

**HTML**



**CSS**



**Fonds, dégradés et ombres**

Rédigé par: Jean-Luc Colson

Date première rédaction: Janvier 2020

## SUIVI DES MODIFICATIONS A LA FICHE

Série	Date	Page(s) modifiée(s)	Raison

## Contents

Modifier la couleur de fond du contenu HTML avec background-color CSS .....	5
Le champ d'application de background-color .....	5
Les valeurs de couleur de background-color .....	5
Exemples d'utilisation de background-color en CSS .....	5
Ajouter un background-color semi transparent à un élément .....	6
Ajouter différentes couleurs de fond à un élément parent et à ses enfants .....	7
Ajouter des images de fond aux éléments HTML avec background-image CSS.....	8
Les couches ou « layers » constituant le fond d'un élément .....	8
Utilisation de la propriété background-image .....	8
Ajouter une image de fond avec background-image .....	8
Ajouter plusieurs images de fond avec background-image .....	9
Une bonne pratique : déclarer une couleur de fond avec une image de fond .....	10
Ajouter différentes images de fond à différents éléments .....	10
Les propriétés nous permettant de gérer le comportement du fond .....	11
Gérer la répétition d'une image de fond avec background-repeat .....	11
Gérer la taille des images de fond avec background-size .....	14
Gérer le défilement d'une image de fond avec background-attachment .....	16
Déterminer la position du fond avec background-origin .....	17
Positionner une image de fond avec background-position .....	18
Déterminer la surface que peut occuper notre fond avec background-clip .....	19
La propriété CSS raccourcie background .....	20
Création de dégradés linéaires en CSS avec linear-gradient.....	23
Qu'est-ce qu'un dégradé linéaire ? .....	23
Direction et couleurs du dégradé .....	23
Définition de la direction d'un dégradé linéaire .....	23
Les "color stops" et l'avancement d'un dégradé .....	24
Exemples de création de dégradés linéaires .....	25
Création de dégradés avec direction et color stops .....	26
Création de dégradés semi-transparents .....	27
Créer plusieurs dégradés en fond d'un élément .....	29

---

La répétition des dégradés .....	30
Création de dégradés radiaux en CSS avec radial-gradient .....	33
Qu'est-ce qu'un dégradé radial ? .....	33
Comment se construit un dégradé radial en CSS.....	33
La forme finale du dégradé .....	34
La taille du dégradé .....	34
Couleurs et color stops du dégradé.....	35
Exemples de création de dégradés radiaux.....	35
Création de dégradés radiaux simples.....	35
Création de dégradés radiaux complexes.....	37
Dégradés radiaux multiples et semi-transparents.....	38
La répétition de dégradés.....	39
Ajouter des ombres aux éléments avec box-shadow en CSS.....	40
Définition d'une ombre avec box-shadow.....	40
Exemples d'utilisation de box-shadow et de création d'ombres en CSS .....	41
Création d'ombres simples.....	41
Ajout de la valeur inset pour créer une ombre interne .....	42
Création d'ombres colorées et / ou semi transparentes .....	43
Ajout d'un flou et ombre centrée autour de la boîte .....	44
Une ombre plus grande ou plus petite que notre boîte .....	45
Création d'ombres multiples autour d'un élément .....	48



# Modifier la couleur de fond du contenu HTML avec background-color CSS

Dans cette nouvelle partie, nous allons nous intéresser à la personnalisation du fond de nos boîtes éléments en CSS et commençant ici avec l'ajout d'une couleur de fond qui va être possible avec la propriété **background-color** ou avec la notation **background** dont nous reparlerons dans la leçon suivante.

## Le champ d'application de background-color

Tout élément HTML est avant tout une boîte rectangulaire qui peut être composée d'un contenu, de marges internes, d'une bordure et de marges externes.

Lorsqu'on ajoute une couleur de fond à un élément, cette couleur va remplir l'espace pris par le contenu et les marges internes. En revanche, la couleur de fond ne s'appliquera pas aux marges externes qui sont considérées comme « en dehors » de l'élément.

Notez par ailleurs une chose intéressante ici : lorsqu'on donne une couleur de fond à un élément parent, on « dirait » que les éléments enfants en héritent. En réalité, ce n'est pas tout à fait correct : les éléments ont par défaut un **background-color : transparent** et c'est la raison pour laquelle la couleur de fond de l'élément parent va également remplir l'espace pris par les éléments enfants sauf si on spécifie une couleur de fond différente pour eux.

## Les valeurs de couleur de background-color

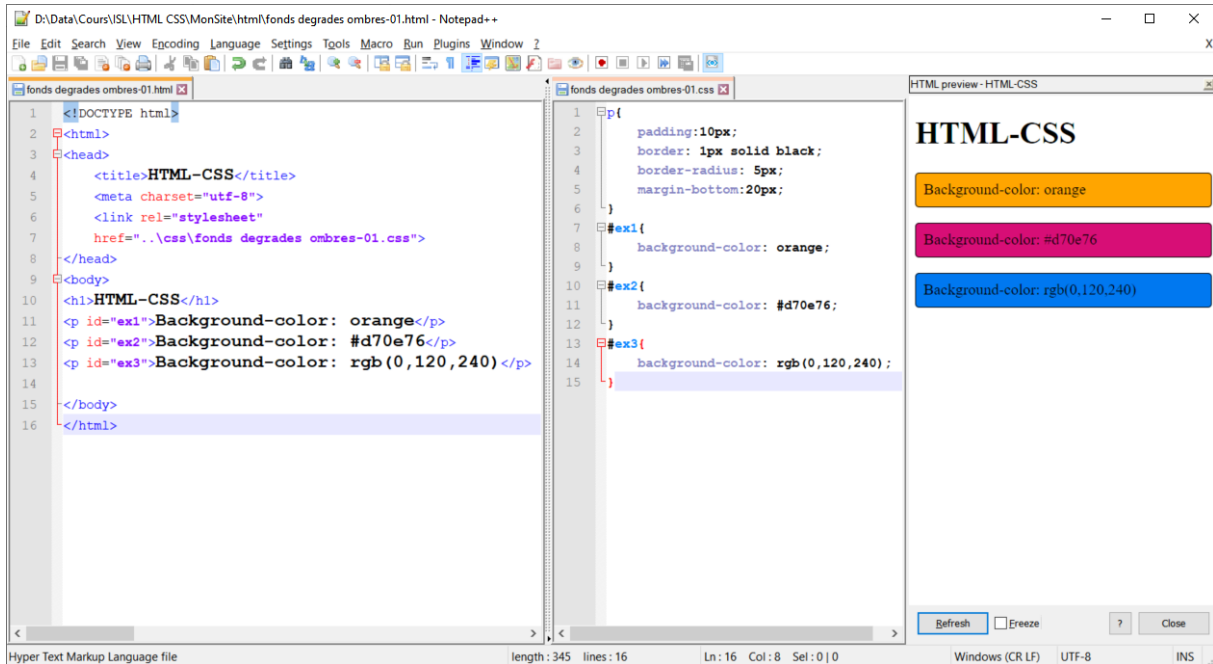
La propriété CSS **background-color** va accepter les mêmes notations de couleurs que la propriété **color** c'est-à-dire toutes les formes de notations de couleurs en CSS. On va ainsi pouvoir lui passer :

- Un nom de couleur de anglais ;
- Une valeur hexadécimale ;
- Une valeur RGB (ou RGBa) ;
- Une valeur HSL (ou HSLa).

## Exemples d'utilisation de background-color en CSS

Attribuer un background-color opaque à un élément

Nous allons pouvoir ajouter une couleur de fond opaque à un élément en utilisant soit une notation de couleur de type nom de couleur, soit une notation hexadécimale, soit une notation RGB ou HSL.



The screenshot shows Notepad++ with two files open: 'fonds degradés ombres-01.html' and 'fonds degradés ombres-01.css'. The HTML file contains a basic structure with a title 'HTML-CSS' and three paragraphs with different background colors. The CSS file defines a base style for padding, border, and margin, and three specific styles for the paragraphs using different color notations: orange, hex code #d70e76, and RGB values (0, 120, 240). A preview window on the right shows the rendered HTML with these colors.

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>HTML-CSS</title>
5   <meta charset="utf-8">
6   <link rel="stylesheet"
7     href="..\css\fonds degradés ombres-01.css">
8 </head>
9 <body>
10  <h1>HTML-CSS</h1>
11  <p id="ex1">Background-color: orange</p>
12  <p id="ex2">Background-color: #d70e76</p>
13  <p id="ex3">Background-color: rgb(0,120,240) </p>
14
15 </body>
16 </html>
  
```

```

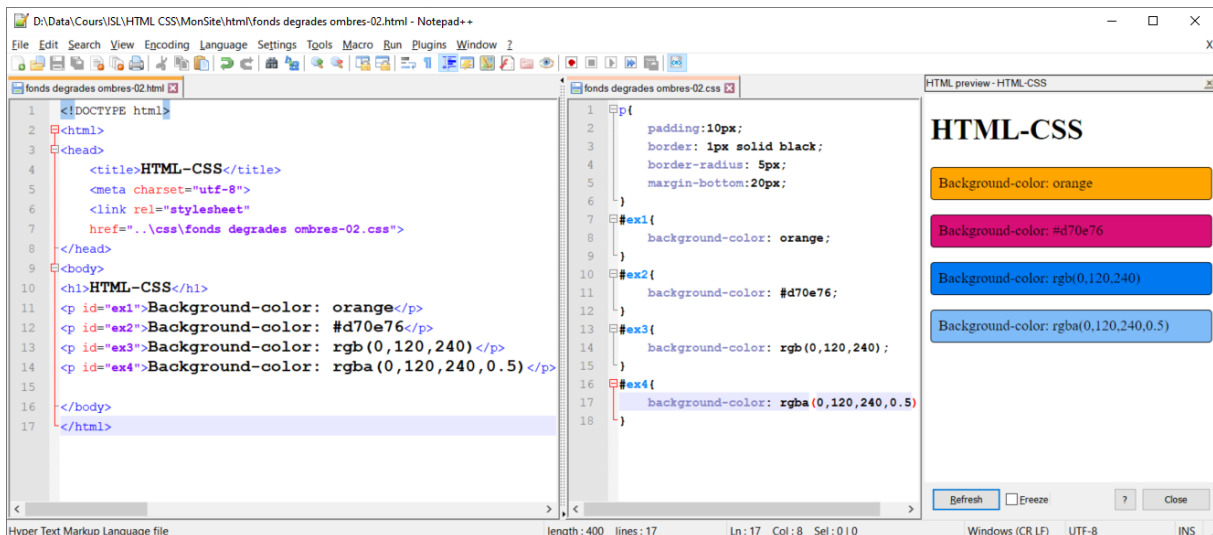
1 p{
2   padding:10px;
3   border: 1px solid black;
4   border-radius: 5px;
5   margin-bottom:20px;
6 }
7 #ex1{
8   background-color: orange;
9 }
10 #ex2{
11   background-color: #d70e76;
12 }
13 #ex3{
14   background-color: rgb(0,120,240);
15 }
  
```

Ici, j'ai ajouté un **padding** c'est-à-dire des marges intérieures, une bordure et des marges externes aux différents paragraphes afin de rendre l'exemple plus visuel et que vous voyiez bien à quoi s'applique le **background-color** en CSS.

## Ajouter un background-color semi transparent à un élément

Nous allons également pouvoir ajouter une couleur de fond semi transparente à un élément en utilisant cette fois-ci les notations RGBA.

Notez bien ici qu'utiliser la propriété **opacity** sur nos éléments ne produirait pas le comportement voulu puisque l'intégralité de nos éléments (contenu et bordure compris) serait alors semi transparent or nous voulons que seulement le fond de l'élément soit transparent.



The screenshot shows Notepad++ with two files open: 'fonds degradés ombres-02.html' and 'fonds degradés ombres-02.css'. The HTML file is similar to the previous one but includes a fourth paragraph with a semi-transparent background color using the RGBA notation. The CSS file adds a fourth style for this paragraph, also using the RGBA notation for the background color.

```

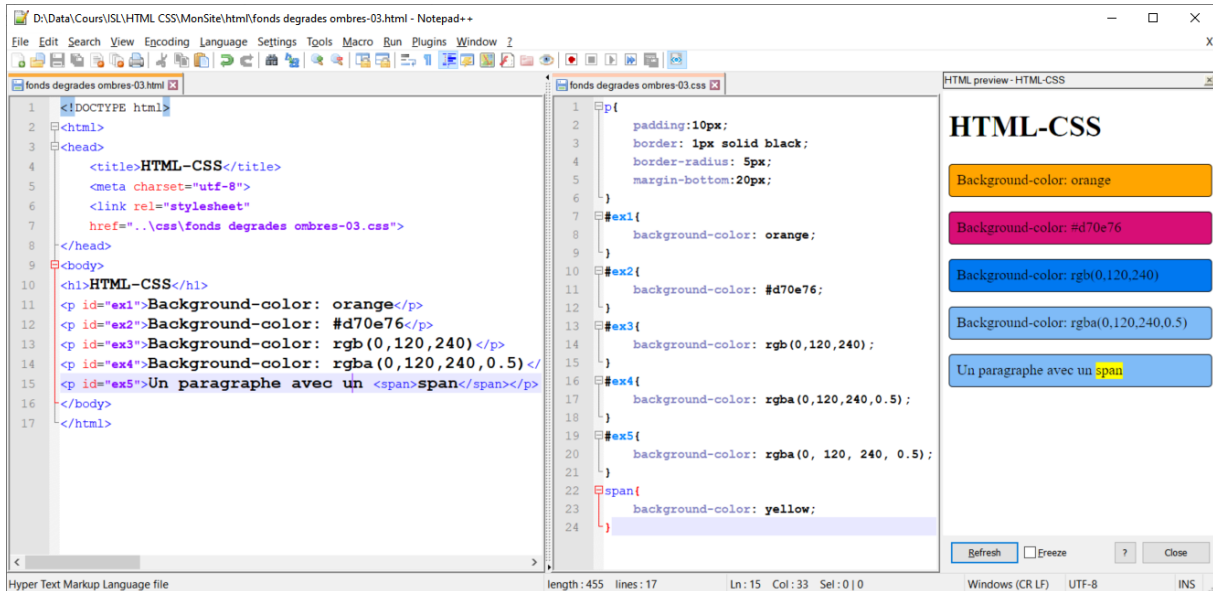
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>HTML-CSS</title>
5   <meta charset="utf-8">
6   <link rel="stylesheet"
7     href="..\css\fonds degradés ombres-02.css">
8 </head>
9 <body>
10  <h1>HTML-CSS</h1>
11  <p id="ex1">Background-color: orange</p>
12  <p id="ex2">Background-color: #d70e76</p>
13  <p id="ex3">Background-color: rgb(0,120,240) </p>
14  <p id="ex4">Background-color: rgba(0,120,240,0.5) </p>
15
16 </body>
17 </html>
  
```

```

1 p{
2   padding:10px;
3   border: 1px solid black;
4   border-radius: 5px;
5   margin-bottom:20px;
6 }
7 #ex1{
8   background-color: orange;
9 }
10 #ex2{
11   background-color: #d70e76;
12 }
13 #ex3{
14   background-color: rgb(0,120,240);
15 }
16 #ex4{
17   background-color: rgba(0,120,240,0.5);
18 }
  
```

## Ajouter différentes couleurs de fond à un élément parent et à ses enfants

Finalement, notez qu'on va tout à fait pouvoir ajouter différents **background-color** à différents éléments imbriqués. Dans ce cas-là, la couleur de fond définie pour chaque élément s'appliquera à l'élément en question.



The screenshot shows a Notepad++ window with two files open: 'fonds degradés ombres-03.html' and 'fonds degradés ombres-03.css'. The HTML file contains a document type declaration, a head section with a title 'HTML-CSS', a meta charset, and a link to the CSS file. The body contains a heading 'HTML-CSS' and five paragraphs, each with a unique ID and a background color. The CSS file defines a base style for the 'p' tag and specific styles for each ID, using various color notations: orange, #d70e76, rgb(0,120,240), and rgba(0,120,240,0.5). A span element is also styled with a yellow background. A preview window on the right shows the rendered HTML with the corresponding background colors.

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>HTML-CSS</title>
5 <meta charset="utf-8">
6 <link rel="stylesheet"
7 href="..\css\fonds degradés ombres-03.css">
8 </head>
9 <body>
10 <h1>HTML-CSS</h1>
11 <p id="ex1">Background-color: orange</p>
12 <p id="ex2">Background-color: #d70e76</p>
13 <p id="ex3">Background-color: rgb(0,120,240)</p>
14 <p id="ex4">Background-color: rgba(0,120,240,0.5)</p>
15 <p id="ex5">Un paragraphe avec un <span>span</span></p>
16 </body>
17 </html>

```

```

1 p{
2 padding:10px;
3 border: 1px solid black;
4 border-radius: 5px;
5 margin-bottom:20px;
6 }
7 #ex1{
8 background-color: orange;
9 }
10 #ex2{
11 background-color: #d70e76;
12 }
13 #ex3{
14 background-color: rgb(0,120,240);
15 }
16 #ex4{
17 background-color: rgba(0,120,240,0.5);
18 }
19 #ex5{
20 background-color: rgba(0,120,240,0.5);
21 }
22 span{
23 background-color: yellow;
24 }

```

Ici, vous pouvez noter que le **background-color** ne s'applique que sur l'arrière-plan du contenu de notre **span** ce qui est tout à fait normal puisqu'un élément **span** est un élément de type **inline** par défaut et ne prend donc que l'espace nécessaire à son contenu en largeur.

## Ajouter des images de fond aux éléments HTML avec background-image CSS

Dans cette nouvelle leçon, nous allons apprendre à insérer une ou plusieurs images en fond d'un élément plutôt qu'une simple couleur. Nous allons pouvoir faire cela avec la propriété **background-image** ou avec la notation raccourcie **background**.

Notez que les dégradés sont considérés comme des images en CSS. Nous allons donc devoir utiliser **background-image** ou **background** pour définir un dégradé en fond d'un élément. Nous allons apprendre à créer des dégradés dans la leçon suivante.

### Les couches ou « layers » constituant le fond d'un élément

Pour bien comprendre comment vont fonctionner et s'appliquer les propriétés **background-image** et **background-color**, vous devez avant tout savoir qu'un élément HTML (ou plus exactement la boîte le représentant) peut avoir plusieurs couches depuis le CSS3.

Cela veut dire qu'on va pouvoir « empiler » différents fonds les uns au-dessus des autres pour un même élément, à une limitation près qui est qu'on ne peut déclarer qu'une seule valeur pour la propriété **background-color** pour un élément et donc qu'un élément ne peut avoir qu'une couche de couleur de fond avec cette propriété.

En revanche, on va tout à fait pouvoir empiler une ou plusieurs images les unes sur les autres et par-dessus une couleur de fond pour un même élément. Cela va pouvoir s'avérer très utile si une image de fond ne peut pas s'afficher pour une raison ou une autre par exemple.

Retenez bien ici l'ordre des couches : la première image déclarée va être la plus proche de l'utilisateur, c'est-à-dire l'image de fond qui sera visible par défaut. Si on spécifie également une couleur de fond, alors cette couche sera placée derrière les couches « images de fond ».

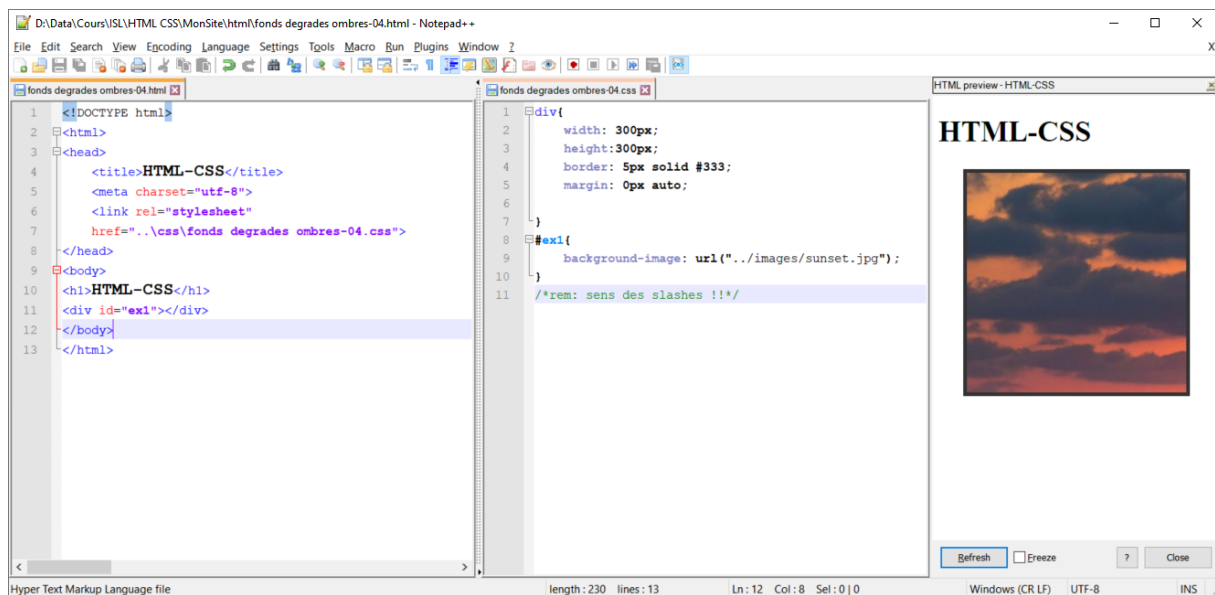
Note : J'ai dit plus haut qu'il était impossible d'ajouter plusieurs couches de couleurs de fond à un élément avec **background-color**. Cependant, rien ne nous empêche d'utiliser les dégradés ou gradient en anglais pour mixer plusieurs couleurs sur la même couche.

### Utilisation de la propriété background-image

La propriété **background-image** va nous permettre d'affecter une ou plusieurs images de fond à un élément. La première image déclarée sera l'image visible par défaut et chacune des autres images possiblement déclarées sera une couche en dessous.

### Ajouter une image de fond avec background-image

Voyons comment ajouter une image de fond :

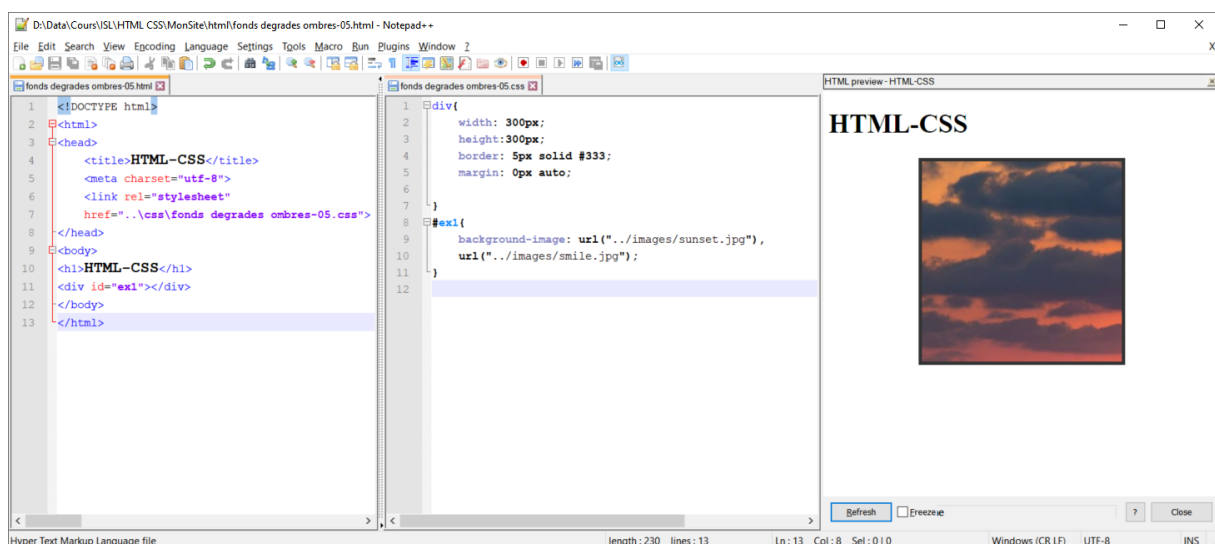


Ici, je me suis contenté d'attribuer une image de fond à mon `div id="ex1"`. Pour cela, je le cible en CSS et j'utilise ma propriété `background-image` en précisant l'adresse relative de mon image (qui est ici dans le même dossier que mes fichiers) au sein d'un paramètre URL.

Notez qu'on va également tout à fait pouvoir passer une adresse absolue pour ajouter une image de fond (une adresse de type `https://...`). Cependant, pour des raisons évidentes, il est déconseillé d'utiliser une image hébergée sur un autre site comme image de fond : si le propriétaire du site supprime son image, votre image de fond n'apparaîtra plus !

### Ajouter plusieurs images de fond avec `background-image`

Pour ajouter plusieurs images de fond avec `background-image`, nous allons tout simplement séparer les différentes déclarations d'images par une virgule comme ceci :



Comme vous pouvez le voir ici, seule la première image déclarée est visible. C'est un comportement tout à fait normal et cela signifie que la première image déclarée a bien été chargée. Cependant, la

deuxième image a bien été placée en arrière-plan également, seulement elle est cachée sous la première.

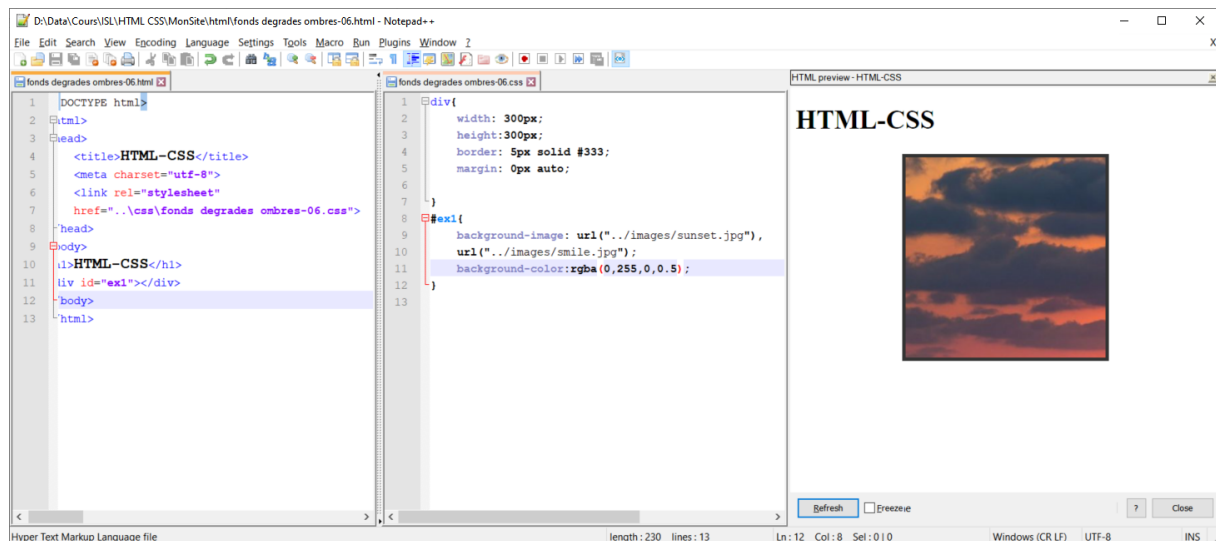
Cette deuxième image va être visible dans deux cas : si la première image ne peut pas être affichée ou si la première image ne remplit pas le fond de l'élément (et que la deuxième image est plus grande que la première). Pas d'inquiétude, nous allons apprendre par la suite à modifier la position et la taille d'une image de fond.

## Une bonne pratique : déclarer une couleur de fond avec une image de fond

Par ailleurs, notez qu'il est considéré comme une bonne pratique de toujours déclarer une couleur de fond avec la propriété **background-color** en plus d'une ou de plusieurs images de fond au cas où celles-ci ne pourraient pas s'afficher.

Une nouvelle fois, si une image peut être chargée normalement, alors la propriété **background-color** n'aura aucun impact puisque la couche « couleur de fond » va se placer sous les couches « images de fond ».

La propriété **background-color** va accepter les mêmes valeurs que la propriété **color**, c'est à dire toutes les valeurs possibles de type « couleur » : nom de couleur, valeur hexadécimale, valeur RGB, RGBA, etc.

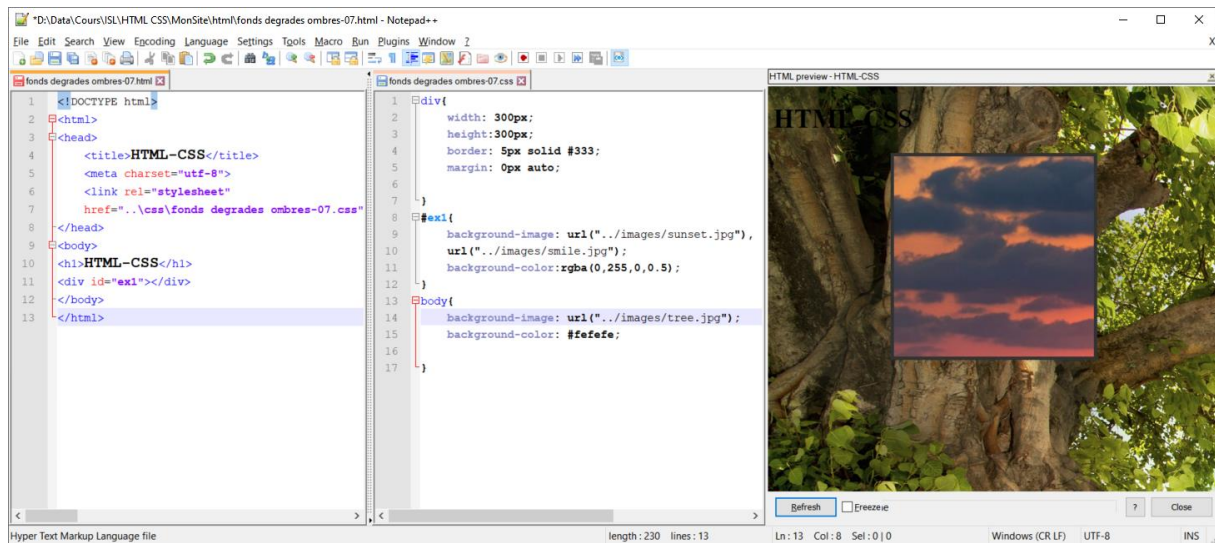


## Ajouter différentes images de fond à différents éléments

Bien évidemment, rien ne nous empêche d'attribuer différentes images de fond à différents éléments. Dans le cas où un élément est inclus dans un autre, l'image ou la couleur de fond qui lui a été attribuée restera au premier plan.

Ainsi, on va par exemple pouvoir ajouter une image de fond à l'élément représentant le contenu visible de notre page à savoir l'élément **body** afin d'ajouter une image de fond « à notre page » en plus de l'image de fond passée à notre **div**.

Notez cependant que pour des raisons évidentes de lisibilité et de clarté nous éviterons généralement de multiplier les images de fond et préférons généralement des couleurs unies notamment pour le fond d'une page.



## Les propriétés nous permettant de gérer le comportement du fond

Nous savons maintenant comment ajouter une ou plusieurs images de fond à nos éléments. Cependant, nous n'avons aucun contrôle sur leur affichage. Pour contrôler le comportement de nos images de fond, nous allons pouvoir utiliser les propriétés suivantes :

- La propriété **background-position** ;
- La propriété **background-size** ;
- La propriété **background-repeat** ;
- La propriété **background-origin** ;
- La propriété **background-clip** ;
- La propriété **background-attachment** ;

Chacune des propriétés citées ci-dessous va nous permettre de préciser un comportement spécifique pour nos images de fond comme la taille, le comportement de répétition, etc.

Notez que toutes ces propriétés peuvent être déclarées d'un coup dans la notation raccourcie **background** que nous étudierons à la fin de cette leçon.

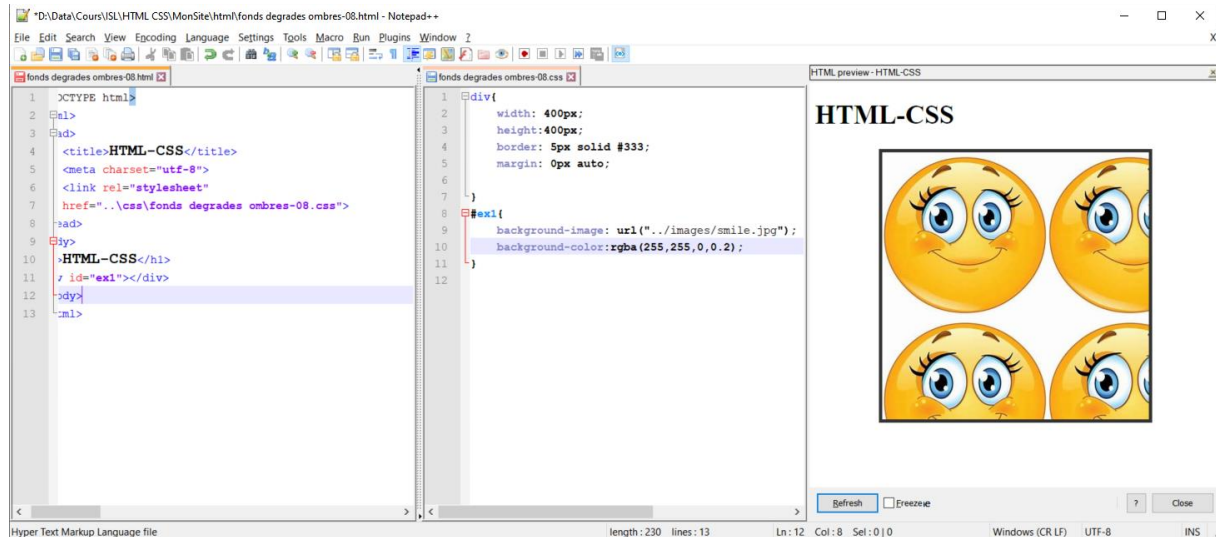
## Gérer la répétition d'une image de fond avec background-repeat

La propriété **background-repeat** va nous permettre de définir si une image d'arrière-plan doit être répétée ou pas et selon quel(s) axe(s) (axe horizontal et / ou vertical).



Par défaut, une image de fond va être rognée à la taille de l'élément en largeur et / ou en hauteur si elle plus grande que celui-ci et au contraire être répétée horizontalement et verticalement si elle est plus petite que l'élément jusqu'à remplir la surface de fond définie.

Regardez plutôt l'exemple suivant :



Ici, mon div fait 800px de large (ou 810px en comptant les bordures) et 400px de haut (ou 410px en comptant les bordures). Mon image « emoji-smile.png » possède quant-à-elle une dimension de 248px\*248px et va donc être par défaut répétée jusqu'à remplir le fond du div à la fois en largeur et en hauteur.

Notez deux choses intéressantes ici :

- Par défaut, l'image ne va pas être redimensionnée pour apparaître un nombre entier de fois : on voit bien ici que les dernières itérations en largeur et en hauteur sont coupées ;
- Mon image gère ici la transparence, c'est pour cela que vous pouvez voir la couleur de fond du **div** en arrière-plan dans les endroits non remplis par celle-ci !

Nous allons pouvoir annuler ce comportement de répétition horizontale et verticale par défaut et définir nos propres règles de répétitions grâce à la propriété **background-repeat**. Pour cela, nous allons passer une ou deux valeurs à la propriété **background-repeat** parmi les valeurs suivantes.

Si une seule valeur est passée, alors elle définira le comportement de répétition de l'image à la fois sur son axe horizontal et sur son axe vertical. Si deux valeurs sont passées, la première valeur définira le comportement de répétition horizontal tandis que la seconde déterminera le comportement de répétition vertical de l'image.

- **repeat** : valeur par défaut. L'image est répétée jusqu'à remplir le fond ;
- **no-repeat** : l'image n'est pas répétée et n'apparaît qu'une fois ;

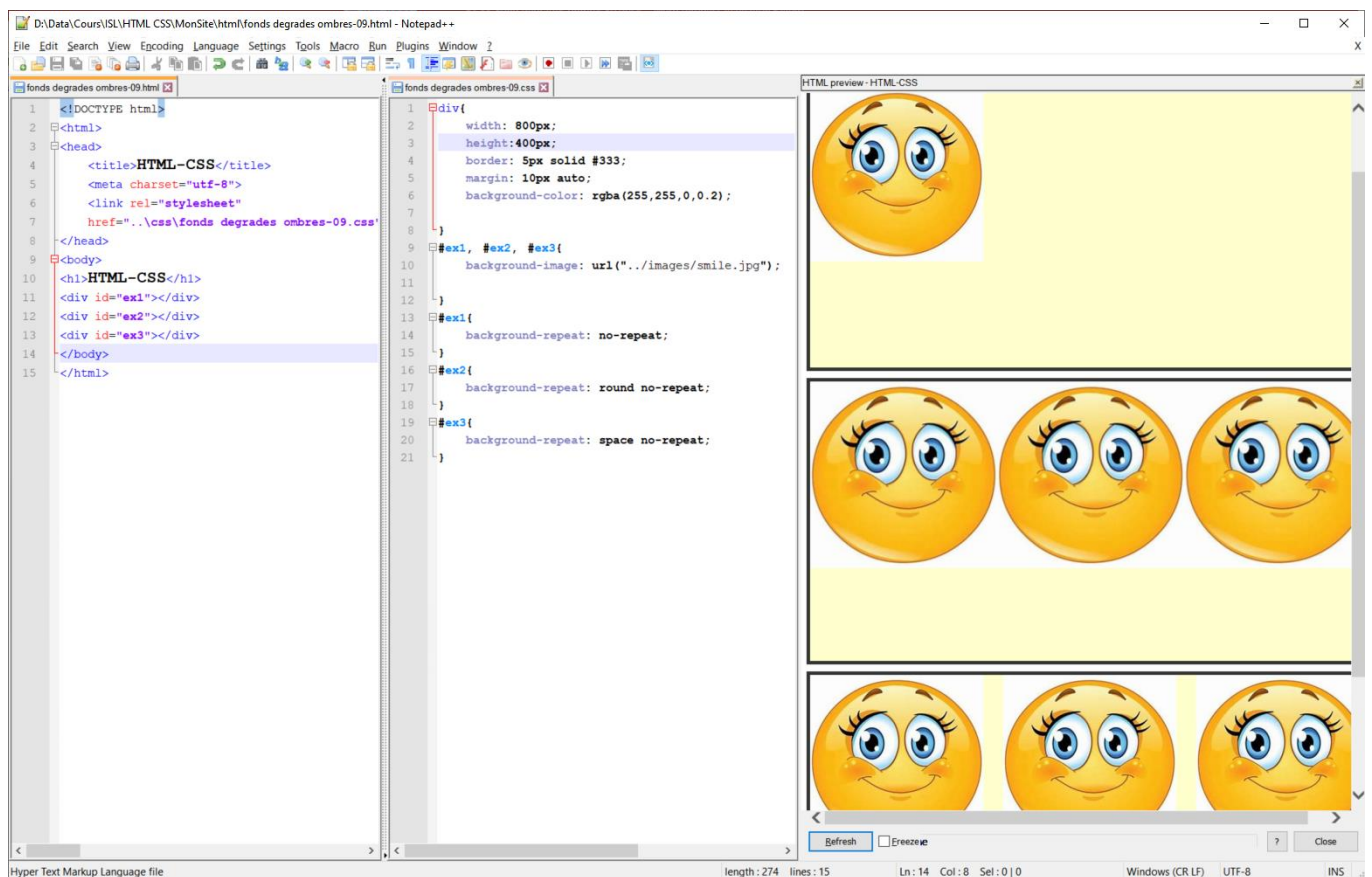


- **round** : l'image est répétée jusqu'à remplir le fond et est étirée si nécessaire de façon à n'avoir qu'un nombre de répétitions complet de l'image (l'image ne pourra pas être rognée);
- **space** : l'image est répétée jusqu'à remplir le fond un nombre complet de fois comme pour la valeur round. Cependant, ici, l'image n'est pas étirée : la première et la dernière répétition de l'image sont collées au bord de l'élément et l'espace va être distribué équitablement entre chaque répétition de l'image.

Attention cependant, les valeurs **round** et **space** ainsi que la syntaxe avec deux valeurs font partie des recommandations du CSS3 et ne sont donc à proprement parler pas encore officielles. Il est donc possible que vous n'ayez pas le comportement attendu selon le navigateur et la version utilisée.

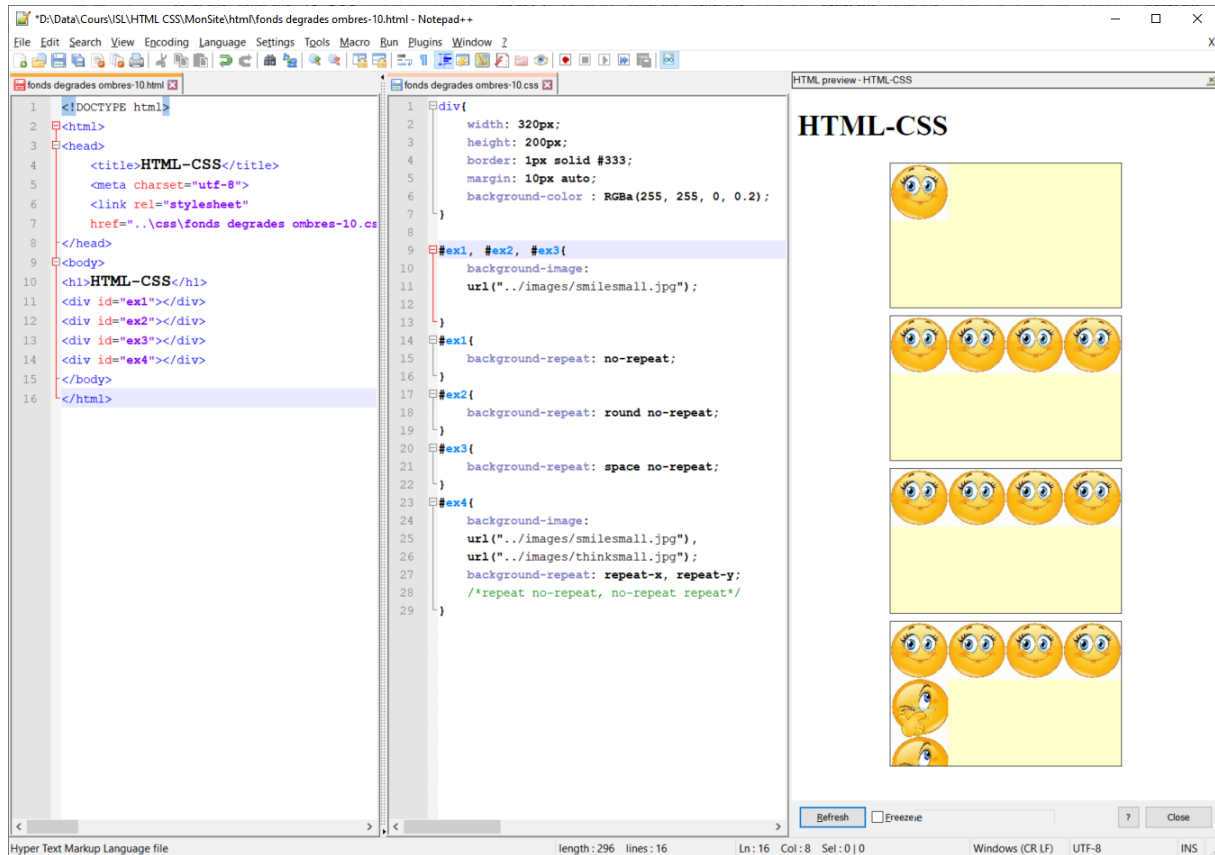
En plus de ces valeurs, il existe également deux autres valeurs qui servent seules à définir les comportements de répétition à la fois horizontal et vertical de l'image :

- **repeat-x** : l'image est répétée horizontalement jusqu'à remplir le fond mais pas verticalement. L'équivalent avec deux valeurs va être **background-repeat : repeat no-repeat ;**
- **repeat-y** : l'image est répétée verticalement jusqu'à remplir le fond mais pas horizontalement. L'équivalent avec deux valeurs va être **background-repeat : no-repeat repeat.**



Si plusieurs images de fond ont été déclarées avec **background-image** pour l'élément, alors on va pouvoir gérer le comportement de répétition de chacune d'entre elles exactement de la même

façon: en passant plusieurs valeurs à **background-repeat** séparées par des virgules. Chaque valeur va s'appliquer à l'image de fond correspondante déclarée avec **background-image**.



## Gérer la taille des images de fond avec background-size

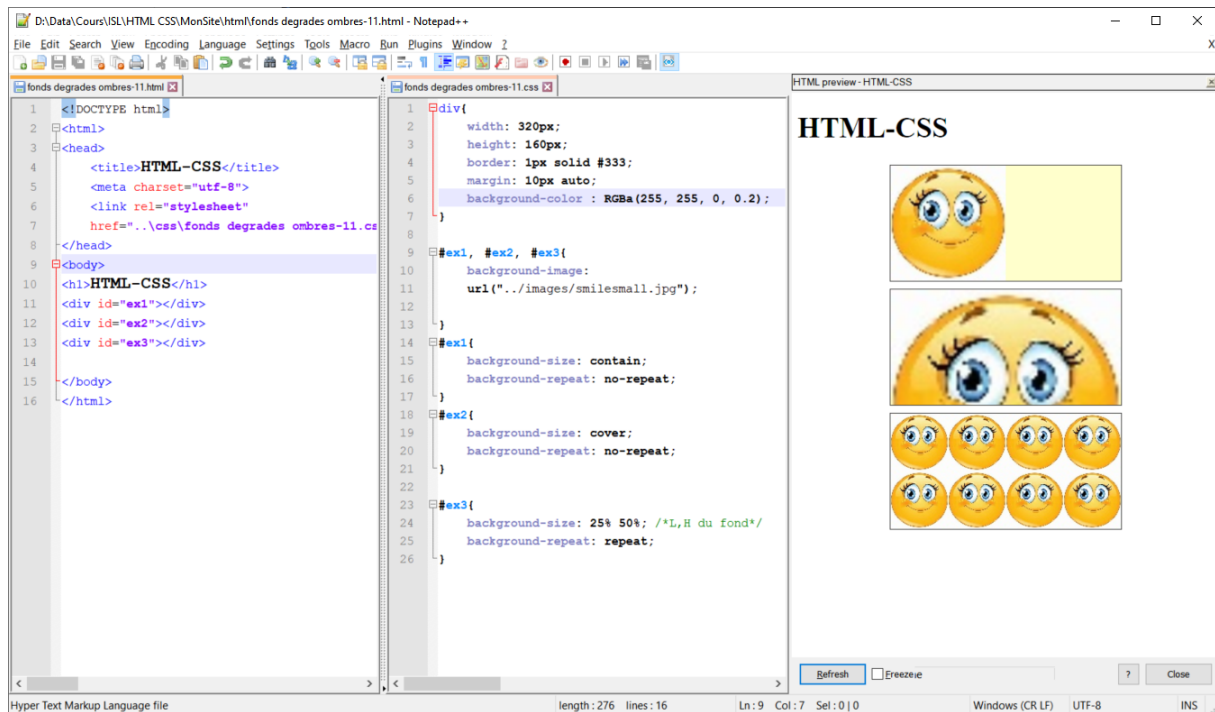
La propriété CSS **background-size** va nous permettre de gérer la taille des images de fond d'un élément. Cette propriété va pouvoir prendre une ou deux valeurs. Les valeurs vont pouvoir être des mots clefs ou des dimensions absolues ou relatives. En cas de valeur relative, la dimension est exprimée en fonction de la taille du fond (du conteneur) et non pas de la taille originale de l'image.

Si on ne fournit qu'une valeur à **background-size**, alors cette valeur servira à déterminer la largeur de l'image de fond et la hauteur sera calculée automatiquement par rapport à la largeur fournie. En passant deux valeurs à **background-size**, la première servira à déterminer la largeur de l'image tandis que la seconde imposera la hauteur de celle-ci. Attention dans ce cas aux proportions de l'image dans le rendu final !

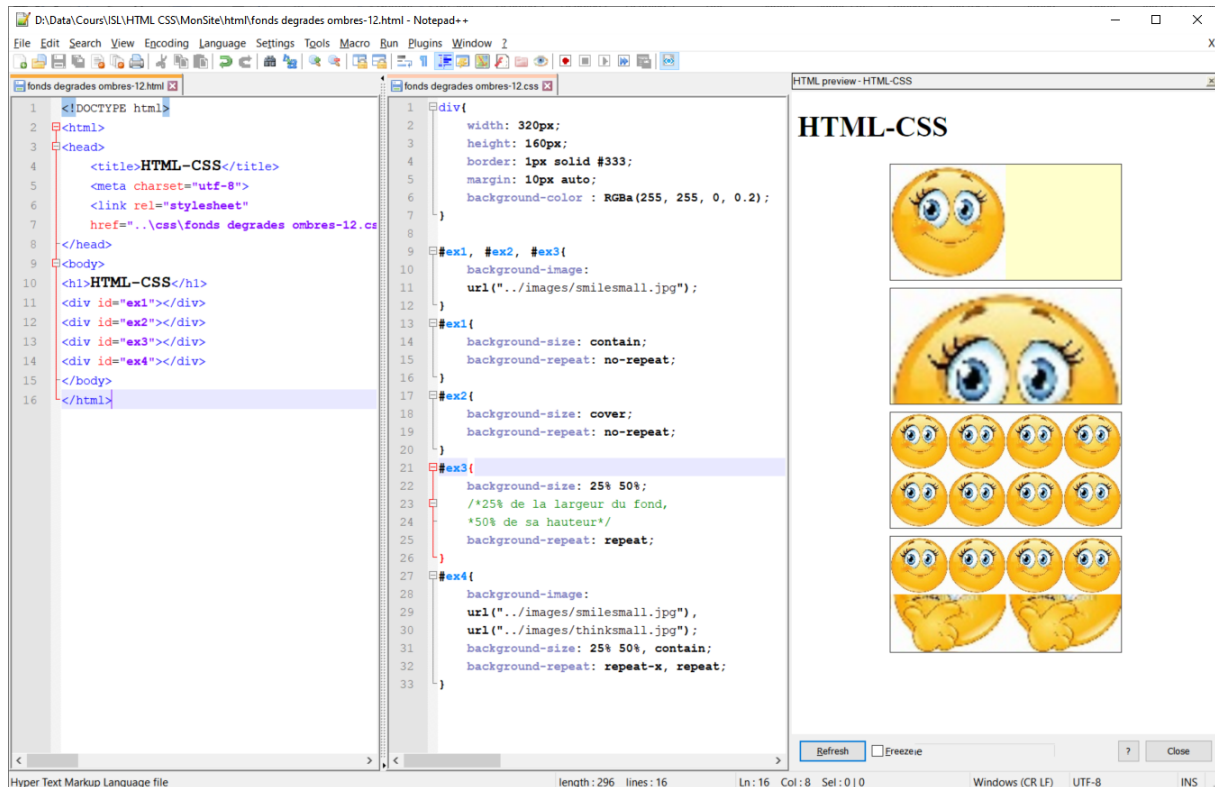
On va également pouvoir fournir l'un de ces deux mots clefs à **background-size** :

- **contain** : l'image va être redimensionnée afin qu'elle occupe le maximum de place dans le conteneur tout en conservant ses proportions de base et sans dépasser du conteneur. L'image va être contenue dans le conteneur ;

- **cover** : l'image va être redimensionnée afin de couvrir tout l'espace (en largeur) dans le conteneur tout en conservant ses proportions de base. L'image va pouvoir déborder du conteneur afin d'occuper toute la surface de fond. Les parties qui dépassent vont être rognées.



Dans le cas où plusieurs images de fond ont été attribuées avec background-image, et si l'on souhaite gérer la taille de chacune de ces images, alors il suffira une nouvelle fois d'indiquer les différentes valeurs de taille pour chaque image dans background-size en les séparant par des virgules.



Attention : La définition et les valeurs de la propriété **background-size** ne sont pour le moment que candidates au statut de recommandation et ne sont donc pas encore tout à fait officielles. Cependant, elles disposent déjà d'une très bonne prise en charge par les versions récentes des navigateurs les plus populaires.

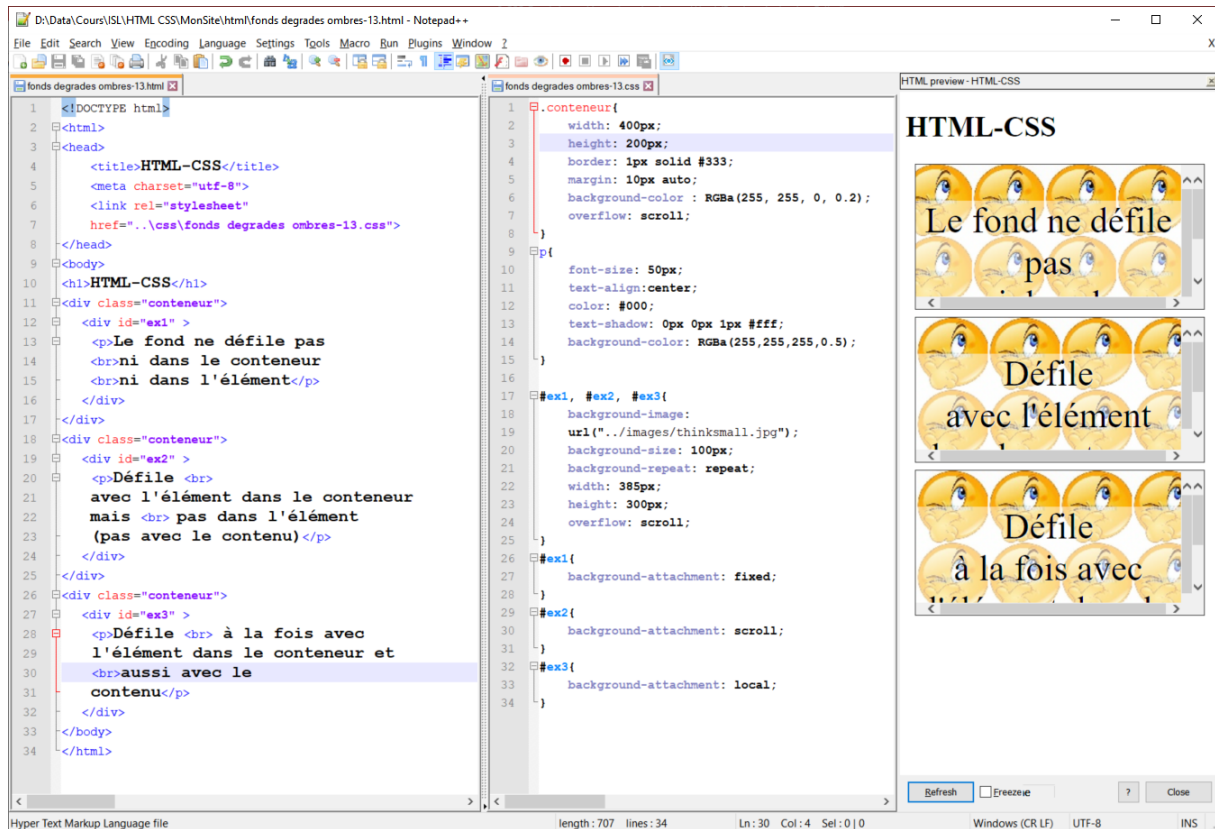
## Gérer le défilement d'une image de fond avec background-attachment

La propriété **background-attachment** va nous permettre de définir si l'image de fond doit être fixe ou défiler dans son conteneur. Cette propriété ne va donc avoir d'impact que dans le cas où nous avons une barre de défilement ou de scrolling dans notre élément.

Nous allons pouvoir passer une valeur parmi les suivantes à **background-attachment** :

- **scroll** : valeur par défaut. L'image de fond ne défile pas avec le contenu de l'élément auquel elle est associée mais défile avec l'élément dans la page ;
- **fixed** : l'image de fond restera fixe par rapport à son conteneur quelle que soit la taille de l'élément auquel elle est associée et même si l'élément en soi possède une barre de défilement ;
- **local** : l'image de fond va se déplacer avec le contenu de l'élément auquel elle est associée et défiler dans la page avec l'élément auquel elle est associée.

Regardez plutôt l'exemple suivant pour bien comprendre le fonctionnement de cette propriété (pensez à bien jouer avec chacune des deux barres de défilement) :



Comme pour les propriétés précédentes, dans le cas où on a plusieurs images de fond, il suffira de donner autant de valeurs à **background-attachment** que d'images de fond déclarées et de séparer les différentes valeurs données par des virgules.

## Déterminer la position du fond avec background-origin

La propriété **background-origin** nous permet d'ajuster la zone dans laquelle notre ou nos images de fond vont pouvoir se placer ou plus exactement de déterminer la position de l'origine (donc de son coin supérieur gauche par défaut) de l'image de fond dans son élément.

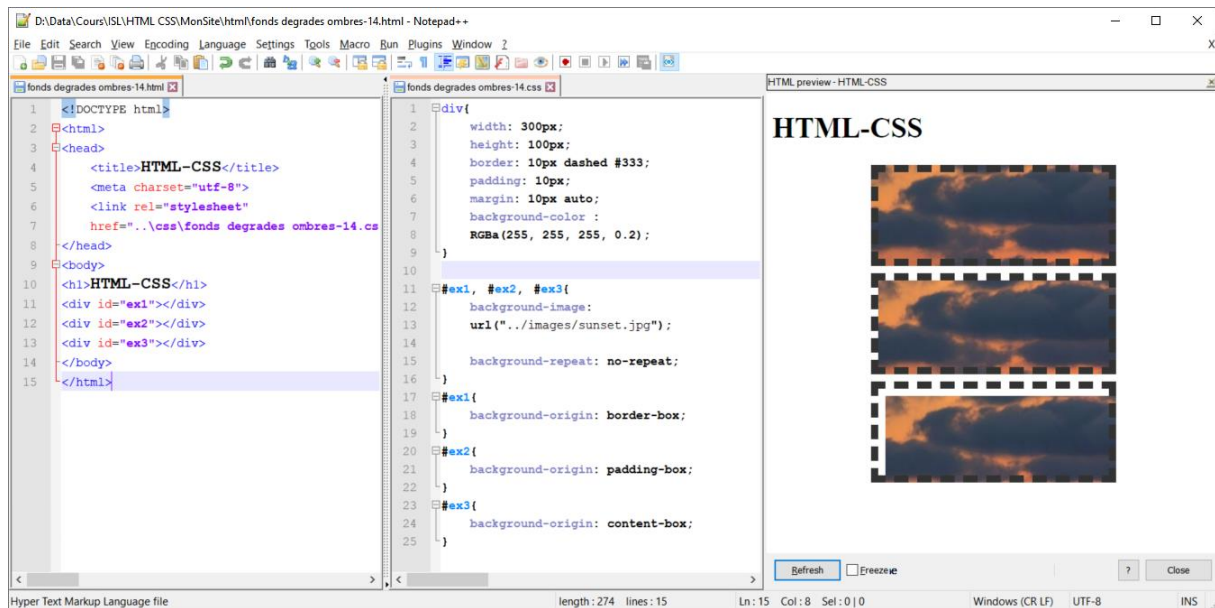
Notez que si on a appliqué un **background-attachment : fixed** à notre élément, alors la propriété **background-origin** sera tout simplement ignorée (l'image de fond sera fixe en toutes circonstances).

On va pouvoir passer l'une de ces trois valeurs à **background-origin** :

- **padding-box** : valeur par défaut. La position de l'origine de l'image va être déterminée par rapport à la boîte représentant le contenu + marges internes de l'élément. Le point d'origine de l'image est donc à l'extérieur des bordures ;
- **border-box** : la position de l'origine de l'image de fond va être déterminée par rapport à la bordure de l'élément. Le coin supérieur gauche de l'image va donc se situer derrière la bordure supérieure gauche par défaut ;



- **content-box** : la position de l'origine de l'image va être déterminée par rapport à la boîte représentant le contenu de l'élément. Elle ne va donc pas recouvrir l'espace supérieur gauche occupé par le **padding** par défaut.



Comme précédemment, si plusieurs images de fond ont été déclarées avec **background-image**, il suffira de passer une valeur pour chaque image à **background-origin** en séparant les différentes valeurs par une virgule.

## Positionner une image de fond avec **background-position**

La propriété CSS **background-position** va nous permettre de définir à partir de quelle position une image de fond doit s'afficher dans la surface de fond de l'élément associé et relativement à la surface de la zone de fond qu'elle peut occuper (qui a été définie avec **background-origin**).

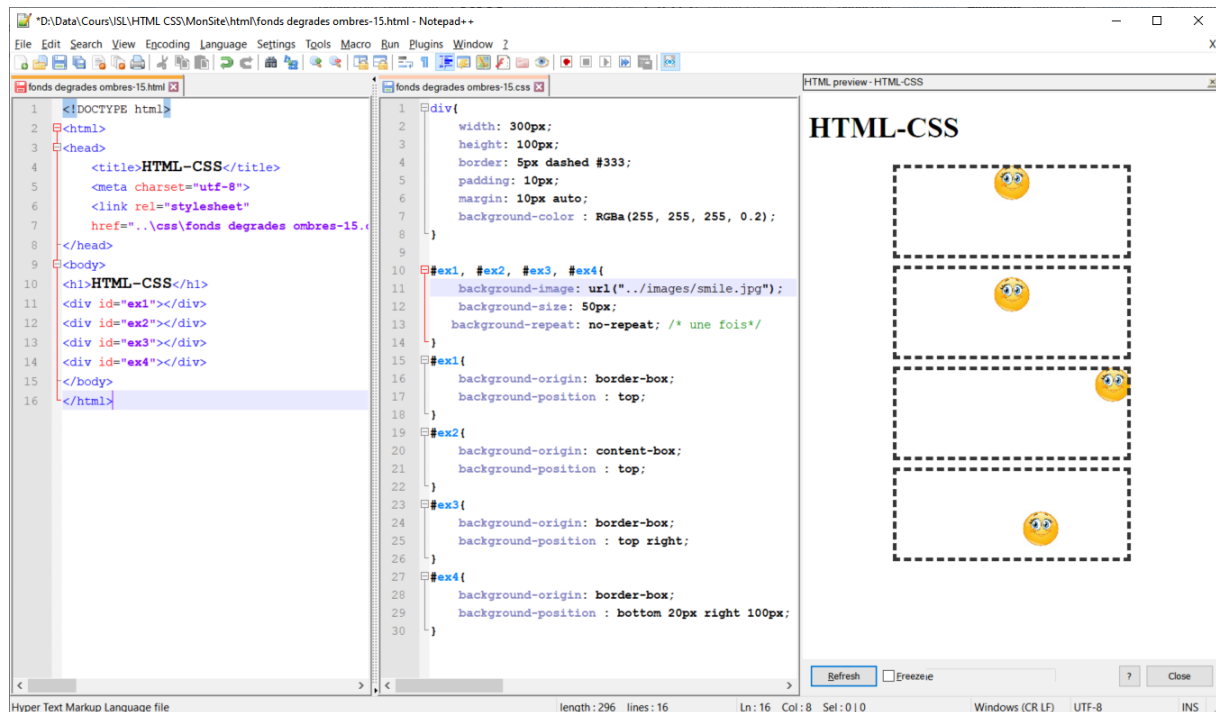
Cette propriété va pouvoir prendre des mots clefs en valeur comme **top**, **right**, **bottom**, **left** ou **center** ou des distances relatives ou absolues comme **25%** ou **50px** ou un mélange des deux comme **bottom 50px** qui indique que le bord bas de l'image de fond sera situé à **50px** du « bas de l'élément » (selon la zone de fond définie avec **background-origin** une nouvelle fois).

On va pouvoir passer une ou deux valeurs (simples comme **bottom** ou composées comme **bottom 50px**) à **background-position**. En ne passant qu'une valeur :

- **background-position : center** va centrer l'image de fond ;
- Les autres mots clefs vont coller l'image de fond au bord correspondant au mot clef et la deuxième position (horizontale ou verticale) sera par défaut déterminée à 50% de la taille de l'élément. Le point de départ de l'affichage de l'image de fond (dans le cas où elle serait plus grande que l'élément associé) sera déterminé par le mot clef ;

- Une longueur absolue ou relative va déterminer la position de l'image de fond par rapport au bord gauche de l'élément. La position verticale calculée sera 50% de la hauteur de l'élément. Le point de départ de l'image sera également son bord gauche.

En passant deux valeurs à **background-position**, la première valeur déterminera le point d'origine de l'image sur l'axe horizontal tandis que la seconde valeur déterminera le point d'origine de l'image dans l'axe vertical.



Une nouvelle fois, dans le cas où l'image de fond est plus grande que son élément associé, le point de départ d'affichage de l'image sera son bord gauche si on passe une longueur ou sera déterminé par le mot clé passé.

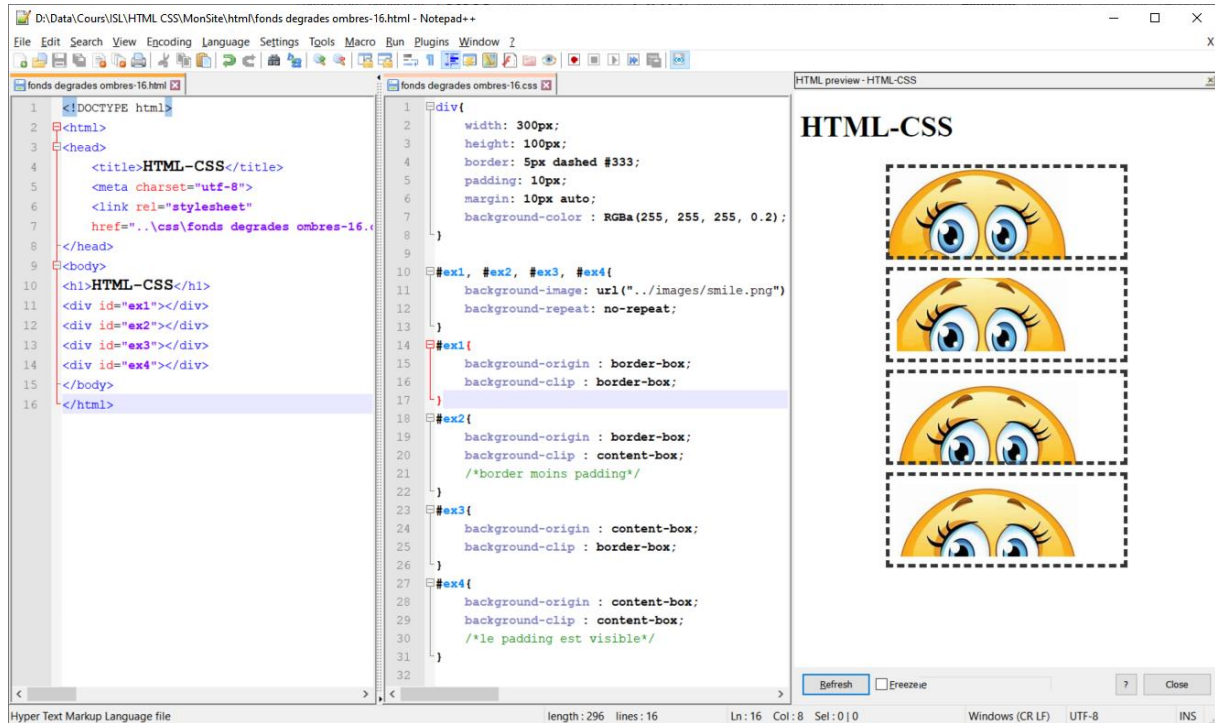
## Déterminer la surface que peut occuper notre fond avec background-clip

La propriété **background-clip** permet de définir la surface du fond de l'élément dans laquelle l'image de fond va être visible ou pas.

Cette propriété va pouvoir prendre les mêmes valeurs que **background-origin** à savoir :

- **border-box** : valeur par défaut. L'image de fond sera visible jusque sous les bordures de l'élément (on pourra la voir dans le cas d'une bordure en pointillés par exemple) ;
- **padding-box** : l'image de fond sera visible jusqu'à la limite de la marge intérieure (padding) mais la partie sous les bordures de l'élément ne le sera pas ;
- **content-box** : l'image de fond ne sera visible que dans l'espace relatif à la boîte entourant le contenu de l'élément.

Il convient cependant de ne pas confondre les propriétés **background-clip** et **background-origin** : la propriété **background-origin** sert à déterminer le point d'origine de l'image de fond, c'est-à-dire à partir d'où celle-ci va être placée tandis que **background-clip** va nous permettre de définir quelle partie du fond va être affichée / visible.



Comme vous pouvez le voir ici, le point d'origine de notre image de fond défini par la propriété **background-origin** se situe à l'extrémité des bordures de l'élément pour les div **id="ex1"** et **div id="ex2"**. Pour le div **id="ex2"**, cependant, on impose le fait que la partie du fond située entre la boîte relative au contenu et la l'extrémité de la bordure ne soit pas affichée avec **background-clip : content-box**.

## La propriété CSS raccourcie background

La propriété CSS **background** est la notation short hand ou notation raccourcie des propriétés liées au fond. En CSS 2, la propriété **background** permettait de définir une couleur et / ou une image de fond pour un élément HTML ainsi que le comportement de répétition, le défilement et la position du fond.

La propriété **background** était donc une propriété condensée des propriétés **background-color**, **background-image**, **background-repeat**, **background-attachment** et **background-position**.

En CSS 3, ses fonctionnalités ont été étendues pour refléter les apports du CSS 3 en termes d'outil de gestion du fond et la propriété **background** dans sa nouvelle définition permet aujourd'hui également de gérer plusieurs fonds d'un coup ainsi que de définir les comportements relatifs aux propriétés **background-size**, **background-origin** et **background-clip** pour un fond.



Attention cependant : même si la plupart des navigateurs supportent parfaitement ces ajouts, je vous rappelle que ce module du CSS 3 n'est pas encore reconnu comme recommandation officielle par le W3C mais est seulement candidat au statut de recommandation et est donc potentiellement sujet à modifications (même si elles sont peu probables).

L'ordre de déclaration des valeurs de la propriété **background** va être le suivant :

1. La valeur relative à la propriété **background-image** ;
2. Les valeurs relatives aux propriétés **background-position** / **background-size** (avec le slash entre les deux valeurs);
3. La valeur relative à la propriété **background-repeat** ;
4. La valeur relative à la propriété **background-attachment** ;
5. La valeur relative à la propriété **background-origin** ;
6. La valeur relative à la propriété **background-clip** ;
7. La valeur relative à la propriété **background-color**.

Notez que si certaines valeurs sont omises, alors les valeurs par défaut des différentes propriétés seront utilisées. Pour rappel, les valeurs par défaut de background vont être : **background : none 0% 0% / auto auto repeat scroll padding-box padding-box transparent**.

Pour déclarer plusieurs fonds, il suffira une nouvelle fois de séparer les déclarations complètes par une virgule. Attention cependant, pour que la propriété **background** fonctionne correctement avec des fonds multiples il faudra absolument réserver la valeur relative à la couleur de fond (**background-color**) pour le dernier fond déclaré.



# Création de dégradés linéaires en CSS avec linear-gradient

En CSS, un dégradé est considéré comme une image qui va progressivement passer d'une couleur de base à une couleur d'arrivée.

Ainsi, nous allons pouvoir créer et utiliser les dégradés en CSS avec toutes les propriétés qui acceptent des images et notamment créer des dégradés en fond de nos éléments en utilisant la propriété **background**.

Il existe deux types de dégradés en CSS :

- Les dégradés linéaires ou **linear-gradient** en anglais dont le principe va être de passer progressivement d'une couleur à une autre de manière linéaire c'est-à-dire selon un axe donné ;
- Les dégradés radiaux ou **radial-gradient** en anglais pour lesquels le passage d'une couleur à une autre va se faire dans toutes les directions à partir d'un point central.

Dans cette leçon, nous allons nous intéresser à la création de dégradés linéaires seulement. Nous verrons comment créer des dégradés radiaux dans la leçon suivante.

## Qu'est-ce qu'un dégradé linéaire ?

Un dégradé linéaire est un dégradé qui va se faire selon un axe ou une direction unique.

Nous allons pouvoir créer un dégradé linéaire en CSS en utilisant la syntaxe `linear-gradient()` en valeur d'une propriété acceptant des images comme `background` par exemple.

Voici un premier exemple de dégradé linéaire :



## Direction et couleurs du dégradé

Pour créer un dégradé linéaire de manière effective, il va falloir préciser un axe ou une direction pour notre dégradé ainsi qu'au moins deux couleurs et éventuellement définir des « color stops » qui vont nous permettre d'indiquer qu'à un certain point le dégradé doit passer par une couleur donnée.

## Définition de la direction d'un dégradé linéaire

Nous allons pouvoir préciser la direction du dégradé de deux manières différentes : soit en utilisant une notation sous forme d'angle en degrés **deg**, soit avec les mots clefs **to top**, **to right**, **to bottom**, **to left** ou les combinaisons **to bottom right**, **to bottom left**, etc.

- Un angle de **0deg** indique que le dégradé se fera selon l'axe vertical à partir du bas vers le haut. L'équivalent de **0deg** sous forme de mot clef va être **to top** ;
- Un angle de **90deg** signifie que le dégradé se fera selon l'axe horizontal de la gauche vers la droite. L'équivalent de **90deg** sous forme de mot clef va être **to right** ;
- Un angle de **180deg** indique que le dégradé se fera selon l'axe vertical à partir du haut vers le bas. L'équivalent de **180deg** sous forme de mot clef va être **to bottom** ;
- Un angle de **270deg** signifie que le dégradé se fera selon l'axe horizontal de la droite vers la gauche. L'équivalent de **270deg** sous forme de mot clef va être **to left** ;

De même, la valeur **to top right** va être équivalente à **45deg** : le dégradé partira d'en bas à gauche de l'élément pour aller en haut à droite et etc.

A noter : Utiliser des notations sous forme d'angles en degrés va nous permettre d'ajuster la direction de nos dégradés de manière beaucoup plus précise qu'en utilisant les mots clefs.

Notez que si on ne précise pas de direction pour un dégradé linéaire, alors la direction du dégradé sera par défaut verticale du haut vers le bas c'est-à-dire **to bottom** ou encore **180deg**.

## Les "color stops" et l'avancement d'un dégradé

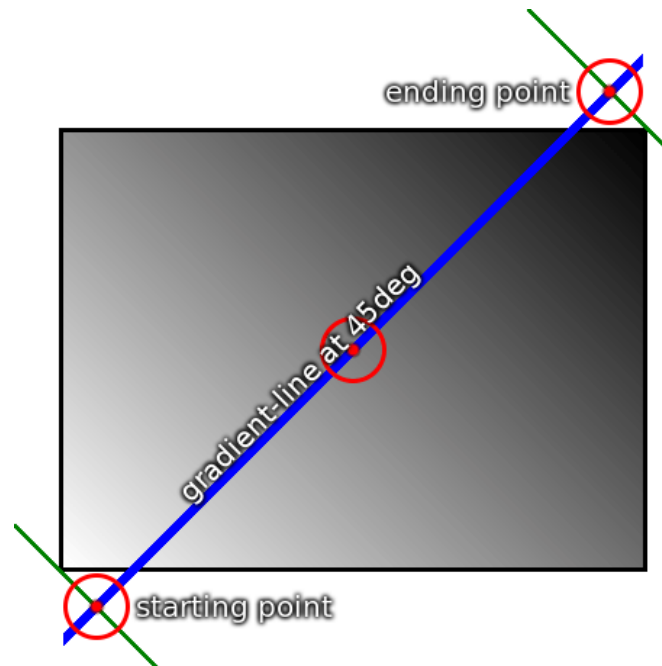
Les « color stops » vont être des points que nous allons définir au niveau desquels notre dégradé doit arriver à une couleur en particulier. Les « color stops » seront généralement situés entre le point de départ et le point d'arrivée de notre dégradé mais cela n'est pas obligatoire (on peut tout à fait définir un color stop en dehors de notre dégradé).

Pour définir un color stop, nous allons attribuer un pourcentage à une couleur. Ce pourcentage représente un niveau d'avancement du dégradé pour lequel il doit arriver à la couleur correspondante.

En effet, un dégradé linéaire se fait selon une direction. Vous pouvez imaginer cette direction sous forme d'une droite.

Le point de départ de notre dégradé linéaire ou le 0% va se trouver au point où la droite perpendiculaire à la direction du dégradé touche l'extrémité de la boîte représentant l'élément dans lequel on crée notre dégradé et de manière identique pour le point d'arrivée de notre dégradé de l'autre côté de la boîte.

Regardez plutôt le schéma suivant pour bien comprendre cela :

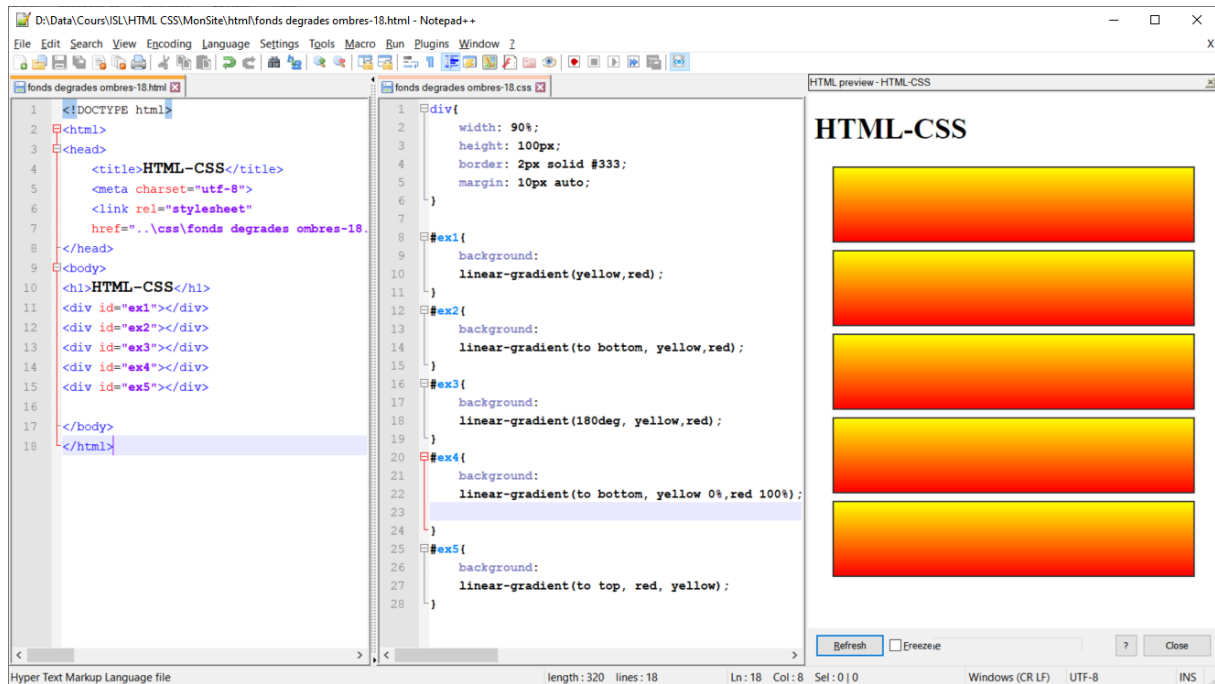


## Exemples de création de dégradés linéaires

Maintenant que nous avons vu la théorie, il est temps de passer à la pratique et de créer nos premiers dégradés linéaires afin de voir comment cela va fonctionner en pratique. Dans les exemples suivants, je vais créer des dégradés linéaires en images de fond de mes éléments.

### Création de dégradés linéaires simples

Commençons avec quelques exemples simples. Essayez de bien retenir la syntaxe des exemples suivants et notamment quand utiliser des virgules ou pas (pour faire simple, nous n'utilisons les virgules que pour séparer les couleurs entre elles et les séparer des autres valeurs à part des color stops qui leur sont associés).



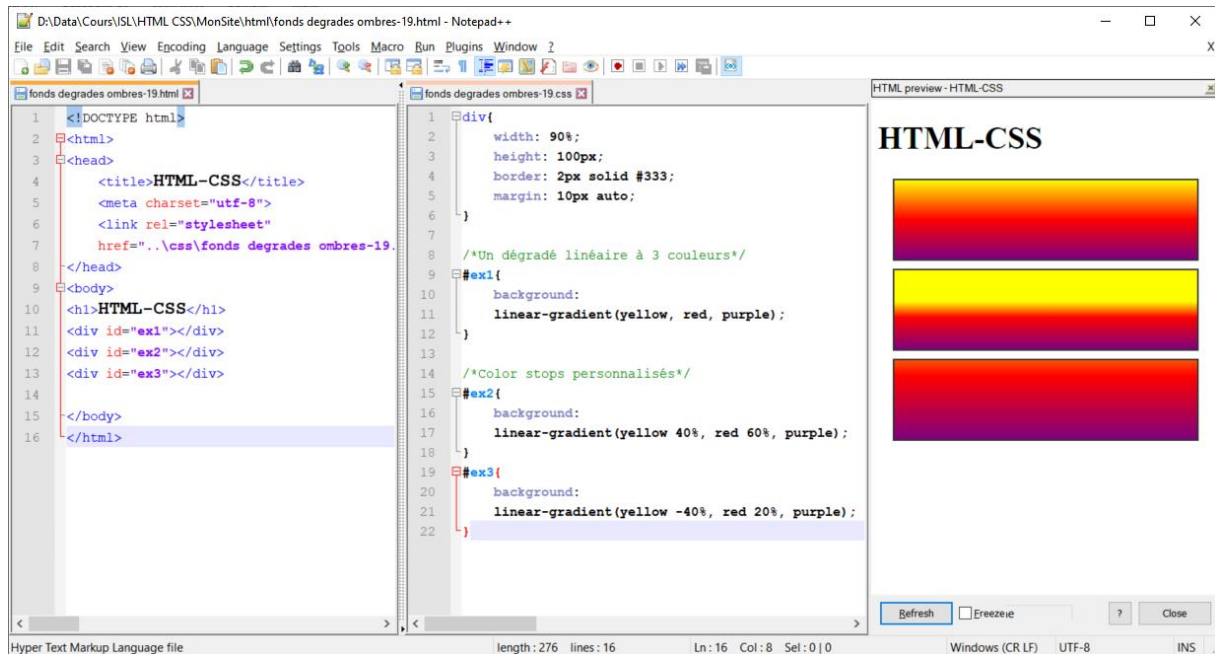
Ici, mes cinq dégradés sont strictement identiques. En effet, je crée un dégradé en image de fond avec la propriété **background** et la notation **linear-gradient** pour mon premier **div id="ex1"** en renseignant une couleur de départ, le jaune (yellow), et une couleur d'arrivée, le rouge (red). Dans ce premier exemple, je n'indique ni direction ni color stop. Les valeurs par défaut vont donc être utilisées à savoir **to bottom** ou **180deg** pour la direction et **0%** en color stop pour ma couleur de départ et **100%** pour ma couleur d'arrivée.

Dans les dégradés suivants, je me contente de renseigner des valeurs qui sont ici superflues puisque ces valeurs sont les valeurs par défaut.

Notez que dans le cinquième exemple j'inverse à la fois la direction du dégradé et les couleurs de départ et d'arrivée ce qui fait que nous nous retrouvons une nouvelle fois avec le même dégradé !

## Création de dégradés avec direction et color stops

Essayons maintenant de créer des dégradés linéaires avec plus de deux couleurs et en définissant des directions et color stops différents :



Dans le premier exemple, on définit un dégradé linéaire à trois couleurs : du jaune vers le rouge puis vers le violet sans préciser de color stop. La couleur jaune sera ainsi par défaut atteinte à 0% du dégradé puis le rouge à 50% et enfin le violet à 100% puisque les color stops sont distribués équitablement par défaut (dans le cas de 4 couleurs, les color stops par défaut seraient 0% 33.33% 66.66% 100%, dans le cas d'un dégradé à 5 couleurs on aurait 0% 25% 50% 75% 100% et etc.).

Dans les deuxième et troisième exemples, en revanche, on définit des color stop personnalisées. Dans le deuxième exemple, on demande à ce que la couleur jaune soit atteinte à 40% du dégradé. Comme le jaune est également la couleur de départ, notre dégradé sera jaune entre 0 et 40%.

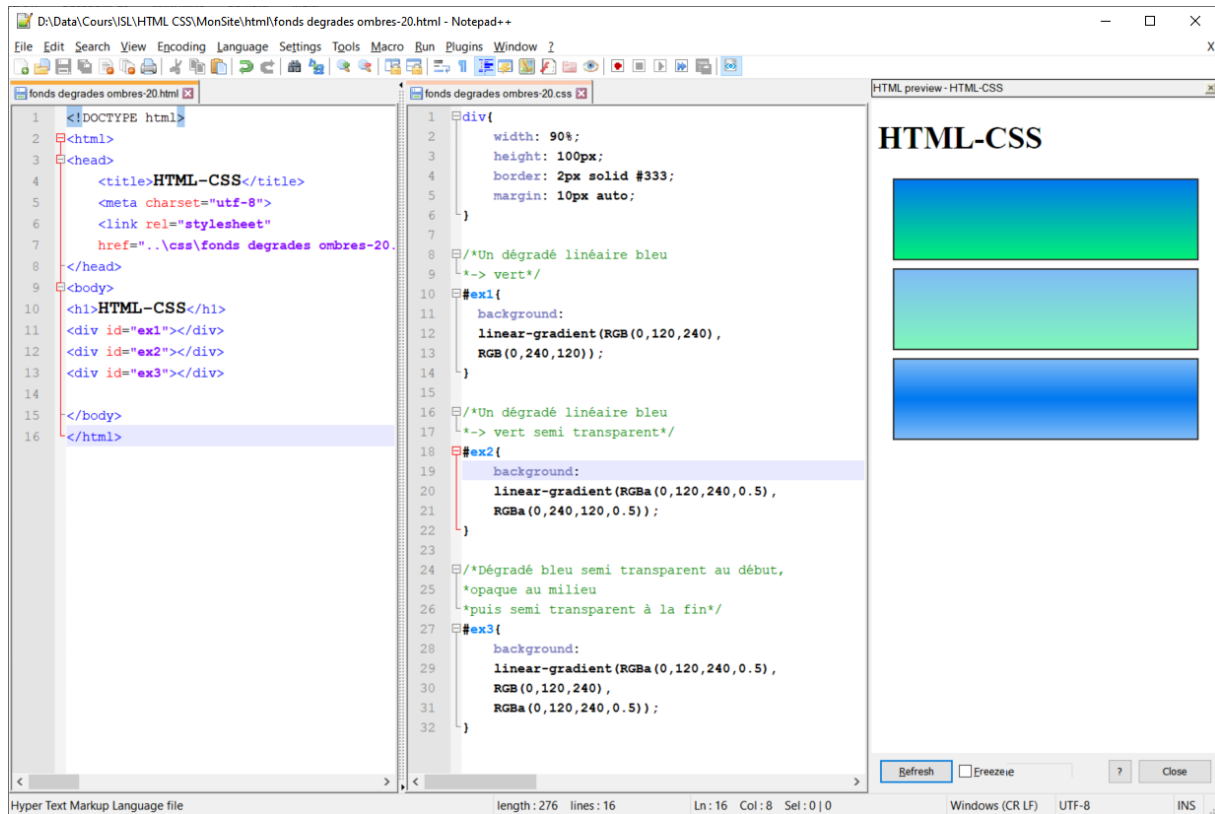
Ensuite, on demande à ce que la couleur rouge soit atteinte à 60%. Il va donc y avoir dégradé du jaune vers le rouge entre 40% et 60% de notre dégradé puis ensuite du rouge vers le violet entre 60% et 100%.

Dans le troisième exemple, on utilise un color stop en dehors de notre de la boîte. On demande à ce que notre dégradé arrive à la couleur jaune à -40%. Cela signifie que si notre ligne de dégradé dans la boîte de l'élément fait 100px de long, on souhaite atteindre le jaune 40px avant notre 0%.

Ensuite on demande à arriver au rouge à 20%. La transition du jaune vers le rouge va donc se faire entre -40% et 20% et donc nous n'allons pas voir le jaune en soi puisque tous les color stops inférieurs à 0% ou supérieurs à 100% se situent en dehors de la boîte représentant l'élément dans lequel on crée notre dégradé.

### Création de dégradés semi-transparents

Nous pouvons tout à fait définir des dégradés semi-transparents en utilisant des notations RGBa ou HSLa pour préciser les couleurs de nos dégradés. Notez par ailleurs qu'on peut très bien définir le début d'un dégradé avec une couleur semi transparente et la fin de celui-ci avec une couleur opaque ou inversement.

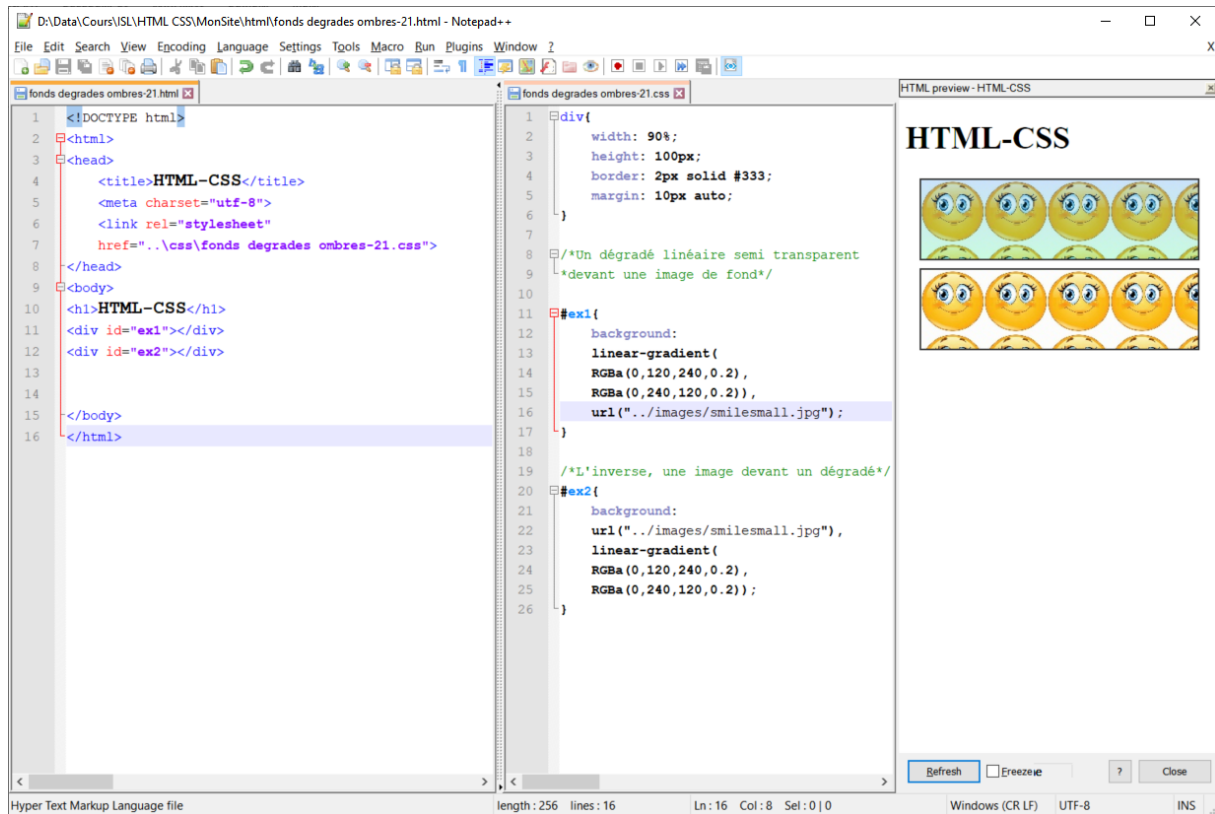


Dans l'exemple ci-dessus, on crée trois dégradés : un premier dégradé complètement opaque, un deuxième dégradé avec des couleurs semi transparentes en utilisant des notations RGBa et un troisième dégradé semi transparent au début, opaque au milieu puis à nouveau semi transparent à la fin.

Définir des dégradés semi-transparents va être véritablement intéressant dans le cas où on souhaite simplement ajouter un effet de couleur par-dessus une image de fond par exemple. En effet, rappelez-vous qu'un dégradé est considéré comme une image en CSS et que la propriété `background` supporte tout à fait plusieurs images de fond.

Ici, il va bien falloir faire attention à l'ordre des déclarations : je vous rappelle que le premier fond déclaré sera toujours au-dessus du deuxième, qui sera lui-même au-dessus du troisième et etc. Ainsi, en déclarant notre dégradé en fond avant notre image, nous allons pouvoir ajouter des couleurs devant notre image de fond.





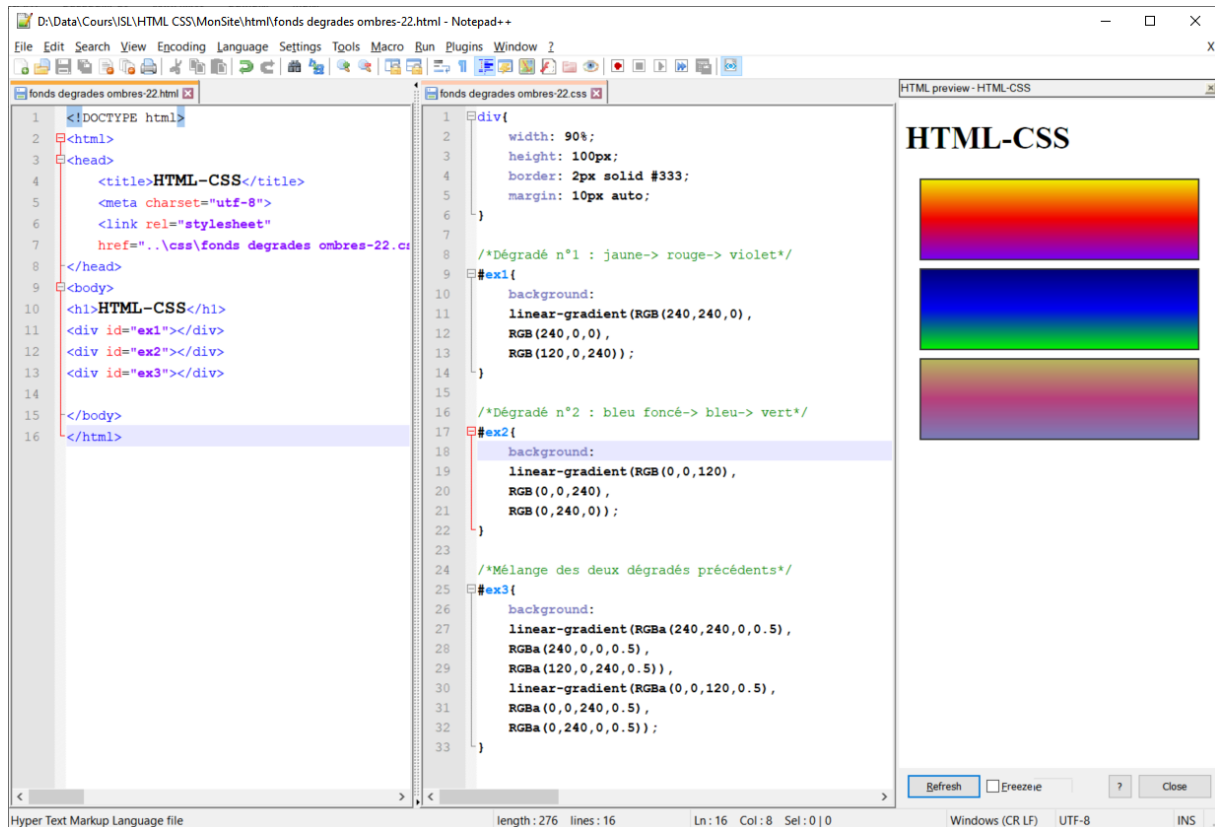
## Créer plusieurs dégradés en fond d'un élément

Nous venons de voir qu'on pouvait tout à fait placer un dégradé et une image en fond d'un même élément. De la même façon, on va pouvoir déclarer plusieurs dégradés en fond d'un élément.

Pour affecter plusieurs dégradés linéaires à notre **background**, il va suffire de séparer les différentes déclarations par une virgule de la même façon qu'on séparait les images de fond.

Le premier dégradé déclaré sera au-dessus du deuxième qui sera lui-même au-dessus du troisième et etc., chaque nouvelle déclaration créant une nouvelle couche de fond pour l'élément concerné. Bien évidemment, si les dégradés sont opaques, alors seul le premier déclaré sera visible (il cachera totalement les autres).

En revanche, si le ou les dégradés du dessus sont semi transparents, alors on va pouvoir voir également les dégradés des couches plus profondes et les couleurs des différents dégradés vont se mélanger ou « fusionner » entre elles.

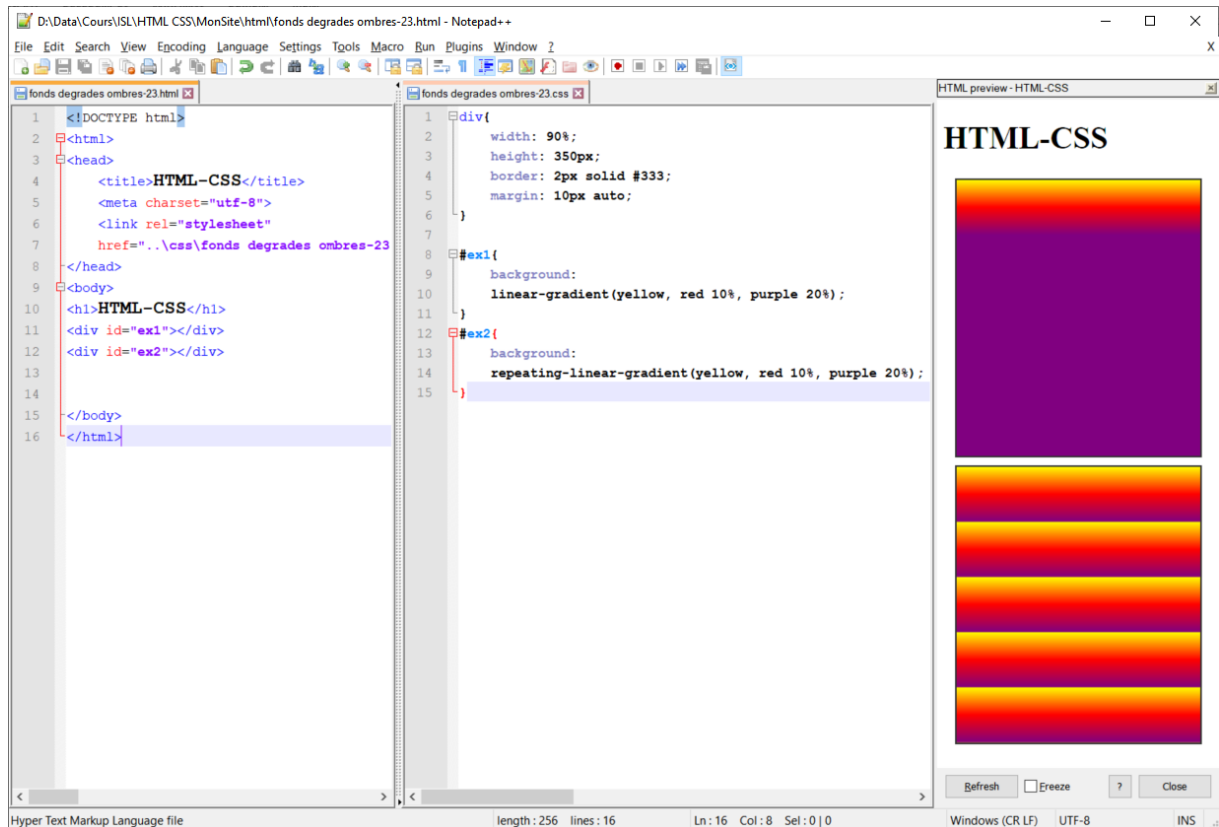


## La répétition des dégradés

Nous avons vu comment créer des dégradés **uniques** linéaires avec la fonction **linear-gradient()**.

Le CSS nous offre également la possibilité de créer des dégradés qui vont pouvoir se répéter grâce en utilisant cette fois-ci plutôt la fonction **repeating-linear-gradient()**.

Bien évidemment, pour voir l'effet d'un dégradé répété il faudra régler les color stops de façon à ce que la première itération du dégradé n'occupe pas 100% de l'espace de base.



Ici, vous pouvez voir que la transition entre la première et la deuxième itération du dégradé est brutale lors de l'utilisation de `repeating-linear-gradient()`.

Ce comportement est tout à fait normal puisqu'ici le color stop relatif au violet est en fin de dégradé et le color stop du jaune est au tout début. On va donc passer directement du violet au jaune sans transition fluide.

*Si on souhaite créer une transition plus fluide entre les différentes itérations de notre dégradé, il faudra créer le dégradé de manière à ce qu'il boucle sur lui-même c'est-à-dire qu'il faudra répéter la couleur de début de dégradé en fin de dégradé.*

D:\Data\Cours\ISL\HTML CSS\MonSite\html\fonds degradés ombres-24.html - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window 2

fonds degradés ombres-24.html

```

1 :!DOCTYPE html>
2 <html>
3 <head>
4 <title>HTML-CSS</title>
5 <meta charset="utf-8">
6 <link rel="stylesheet"
7 href="..\css\fonds degradés ombres-24.css">
8 </head>
9 <body>
10 <h1>HTML-CSS</h1>
11 <div id="ex1"></div>
12 <div id="ex2"></div>
13 </body>
14 </html>

```

fonds degradés ombres-24.css

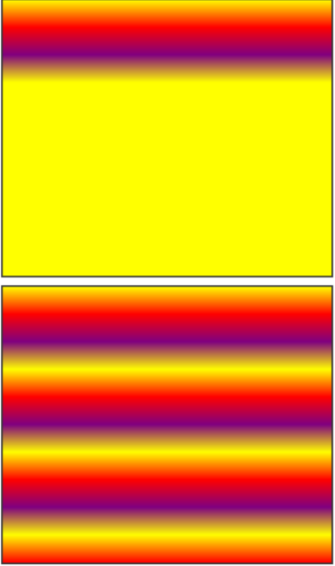
```

1 div{
2 width: 90%;
3 height: 350px;
4 border: 2px solid #333;
5 margin: 10px auto;
6 }
7
8 #ex1{
9 background: linear-gradient(
10 yellow, red 10%, purple 20%, yellow 30%);
11 }
12 #ex2{
13 background:
14 repeating-linear-gradient(
15 yellow, red 10%, purple 20%, yellow 30%);
16 }

```

HTML preview - HTML-CSS

## HTML-CSS



Refresh Freeze ? Close

Hyper Text Markup Language file length: 256 lines: 16 Ln: 16 Col: 8 Sel: 0 | 0 Windows (CR LF) UTF-8 INS

# Création de dégradés radiaux en CSS avec radial-gradient

En CSS, un dégradé est considéré comme une image qui va progressivement passer d'une couleur de base à une couleur d'arrivée.

Ainsi, nous allons pouvoir créer et utiliser les dégradés en CSS avec toutes les propriétés qui acceptent des images et notamment créer des dégradés en fond de nos éléments en utilisant la propriété **background**.

Nous allons pouvoir créer deux types de dégradés en CSS : des dégradés linéaires et des dégradés radiaux. Dans cette nouvelle leçon, nous allons apprendre à créer des dégradés radiaux.

## Qu'est-ce qu'un dégradé radial ?

Un dégradé radial est un dégradé qui va partir d'un point central et se propager dans toutes les directions à partir de ce point.

Pour créer un dégradé radial, nous allons utiliser la notation **radial-gradient()** en valeur d'une propriété CSS acceptant des images comme **background** par exemple.

Voici un premier exemple de dégradé radial :



## Comment se construit un dégradé radial en CSS

Pour créer un dégradé radial ou **radial-gradient**, nous allons devoir définir le **point central** ou point de départ de notre dégradé, les **couleurs du dégradé** ainsi que la **taille** de la **forme finale** de celui-ci. Nous allons également pouvoir fournir des color stops de la même façon qu'avec nos dégradés linéaires.

Le point d'origine ou point central du dégradé

Nous avons deux méthodes pour définir le point d'origine ou point central de notre **radial-gradient** : nous allons soit pouvoir utiliser des mots clefs, soit des valeurs absolues ou relatives. A noter que si on omet de préciser la position de départ pour notre dégradé radial, alors la position par défaut **center** sera utilisée.

Les mots clefs vont pouvoir être les mêmes qu'avec nos dégradés linéaires, à savoir **top**, **right**, **bottom**, **left** et **center** mais devons être cette fois-ci être précédés du mot **at**.

Pour préciser les coordonnées d'un point, il faut normalement une valeur dans l'axe horizontal et une autre dans l'axe vertical et c'est la raison pour laquelle nous utiliserons souvent des mots clefs composées comme **at top left**. Si on ne précise que la valeur d'un axe (at top ou at left par exemple), alors la deuxième valeur calculée sera par défaut **center**.

Si on choisit d'utiliser des valeurs absolues ou relatives pour déterminer le point de départ de notre dégradé radial, alors il suffit de se rappeler que ces valeurs vont représenter l'écartement du point central du dégradé par rapport au coin supérieur gauche de son élément. Dans le cas où une seule valeur est passée, elle représentera l'écartement par rapport au bord gauche et par rapport au bord supérieur.

Si deux valeurs sont passées, la première représente l'écartement du point central par rapport au bord gauche tandis que la seconde représente l'écartement du point central du dégradé radial par rapport au coin supérieur de l'élément dans lequel on le crée.

## La forme finale du dégradé

Pour définir la forme finale d'un dégradé radial, nous allons pouvoir utiliser les mots clefs **circle** (le dégradé aura une forme de cercle) soit **ellipse** (le dégradé sera une ellipse). Si aucune valeur n'est précisée, le dégradé sera un cercle dans le cas où une seule dimension a été précisée pour sa taille ou une ellipse dans le cas contraire.

## La taille du dégradé

Nous allons également pouvoir indiquer la **taille** totale de notre dégradé radial soit de manière explicite en utilisant des valeurs soit en utilisant des mots clefs. La « taille du dégradé » correspond à la distance (ou aux distances) entre son point central et l'endroit où l'on va arriver à la dernière couleur déclarée dans le dégradé. Le reste de l'élément si le dégradé en soi possède une taille plus petite que lui sera rempli avec cette dernière couleur.

En ne passant qu'une valeur explicite, notre dégradé aura de fait la forme d'un cercle et la valeur passée sera la taille du rayon du dégradé. Si deux valeurs sont passées, la première représentera le « rayon » de l'ellipse dans l'axe horizontal tandis que la seconde représentera le « rayon » de l'ellipse dans l'axe vertical. Bien évidemment, on évitera de déclarer notre dégradé comme cercle et de passer deux valeurs différentes en taille puisque cela n'aurait aucun sens.

Les mots clefs relatifs à la gestion de la taille d'un dégradé radial à notre disposition sont les suivants:

- **closest-side** : Le dégradé va stopper dès qu'il va rencontrer le premier bord (le bord le plus proche) de l'élément auquel il est appliqué s'il a une forme de cercle. Si il a une forme d'ellipse, alors il s'arrêtera au moment où il va toucher les deux côtés les plus proches dans les deux axes horizontal et vertical (il les touchera toujours en même temps par définition) ;
- **farthest-side** : Le principe est le même que pour **closest-side**, à la différence que le dégradé va stopper lorsqu'il touchera le bord le plus lointain de l'élément auquel on l'applique cette fois-ci ;
- **closest-corner** : Le dégradé va stopper dès qu'il va rencontrer le premier coin (le coin le plus proche) de l'élément auquel il est appliqué ;

- **farthest-corner** : Même principe que pour **closest-corner** à la différence que la dégradé ne stoppera qu'une fois qu'il aura atteint l'angle le plus lointain de l'élément auquel on l'applique.

Notez que si on ne précise pas de taille pour notre dégradé radial alors c'est la valeur **farthest-corner** qui sera utilisée par défaut.

## Couleurs et color stops du dégradé

Finalement, par rapport à la position des color stops, notre **0%** correspond au point central ou point d'origine du dégradé radial tandis que **100%** va correspondre à l'endroit où le dégradé se termine.

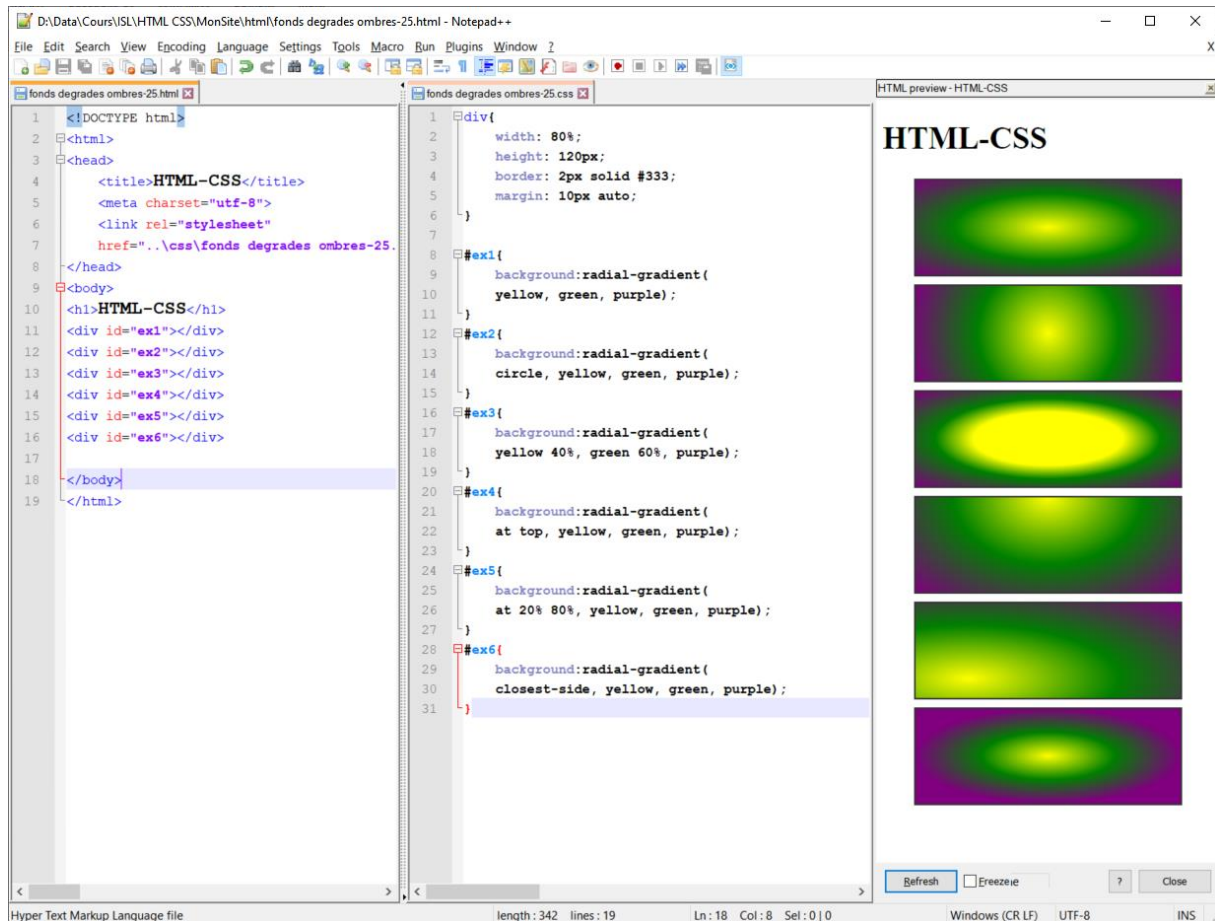
## Exemples de création de dégradés radiaux

Dans les exemples suivants, je vais créer des dégradés radiaux que je vais placer en fond de mes éléments. Avant tout, je vous rappelle les valeurs par défaut qui seront appliquées au dégradé si elles sont omises :

- Forme du dégradé : ellipse par défaut sauf si une seule valeur (ou deux valeurs identiques) ont été précisées en taille du dégradé ;
- Position du point central : center (centré) dans l'élément par défaut ;
- Taille du dégradé : farthest-corner par défaut, ce qui signifie que la fin du dégradé va venir toucher le coin de l'élément le plus éloigné de son centre.

## Création de dégradés radiaux simples

Commençons avec quelques exemples simples de création de dégradés radiaux avec **radial-gradient**. Prenez le temps une nouvelle fois de faire l'effort de retenir la syntaxe qui ressemble à celle déjà utilisée avec les dégradés linéaires.



Dans le premier exemple, on crée un dégradé radial d’arrière-plan avec **background : radial-gradient(yellow, green, purple)** pour notre premier div. Comme on ne précise que les couleurs du dégradé, notre dégradé sera par défaut une ellipse dont le point central est centré dans l’élément et qui va aller toucher *le coin le plus éloigné de l’élément* (les quatre coins en l’occurrence vu que notre dégradé est centré).

Dans le deuxième exemple, on utilise le mot clef **circle** pour que notre dégradé ait une forme de cercle. Comme la taille de celui-ci est toujours réglée sur **farthest-corner** et comme notre élément est plus large que haut, les parties haute et basse du dégradé vont être rognées.

Dans notre troisième exemple, on utilise des color stops juste pour vous montrer que ceux-ci vont fonctionner exactement de la même façon qu’avez les dégradés linéaires à la différence qu’ici notre dégradé a une forme d’ellipse. Pour bien calculer les color stops, il va falloir visualiser le « rayon » du dégradé. Dans le cas présent vous pouvez imaginer une ligne partant du centre de l’élément (puisque notre dégradé est centré par défaut) et allant toucher un de ses coins (puisque la taille du dégradé est par défaut **farthest-side** et qu’il est centré).

Dans nos quatrième et cinquième exemples, on définit des points de départ ou point centraux pour nos dégradés personnalisés. Dans le quatrième exemple, on indique simplement **top**. Le point central du dégradé va donc se trouver contre la bordure supérieure de l’élément et être centré par défaut en



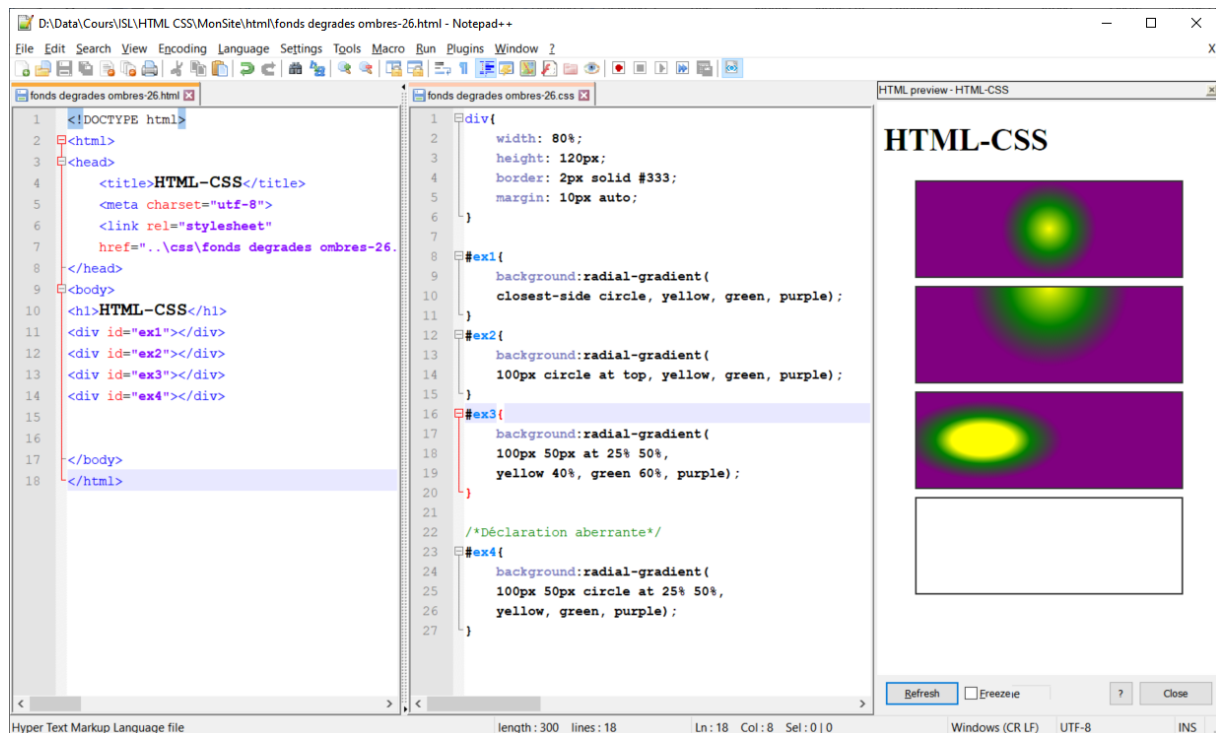
largeur.

Dans le cinquième exemple, on précise deux coordonnées pour définir un point d'origine totalement personnalisé avec **20% 80%**. Le point central du dégradé sera donc situé à une distance de 20% de la bordure gauche de l'élément et à 80% de la bordure haute. Les pourcentages donnés ici sont exprimés en fonction de la taille de la boîte représentant l'élément.

Dans le dernier exemple, on indique la valeur **closest-side** en taille de notre dégradé ce qui va le contraindre dans notre boîte puisque dès qu'il va toucher un bord de l'élément il va s'arrêter.

### Création de dégradés radiaux complexes

Nous allons également bien évidemment pouvoir créer des dégradés radiaux plus complexes et complets en définissant à la fois leur taille totale, la position de leur point central, la forme de ceux-ci etc. Faites juste attention ici à ne pas écrire de déclarations aberrantes (par exemple tenter d'imposer une forme de cercle à un dégradé radial tout en lui passant deux valeurs de taille différentes).



Ici, notre premier dégradé radial a une forme de cercle et un rayon d'une taille égale à la distance entre le point central du dégradé et le bord de l'élément le plus proche de ce point central.

Notre deuxième dégradé a un rayon de 100px et on place le point central du dégradé contre le bord supérieur de l'élément. Comme on ne précise pas de position dans l'axe horizontal, le dégradé sera centré horizontalement par défaut.

On donne deux dimensions (largeur / hauteur du « rayon ») différentes à notre troisième dégradé qui va donc de fait prendre la forme d'une ellipse. On place également le point central à une distance de

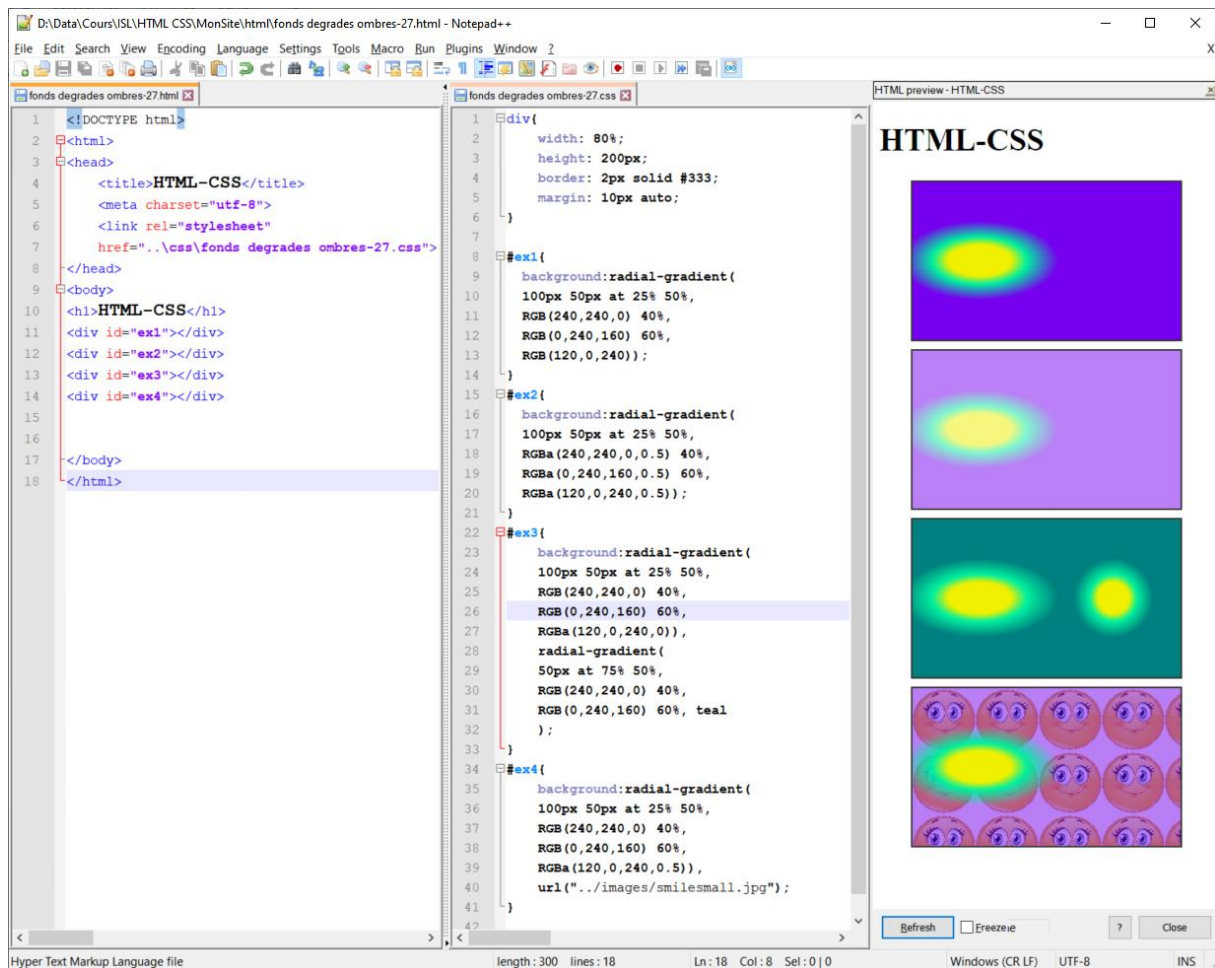
la bordure gauche de l'élément égale à 25% de la largeur de la boîte de l'élément et à une distance de la bordure haute de l'élément égale à 50% de la hauteur de la boîte de l'élément.

Dans notre dernier div, on tente de créer un dégradé avec un « rayon » de 100px de large pour 50px de haut tout en lui imposant une forme de cercle. Cela n'est évidemment pas possible et va résulter en une erreur et le dégradé ne sera pas affiché.

## Dégradés radiaux multiples et semi-transparents

La propriété **background** est tout à fait capable de gérer des fonds multiples. Nous allons donc pouvoir placer plusieurs dégradés en fond de nos éléments et les faire fusionner entre eux en utilisant des notations **RGBa** ou **HSLa()** qui vont nous permettre de rendre nos dégradés semi transparents.

Nous allons également de la même manière pouvoir placer un dégradé radial et une image en fond d'un élément. Je vous rappelle ici que le premier élément de fond déclaré en CSS sera au-dessus du second et etc.



The screenshot shows a web browser window titled "HTML preview - HTML-CSS". The browser displays a page with four divs, each containing a different radial gradient background. The first three divs show gradients with different colors and positions. The fourth div shows a radial gradient with a semi-transparent image of a smiley face. The browser's status bar at the bottom indicates the page length is 300 lines, with the cursor at line 18, column 8.

Notre premier dégradé est ici totalement opaque. Le dégradé de notre deuxième **div** utilise les mêmes paramètres que celui du premier à la différence que nous utilisons des notations **RGBa()** et qu'on lui attribue des couleurs semi transparentes.

Ensuite, on place deux dégradés radiaux en fond de notre troisième **div**. Afin qu'on puisse voir les deux, ici, j'opte pour un premier dégradé d'une taille relativement petite et utilise les notations **RGBa()** en déclarant une couleur d'arrivée de mon dégradé totalement transparente pour ne pas « polluer » le dégradé du dessous.

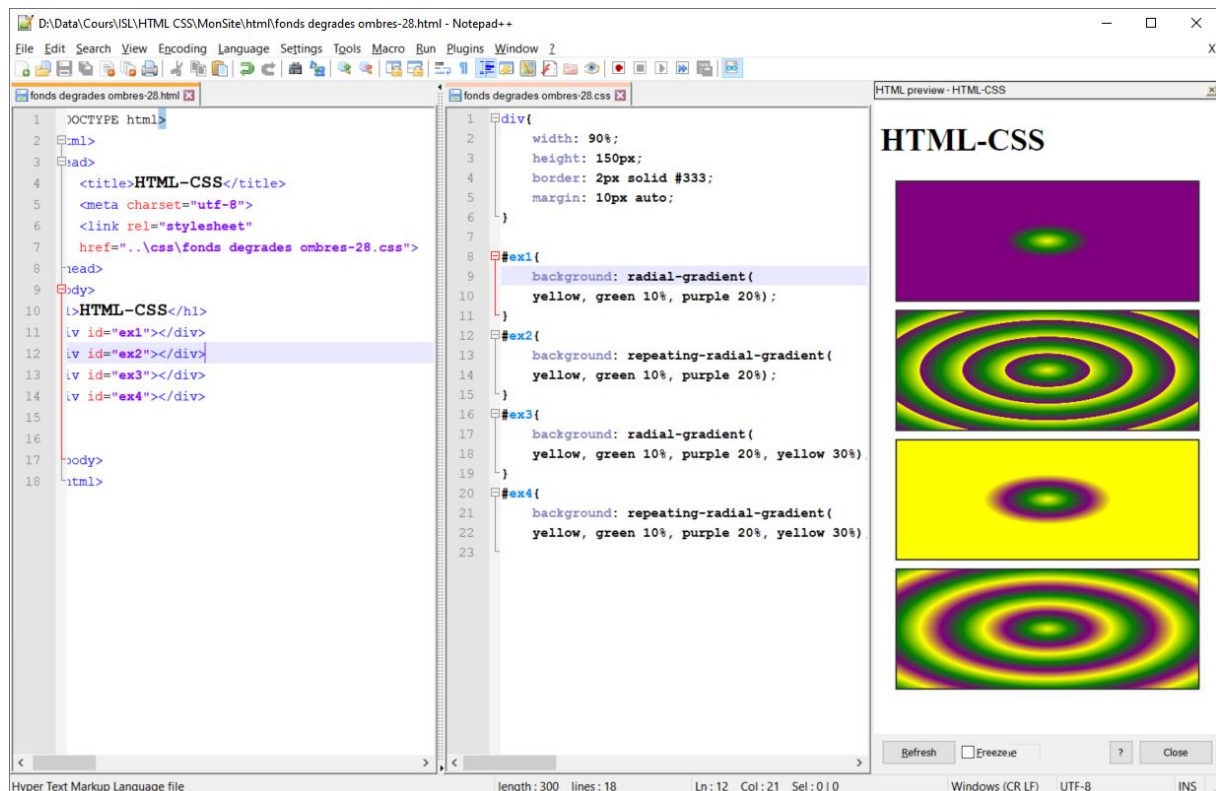
Je décale suffisamment mon deuxième dégradé dans mon élément afin qu'il ne se trouve pas au même endroit que le premier. C'est la couleur de fin du deuxième dégradé qui va remplir le **div**.

Finalement, on utilise un dégradé radial en semi transparence avec une image de fond pour notre quatrième **div**.

## La répétition de dégradés

Nous avons vu dans la leçon précédente qu'on pouvait créer des dégradés linéaires qui vont se répéter avec la fonction **repeating-linear-gradient()**. Nous allons avoir accès à la même fonctionnalité avec les dégradés radiaux en utilisant cette fois-ci plutôt **repeating-radial-gradient()**.

Cette fonction va s'utiliser de manière analogue à sa sœur servant à créer des dégradés linéaires. Nous allons une nouvelle fois pouvoir créer des transitions fluides en faisant boucler un dégradé radial sur lui-même, c'est-à-dire en lui donnant la même couleur de départ et d'arrivée.



# Ajouter des ombres aux éléments avec box-shadow en CSS

La propriété CSS **box-shadow** va nous permettre de créer des ombres dites « portées » autour de nos éléments (ou plus exactement des boîtes qui représentent les éléments). Nous allons pouvoir projeter l'ombre dans n'importe quelle direction et même vers l'intérieur.

Il convient de ne pas confondre **box-shadow** et **text-shadow** même si ces deux propriétés vont s'utiliser de façon similaire : la propriété **text-shadow** va nous permettre de créer des ombres derrière des textes tandis que **box-shadow** sert à créer des ombres autour de la boîte de l'élément.

Vous pouvez voir ci-dessous la différence entre **text-shadow** (ombre bleue) et **box-shadow** (ombre rouge) :

Les propriétés **text-shadow** et **box-shadow**

Les propriétés **text-shadow** et **box-shadow**

## Définition d'une ombre avec box-shadow

Pour générer une ombre portée, nous allons pouvoir passer jusqu'à 6 valeurs à la propriété **box-shadow**. Cependant, seules deux de ces valeurs vont être strictement obligatoires : il s'agit des valeurs définissant le décalage horizontal et du décalage vertical de l'ombre.

La liste complète des valeurs que va pouvoir accepter box-shadow est la suivante (valeurs dans leur ordre d'écriture) :

1. **L'inset** : si on précise le mot clef **inset**, alors l'ombre sera tournée vers l'intérieur de la boîte c'est à dire interne à l'élément. Si on omet la valeur (cas par défaut), alors l'ombre sera externe ;
2. **Le décalage horizontal de l'ombre** (valeur requise). Cette valeur est une longueur qu'on va donc pouvoir préciser en unités absolues (px par exemple) ou relatives (em par exemple ; attention les pourcentages ne sont pas acceptés). Si la valeur passée est positive, l'ombre sera projetée sur une distance égale à la valeur passée vers la droite en partant de l'extrémité droite de la boîte. Dans le cas contraire, l'ombre sera projetée à gauche de la boîte en partant de son extrémité gauche ;
3. **Le décalage vertical de l'ombre** (valeur requise). Cette valeur est également une longueur et va pouvoir prendre les mêmes unités que la décalage horizontal de l'ombre. Si la valeur passée est positive, l'ombre sera projetée sur une distance égale à la valeur passée vers le bas en partant du bas de la boîte. Dans le cas contraire, l'ombre sera projetée vers le haut de la boîte en partant de son extrémité supérieure ;

4. **Le rayon de flou.** Cette valeur est encore une longueur et va nous permettre de rendre notre ombre floue en lui appliquant un flou Gaussien. La formule de création d'un flou Gaussien est assez complexe; retenir simplement qu'une moyenne entre les couleurs des pixels environnants va être établie en mixant ces couleurs ensemble. Plus la valeur du flou va être grande, plus le flou de l'ombre sera diffus et l'ombre étalée. La valeur du flou vient s'ajouter aux décalage horizontal et vertical de l'ombre ;
5. **Le rayon d'étendue de l'ombre.** Cette valeur est à nouveau une longueur et va servir à agrandir ou à rétrécir la taille de l'ombre. Par défaut, sa valeur est 0 et cela correspond à une ombre de même taille que la boîte. Passer une valeur positive augmentera d'autant la taille de l'ombre tandis que passer une valeur négative la rétrécira d'autant ;
6. **La couleur de l'ombre.** Nous allons pouvoir définir la couleur de notre ombre. Toutes les valeurs de couleurs sont ici acceptées : nom de couleur, hex, RGB, RGBA...

Avant de passer à la création d'ombres avec **box-shadow**, j'aimerais attirer votre attention sur un point qui me semble essentiel de connaître pour bien comprendre ensuite le comportement des ombres créées : la propriété **box-shadow** à elle seule va suffire à créer une ombre qui va par défaut faire la taille de la boîte et être centrée derrière la boîte. Cette ombre va donc être éclipsée par la boîte et être invisible mais elle est bien présente.

Les valeurs données à **box-shadow** ne vont que nous permettre d'agrandir ou de rétrécir cette ombre, de la décaler par rapport à la boîte ou de lui appliquer un effet de flou afin d'obtenir une ombre visible autour de notre boîte. Savoir cela va être très important notamment pour comprendre le résultat lié à la définition d'un rayon de flou pour une ombre.

## Exemples d'utilisation de box-shadow et de création d'ombres en CSS

Voyons immédiatement comment utiliser la propriété **box-shadow** en pratique. Dans les exemples suivants, nous allons commencer par ajouter des ombres très simples puis nous créerons des ombres de plus en plus complexes avec **box-shadow**.

### Création d'ombres simples

Nous allons pouvoir créer des ombres très basiques autour de nos boîtes en ne renseignant que deux valeurs à la propriété **box-shadow** : la distance de projection ou de décalage de l'ombre sur le plan horizontal et sa distance de projection sur l'axe vertical.

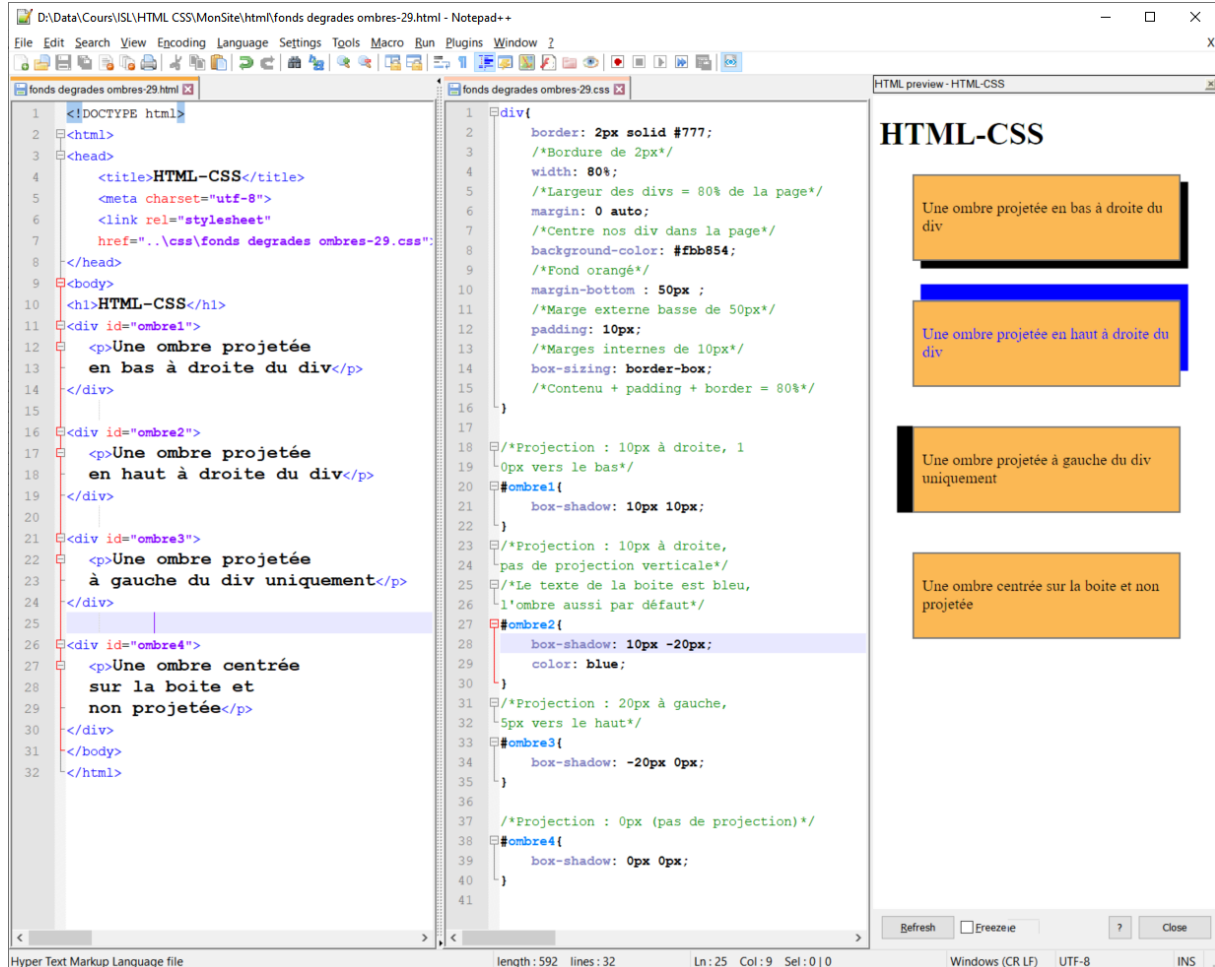
En donnant une valeur positive au paramètre « décalage horizontal », l'ombre sera projetée à droite de la boîte. Dans le cas contraire, elle sera projetée à gauche.

En donnant une valeur positive au paramètre « décalage vertical », l'ombre sera projetée en dessous de la boîte. Dans le cas contraire, elle sera projetée au-dessus.

En passant des valeurs nulles, l'ombre ne sera pas projetée et restera derrière la boîte. Attention : l'ombre existe quand même, elle n'est juste pas visible.

Notez que si on ne donne pas explicitement une couleur à l'ombre, alors elle sera de la même couleur que le texte de la boîte.

Regardez plutôt les exemples suivants (les styles appliqués aux différents div ne sont qu'esthétiques, préoccupez-vous seulement des ombres définies avec box-shadow :



The screenshot shows a Notepad++ window with two files open: 'fonds degrades ombres-29.html' and 'fonds degrades ombres-29.css'. The HTML file contains four divs with different box-shadow effects. The CSS file defines the styles for these divs, including border, background-color, margin, padding, and box-shadow. A preview window on the right shows the resulting HTML-CSS output, displaying four orange boxes with different shadow effects: a box with a shadow to the bottom right, a box with a shadow to the top right, a box with a shadow to the left, and a box with a shadow to the right.

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>HTML-CSS</title>
5   <meta charset="utf-8">
6   <link rel="stylesheet"
7     href="..\css\fonds degrades ombres-29.css">
8 </head>
9 <body>
10  <h1>HTML-CSS</h1>
11  <div id="ombre1">
12    <p>Une ombre projetée
13      en bas à droite du div</p>
14  </div>
15
16  <div id="ombre2">
17    <p>Une ombre projetée
18      en haut à droite du div</p>
19  </div>
20
21  <div id="ombre3">
22    <p>Une ombre projetée
23      à gauche du div uniquement</p>
24  </div>
25
26  <div id="ombre4">
27    <p>Une ombre centrée
28      sur la boîte et
29      non projetée</p>
30  </div>
31 </body>
32 </html>

```

```

1 div{
2   border: 2px solid #777;
3   /*Bordure de 2px*/
4   width: 80%;
5   /*Largeur des divs = 80% de la page*/
6   margin: 0 auto;
7   /*Centre nos div dans la page*/
8   background-color: #fbb854;
9   /*Fond orangé*/
10  margin-bottom: 50px;
11  /*Marge externe basse de 50px*/
12  padding: 10px;
13  /*Marges internes de 10px*/
14  box-sizing: border-box;
15  /*Contenu + padding + border = 80%*/
16 }
17
18 /*Projection : 10px à droite, 1
19 0px vers le bas*/
20 #ombre1{
21   box-shadow: 10px 10px;
22 }
23
24 /*Projection : 10px à droite,
25 pas de projection verticale*/
26 /*Le texte de la boîte est bleu,
27 l'ombre aussi par défaut*/
28 #ombre2{
29   box-shadow: 10px -20px;
30   color: blue;
31 }
32
33 /*Projection : 20px à gauche,
34 5px vers le haut*/
35 #ombre3{
36   box-shadow: -20px 0px;
37 }
38
39 /*Projection : 0px (pas de projection)*/
40 #ombre4{
41   box-shadow: 0px 0px;
42 }

```

HTML-CSS

Une ombre projetée en bas à droite du div

Une ombre projetée en haut à droite du div

Une ombre projetée à gauche du div uniquement

Une ombre centrée sur la boîte et non projetée

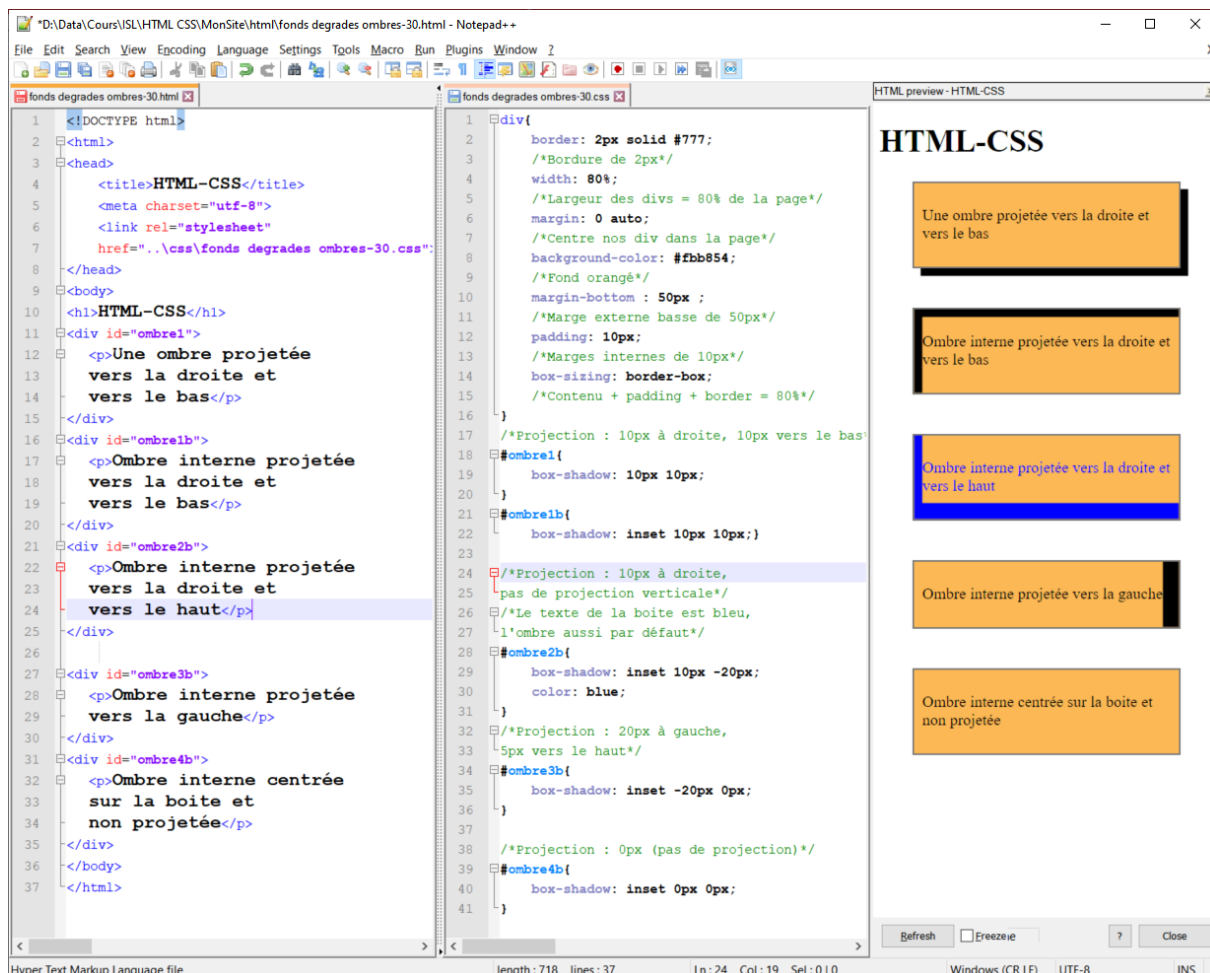
Refresh Freeze ? Close

Hyper Text Markup Language file length: 592 lines: 32 Ln: 25 Col: 9 Sel: 0 | 0 Windows (CR LF) UTF-8 INS

### Ajout de la valeur inset pour créer une ombre interne

La propriété **box-shadow** va également nous permettre de créer des ombres internes à nos boîtes, c'est-à-dire de projeter l'ombre à l'intérieur de l'élément plutôt qu'à l'extérieur en lui ajoutant la valeur **inset** en première valeur comme cela :





```

1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>HTML-CSS</title>
5   <meta charset="utf-8">
6   <link rel="stylesheet"
7     href="..\css\fonds degreates ombres-30.css">
8 </head>
9 <body>
10  <h1>HTML-CSS</h1>
11  <div id="ombre1">
12    <p>Une ombre projetée
13      vers la droite et
14      vers le bas</p>
15  </div>
16  <div id="ombre1b">
17    <p>Ombre interne projetée
18      vers la droite et
19      vers le bas</p>
20  </div>
21  <div id="ombre2b">
22    <p>Ombre interne projetée
23      vers la droite et
24      vers le haut</p>
25  </div>
26  <div id="ombre3b">
27    <p>Ombre interne projetée
28      vers la gauche</p>
29  </div>
30  <div id="ombre4b">
31    <p>Ombre interne centrée
32      sur la boîte et
33      non projetée</p>
34  </div>
35 </body>
36 </html>

```

```

1 div{
2   border: 2px solid #777;
3   /*Bordure de 2px*/
4   width: 80%;
5   /*Largeur des divs = 80% de la page*/
6   margin: 0 auto;
7   /*Centre nos div dans la page*/
8   background-color: #fbb854;
9   /*Fond orangé*/
10  margin-bottom: 50px;
11  /*Marge externe basse de 50px*/
12  padding: 10px;
13  /*Marges internes de 10px*/
14  box-sizing: border-box;
15  /*Contenu + padding + border = 80%*/
16 }
17 /*Projection : 10px à droite, 10px vers le bas*/
18 #ombre1{
19   box-shadow: 10px 10px;
20 }
21 #ombre1b{
22   box-shadow: inset 10px 10px;
23 }
24 /*Projection : 10px à droite,
25   pas de projection verticale*/
26 /*Le texte de la boîte est bleu,
27   l'ombre aussi par défaut*/
28 #ombre2b{
29   box-shadow: inset 10px -20px;
30   color: blue;
31 }
32 /*Projection : 20px à gauche,
33   5px vers le haut*/
34 #ombre3b{
35   box-shadow: inset -20px 0px;
36 }
37 /*Projection : 0px (pas de projection)*/
38 #ombre4b{
39   box-shadow: inset 0px 0px;
40 }
41

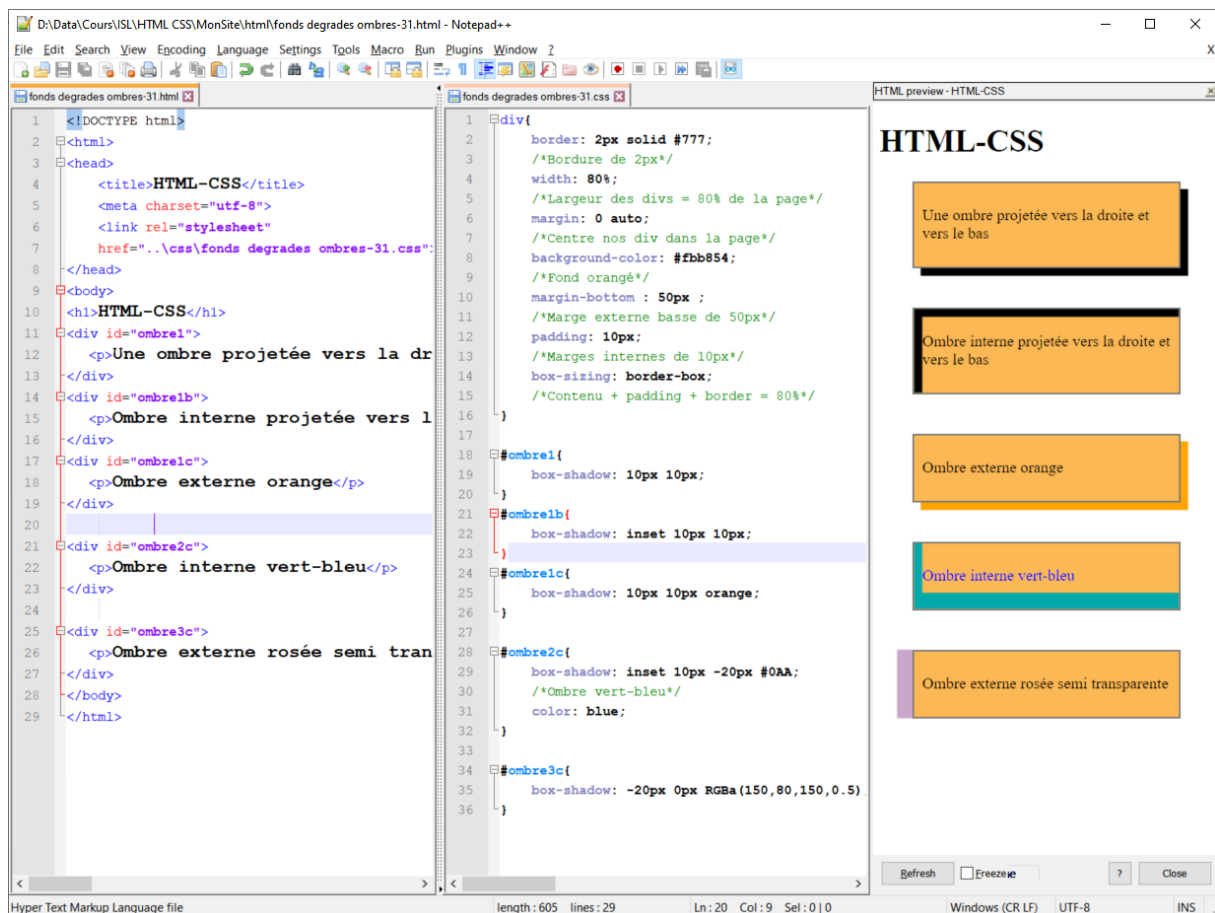
```

Notez bien ici que la direction des ombres est toujours la même. Cependant, comme on demande à ce que l'ombre soit projetée à l'intérieur de la boîte, l'ombre va être visible de l'autre côté de la boîte par rapport à une ombre externe définie de la même façon. C'est simplement le « point de départ » de l'ombre qui change.

### Création d'ombres colorées et / ou semi transparentes

Par défaut, l'ombre créée par **box-shadow** va être de même couleur que le texte de la boîte à laquelle on applique l'ombre.

Nous allons cependant pouvoir choisir la couleur de notre ombre. Pour cela, il suffira de la renseigner en valeur de **box-shadow**. Cette propriété accepte toutes les notations de couleurs, que ce soit une couleur nommée, une notation RGB, hexadécimale ou même une notation RGBA qui va nous permettre de créer des ombres semi transparentes.



Note : La valeur « couleur » de l'ombre sera la dernière valeur à indiquer dans **box-shadow** en CSS.

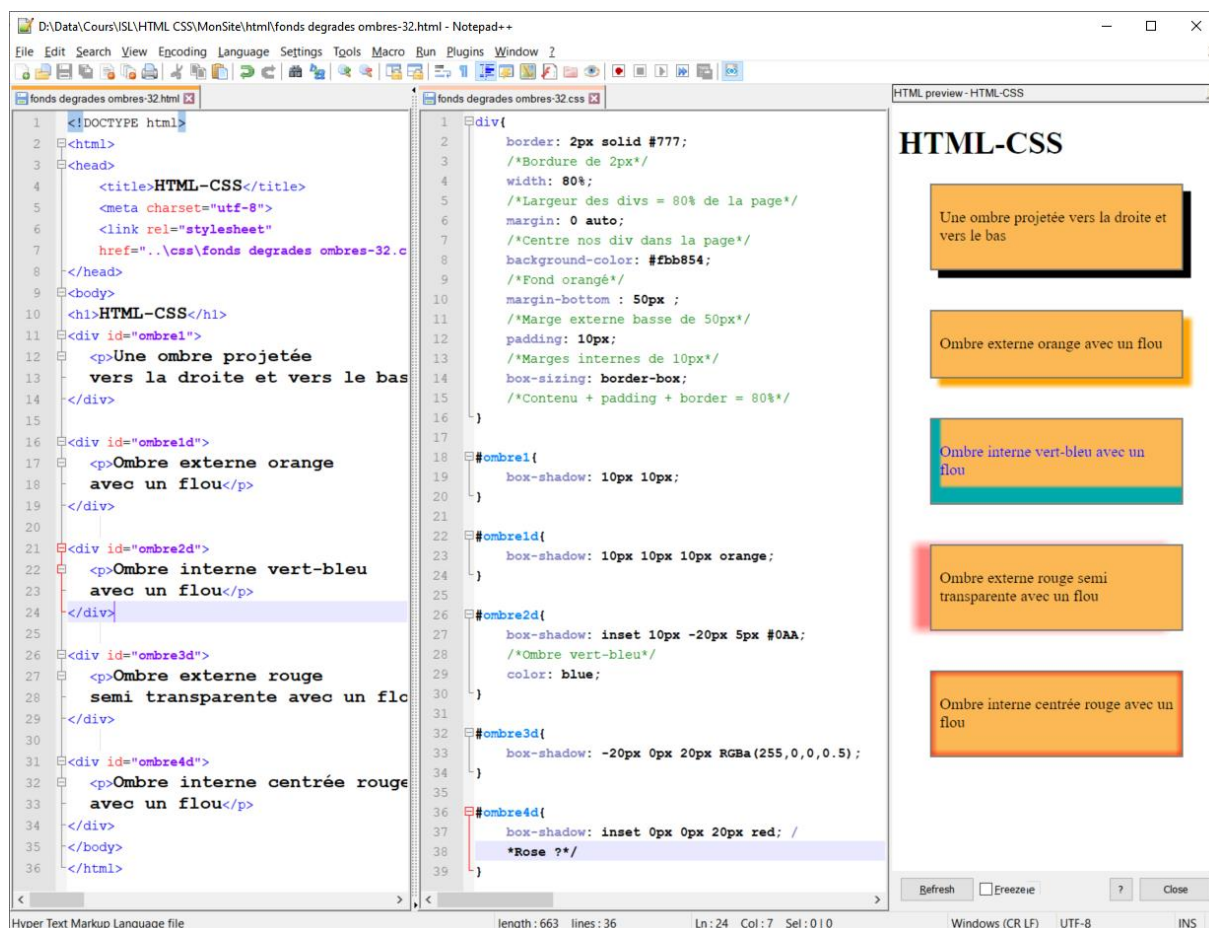
### Ajout d'un flou et ombre centrée autour de la boîte

Nous avons vu que pour centrer une ombre par rapport à la boîte, il suffisait d'indiquer des décalages horizontal et vertical de 0px lors de la définition de l'ombre avec **box-shadow**. Cela fait sens : par défaut, l'ombre d'une boîte se trouve derrière la boîte et la propriété **box-shadow** va nous permettre, entre autres, de la décaler.

Jusqu'à présent, nos ombres centrées n'étaient pas visibles puisqu'elles faisaient exactement la taille de notre boîte. Nous allons cependant pouvoir les faire apparaître en ajoutant un flou à notre ombre (la « taille » du flou va s'ajouter à la taille de l'ombre ou en modifiant la taille de celle-ci.

Essayons déjà d'ajouter un flou à nos ombres en ajoutant une nouvelle valeur à **box-shadow** et regardons les résultats ensemble. La valeur du flou va se placer après les valeurs liées aux décalages dans la propriété CSS.





Pour créer cet effet de flou, une moyenne entre les couleurs des pixels environnants va être établie en mixant ces couleurs ensemble. Plus la valeur du flou va être grande, plus le flou de l'ombre sera diffus et l'ombre étalée. Notez que le flou s'applique tout autour de l'ombre.

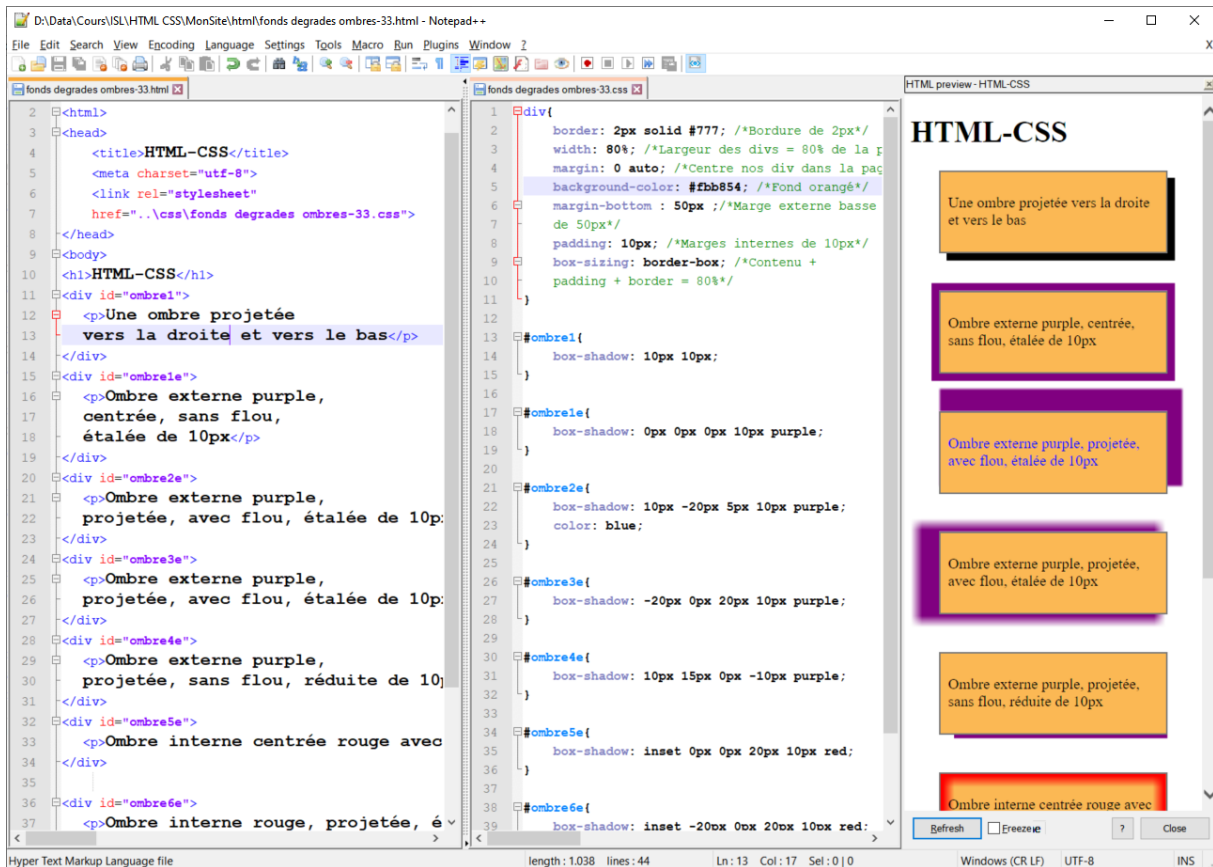
Ici, vous pouvez observer que notre la partie floue de notre ombre apparait également au-dessus et en dessous de notre avant dernière boite qui possède une ombre centrée verticalement par rapport à la boite. Ce comportement est dû au fait que la valeur du flou vient s'ajouter à la taille de l'ombre.

Comme l'ombre fait exactement la taille de la boite par défaut, si celle-ci est centrée et si on ajoute un flou autour de l'ombre, alors la partie floue de l'ombre dépassera de la boite et sera visible. C'est la même chose qui se passe dans notre dernier exemple : vous pouvez remarque que seule la partie floue de l'ombre est visible.

### Une ombre plus grande ou plus petite que notre boite

Finalement, nous allons pouvoir modifier la taille de départ de notre ombre pour que celle-ci dépasse par défaut de la boite ou, au contraire, soit plus petite qu'elle.

Les valeurs positives agrandiront (ou étaleront) l'ombre et les valeurs négatives rétréciront l'ombre. Nous allons renseigner cette valeur correspondant au rayon d'étalement de l'ombre après la valeur du flou dans box-shadow.



HTML-CSS

Une ombre projetée vers la droite et vers le bas

Ombre externe purple, centrée, sans flou, étalée de 10px

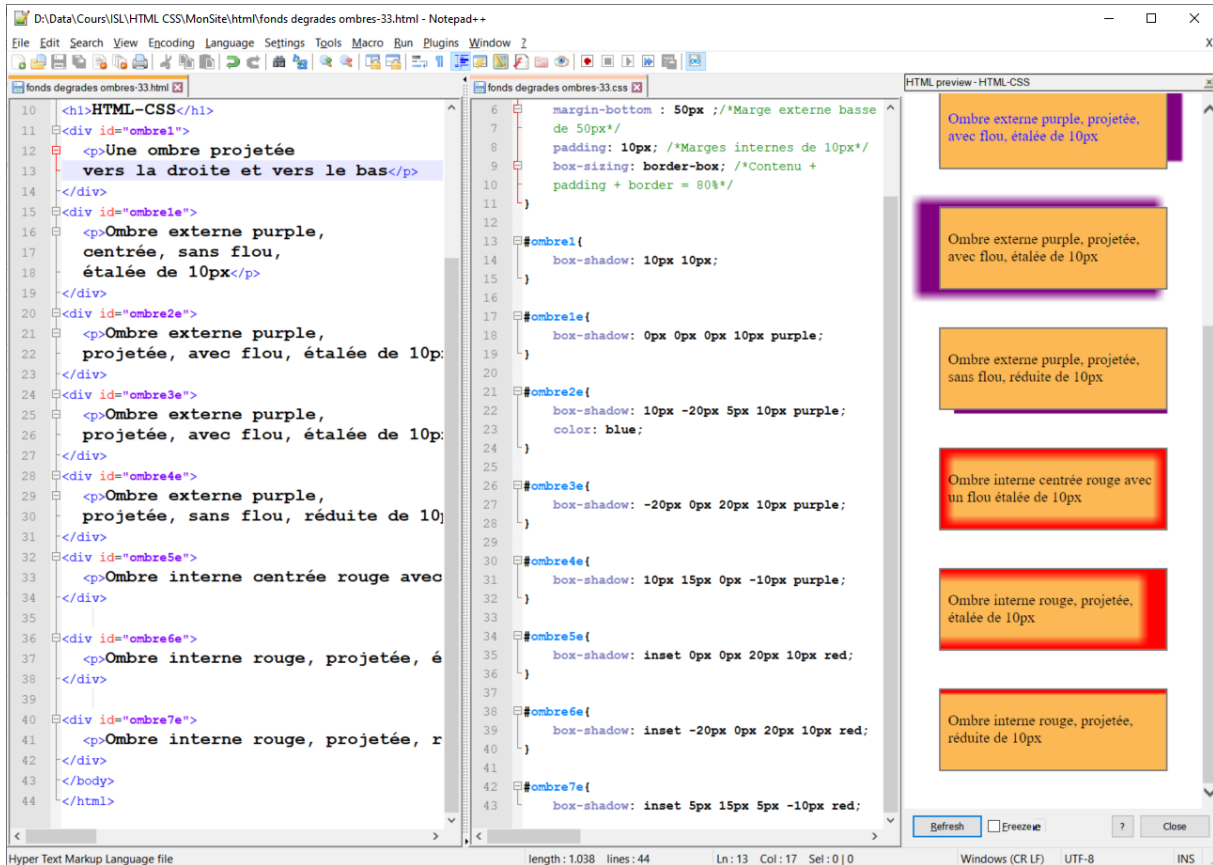
Ombre externe purple, projetée, avec flou, étalée de 10px

Ombre externe purple, projetée, avec flou, étalée de 10px

Ombre externe purple, projetée, sans flou, réduite de 10px

Ombre interne centrée rouge avec

Ombre interne rouge, projetée, é



```

10 <h1>HTML-CSS</h1>
11 <div id="ombre1">
12 <p>Une ombre projetée
13 vers la droite et vers le bas</p>
14 </div>
15 <div id="ombre1e">
16 <p>Ombre externe purple,
17 centrée, sans flou,
18 étalée de 10px</p>
19 </div>
20 <div id="ombre2e">
21 <p>Ombre externe purple,
22 projetée, avec flou, étalée de 10px</p>
23 </div>
24 <div id="ombre3e">
25 <p>Ombre externe purple,
26 projetée, avec flou, étalée de 10px</p>
27 </div>
28 <div id="ombre4e">
29 <p>Ombre externe purple,
30 projetée, sans flou, réduite de 10px</p>
31 </div>
32 <div id="ombre5e">
33 <p>Ombre interne centrée rouge avec
34 flou</p>
35 </div>
36 <div id="ombre6e">
37 <p>Ombre interne rouge, projetée, étalée de 10px</p>
38 </div>
39 <div id="ombre7e">
40 <p>Ombre interne rouge, projetée, réduite de 10px</p>
41 </div>
42 </body>
43 </html>
44
6 margin-bottom : 50px ;/*Marge externe basse
7 de 50px*/
8 padding: 10px; /*Marges internes de 10px*/
9 box-sizing: border-box; /*Contenu +
10 padding + border = 80%*/
11
12
13 #ombre1{
14 box-shadow: 10px 10px;
15 }
16
17 #ombre1e{
18 box-shadow: 0px 0px 0px 10px purple;
19 }
20
21 #ombre2e{
22 box-shadow: 10px -20px 5px 10px purple;
23 color: blue;
24 }
25
26 #ombre3e{
27 box-shadow: -20px 0px 20px 10px purple;
28 }
29
30 #ombre4e{
31 box-shadow: 10px 15px 0px -10px purple;
32 }
33
34 #ombre5e{
35 box-shadow: inset 0px 0px 20px 10px red;
36 }
37
38 #ombre6e{
39 box-shadow: inset -20px 0px 20px 10px red;
40 }
41
42 #ombre7e{
43 box-shadow: inset 5px 15px 5px -10px red;
44 }

```

HTML preview - HTML-CSS

- Ombre externe purple, projetée, avec flou, étalée de 10px
- Ombre externe purple, projetée, avec flou, étalée de 10px
- Ombre externe purple, projetée, sans flou, réduite de 10px
- Ombre interne centrée rouge avec un flou étalée de 10px
- Ombre interne rouge, projetée, étalée de 10px
- Ombre interne rouge, projetée, réduite de 10px

Refresh Freeze ? Close

Hyper Text Markup Language file length: 1.038 lines: 44 Ln: 13 Col: 17 Sel: 0 | 0 Windows (CR LF) UTF-8 INS

Nous pouvons voir nettement dans le premier exemple que la taille de notre ombre a été agrandie puisque celle-ci est centrée et ne possède pas de flou mais dépasse bien de la boîte.

Parmi ces exemples, j'attire votre attention sur les 4<sup>e</sup> et dernier, c'est à dire sur les deux ombres qu'on a rétréci. En effet, ces deux ombres sont censées être décalées vers le bas et vers la droite. Or, on ne voit pas le décalage droit. Pour comprendre pourquoi, il suffit d'analyser le contenu de notre propriété **box-shadow** dans chacun des cas.

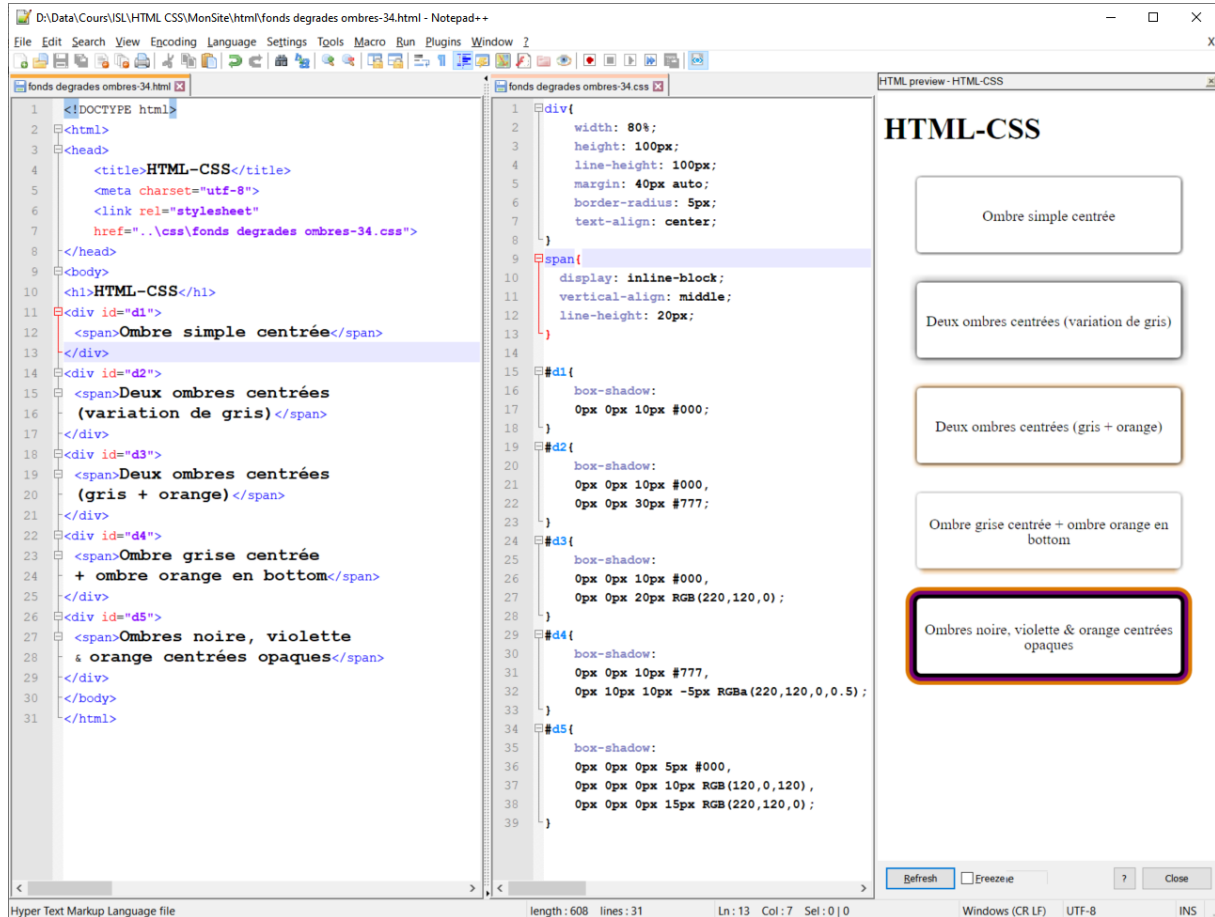
Dans le 4<sup>e</sup> exemple, l'ombre possède un décalage vers la droite de 10px et pas de flou. Or, on demande également à ce que l'ombre soit rétrécie de 10px. Ainsi, l'ombre va bien être décalée de 10px vers la droite mais comme elle fait désormais 10px de moins que la boîte, elle ne va pas dépasser.

Dans notre dernier exemple, on définit un décalage vers la droite de 5px pour notre ombre. De plus, on lui ajoute un flou de 5px qui vient s'ajouter à la taille de l'ombre. Notre ombre devrait donc à nouveau dépasser de 10px. Cependant, ici aussi, on demande à ce que l'ombre soit rétrécie de 10px. L'ombre va donc bien à nouveau être décalée de 5px et un flou de 5px va bien être appliqué mais tout cela ne sera pas visible car compensé par le rétrécissement de l'ombre.

## Création d'ombres multiples autour d'un élément

Nous allons tout à fait pouvoir définir plusieurs ombres autour de nos éléments en utilisant **box-shadow**. Pour cela, il suffit de séparer nos différentes ombres par des virgules lors de leur déclaration avec **box-shadow**.

Appliquer plusieurs ombres d'un coup à un élément peut nous permettre de créer différents niveaux d'opacité d'ombre et de créer des effets 3D intéressants.



The screenshot shows a Notepad++ editor with two files open: 'fonds degrades ombres-34.html' and 'fonds degrades ombres-34.css'. The HTML file contains a series of `<div>` and `<span>` elements, each with a unique ID and descriptive text. The CSS file defines styles for these elements, including a base style for a `div` and specific `box-shadow` rules for each `span` ID. The preview window on the right, titled 'HTML-CSS', displays the rendered output, showing five distinct boxes with various shadow effects: a simple centered shadow, two centered shadows with different gray shades, two centered shadows with different colors (gray and orange), a centered gray shadow with an orange shadow at the bottom, and a combination of black, violet, and orange centered shadows.

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>HTML-CSS</title>
5   <meta charset="utf-8">
6   <link rel="stylesheet"
7     href="..\css\fonds degrades ombres-34.css">
8 </head>
9 <body>
10  <h1>HTML-CSS</h1>
11  <div id="d1">
12    <span>Ombre simple centrée</span>
13  </div>
14  <div id="d2">
15    <span>Deux ombres centrées
16      (variation de gris)</span>
17  </div>
18  <div id="d3">
19    <span>Deux ombres centrées
20      (gris + orange)</span>
21  </div>
22  <div id="d4">
23    <span>Ombre grise centrée
24      + ombre orange en bottom</span>
25  </div>
26  <div id="d5">
27    <span>Ombres noire, violette
28      & orange centrées opaques</span>
29  </div>
30 </body>
31 </html>

```

```

1 div{
2   width: 80%;
3   height: 100px;
4   line-height: 100px;
5   margin: 40px auto;
6   border-radius: 5px;
7   text-align: center;
8 }
9 span{
10  display: inline-block;
11  vertical-align: middle;
12  line-height: 20px;
13 }
14
15 #d1{
16  box-shadow:
17    0px 0px 10px #000;
18 }
19 #d2{
20  box-shadow:
21    0px 0px 10px #000,
22    0px 0px 30px #777;
23 }
24 #d3{
25  box-shadow:
26    0px 0px 10px #000,
27    0px 0px 20px RGB(220,120,0);
28 }
29 #d4{
30  box-shadow:
31    0px 0px 10px #777,
32    0px 10px 10px -5px RGBa(220,120,0,0.5);
33 }
34 #d5{
35  box-shadow:
36    0px 0px 0px 5px #000,
37    0px 0px 0px 10px RGB(120,0,120),
38    0px 0px 0px 15px RGB(220,120,0);
39 }

```

Ici, nous réutilisons simplement ce que nous avons vu jusqu'à présent pour créer des ombres. Simplement, nous déclarons plusieurs ombres d'aspects et de tailles différents autour de nos éléments **div**.