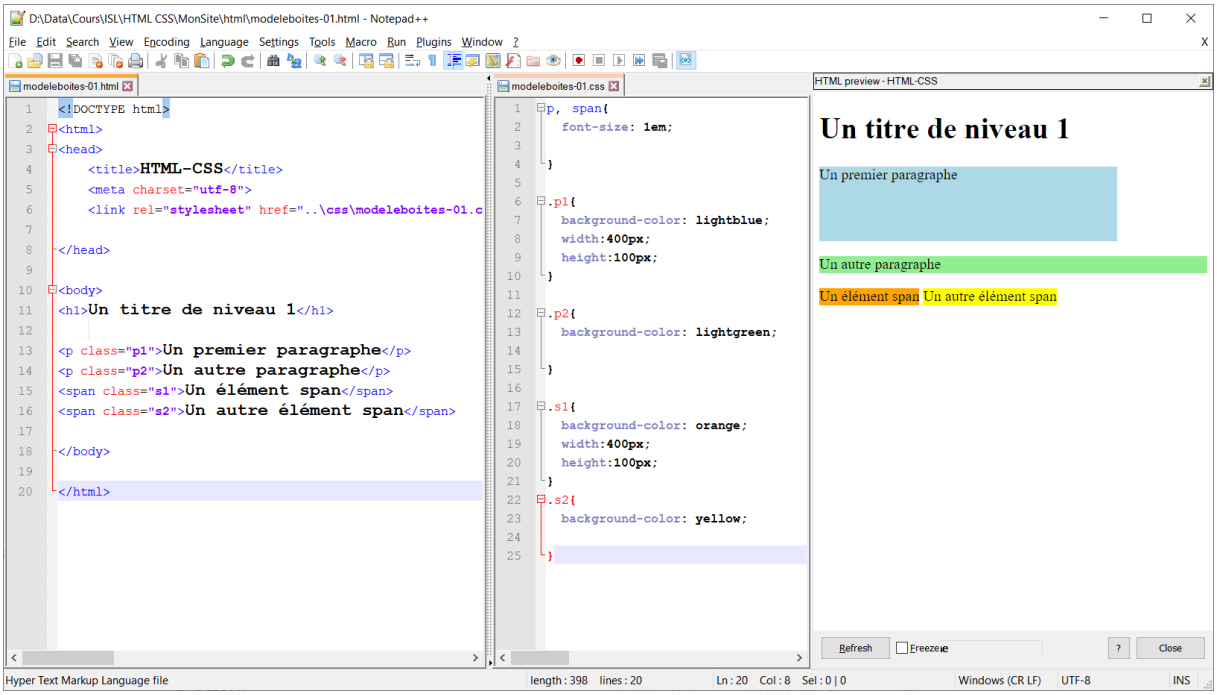
En effet, je vous rappelle que les éléments HTML peuvent être affichées de deux grandes façons différentes : soit sous forme de bloc, soit en ligne.

Les dimensions par défaut du contenu des éléments HTML vont avant tout être déterminées par le type d'affichage des éléments : *en effet, les éléments de type block occuperont par défaut toute la largeur disponible dans leur élément parent tandis que les éléments de type inline n'occuperont que la largeur nécessaire à leur contenu.*

Vous pouvez ainsi déjà retenir ici que nous n'allons pas pouvoir modifier la taille de l'espace pris par le contenu des éléments de type inline avec les propriétés width et height : les valeurs qu'on va pouvoir définir vont tout simplement être ignorées.

En effet, le principe de base d'un élément de type inline est que l'espace pris par sa boite « contenu » soit toujours égal à l'espace strictement nécessaire à l'affichage de ce contenu.

Regardez l'exemple ci-dessous pour vous en convaincre. Je vous rappelle ici qu'un élément p possède un type d'affichage block tandis qu'un élément span possède un type d'affichage inline.



Comme vous pouvez le constater, les valeurs passées à width et à height sont ignorées pour mon span s1.

Notez bien ici que modifier la taille de la boite du contenu d'un élément de type block ne change pas les propriétés fondamentales de celui-ci. J'entends par là qu'un élément de type block commencera toujours sur une nouvelle ligne et ne viendra jamais se positionner à côté d'un autre élément de type block quelle que soit sa taille.

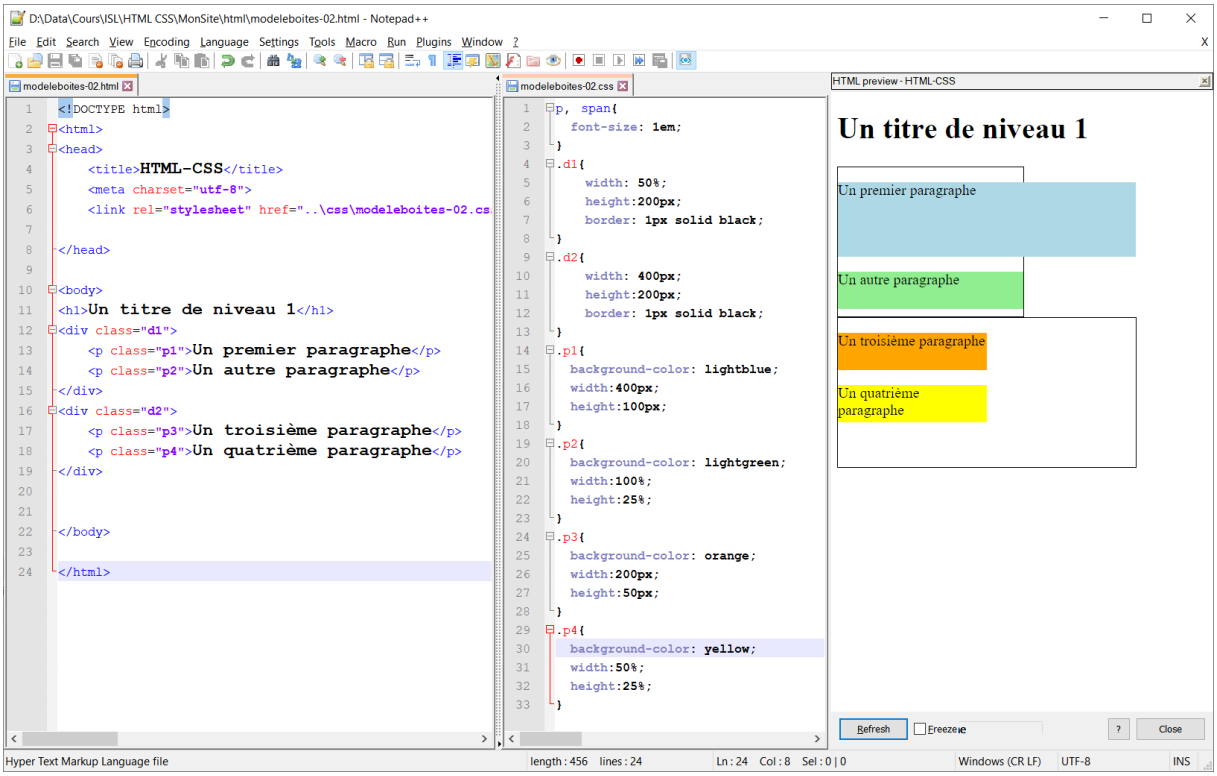
Les types de valeur des propriétés width et height et les problèmes de dépassement

Les propriétés width et height vont pouvoir accepter plusieurs types de valeurs :

- Des valeurs de type « longueur » qui vont être généralement exprimées en px ou en em ;
- Des valeurs en pourcentage, auquel cas le pourcentage donné sera relatif à la dimension de l'élément parent.

Problème de dépassement n°1 : l'élément dépasse de son parent

Pour bien identifier ce premier problème de dépassement, prenons immédiatement un exemple qui va également nous permettre de comprendre pourquoi il est généralement déconseillé de mélanger des unités fixes et relatives lorsqu'on définit des dimensions.

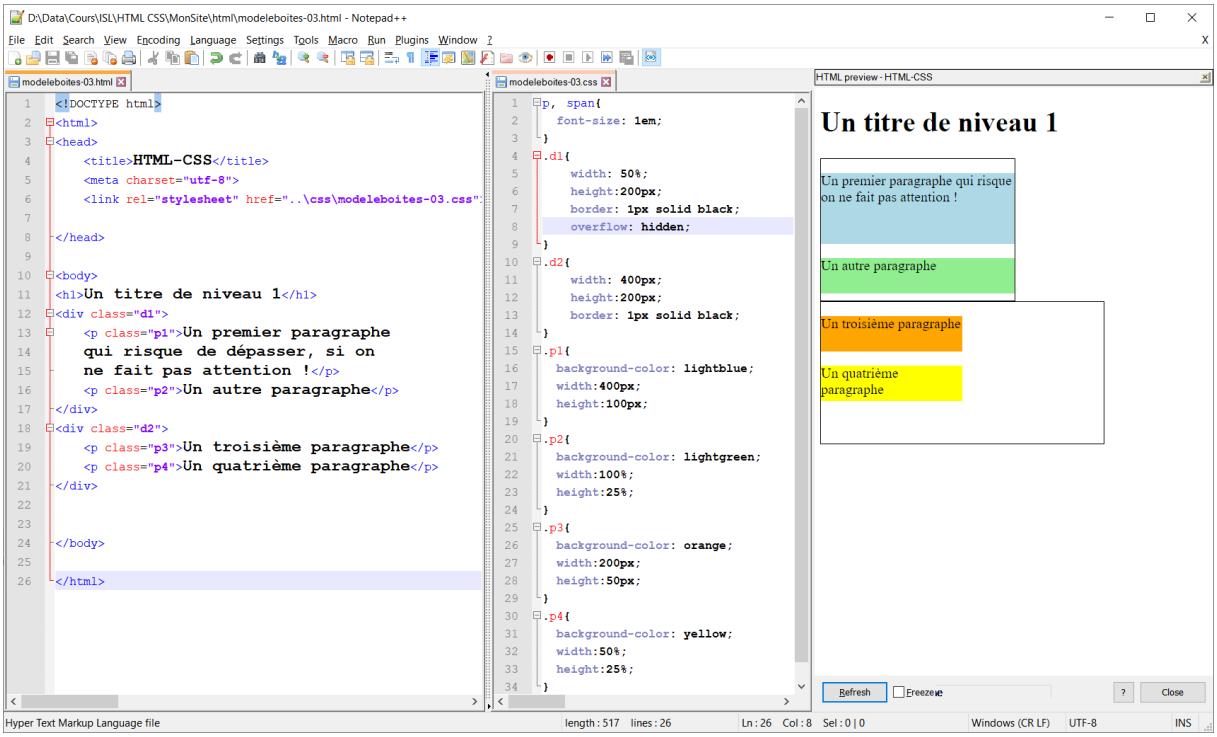


Dans cet exemple, la largeur de notre premier conteneur div class="d1" est définie de manière relative par rapport à son parent (qui est le body). La largeur du div va donc changer en fonction de la largeur de la fenêtre.

Ensuite, dans ce div conteneur, on spécifie une largeur pour notre premier paragraphe en unités absolues et fixes. Le problème ici va être que pour des fenêtres trop petites notre élément conteneur div va être plus petit que le paragraphe qu'il contient et cela va poser de nombreux problèmes de design.

Pour éviter ce genre de problèmes, la première règle est de bien faire attention lorsqu'on définit des dimensions aux différents types de valeurs utilisés et aux interactions entre ces différentes valeurs.

Ensuite, afin d'être certain que le design général de notre page ne sera pas impacté, on peut également utiliser la propriété overflow et en particulier sa valeur hidden sur l'élément parent. Cela va avoir pour effet de tronquer tout le contenu qui dépasse de l'élément. Si l'on souhaite que le contenu reste accessible, on peut utiliser overflow : scroll qui va proposer une barre de défilement dans l'élément parent.

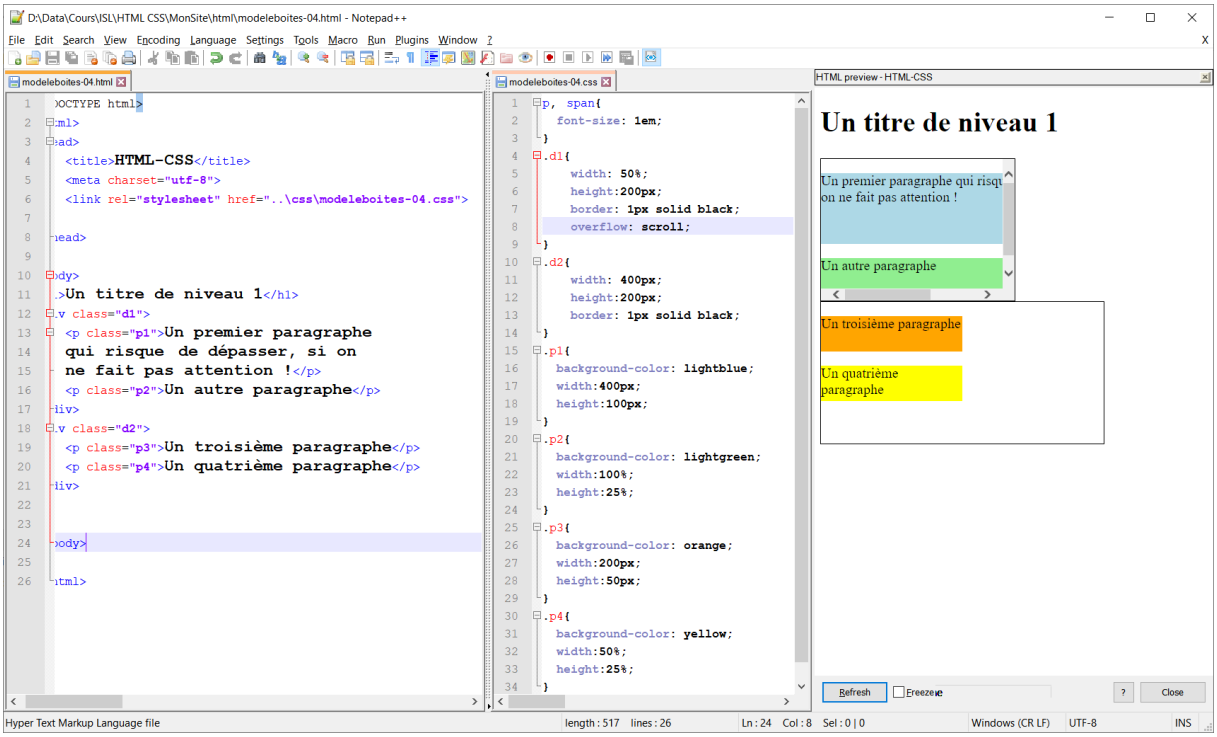


Problème de dépassement n°2 : le contenu dépasse de l'élément

Un autre problème de dépassement courant va être le problème « contraire » au précédent, à savoir le cas où le contenu de mon élément dépasse de sa boite.

Ce problème va survenir si on définit une taille trop petite pour la boite de contenu de l'élément par rapport au contenu de l'élément.

Dans ce cas-là, il va falloir utiliser la propriété overflow :scroll ; sur l'élément en soi afin que le contenu qui dépasse soit caché.

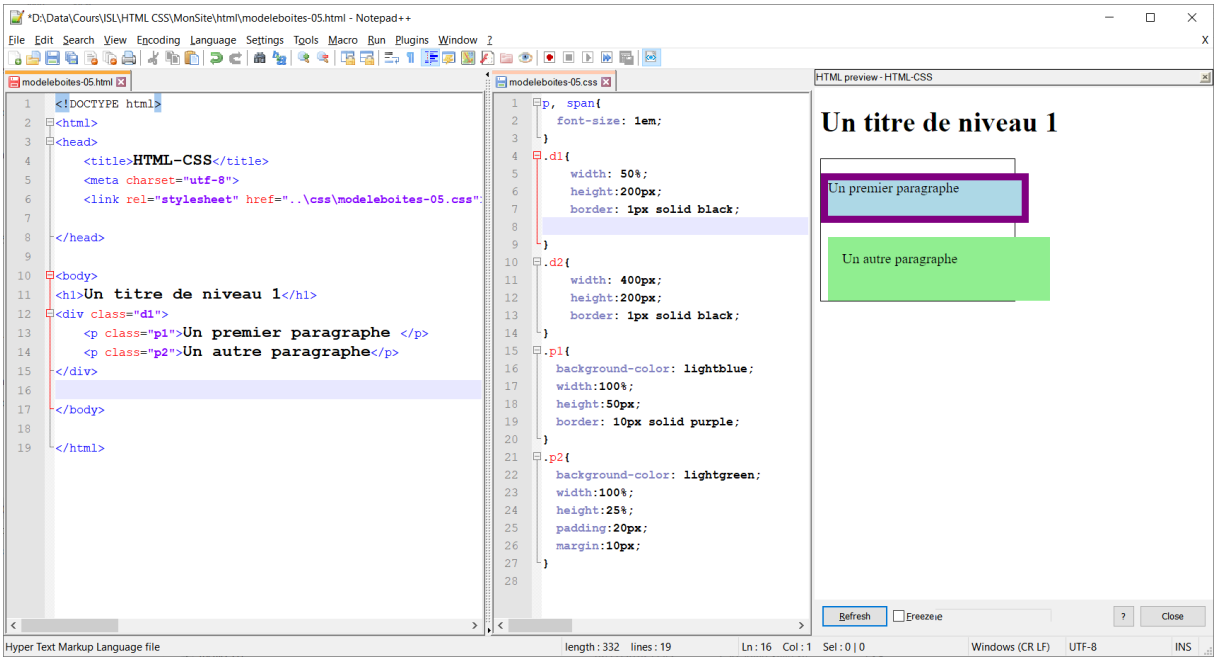


Problème de dépassement n°3 : l'élément dépasse globalement de son parent

Une autre erreur couramment faite se situe dans la mauvaise compréhension du modèle des boites. En effet, il faut bien ici se rappeler que les propriétés width et height ne servent qu'à définir les dimensions de la boite liée au contenu d'un élément.

A ces dimensions vont venir s'ajouter les tailles des marges intérieures, des bordures et celles des marges extérieures pour former la taille totale de l'élément. Ainsi, si l'on ne fait pas attention, on peut très rapidement se retrouver avec des éléments enfants plus grands et qui vont dépasser de leur parent.

Imaginons par exemple qu'on fixe la largeur d'un élément enfant à 100% de celle de son parent. Si on attribue ensuite une quelconque marge interne, bordure, ou marge externe à l'élément, celui-ci va dépasser de son élément parent.




```html
<!DOCTYPE html>
<html>
<head>
    <title>HTML-CSS</title>
    <meta charset="utf-8">
    <link rel="stylesheet" href="..\css\modeleboites-05.css">

</head>

<body>
<h1>Un titre de niveau 1</h1>
<div class="d1">
    <p class="p1">Un premier paragraphe </p>
    <p class="p2">Un autre paragraphe</p>
</div>

</body>

</html>
```

```css
p, span{
  font-size: 1em;
}
.d1{
    width: 50%;
    height:200px;
    border: 1px solid black;

}
.d2{
    width: 400px;
    height:200px;
    border: 1px solid black;
}
.p1{
  background-color: lightblue;
  width:100%;
  height:50px;
  border: 10px solid purple;
}
.p2{
  background-color: lightgreen;
  width:100%;
  height:25%;
  padding:20px;
  margin:10px;
}
```

Gestion des marges internes ou padding en CSS

En CSS, nous allons devoir distinguer deux types de marges : les marges intérieures (“padding”) et les marges extérieures (“margin”).

Les marges intérieures se trouvent entre le contenu de l’élément et sa bordure. Ainsi, définir une marge intérieure importante va éloigner la bordure de l’élément de son contenu. Si on définit une couleur de fond pour notre élément, celle-ci s’applique également dans l’espace correspondant aux marges intérieures.

La propriété CSS padding

Nous allons pouvoir ajouter des marges internes à un élément grâce à la propriété CSS padding. Notez déjà que la propriété padding est la version raccourcie des propriétés padding-top, padding-left, padding-bottom et padding-right qui vont servir à définir les marges internes de chaque côté d’un élément.

Ces propriétés vont pouvoir accepter deux types de valeurs :

- Des valeurs de type longueur, généralement en px ou en em ;
- Des valeurs de type pourcentage. Dans ce cas, le % indiqué est calculé par rapport à la taille de l’élément parent.

Notez qu’il n’est pas possible de passer des valeurs de padding négatives.

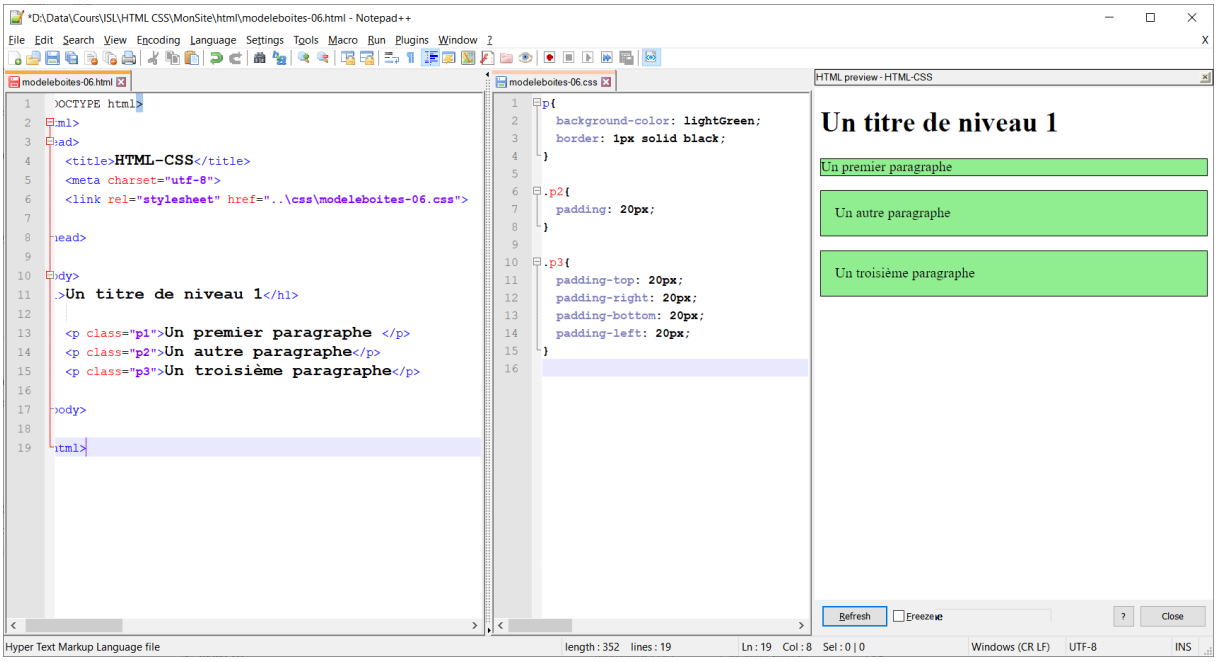
Ajouter une même marge interne de chaque côté d’un élément

Pour définir une même marge interne de chaque côté d’un élément, il suffit de passer la valeur de la marge intérieure que l’on souhaite appliquer à l’élément à la propriété padding en CSS.

Par exemple, pour appliquer une marge intérieure de 25px de chaque côté d’un élément, on écrira en CSS padding: 25px. Pour que la marge intérieure soit égale à 10% de la taille de l’élément parent de notre élément, on écrira padding: 10% tout simplement.

Pour bien constater l’effet de la propriété padding, je vous conseille d’ajouter une couleur de fond ou une bordure aux éléments pour les exemples suivants.

Notez qu’on va tout aussi bien pouvoir utiliser les propriétés padding-top, padding-left, padding-bottom et padding-right et leur passer la même valeur pour arriver au même résultat. C’est juste plus long à écrire !



Définir des marges internes différentes de chaque côté d'un élément

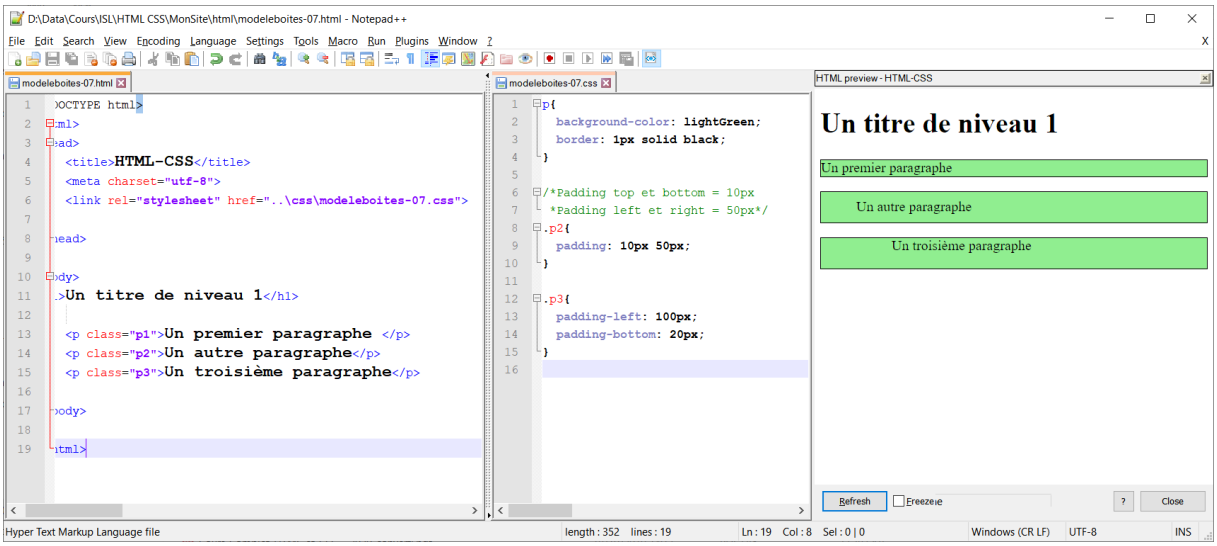
Nous allons également pouvoir appliquer des marges intérieures de taille différentes de chaque côté d'un élément.

Pour cela, nous avons deux façons de faire : soit en passant plusieurs valeurs à la propriété padding, soit en utilisant les propriétés padding-top, padding-left, padding-bottom et padding-right.

On va en effet pouvoir passer entre 1 et 4 valeurs à la propriété raccourcie padding :

- En passant une valeur à padding, la valeur passée définira le comportement des 4 marges intérieures de l'élément ;
- En passant deux valeurs à padding, la première valeur passée définira le comportement des marges intérieures supérieure et inférieure de l'élément tandis que la seconde valeur définira le comportement des marges intérieures gauche et droite de l'élément ;
- En passant trois valeurs à padding, la première valeur passée définira le comportement de la marge interne supérieure, la deuxième définira le comportement des marges intérieures gauche et droite tandis que la troisième définira le comportement de la marge interne basse ;
- En passant quatre valeurs à padding, la première valeur passée définira le comportement de la marge interne supérieure, la deuxième définira le comportement de la marge interne droite, la troisième celui de la marge interne basse et la quatrième celui de la marge interne gauche.

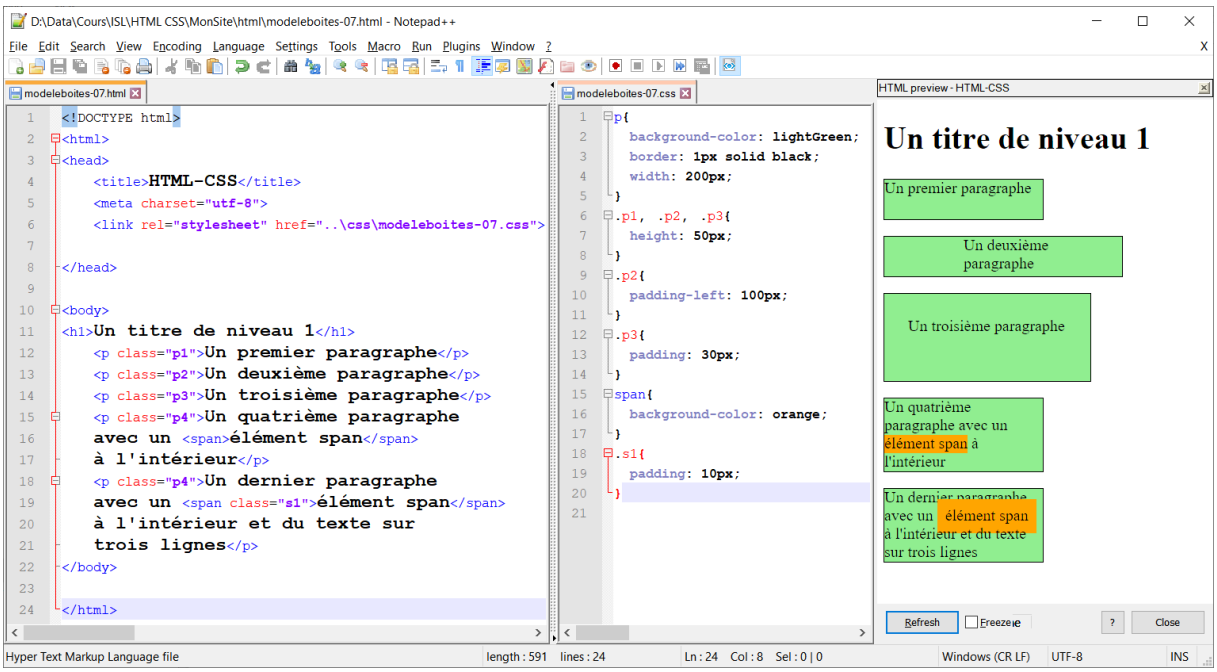
Si on choisit d'utiliser les propriétés padding-top, padding-left, padding-bottom et padding-right, il suffit de suivre le nom pour comprendre à quelle marge interne est liée chaque propriété (top = haute, left = gauche, bottom = basse, right = droite).



L'effet du padding sur les éléments environnants

L'ajout de marges internes ou padding va augmenter la taille totale de l'élément. En effet, la valeur donnée au padding va venir s'ajouter à celles des propriétés width et height par défaut.

Regardez plutôt le code ci-dessous pour vous en convaincre :



Ici, on voit clairement que le padding a un impact sur les dimensions totales des éléments, et ceci qu'ils soient de type block ou inline.

Cependant, l'impact sur les éléments environnants ne va pas être le même selon le type d'élément auquel on applique un padding et selon le padding appliqué.

Ici, il y a notamment un cas à noter : celui des marges internes haute et basse d'un élément de type inline. Comme vous pouvez le voir ci-dessus, les marges internes haute et basse sont bien appliquées à mon élément span s1 (on le voit grâce à la couleur en fond de mon élément).

Cependant, le navigateur ne va pas en tenir compte pour l'affichage et le positionnement des autres éléments. C'est la raison pour laquelle notre élément span chevauche le texte des lignes précédente et suivante du paragraphe.

Notez par ailleurs ici les différentes « couches » de texte : la ligne de texte précédent celle du span va se trouver en dessous de lui tandis que la suivante va être au-dessus (on voit que la couleur de fond du span cache le texte précédent mais est sous le texte de la ligne suivante).

Encore une fois, notez que le padding est bien appliqué de chaque côté pour chaque type d'éléments, cependant les padding haut et bas n'auront pas d'impact sur le positionnement d'un élément inline ni sur celui des éléments autour de lui.

Gestion des bordures avec la propriété CSS border

Nous allons pouvoir définir des bordures de largeurs, de styles ou de couleurs différents autour de nos éléments HTML en CSS.

L'espace pris par la bordure va se trouver entre la marge intérieure et la marge extérieure d'un élément HTML.

Nous pouvons définir les bordures d'un élément de différentes manières en CSS : soit en utilisant les trois propriétés border-width, border-style et border-color, soit un utilisant directement la notation raccourcie border.

Nous allons également pouvoir créer des bordures arrondies à l'aide de la propriété border-radius que nous étudierons plus tard dans ce cours car elle est relativement complexe à comprendre et à maitriser.

Les caractéristiques des bordures en CSS

Les bordures vont être définies par trois caractéristiques en CSS : une épaisseur (ou largeur), un style et une couleur.

Nous allons pouvoir définir ces différentes caractéristiques d'un coup au sein de la propriété raccourcie border ou les définir une à une avec chacune des propriétés de type border-. Plus précisément :

- La propriété border-width va nous permettre de définir la largeur (ou « l'épaisseur) d'une bordure ;
- La propriété border-style va nous permettre de définir le style d'une bordure ;
- La propriété border-color va nous permettre de définir la couleur d'une bordure.

Définir la largeur ou l'épaisseur d'une bordure

Nous allons pouvoir utiliser différents types de valeurs pour définir la largeur d'une bordure :

- Une valeur de type « mot clef » à choisir parmi thin (bordure fine), medium (bordure moyenne) et thick (bordure épaisse) ;
- Une valeur de type « longueur » en px ou en em par exemple.

Généralement, nous utiliserons des unités en px pour définir la largeur de nos bordures.

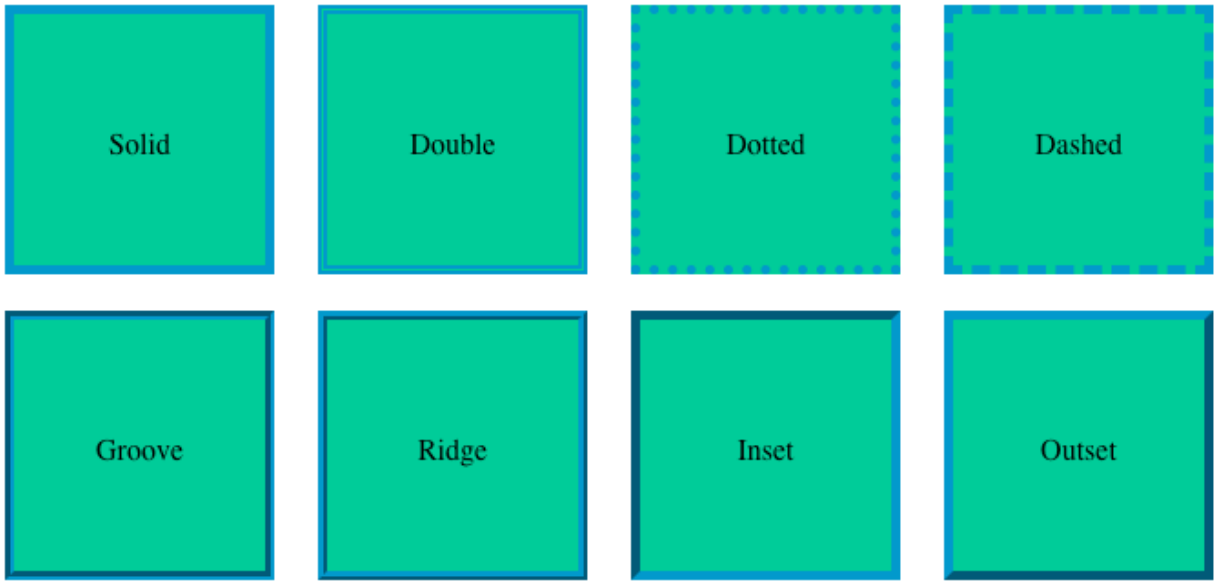
Choisir le style d'une bordure

Le « style » d'une bordure correspond à son aspect : une bordure peut prendre la forme d'un simple trait, d'un trait double, ou être constituée de pointillés, avoir un effet 3D, etc.

Pour définir le style d'une bordure, nous allons devoir choisir parmi les mots clefs suivants :

Valeur	**Description**
solid	Bordure solide simple (un trait)
double	Bordure solide double
dotted	Bordure en pointillés
dashed	Bordure constituée de tirets
groove	Bordure incrustée avec effet 3D. L'effet produit est l'inverse de ridge
ridge	Bordure en relief avec effet 3D. L'effet produit est l'inverse de groove
inset	La bordure donne l'effet que la boite représentant l'élément est enfoncée dans la page. L'effet produit est l'inverse de outset
outset	La bordure donne l'effet que la boite représentant l'élément est en relief par rapport au reste de la page. L'effet produit est l'inverse de inset

Chaque mot clef va créer un type de bordure différent. Voici à quoi correspond chaque effet visuellement :



Définir la couleur d'une bordure

Finalement, nous allons devoir définir la couleur d'une bordure. Pour cela, nous allons pouvoir piocher parmi toutes les valeurs de type « couleur » connues et notamment :

- Les notations de type « nom de couleur » ;
- Les notations hexadécimales ;
- Les notations RGB() et RGBa() ;
- Les notations HSL() et HSLa().

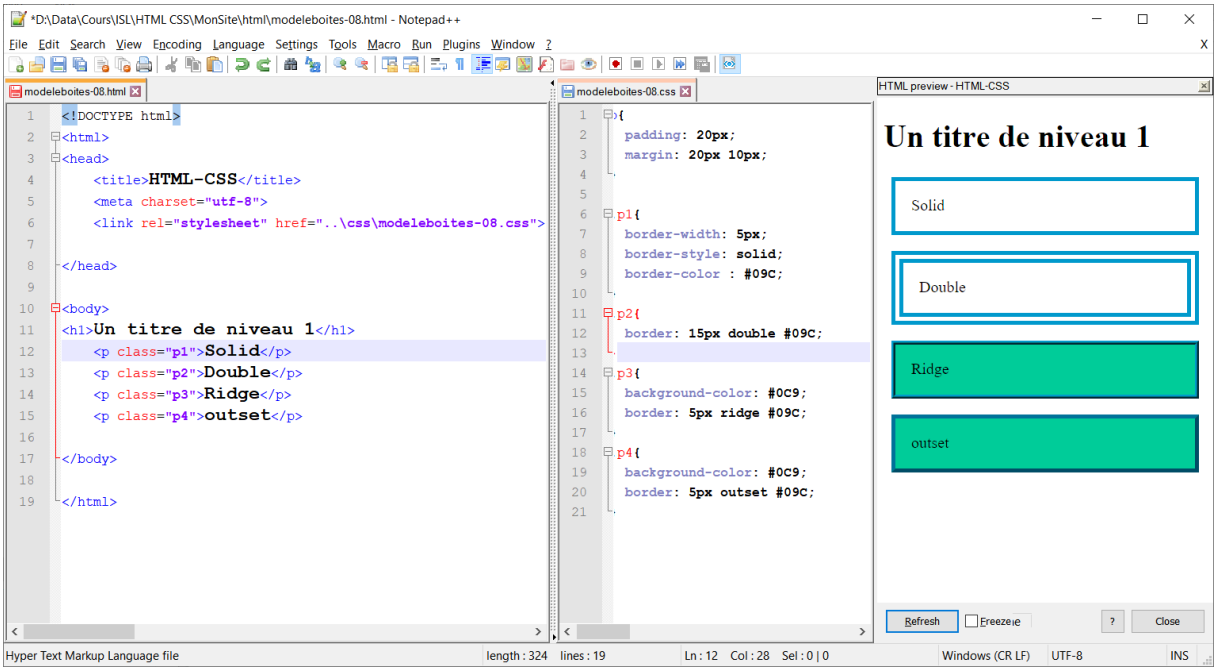
Exemples de création de bordures en CSS

La suite de cette leçon va être pour nous l'occasion de s'exercer et de créer toutes sortes de bordures. Je vous rappelle ici que nous aborderons les bordures arrondies dans la prochaine leçon.

Commençons déjà avec des exemples simples de bordures créées en CSS en appliquant ce que nous avons vu ci-dessous.

Ici, nous allons créer 4 types de bordures différentes : une bordure simple, une bordure double, une avec effet 3D de type « ridge » et une avec un style « outset ».

Nous allons pouvoir faire cela en utilisant soit les propriétés border-width, border-style et border-color, soit la notation raccourcie border.



Ici, nous nous contentons d'ajouter des bordures différentes autour de nos paragraphes. J'ai également rajouté une couleur de fond aux deux derniers paragraphes afin que l'on puisse bien observer l'effet de 3D des deux dernières bordures.

Notez une chose intéressante par rapport à notre bordure double : la taille de la bordure correspond à la taille totale de la bordure, c'est-à-dire dans ce cas du double trait et de l'espace entre ces deux traits. La largeur totale va être répartie régulièrement : en indiquant une bordure double de 15px, chaque élément de la bordure (les deux traits plus l'espace entre les traits) va faire 5px de large (15px / 3).

Définir des bordures différentes pour chaque côté d'un élément

Le CSS va également nous permettre de définir chacune des quatre bordures de nos éléments indépendamment les unes des autres afin de pouvoir appliquer des effets intéressants.

Nous avons deux façons de faire cela selon que nous utilisions la notation raccourcie border pour définir nos bordures ou que nous utilisions chacune des propriétés border-.

Dans le cas où nous souhaitons utiliser border, nous allons devoir utiliser 4 sous propriétés CSS qui sont :

- border-top pour définir l'aspect (taille, style et couleur) de la bordure supérieure de l'élément ;
- border-right pour définir l'aspect (taille, style et couleur) de la bordure droite de l'élément ;
- border-bottom pour définir l'aspect (taille, style et couleur) de la bordure inférieure de l'élément ;

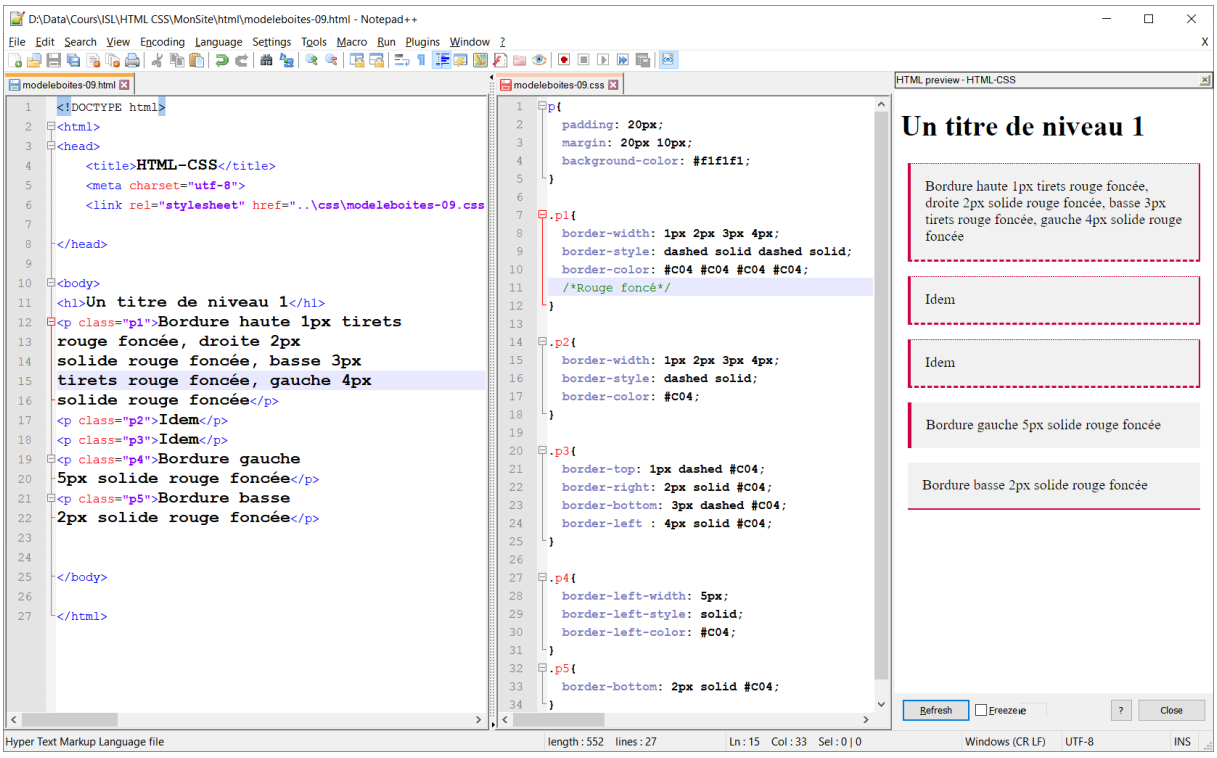
- border-left pour définir l'aspect (taille, style et couleur) de la bordure gauche de l'élément.

Bon à savoir : Les propriétés border-top, border-right, border-bottom et border-left sont à nouveau des écritures raccourcies des propriétés CSS border-top-width, border-top-style, border-top-color, etc.

Dans le cas où nous utilisons les propriétés border-width, border-style et border-color alors nous pourrons indiquer 4 valeurs à la suite qui définiront le comportement des bordures supérieure, droite, inférieure et gauche dans cet ordre.

Notez ici qu'il est tout à fait possible de ne renseigner qu'une valeur pour l'une de ces 3 propriétés si vous voulez que vos 4 bordures aient le même comportement. Vous pouvez également ne mentionner que deux valeurs : dans ce cas, la première valeur définira le comportement des bordures supérieure et inférieure tandis que la seconde définira le comportement des bordures droite et gauche de l'élément.

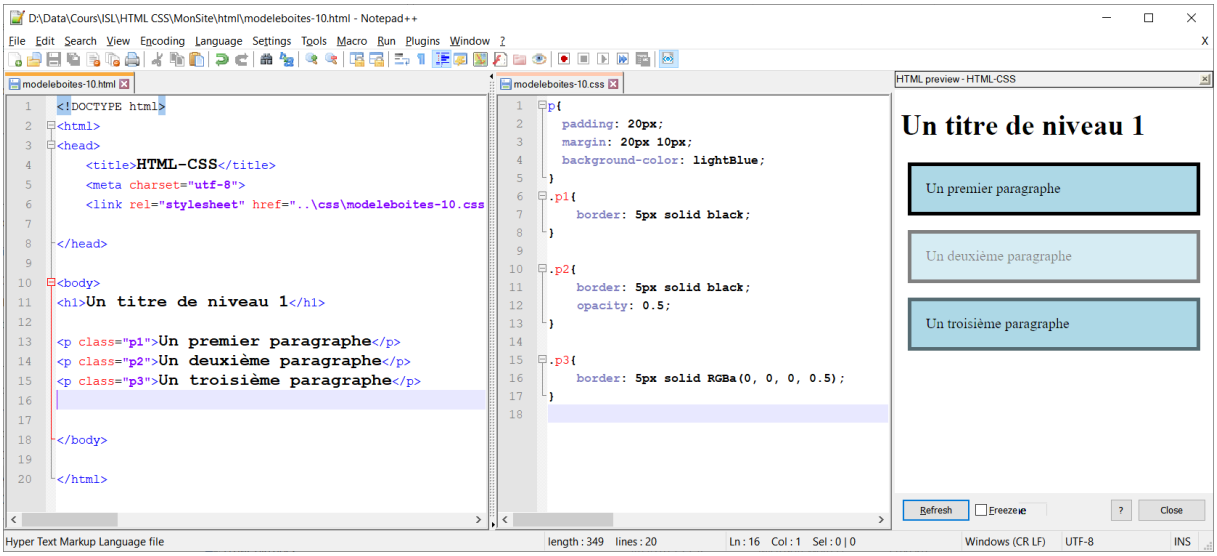
Regardez plutôt les exemples ci-dessous qui résument toutes les situations possibles pour bien comprendre :



Créer des bordures semi transparentes

Nous allons encore pouvoir ajouter un effet de transparence à nos bordures en utilisant tout simplement une notation RGBa lorsque l'on précisera la paramètre couleur de notre bordure.

Notez ici qu'utiliser la propriété CSS opacity ne produirait pas le comportement voulu puisque l'effet de transparence serait appliqué à l'élément entier et non pas seulement à la bordure comme on le souhaite.



Les bordures dans le modèle des boites

Les bordures d'un élément HTML se situent entre les marges internes et externes de l'élément.

La taille des bordures va par défaut venir s'ajouter aux dimensions de l'élément définies avec les propriétés width et height ainsi qu'aux marges internes.

Notez que nous allons pouvoir définir des bordures de la même façon pour des éléments de type block et inline et que ces bordures auront exactement le même comportement.

Gestion des marges externes (margin) en CSS

En CSS, nous allons devoir distinguer deux types de marges : les marges intérieures (“padding”) et les marges extérieures (“margin”).

Les marges intérieures se trouvent entre le contenu de l’élément et sa bordure. Les marges extérieures, au contraire, vont définir la taille de l’espace autour d’un élément.

Les marges extérieures se trouvent en dehors des bordures d’un élément et servent généralement à éloigner un élément d’un autre. Lorsqu’on définit une couleur de fond pour un élément, cette couleur de fond ne va pas s’appliquer dans l’espace des marges extérieures car encore une fois celles-ci se trouvent « en dehors » de l’élément.

La propriété CSS margin

Nous allons pouvoir ajouter des marges externes à un élément grâce à la propriété CSS margin. Cette propriété est en fait la notation raccourcie des propriétés margin-top, margin-left, margin-bottom et margin-right qui vont servir à définir les marges externes de chaque côté d’un élément.

Ces propriétés vont pouvoir accepter différents types de valeurs :

- Des valeurs de type longueur, généralement en px ou en em ;
- Des valeurs de type pourcentage. Dans ce cas, le % indiqué est calculé par rapport à la taille de l’élément parent ;
- Le mot clef auto. Ce mot clef va surtout nous servir pour centrer des blocs.

Notez qu’à la différence de la propriété padding, nous allons pouvoir passer des valeurs négatives à margin pour créer des marges externes négatives même si en pratique cela est généralement déconseillé pour les problèmes d’ergonomie et de consistance du design que ça peut causer au niveau de la page.

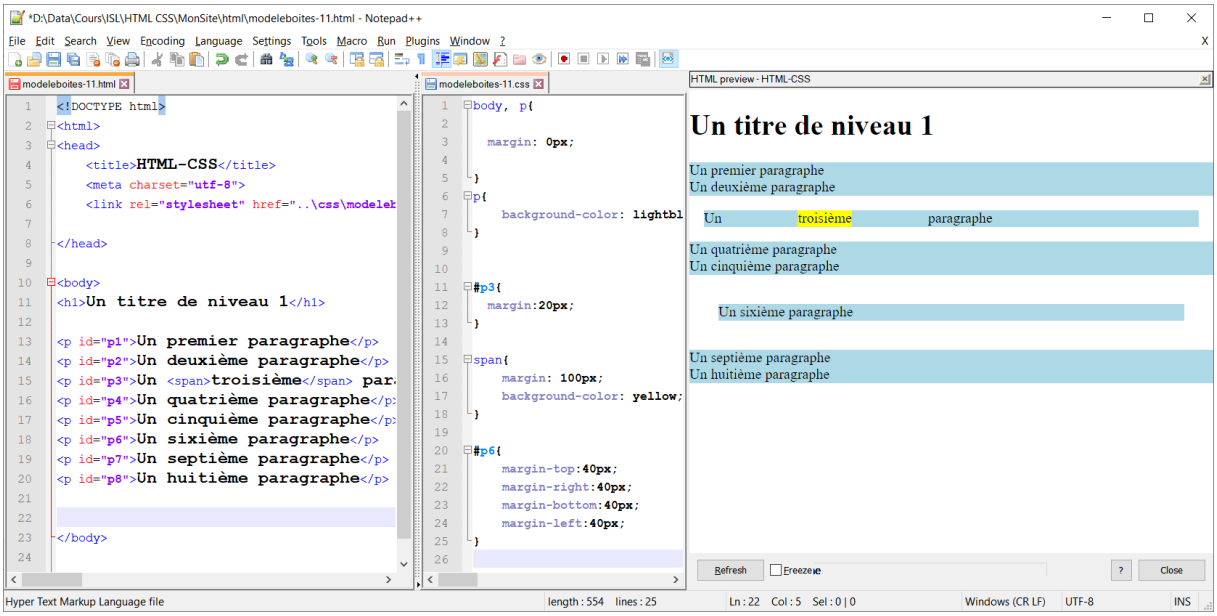
Notez également que la plupart des navigateurs définissent des marges par défaut pour de nombreux éléments. Il faudra donc souvent effectuer un reset des marges dans nos feuilles de styles (en définissant des marges nulles pour les différents éléments) pour être certain d’obtenir le résultat attendu.

Ajouter une même marge externe de chaque côté d’un élément

Pour définir une même marge externe de chaque côté d’un élément, il suffit de passer la valeur de la marge extérieure que l’on souhaite appliquer à l’élément à la propriété margin en CSS.

Par exemple, pour appliquer une marge extérieure de 25px de chaque côté d'un élément, on écrira en CSS margin : 25px. Pour que la marge extérieure représente 10% de la taille de l'élément parent de notre élément, on écrira margin : 10% tout simplement.

Nous allons bien évidemment également pouvoir définir des marges externes égales pour chaque côté d'un élément en utilisant les propriétés margin-top, margin-left, margin-bottom et margin-right et en leur passant la même valeur à chacune.



Ici, vous pouvez noter que les bordures d'un élément définissent la « limite » de cet élément. Les marges externes sont un espace entre les éléments et c'est la raison pour laquelle nous ne pouvons pas appliquer de couleur de fond (entre autres) dans l'espace des marges externes de l'élément.

Notez également qu'on ne va pas pouvoir appliquer de marges haute ou basse aux éléments de type inline comme c'était déjà le cas pour les marges internes. En revanche, les marges externes droite et gauche s'appliquent normalement.

Définir des marges externes différentes de chaque côté d'un élément

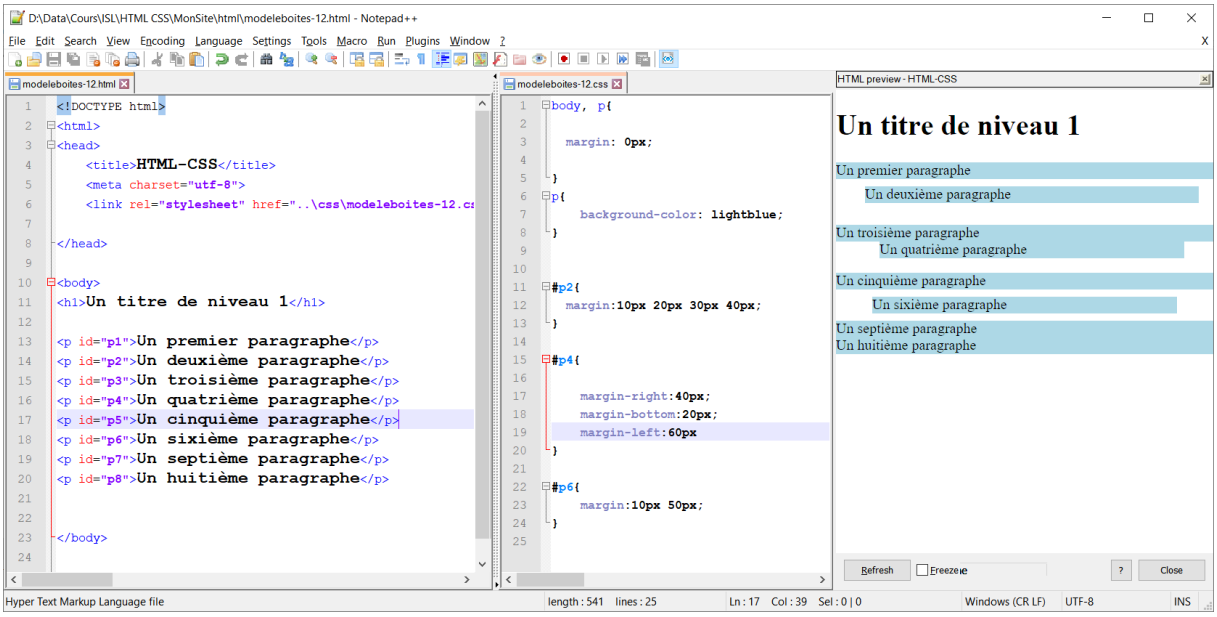
Nous allons également pouvoir appliquer des marges extérieures de taille différentes de chaque côté d'un élément.

Pour cela, nous avons deux façons de faire : soit en passant plusieurs valeurs à la propriété margin, soit en utilisant les propriétés margin-top, margin-left, margin-bottom et margin-right.

On va en effet pouvoir passer entre 1 et 4 valeurs à la propriété raccourcie margin :

- En passant une valeur à margin, la valeur passée définira le comportement des 4 marges extérieures de l'élément ;

- En passant deux valeurs à margin, la première valeur passée définira le comportement des marges extérieures supérieure et inférieure de l'élément tandis que la seconde valeur définira le comportement des marges extérieures gauche et droite de l'élément ;
- En passant trois valeurs à margin, la première valeur passée définira le comportement de la marge externe supérieure, la deuxième définira le comportement des marges extérieures gauche et droite tandis que la troisième définira le comportement de la marge externe basse;
- En passant valeurs à margin, la première valeur passée définira le comportement de la marge externe supérieure, la deuxième définira le comportement de la marge externe droite, la troisième celui de la marge externe basse et la quatrième celui de la marge externe gauche.



La fusion ou « collapsing » des marges externes verticales

Une des grandes différences entre l'implémentation des marges internes et des marges externes est que les marges externes haute et basse de deux éléments vont pouvoir « fusionner » selon certaines conditions.

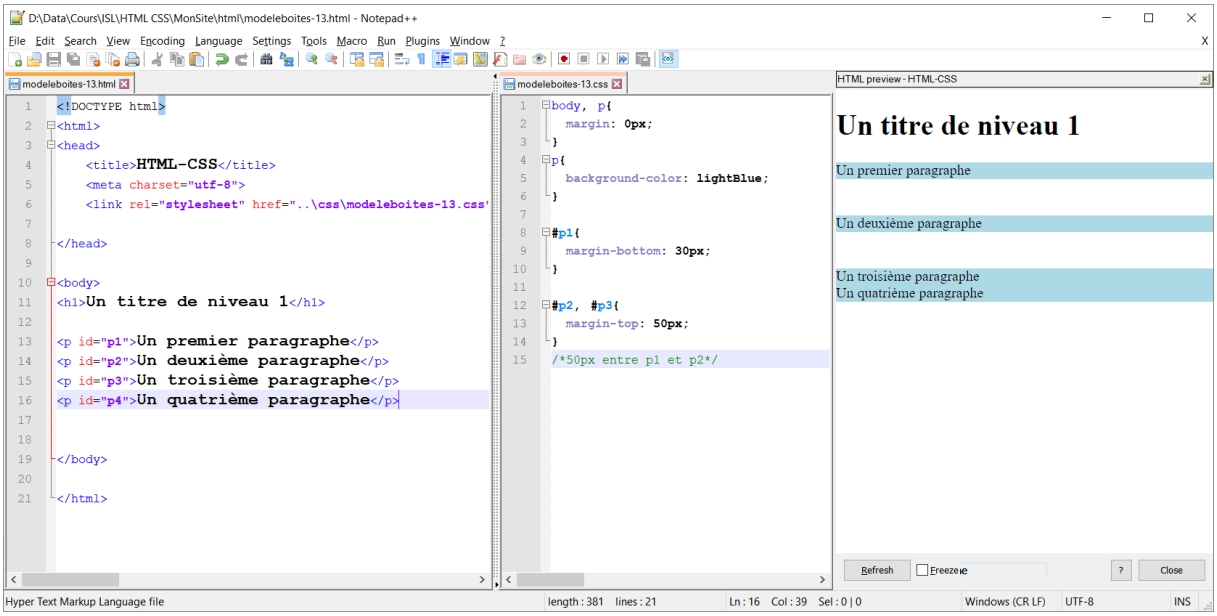
La fusion des marges externes d'éléments consécutifs

Si on définit une marge basse à un premier élément de type block et une marge haute à l'élément suivant, alors les hauteurs des deux marges ne vont pas s'additionner mais vont « fusionner ». En effet, seule la marge la plus importante sera appliquée.

Par exemple, imaginons qu'on ait défini une margin-bottom : 30px pour notre premier élément et une margin-top : 50px pour l'élément suivant. Les deux éléments ne seront pas séparés de 30 + 50 = 80px mais seulement de la taille de la marge la plus importante parmi les deux, à savoir ici 50px.

De même, si deux éléments consécutifs possèdent deux marges négatives, alors seule la marge négative la plus importante sera conservée.

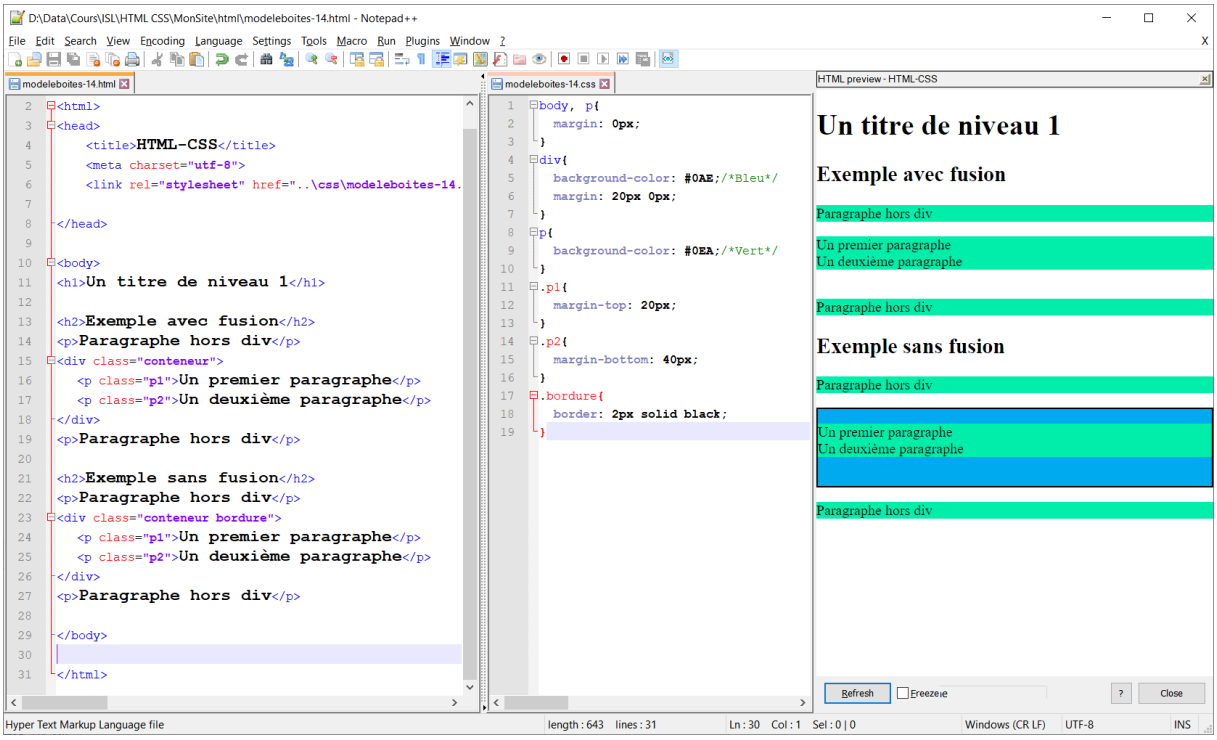
Dans le cas où un élément possède une marge basse externe positive et le suivant possède une marge haute externe négative (ou le contraire), alors la marge négative va se soustraire à la marge positive. Par exemple, si le premier élément possède une margin-bottom : 50px et le deuxième élément a une margin-top : -30px, alors la marge appliquée sera de 50 – 30 = 20px.



La fusion des marges entre un élément parent et son premier et dernier enfant

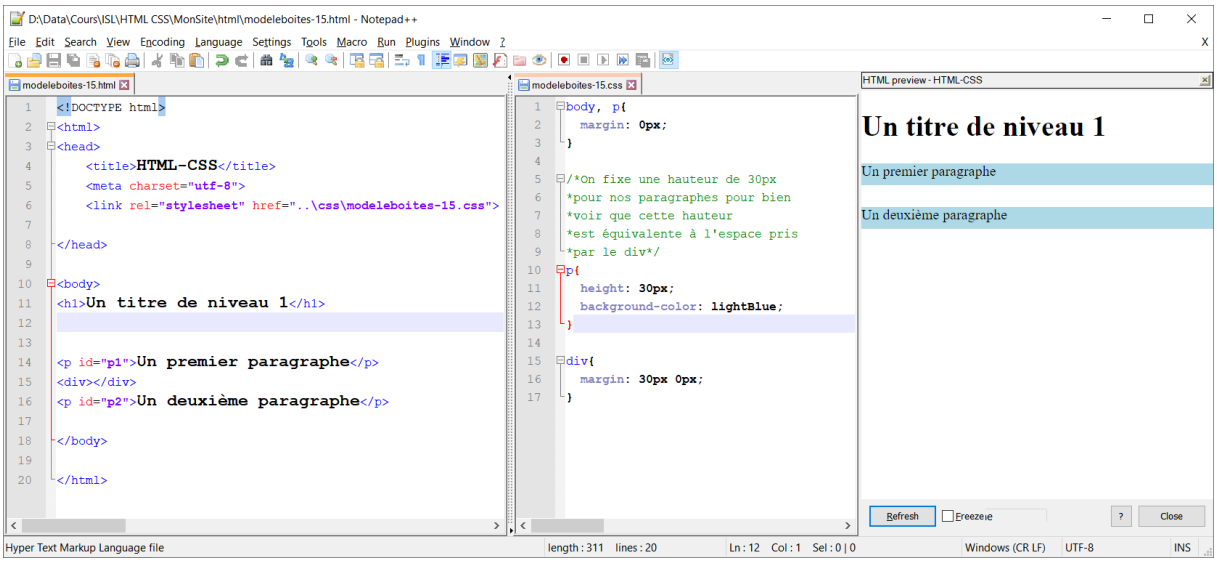
Cette deuxième situation de fusion des marges est plus complexe que la première. Pour faire très simple, vous pouvez retenir qu'il va y avoir fusion entre un élément parent et son premier enfant s'il n'y a aucune bordure ou remplissage entre sa marge haute et la marge haute de son enfant.

De même, il y aura fusion entre la marge basse du parent et la marge basse de son dernier enfant s'il n'y a aucune bordure ou remplissage pour les séparer et si aucune propriété height, min-height ou max-height n'est définie bien évidemment.



Fusion des marges d'un bloc vide

Finalement, si un élément de type block est vide, c'est-à-dire ne possède aucun contenu, bordure, remplissage et qu'on ne lui a pas défini de hauteur fixe alors ses marges hautes et basses vont également fusionner.



Notez bien à nouveau que cet effet de collapse ne va pouvoir avoir lieu qu'avec les marges supérieure et inférieure des éléments et ne va pas se manifester pour les marges gauches et droite.

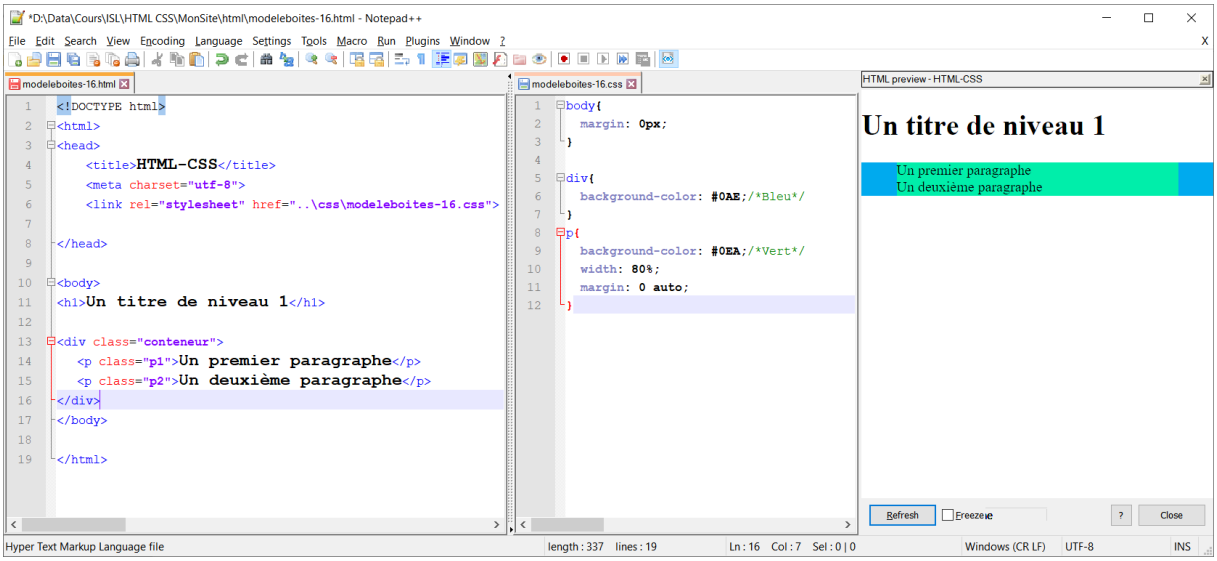
Utiliser margin pour centrer un élément dans son parent

La propriété margin va souvent être utilisée pour centrer horizontalement un élément dans son parent.

Pour faire cela, nous allons devoir définir des marges externes auto à gauche et à droite de l'élément qui devra être centré.

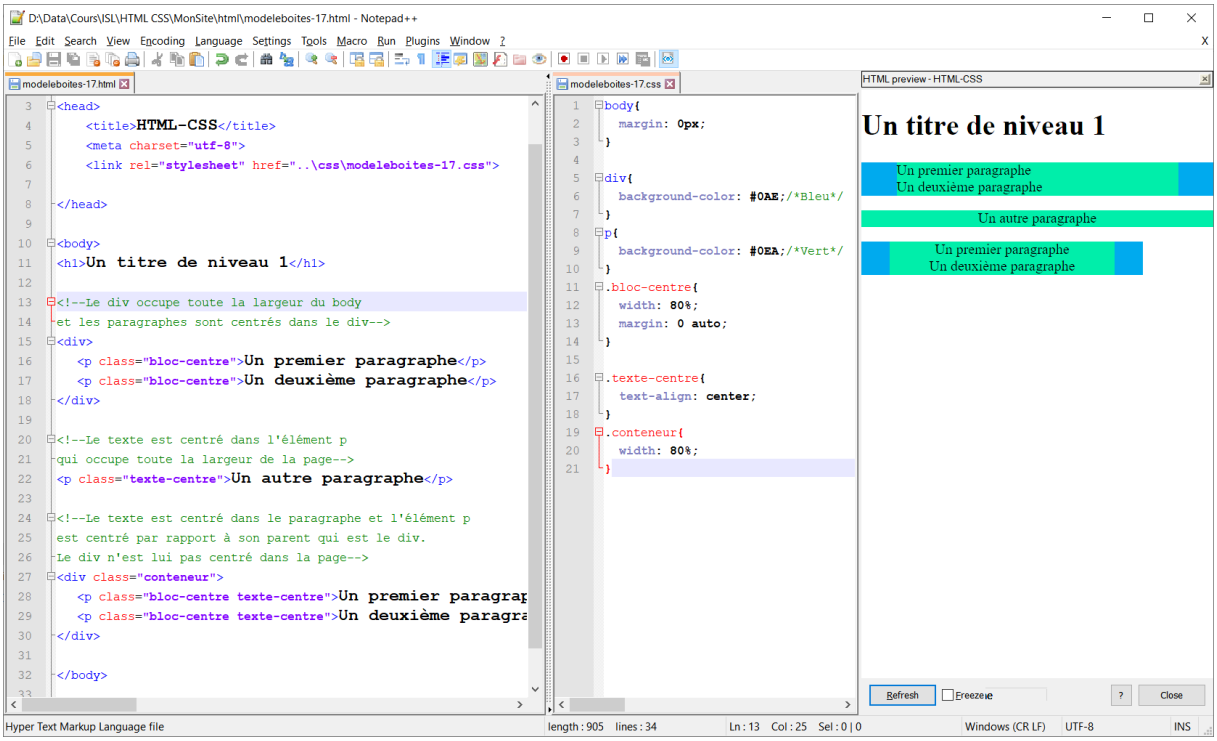
Attention toutefois : cela ne va fonctionner que sur des éléments de type block car centrer des éléments inline (en ligne) dans leur parent n'aurait aucun sens.

De plus, afin de voir l'effet du centrage, il va falloir définir explicitement une largeur pour l'élément qui devra être centré avec la propriété width et lui passer une largeur plus petite que celle de son parent.



Notez bien ici que c'est l'élément entier qui est centré dans son conteneur et non pas le contenu de l'élément qui est centré dans l'élément. Pour centrer horizontalement le contenu d'un élément dans l'élément en soi, on utilisera la propriété text-align et sa valeur center.

Pour centrer l'élément et son contenu, il suffit donc d'utiliser les propriétés margin et text-align ensemble.


D:\Data\Cours\ISL\HTML CSS\MonSite\html\modeleboites-17.html - Notepad++
modeleboites-17.html
modeleboites-17.css
HTML preview - HTML-CSS
Un titre de niveau 1
Un premier paragraphe
Un deuxième paragraphe
Un autre paragraphe
Un premier paragraphe
Un deuxième paragraphe
Refresh
Freeze
Close

La propriété CSS box-sizing

La propriété box-sizing va nous permettre de définir quelles boites doivent être inclues dans le calcul de la largeur et de la hauteur d'un élément.

Cette propriété va être très utile pour éviter qu'un élément ne dépasse de son parent à cause de bordures trop large ou de marges internes trop grandes par exemple.

Les interactions entre les différentes propriétés du modèle des boites

Commençons déjà par souligner comment fonctionnent les propriétés liées au modèle des boites ensemble.

Vous pouvez retenir que lorsqu'aucune largeur n'est explicitement définie pour un bloc, alors l'ajout de marges (externes comme internes) et de bordures va compresser le contenu ou les boites internes afin que l'élément ne dépasse pas de son parent conteneur.

En revanche, dès qu'on définit une largeur pour l'élément auquel on applique des marges et / ou des bordures, les différentes tailles des marges et bordures vont venir s'ajouter par défaut à la taille définie et l'élément va ainsi pouvoir potentiellement dépasser de son conteneur.

Le fonctionnement de la propriété box-sizing

La propriété box-sizing va nous permettre d'indiquer que l'on souhaite inclure les marges internes et les bordures dans le calcul de la taille d'un élément.

Nous allons pouvoir fournir l'un des mots clefs suivants à cette propriété :

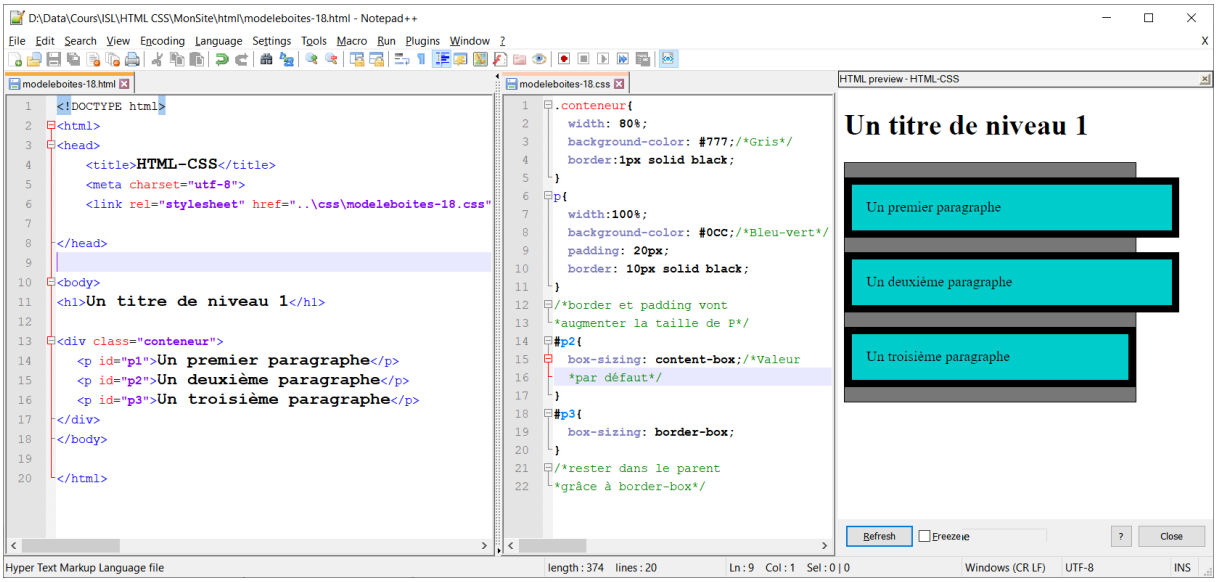
- content-box : valeur par défaut. Les dimensions définies pour l'élément vont s'appliquer à sa boite de contenu. Toute marge interne ou bordure ajoutées ensuite vont augmenter la taille de l'élément ;
- border-box : les dimensions définies pour l'élément vont s'appliquer à la boite contenant le contenu + le padding + les bordures. En ajoutant ou en augmentant la taille des marges internes ou des bordures, la taille de l'élément ne change pas mais son contenu sera compressé.

Exemple d'utilisation de la propriété box-sizing

Dans l'exemple ci-dessous, nos paragraphes sont tous des enfants d'un div class="conteneur". La largeur des éléments p est fixée à 100% de la taille de leur élément parent.

Par défaut, la largeur de la boite de contenu des paragraphes sera égale à celle du div parent. Ensuite, nous ajoutons des marges internes et des bordures à nos paragraphes. *La taille des marges et de la bordure va par défaut s'ajouter à la taille définie avec width pour nos paragraphes et ceux-ci vont donc dépasser de leur parent.*

Nous allons donc utiliser la propriété box-sizing et sa valeur border-box pour que la largeur définie inclue les marges internes et bordures dans son calcul et afin que notre paragraphe ne dépasse pas de son élément parent.



Créer des bordures arrondies avec border-radius en CSS

Le module CSS3 relatif aux bordures a apporté de nouvelles fonctionnalités nous permettant de personnaliser encore davantage nos bordures. Parmi celles-ci, l'une des nouveautés les plus attendues était la possibilité de créer des bordures arrondies.

Nous allons donc pouvoir créer des bordures arrondies en CSS (ou plus exactement arrondir les bords d'un élément HTML) en utilisant la propriété border-radius.

Notez que l'arrondi créé va s'appliquer aux bords de l'élément en soi (ou à son arrière-plan si vous préférez). Cela signifie que nous n'avons pas forcément besoin de définir une quelconque bordure avec border pour que border-radius s'applique.

Comment sont définis les arrondis avec border-radius ?

La propriété border-radius va prendre deux valeurs représentant la dimension du « rayon » sur l'axe des X (axe horizontal) et la dimension du « rayon » sur l'axe des Y (axe vertical) d'une ellipse qui va servir à définir la forme de l'arrondi.

Ces deux valeurs dont être séparées par un slash comme ceci : border-radius :X/Y. La deuxième valeur est facultative et si elle est omise elle sera considérée comme égale à la première par défaut.

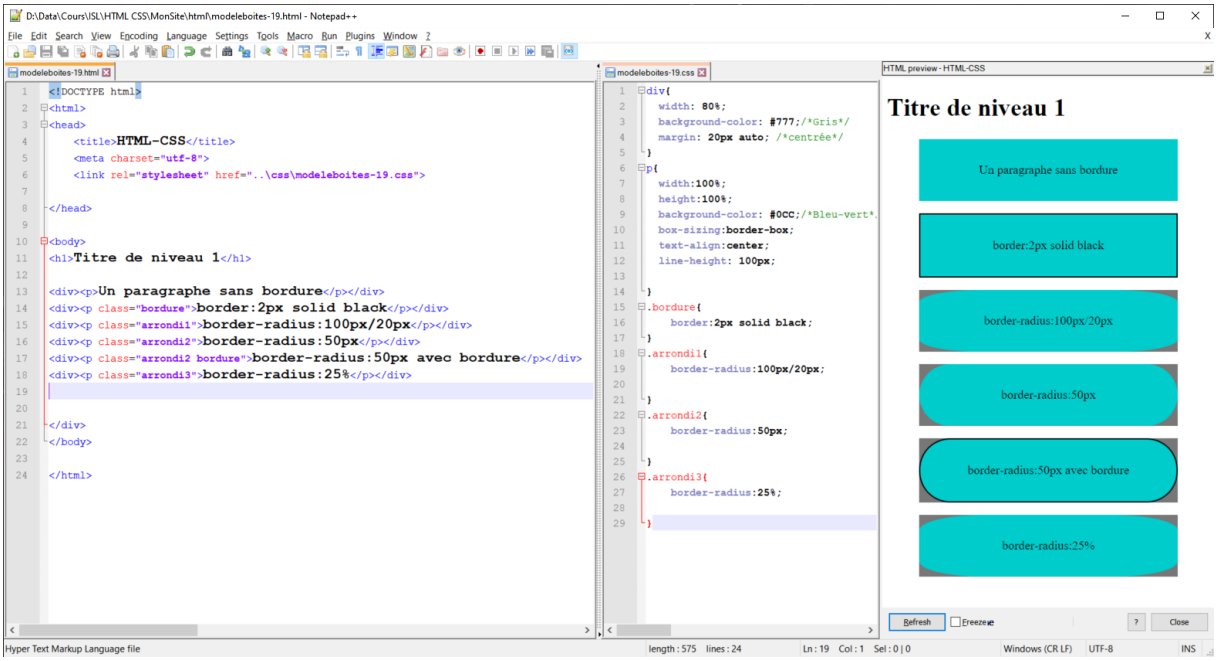
Note : ici, j'appelle « rayon » sur l'axe de X et des Y les longueurs égales à la moitié de la largeur et à la moitié de la hauteur totale de l'ellipse, c'est à dire la distance du point central de l'ellipse à son bord gauche ou droit dans le plan horizontal et la distance du point central de l'ellipse à son bord supérieur ou inférieur dans le plan vertical. Bien évidemment, parler de rayon pour une ellipse n'a aucun sens d'un point de vue mathématique, ce n'est que pour vous donner une image.

Ces deux valeurs vont pouvoir être des valeurs de type longueur (en px par exemple) ou des pourcentages (%).

Pour être tout à fait précis, notez déjà que ce sont 4 ellipses qui vont être utilisées pour définir les bordures arrondies d'un élément (une ellipse pour chaque coin).

Pour le moment, nous n'allons mentionner qu'une valeur de rayon horizontale et une valeur de rayon vertical avec border-radius ce qui fait que les 4 ellipses vont être de tailles identiques (c'est la raison pour laquelle je parlerai de « l'ellipse » plutôt que « des ellipses »). Nous verrons par la suite comment définir des bordures arrondies différentes pour chaque côté d'un élément.

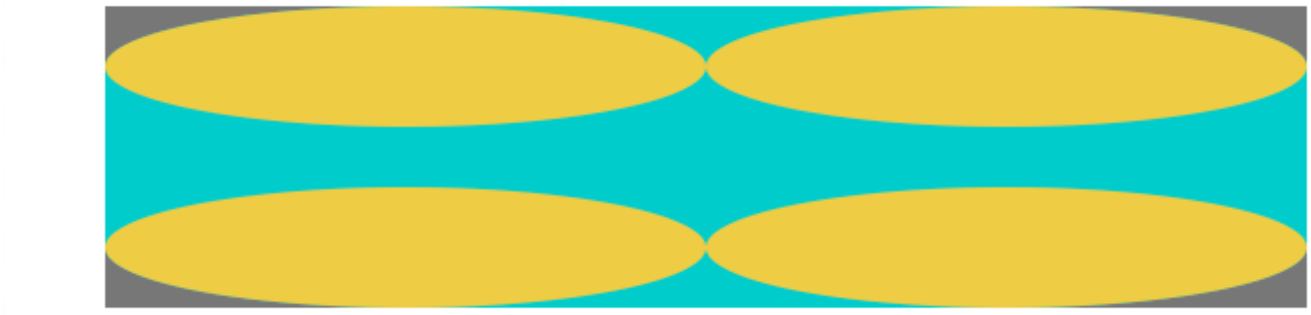
Prenons directement un premier exemple pour illustrer tout ça :



Ici, nous voulons appliquer des bordures arrondies à nos différents paragraphes. *On commence par placer chaque paragraphe dans un div et par donner une couleur de fond à nos éléments div et p pour bien voir la bordure arrondie par la suite.*

On attribue également une largeur et une hauteur pour nos différents paragraphes égales à celles de leur parent div pour pouvoir ensuite utiliser la propriété box-sizing et régler d'ores-et-déjà les potentiels problèmes de dépassement des bordures définies avec border.

Ensuite, on définit 3 arrondis différents avec border-radius. Le premier arrondi est défini avec border-radius : 100px / 20px. Cela signifie que chacun des bords arrondis de notre paragraphe va être formé à partir d'une ellipse de largeur maximale de 200px (le « rayon » sur l'axe horizontal * 2) et de hauteur maximale de 40px (le « rayon » sur l'axe vertical * 2). Cela devient très net si on trace ces ellipses à l'intérieur de notre paragraphe :



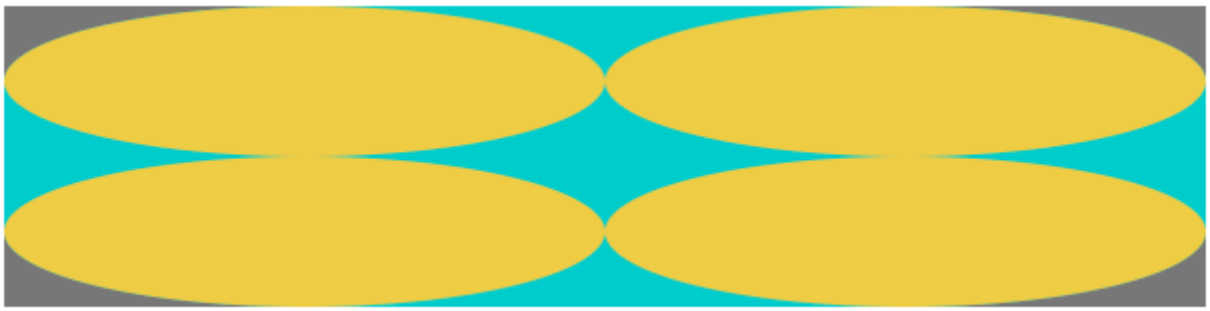
Pour notre deuxième arrondi, on ne précise qu'une valeur pour notre propriété border-radius en pixels avec border-radius : 50px. Par défaut, la deuxième valeur va donc être également fixée à 50px.

Dans ce cas, les bords de notre paragraphe vont être arrondis selon la forme d'une ellipse de rayon horizontal et vertical de 50px, ce qui n'est rien d'autre qu'un cercle.

En effet, lorsque les deux valeurs passées à border-radius sont les mêmes, alors nous sommes dans le cas particulier où notre ellipse est un cercle (à l'exception des valeurs en % dont nous allons parler par la suite). Ici, vous devez donc vous imaginer le contour d'un cercle de diamètre = 100px.

Nous utilisons deux fois cet arrondi : une fois dans un paragraphe sans bordures et une fois avec un paragraphe qui possède des bordures. Ici, on précise border-radius : 25% ce qui est équivalent à border-radius : 25%/ 25%. L'arrondi va ainsi être créé à partir d'une ellipse de rayon horizontal égal à 25% de la largeur du paragraphe et de rayon vertical égal à 25% de la hauteur du paragraphe.

Finalement, notre dernier arrondi utilise des valeurs en pourcentage. Ici, vous devez bien comprendre que le pourcentage défini va être un pourcentage de dimension associée (largeur ou hauteur) du paragraphe.



La propriété border-radius et les valeurs en pourcentage (%)

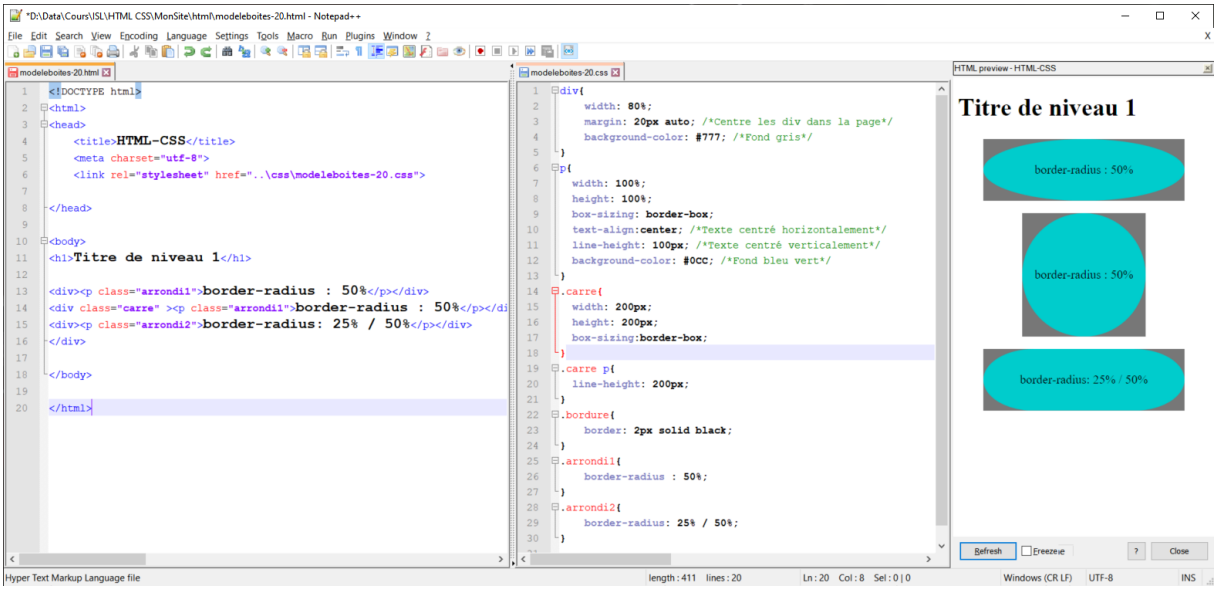
Il faut faire très attention lorsqu'on définit un border-radius avec des valeurs en pourcentage car le ou les pourcentage(s) donné(s) seront exprimés en fonction de la largeur et de la hauteur de la boite représentant l'élément auquel on applique la bordure.

Ainsi, écrire border-radius : 25% signifie que le « rayon » sur l'axe horizontal de l'ellipse utilisée pour créer la bordure sera égal à 25% de la largeur de l'élément et que le « rayon » sur l'axe vertical de l'ellipse utilisée pour créer la bordure sera égal à 25% de la hauteur de l'élément. Ainsi, si notre

élément a par exemple une largeur de 400px et une hauteur de 100px, il est équivalent d'écrire border-radius : 25% et border-radius : 100px / 25px.

Ici, vous pouvez retenir la chose suivante : en ne passant qu'une seule valeur à border-radius (que ce soit des px, em, in, viewport related units, cm…), les bordures créées seront toujours issues d'un cercle EXCEPTE dans le cas où l'on utilise une valeur en %.

Bien évidemment, si la boite représentant notre élément est carrée (largeur = hauteur), alors l'arrondi créé sera toujours créé à partir d'un cercle si on ne passe qu'une valeur à border-radius et ceci même si la valeur passée est une valeur en %.



Définir des bordures arrondies différentes

On va tout à fait pouvoir définir un arrondi différent pour chacune des bordures d'un élément. Pour cela, il suffira de passer 2, 3 ou 4 valeurs à la propriété border-radius :

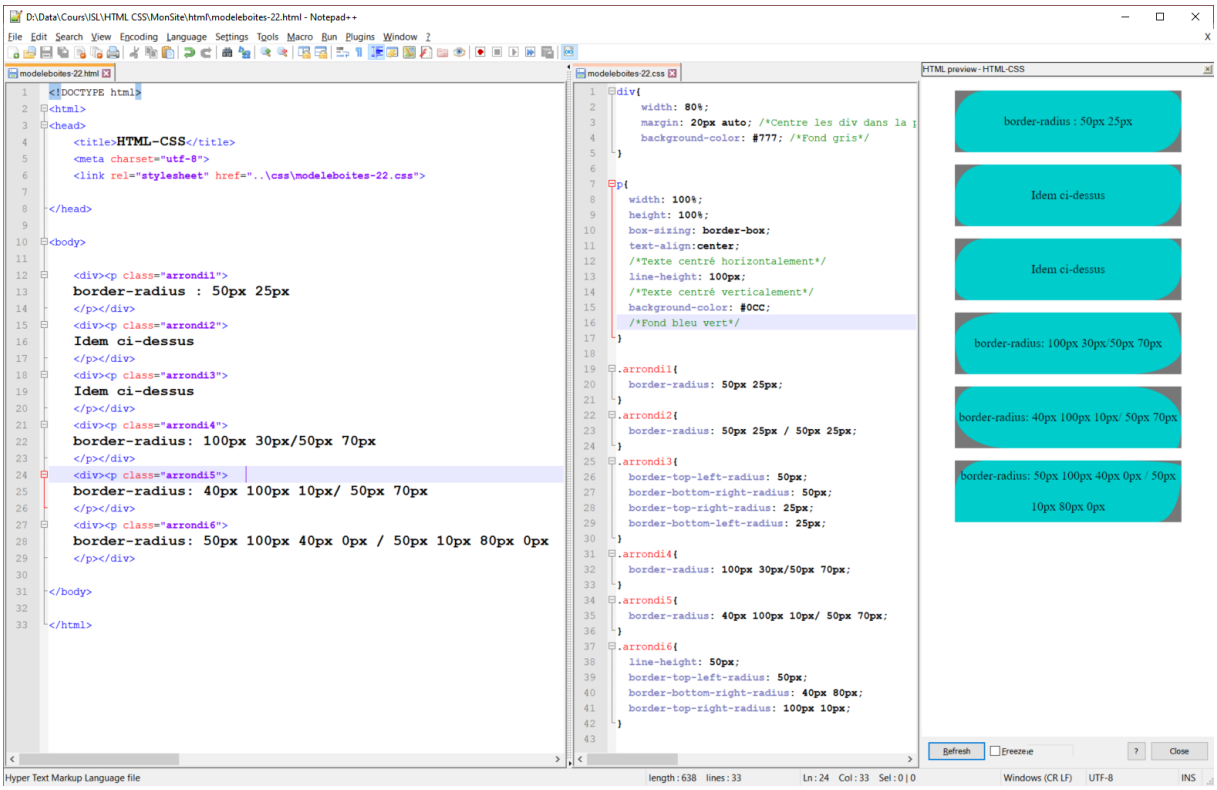
- En passant une valeur : la valeur va définir les arrondis des 4 bords de l'élément ;
- En passant deux valeurs : la première valeur va définir l'arrondi des angles supérieur gauche et inférieur droit de l'élément tandis que la seconde valeur va définir l'arrondi des angles supérieur droit et inférieur gauche de l'élément ;
- En passant trois valeurs : la première valeur définit l'arrondi de l'angle supérieur gauche de l'élément, la deuxième valeur définit l'arrondi des angles supérieur droit et inférieur gauche de l'élément tandis que la troisième valeur définit l'arrondi de l'angle inférieur droit de l'élément ;
- En passant quatre valeurs : la première valeur définit l'arrondi de l'angle supérieur gauche de l'élément, la deuxième valeur définit l'arrondi de l'angle supérieur droit, la troisième valeur définit l'arrondi de l'angle inférieur droit tandis que la quatrième valeur définit l'arrondi de l'angle inférieur gauche.

Notez par ailleurs que la propriété border-radius est une notation raccourcie des propriétés suivantes :

- border-top-left-radius : Définit l'arrondi de l'angle supérieur gauche de l'élément ;
- border-top-right-radius : Définit l'arrondi de l'angle supérieur droit de l'élément ;
- border-bottom-right-radius : Définit l'arrondi de l'angle inférieur droit de l'élément ;
- border-bottom-left-radius : Définit l'arrondi de l'angle inférieur gauche de l'élément.

Une nouvelle fois, nous allons pouvoir mentionner deux « rayons » différents pour chacune des bordures arrondies. Dans le cas où on utilise la notation raccourcie border-radius, il va cependant falloir faire bien attention à l'ordre des valeurs lorsqu'on souhaite définir des arrondis différents car celui-ci peut sembler contre intuitif à première vue : nous allons devoir commencer par passer toutes les valeurs de « rayons » horizontaux pour nos arrondis PUIS les valeurs de « rayons » verticaux, en séparant ces deux groupes par un slash.

Notez également qu'en utilisant les propriétés complètes border-top-left-radius, border-top-right-radius, etc. plutôt que la notation raccourcie border-radius il ne faudra PAS indiquer de slash pour séparer les valeurs des deux « rayons » de l'ellipse si on souhaite en préciser deux différentes.



Pour notre premier arrondi, on définit un border-radius : 50px pour les angles supérieur gauche et inférieur droit de l'élément et un border-radius : 25px pour les angles supérieur droit et inférieur gauche de l'élément.

Notre deuxième arrondi est l'équivalent du premier mais écrit en précisant les valeurs des deux rayons pour chacun de nos angles (ici, les valeurs des deux rayons sont identiques pour chaque angle).

Notre troisième arrondi correspond à une troisième façon d'arriver au même résultat que notre premier arrondi, en utilisant cette fois-ci les propriétés complètes border-top-left-radius, border-top-right-radius, etc. plutôt que la notation raccourcie.

Pour notre quatrième arrondi, les angles supérieur gauche et inférieur droit de l'élément seront formés à partir d'une ellipse de rayon 100px (horizontal) et 50px (vertical). Les angles supérieur droit et inférieur gauche de l'élément emprunteront eux leurs arrondis à une ellipse de rayon horizontal de 30px et de rayon vertical de 70px.

Notre cinquième déclaration d'arrondi est un peu plus complexe à appréhender. Ici, il faut bien comprendre qu'on définit 3 rayons horizontaux pour nos bordures arrondis de 40px, 100px et 10px. La première valeur sera utilisée pour définir l'arrondi horizontal de l'angle supérieur gauche de l'élément, la seconde pour les angles supérieur droit et inférieur gauche et la troisième pour l'angle inférieur droit.

Ensuite, on ne définit que 2 valeurs de rayons verticaux de 50px et 70px. La première valeur sera utilisée pour les angles supérieur gauche et inférieur droit tandis que la deuxième valeur sera utilisée pour les angles supérieur droit et inférieur gauche.

Finalement, le dernier exemple d'arrondi sert à illustrer comment créer des arrondis utilisant deux rayons différents avec nos propriétés complètes border-top-left-radius, border-top-right-radius, etc. Je vous rappelle qu'il ne faut pas préciser de slash en utilisant ces propriétés mais écrire les deux valeurs de rayons à la suite.

La gestion des valeurs d'arrondis trop grandes (valeurs aberrantes)

La création de bordures arrondies se fait au moyen d'ellipses ou, dans certains cas particuliers, de cercles. La création de bordures arrondies se fait au moyen d'ellipses ou, dans certains cas particuliers, de cercles. Parfois, cependant, il peut arriver que des valeurs aberrantes soient fournies.

Définition d'une valeur aberrante pour border-radius

Les valeurs passées à la propriété border-radius vont être considérées comme aberrantes et les bordures arrondies vont être redimensionnées dès que la règle « deux bordures adjacentes ne doivent pas se chevaucher » ne sera plus respectée c'est-à-dire dans les situations suivantes :

- Si l'une (ou chacune) des deux sommes des deux « rayons » dans l'axe vertical des ellipses servant à créer les bordures supérieure gauche et inférieure gauche et supérieure droite et inférieure droite dépasse la valeur de la hauteur de la boite de l'élément ;

- Si l'une (ou chacune) des deux sommes des deux « rayons » dans l'axe horizontal des ellipses servant à créer les bordures supérieure gauche et supérieure droite et inférieure gauche et inférieure droite dépasse la valeur de la largeur de la boite de l'élément.

Dans ce cas-là, les valeurs passées à border-radius vont être réduites de manière proportionnelle jusqu'à ce que la transition entre les deux bordures arrondies soit fluide (pour être tout à fait exact d'un point de vue mathématique, il faudrait dire « jusqu'à ce qu'on puisse tracer une tangente qui soit parallèle au côté de la boite de l'élément »).

Trouver les valeurs corrigées automatiquement par le CSS à partir des valeurs aberrantes

Imaginons par exemple qu'on ait un paragraphe avec les dimensions suivantes width : 400px; height : 100px et qu'on tente de lui appliquer les bordures arrondies suivantes : border-radius : 100px 250px 200px 300px / 10px 30px 70px 80px.

Cela est équivalent à définir les bordures arrondies suivantes :

- border-top-left-radius : 100px 10px ;
- border-top-right-radius : 250px 30px ;
- border-bottom-right-radius : 200px 70px ;
- border-bottom-left-radius : 300px 80px;

Faisons maintenant les sommes des rayons 2-à-2 de nos bordures dans les axes horizontal et vertical:

- Axe horizontal (top-left + top-right) : 100 + 250 = 350px ;
- Axe horizontal (bottom-right + bottom-left) : 200 + 300 = 500px ;
- Axe vertical (top-left + bottom-left) : 10 + 80 = 90px ;
- Axe vertical (top-right + bottom-right) : 30 + 70 = 100px.

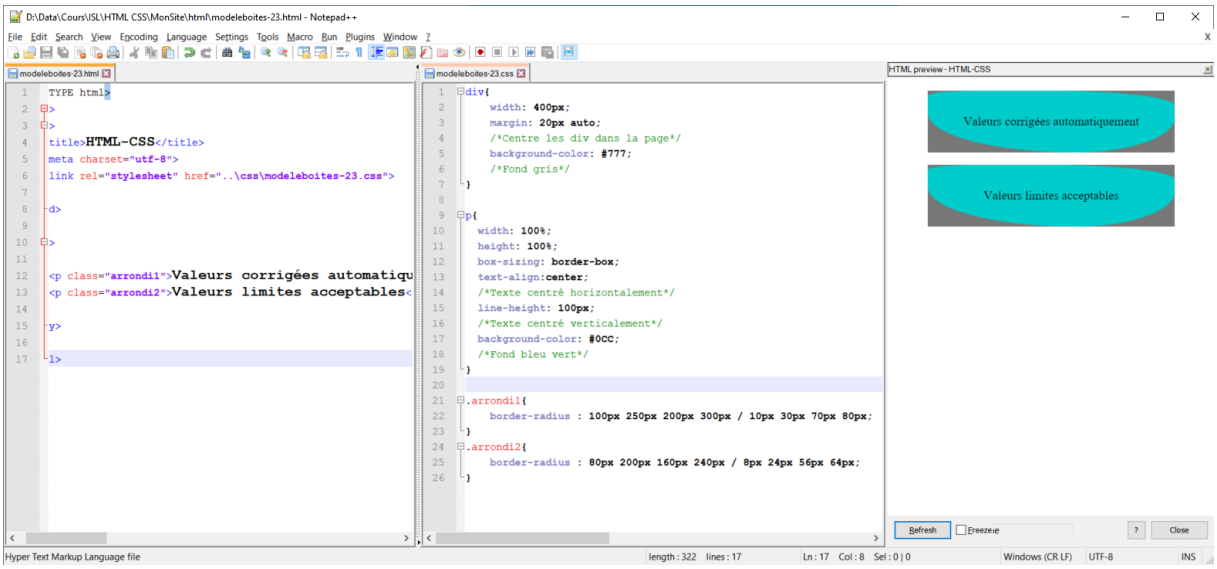
Ici, nous allons avoir un problème pour l'axe horizontal avec les bordures inférieures droite et gauche puisque la valeur totale des deux rayon, 500px, est supérieure à la largeur de l'élément qui est 400px.

Ainsi, TOUTES les bordures arrondies de l'élément vont être redimensionnées de façon proportionnelle jusqu'à arriver à la première valeur acceptable pour les valeurs qui posent problème, c'est-à-dire lorsque la somme des rayons (axe horizontal) des deux bordures inférieures droite et gauche fera 400px.

Nous n'avons donc qu'à faire un calcul de proportionnalité en utilisant une règle de 3 pour connaitre les valeurs qui vont être attribuées. Pour cela, il faut déjà trouver le coefficient de proportionnalité qui nous permet de passer de 500 à 400. On a donc : 400 * 100 / 500 = 0,8.

Il ne nous reste donc plus qu'à multiplier toutes les dimensions données à border-radius par 0,8 pour obtenir les valeurs corrigées automatiquement par le CSS soit : border-radius : 80px 200px 160px 240px / 8px 24px 56px 64px

Regardez plutôt le résultat ci-dessous pour vous en convaincre. J'ai également recréé les bordures dans le dernier exemple telles qu'elles seraient si elles n'avaient pas été ajustées :



Vous pouvez également noter ici que :

- Si nous avons plusieurs valeurs aberrantes, alors nous utiliserons le coefficient de proportionnalité de la valeur la plus aberrante afin que tous les arrondis soient de tailles acceptables ;
- Si une valeur est aberrante en soi (c'est-à-dire si une la valeur d'un rayon de l'ellipse servant à définir bordure arrondie est déjà supérieure à la taille de la boite de l'élément), nous procéderons exactement de la même manière que précédemment pour trouver la valeur qui sera définie automatiquement en CSS ;
- Ces règles de correction des valeurs s'appliquent pour tous types de valeurs passées, que ce soit des valeurs en px ou en %.