Apprendre à coder en HTML - CSS

Table des matières

Les langages HTML et CSS sont véritablement le socle de tout projet de développement web. Que vous vouliez créer un site e-commerce, un blog, une application mobile ou quoique ce soit d'autre, vous serez obligé de passer par les langages HTML et CSS.

Apprendre le HTML et le CSS signifie entrer dans le monde des programmeurs et c'est donc commencer à les comprendre et à parler comme eux.

1. Introduction

HTML est l'abréviation de HyperText Markup Language, soit en français « langage hypertexte de balisage ». Ce langage a été créé en 1991 et a pour fonction de structurer et de donner du sens à du contenu.

CSS signifie Cascading StyleSheets, soit « feuilles de style en cascade ». Il a été créé en 1996 et a pour rôle de mettre en forme du contenu en lui appliquant ce qu'on appelle des styles.

Il ne faut JAMAIS utiliser le HTML pour faire le travail du CSS :

- HTML est utilisé pour baliser un contenu, c'est à dire pour le structurer et lui donner du sens. Le HTML sert, entre autres choses, à indiquer aux navigateurs quel texte doit être considéré comme un paragraphe, quel texte doit être considéré comme un titre, que tel contenu est une image ou une vidéo.
- CSS est utilisé pour appliquer des styles à un contenu, c'est-à-dire à le mettre en forme. Ainsi, avec le CSS, on pourra changer la couleur ou la taille d'un texte, positionner tel contenu à tel endroit de notre page web ou ajouter des bordures ou des ombres autour d'un contenu.

Les versions actuelles du HTML et du CSS sont HTML5 et CSS3. Il faut savoir que, contrairement aux idées reçues, ce sont encore des versions non finalisées. Cela signifie que ces versions sont toujours en développement et qu'elles sont toujours en théorie sujettes à changements et à bugs.

La version 5 d'HTML, tout comme la version 3 de CSS introduisent de nombreuses nouvelles fonctionnalités longtemps attendues par les développeurs et il serait donc dommage de s'en priver.

Parmi celles-ci, l'insertion simplifiée de vidéos et de contenus audio et de nouvelles balises améliorant la sémantique (on va y revenir, pas d'inquiétude !) pour mieux structurer nos pages en HTML ou encore la possibilité de créer des bordures arrondies en CSS.

Pour coder en HTML et en CSS, nous n'avons besoin que d'un éditeur de text :

- NotePad++, sous Windows; • Komodo, pour Mac ou Linux;
- TextWrangler, pour Linux.

2. Les bases en HTML et CSS

2.1. Les bases du HTML

2.1.1. Éléments, balises et attributs

Il y a trois termes dont vous devez absolument comprendre le sens en HTML. Ce sont les termes élément, balise et attribut.

Les éléments, tout d'abord, vont nous servir à définir des objets dans notre page. Grâce aux éléments, nous allons pouvoir définir un paragraphe (élément p), des titres d'importances diverses (éléments h1, h2, h3, h4, h5 et h6) ou un lien (élément a).

Les éléments sont constitués de balises. Dans la majorité des cas, un élément est constitué d'une paire de balises : une balise ouvrante et une balise fermante.

Les balises reprennent le nom de l'élément et sont entourées de chevrons. La balise fermante possède en plus un slash qui précède le nom de l'élément.

Cependant, certains éléments ne sont constitués que d'une balise qu'on appelle alors balise orpheline. C'est par exemple le cas de l'élément br qui va nous servir à créer un retour à la ligne.

Notez que, dans le cas des balises orphelines, le slash se situe après le nom de l'élément. Notez également que ce slash n'est pas obligatoire et que certains développeurs l'omettent volontairement.

Je vous conseille cependant, pour des raisons de propreté de code et de compréhension, de mettre le slash dans un premier temps.

Les attributs, enfin, vont venir compléter les éléments en leur donnant des indications ou des instructions supplémentaires.

Par exemple, dans le cas d'un lien, on va se servir d'un attribut pour indiquer la cible du lien, c'est à dire vers quel site le lien doit mener.

Notez que les attributs se placent toujours à l'intérieur de la balise ouvrante d'un élément (ou de la balise orpheline le cas échéant).

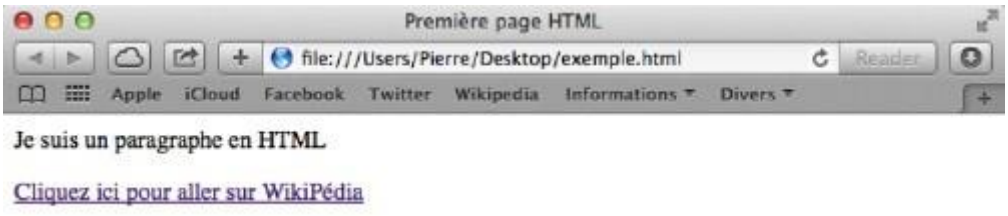
Dans l'exemple ci-dessus, on discerne l'élément a composé :

- d'une balise ouvrante elle-même composée d'un attribut href ;
- d'une ancre textuelle ;
- d'une balise fermante.

L'attribut href (initiales de « Hypertexte Reference ») sert à indiquer la cible de notre lien, en l'occurrence le site Wikipédia.

L'ancre textuelle correspond au texte entre les balises. Ce sera le texte sur lequel vos visiteurs devront cliquer pour se rendre sur Wikipédia et également l'unique partie visible pour eux.

Voici ce que les visiteurs de votre site verront :



Si vous vous sentez un peu perdu pour le moment, ne vous inquiétez pas, c'est tout-à-fait normal ! Continuez le cours, vous allez voir : avec un peu de pratique, tout cela va très vite se décanter et vous aurez tout compris avant même de vous en rendre compte !

2.1.2. Structure de base d'une page en HTML5

En programmation comme dans beaucoup d'autres disciplines, vous l'aurez compris, il y a des règles.

Ainsi, toute page écrite en HTML5 doit comporter une certaine structure, un « squelette » qui sera toujours le même. Ce squelette est bien évidemment constitué de divers éléments.

Tout d'abord, toute page HTML5 doit commencer par la déclaration de ce qu'on appelle un « doctype ». Le doctype, comme son nom l'indique, sert à indiquer le type du document. Dans notre cas, le type de document est HTML. On écrira donc :

```
<!DOCTYPE html>
```

Notez bien le point d'exclamation, obligatoire, au début de cet élément un peu particulier.

Ensuite, pour que notre page HTML5 soit valide, il va nous falloir indiquer trois nouveaux éléments : html, head (« en-tête ») et body (« corps de page »).

L'élément html va contenir toute la page.

L'élément head contiendra entre autres, le titre de la page, l'encodage utilisé et des meta-data (des données invisibles pour les utilisateurs mais essentielles – nous en reparlerons plus tard).

L'élément body contiendra tout le contenu de notre page (paragraphes, images, tableaux, etc.).

Voilà où nous en sommes rendus pour le moment :

```
1  <!DOCTYPE html>
2      <html>
3          <head>
4
5          </head>
6          <body>
7
8          </body>
9      </html>
10
```

Mais ce n'est pas fini ! Pour que la page soit valide, l'élément head doit lui même contenir un élément title, qui correspond au titre de la page et le meta charset qui prendra comme valeur le type d'encodage choisi.

Pour les langues latines, nous choisirons généralement la valeur « utf-8 ».

Voici à quoi vous devriez arriver en pratique :

```
1  <!DOCTYPE html>
2      <html>
3          <head>
4              <title>Ma première page valide en HTML5</title>
5              <meta charset="utf-8"/>
6          </head>
7          <body>
8
9          </body>
10     </html>
11
```

Et voilà, vous venez, sans vous en rendre compte, de créer votre première page valide en HTML5 ! Retenez bien ce schéma, il sera toujours le même quelque soit la page HTML5 que vous créerez.

2.1.3. Afficher un document en HTML5 dans le navigateur

La première chose à faire, une fois un nouveau fichier ouvert dans votre éditeur de texte, est d'en changer le type pour mettre votre document en « html ». Pour cela, deux solutions : soit vous trouvez l'onglet où vous pourrez définir le type de votre document, soit vous enregistrez-sous votre document en lui donnant un nom du type : document.html. Cela aura pour effet de changer le type de document en un document HTML.

Ensuite, pour visualiser votre fichier dans un navigateur, vous avez à nouveau deux choix. Soit votre éditeur possède une option vous proposant de pré-visualiser votre document dans le navigateur de votre choix (c'est le cas de Komodo), soit vous devrez enregistrer votre document (sur votre bureau, par exemple), puis double cliquer dessus, tout simplement, afin que la page s'ouvre dans votre navigateur par défaut.

Pensez bien à enregistrer dans votre éditeur votre document à chaque fois que vous souhaitez l'ouvrir dans votre navigateur et à rafraîchir la page dans votre navigateur si votre document était déjà ouvert (ctrl+r ou cmd+r) sans quoi vous ne verrez aucun changement.

Cela semble évident mais si vous saviez le nombre d'heures que nous avons tous perdu en cherchant pourquoi telle ou telle chose ne fonctionnait pas… Alors que nous avions simplement oublié d'enregistrer les changements effectués au préalable ou de rafraîchir notre document !

2.1.4. Bonnes pratiques, règles et commentaires

Le HTML permet d'imbriquer des éléments les uns à l'intérieur des autres. C'est même l'une des possibilités qui font toute sa force.

Dans l'exemple précédent, par exemple, notre élément title était à l'intérieur de notre élément head, lui même contenu dans un élément html.

```
<!DOCTYPE html>
    <html>
        <head>
            <title>Ma première page valide en HTML5</title>
            <meta charset="utf-8"/>
        </head>
        <body>

        </body>
    </html>
```

Toutefois, si le HTML permet d'imbriquer des éléments les uns dans les autres, vous devrez toujours faire bien attention de refermer les balises dans le bon ordre : le dernier élément ouvert est toujours le premier refermé.

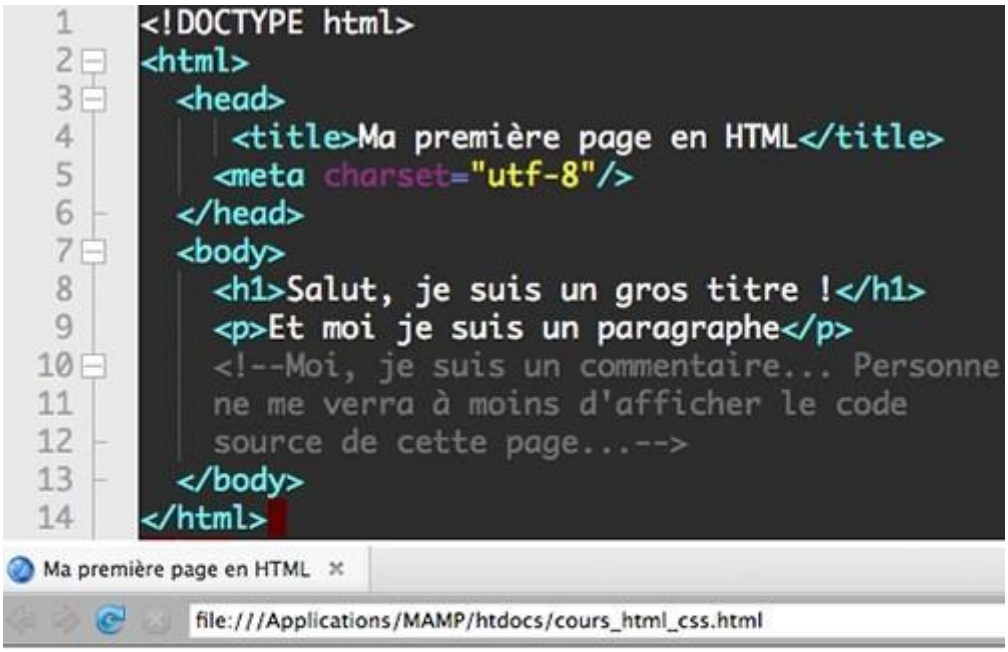
Autre bonne pratique : l'indentation. Indenter son code, c'est tout simplement l'aérer en ajoutant des espaces au début de certaines lignes afin de le rendre plus lisible pour vous comme pour les autres.

Il n'y a pas de règle absolue concernant l'indentation, certains accentuant plus ou moins le retrait en début de ligne. Pour ma part, j'utilise une tabulation (la touche à gauche du a) à chaque fois que j'ouvre une nouvelle balise à l'intérieur d'un élément. Cela permet de bien hiérarchiser son code et de voir immédiatement quel élément est imbriqué dans quel autre.

Troisième et dernière bonne pratique : commenter son code.

Commenter son code, c'est tout simplement y ajouter des commentaires. Ces commentaires sont spéciaux : il ne seront pas visibles par vos visiteurs (à moins que ceux-ci n'affichent le code source de la page).

Voici comment on écrit un commentaire en HTML :



Pourquoi commenter son code :

1. Tout d'abord, pour vous. En effet, lorsque vous commencerez à véritablement savoir coder, vos pages vont s'allonger et se complexifier. Commenter va alors devenir indispensable pour vous repérer dans votre page web et pour vous rappeler pourquoi vous avez fait telle chose de telle façon.
2. Ensuite, pour les personnes à qui vous pourriez distribuer votre code. Rappelez vous qu'il existe autant de manières de coder que d'esprits humains ou presque et qu'il est donc essentiel de donner des indications sur votre code afin que les personnes puissent le comprendre plus rapidement et plus facilement.

En bref : commenter n'est pas un luxe mais souvent ce qui sépare un développeur moyen d'un bon développeur, tout simplement.

Attention à ne jamais mettre d'informations sensibles en commentaire, comme des mots de passe par exemple. En effet, rappelez vous que tout le monde peut avoir accès au code source de votre page, et donc à votre code HTML.

Pour vous en convaincre, n'hésitez pas à aller sur n'importe quel site (Wikipédia par exemple), puis à faire un clic droit sur la page et choisir l'option « inspecter l'élément » ou « afficher le code source » selon votre navigateur.

Vous aurez alors accès au code HTML de la page, quelque soit le site.

2.1.5. Les éléments Heading, Paragraph et Break

Le HTML sert à différencier d'un point de vue sémantique les différents objets que l'on peut rencontrer et dont on peut avoir besoin (titre, paragraphe, espace, image, etc.).

Pour faire cela, le HTML va utiliser des éléments. Et le moment est arrivé pour nous d'apprendre à créer des paragraphes, des titres, et à faire des retours à la lignes en HTML.

Pour créer des paragraphes, tout d'abord, on va utiliser l'élément p. On peut créer autant de paragraphes que l'on souhaite dans une page. Pour chaque nouveau paragraphe, il faut rouvrir une balise <p>. A chaque nouveau paragraphe, un retour à la ligne est automatiquement effectué.

Si maintenant vous désirez créer un retour à la ligne à l'intérieur même d'un paragraphe, il faudra utiliser l'élément br (diminutif de break).

Cet élément est constitué d'une seule balise orpheline, qu'on notera donc
.

Pour créer des titres, on va utiliser les éléments h1, h2, h3, h4, h5 et h6 pour créer des titres de diverses importances. Ainsi, un titre h1 sera considéré comme un titre très important tandis qu'un titre h6 sera considéré comme très peu important.

Voilà ce que ça donne en image :

Les titres apparaissent en gras et ont des tailles différentes selon leur degré d'importance. Et c'est précisément là où j'en reviens à mon premier point : vous ne devez JAMAIS utiliser le langage HTML pour mettre en forme le contenu.

Ce que vous voyez n'est qu'une mise en forme automatique faîte par votre navigateur, une interprétation visuelle de ce que vous avez donné à votre navigateur. Cependant, vous ne devez jamais utiliser un titre de niveau h1 pour mettre un texte en gros et en gras.

Le résultat d'une interprétation et n'est qu'une mise en forme automatique. Le CSS permet de faire réellement tout ce que l'on désire d'un point de vue visuel : créer des styles tout comme enlever des styles automatiques.

2.1.6. Les éléments Strong, Emphasis et Mark

L'élément strong est utilisé pour indiquer aux moteurs de recherche qu'un contenu est particulièrement important, afin que celui-ci soit traité avec plus d'importance. Le résultat visuel est une mise en forme automatique en gras. Mais encore une fois, on n'utilise pas l'élément strong pour mettre un texte en gras !

L'élément em, pour emphasis (« emphase » en français) est proche de l'élément strong. Il sert lui aussi à signifier qu'un contenu est important, mais moins important tout de même qu'un contenu entre des balises strong.

L'intérêt de ces éléments, il est avant tout dans l'optimisation du référencement du site. En effet, normalement, vous devriez avoir ciblé des mots clefs et essayer d'être bien référencé sur ces mots là. Les balises strong et em vous aident à atteindre ce but, entre autres.

Enfin, l'élément mark est utilisé pour faire ressortir du contenu. Ce contenu ne sera pas forcément considéré comme important, mais cette balise peut servir dans le cas de l'affichage de résultats suite à une recherche d'un de vos visiteurs par exemple.

2.1.7. Listes ordonnées et non-ordonnées

Les listes servent à ordonner du contenu, à lui donner un ordre cohérent.

Visuellement, les listes en HTML correspondent à ce que vous créez lorsque vous utilisez des puces dans un document LibreOffice :

- Élément numéro 1
- Élément numéro 2
- Élément numéro 3

En HTML, les listes vont avoir deux grands intérêts : on va pouvoir les utiliser pour créer des menus ou, dans leur forme brute, pour mieux présenter du contenu pour un blog par exemple.

Il existe trois grands types de listes en HTML : les listes ordonnées, les listes non-ordonnées et un dernier type un peu particulier : les listes de définition. Nous allons commencer avec les listes ordonnées et non-ordonnées.

La différence entre les listes ordonnées et non-ordonnées est que les listes ordonnées possèdent un aspect de subordination, d'ordre logique, de classement tandis que ce n'est pas le cas pour les listes non-ordonnées.

Pour créer une liste non-ordonnée, on va avoir besoin de deux nouveaux éléments : l'élément ul (abréviation de unordered list), qui va contenir toute la liste et l'élément li (pour list item) que l'on va utiliser pour créer chaque élément de la liste.

```
<html>
    <head>
        <title>L'univers merveilleux des listes</title>
        <meta charset="utf-8"/>
    </head>
    <body>
        <h1>Aujourd'hui, on parle de listes non-ordonnées</h1>
        <ul>
            <li>Fraise</li>
            <li>Route</li>
            <li>maison</li>
            <li>Chaussure</li>
        </ul>
    </body>
</html>
```

L'univers merveilleux des listes

file:///Users/Pierre/Desktop/Cours%20HTML/livre.html

Aujourd'hui, on parle de listes non-ordonnées

- Fraise
- Route
- maison
- Chaussure

Dans l'exemple précédent, les différents éléments de la liste n'ont pas de cohérence entre eux et on ne peux donc pas les classer, les ordonner (sans contexte du moins). C'est pourquoi on a utilisé une liste non-ordonnée.

Pour créer une liste ordonnée maintenant, nous allons simplement remplacer l'élément ul par l'élément ol (pour ordered list).

```
    <head>
        <title>L'univers merveilleux des listes</title>
        <meta charset="utf-8"/>
    </head>
    <body>
        <h1>Il est temps de traiter le cas des listes ordonnées</h1>
        <ol>
            <li>Introduction</li>
            <li>Partie I</li>
            <li>Partie II</li>
            <li>Partie III</li>
            <li>Conclusion</li>
        </ol>
    </body>
</html>
```

L'univers merveilleux des listes

file:///Users/Pierre/Desktop/Cours%20HTML/livre.html

Il est temps de traiter le cas des listes ordonnées

1. Introduction
2. Partie I
3. Partie II
4. Partie III
5. Conclusion

Cette fois-ci, il y a une relation de subordination, un ordre logique et naturel entre les éléments de la liste (on met généralement l'introduction avant la conclusion) ; on utilise donc une liste ordonnée.

2.1.8. Listes de définitions

Dernier grand type de listes que nous allons voir ensemble, les listes de définition son utilisées pour définir des termes.

Pour créer une liste de définition, il va nous falloir utiliser l'élément dl (pour definition list), l'élément dt (pour definition term) et l'élément dd pour la définition en soi.

Il n'y a qu'une règle à respecter lorsque l'on crée une liste de définitions : vous devez toujours placer l'élément dt avant l'élément dd, c'est-à-dire le terme à définir avant sa définition. Cela est assez intuitif et ne devrait donc pas vous poser de problème.

En revanche, il est tout à fait possible d'associer plusieurs termes à une définition ou même plusieurs définitions à un même terme.

```
<!DOCTYPE html>
    <html>
        <head>
            <title>L'univers merveilleux des listes</title>
            <meta charset="utf-8"/>
        </head>
        <body>
            <h1>Les listes de définition</h1>
            <dl>
                <dt>HTML</dt>
                <dd>HTML signifie HyperText Markup Language. Créé en 1991, le HTML...</dd>
            </dl>
        </body>
    </html>
```

Les listes de définition

HTML
HTML signifie HyperText Markup Language. Créé en 1991, le HTML...

2.1.9. Listes imbriquées

HTML nous offre la possibilité d'imbriquer les listes les unes dans les autres très simplement.

Pour imbriquer des listes, il suffit de commencer une liste, puis d'en ouvrir une seconde à l'intérieur d'un élément de la première (on peut évidemment imbriquer plus de deux listes en répétant le même processus).

Voici une illustration, en imbriquant par exemple une liste non-ordonnée à l'intérieur d'une liste ordonnée :

```html
<!DOCTYPE html>
    <html>
        <head>
            <title>L'univers merveilleux des listes</title>
            <meta charset="utf-8"/>
        </head>
        <body>
            <h1>Les listes imbriquées</h1>
            <ol>
                <li>Natation
                    <ul>
                        <li>Maillot</li>
                        <li>Bonnet</li>
                        <li>Lunettes</li>
                    </ul>
                </li>
                <li>Vélo</li>
                <li>Course à pied</li>
            </ol>
        </body>
    </html>
```

Les listes imbriquées

1. Natation
    - Maillot
    - Bonnet
    - Lunettes
2. Vélo
3. Course à pied

Comme vous le voyez, si on n'oublie pas d'élément, les listes restent très simples à utiliser et à manipuler. J'espère que vous comprenez mieux désormais l'importance de bien indenter son code !

2.1.10. Liens internes et liens externes

Tout d'abord, il faut savoir qu'il existe différents types de liens. Pour l'instant, nous allons nous concentrer sur les eux types les plus « classiques » : les liens internes et les liens externes.

Quelle différence entre ces deux types de liens ? Un lien interne est un lien créé entre deux pages d'un même site web tandis qu'un lien externe est un lien menant d'un site web vers un autre site web.

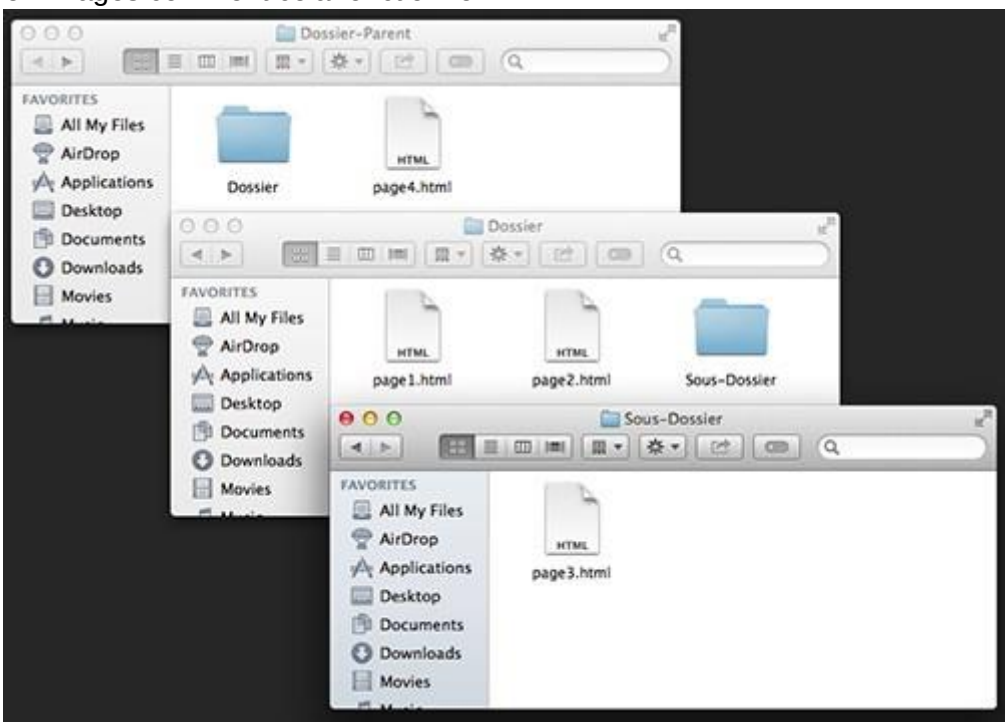
Dans tous les cas, pour créer un lien, nous allons utiliser l'élément « a » accompagné de son attribut href (pour Hypertext Reference) qui sert à indiquer la cible (c'est à dire la destination) du lien.

Quel que soit le type de liens créés, la seule chose qui va changer va être ce que l'on va indiquer en valeur pour l'attribut href.

Commençons donc avec les liens internes. Nous avons trois cas à distinguer :

1. La page à partir de laquelle on fait un lien et celle vers laquelle on fait un lien se trouvent dans le même dossier. Dans ce premier cas, il suffit de préciser le nom de la page dans l'attribut href.
2. La page vers laquelle on souhaite faire un lien se trouve dans un sous-dossier. Dans ce cas, il faudra en tenir compte et inclure le nom du sous-dossier dans la valeur de l'attribut href.
3. La page vers laquelle on veut faire un lien se trouve dans un dossier parent. Dans ce cas, nous devrons rajouter « ../ » dans la valeur de l'attribut href.

Voilà donc en images comment cela fonctionne :

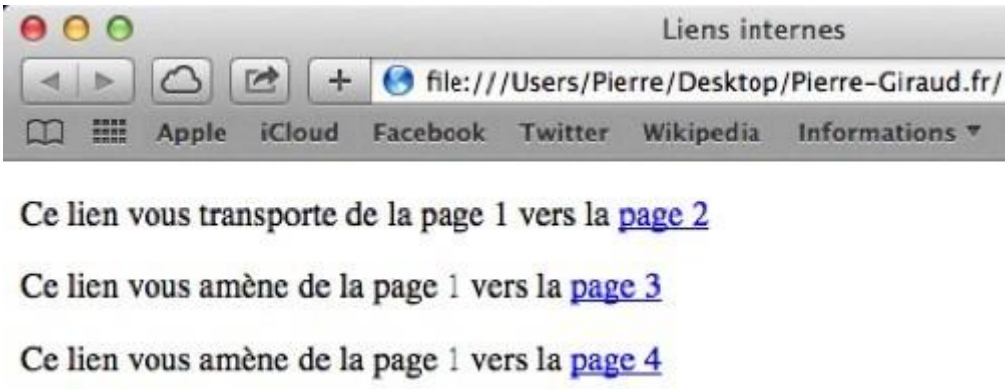


On a créé quatre pages en HTML (avec une structure minimale pour chacune d'entre elles afin de les rendre valide). On a placé les pages page1.html et page2.html dans le même dossier, la page page3.html dans un sous-dossier relativement à ma page page1.html et la page page4.html dans un dossier parent par rapport à ma page page1.html.

On va donc maintenant créer des liens de ma page page1.html vers mes pages page2.html, page3.html et page4.html grâce à l'élément a et à l'attribut href :

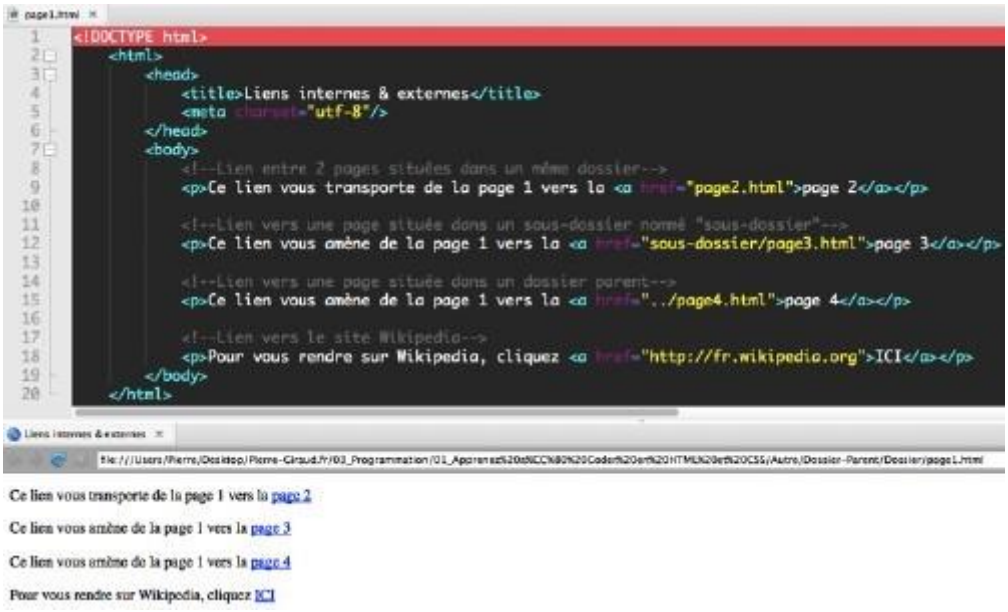
```
<!DOCTYPE html>
    <html>
        <head>
            <title>Liens internes</title>
            <meta charset="utf-8"/>
        </head>
        <body>
            <!--Lien entre 2 pages situées dans un même dossier-->
            <p>Ce lien vous transporte de la page 1 vers la <a href="page2.html">page 2</a></p>

            <!--Lien vers une page située dans un sous-dossier nommé "sous-dossier"-->
            <p>Ce lien vous amène de la page 1 vers la <a href="sous-dossier/page3.html">page 3</a></p>

            <!--Lien vers une page située dans un dossier parent nommé "dossier-parent"-->
            <p>Ce lien vous amène de la page 1 vers la <a href="../page4.html">page 4</a></p>
        </body>
    </html>
```

Si vous ouvrez votre page page1.html, celle-ci doit maintenant ressembler à cela :



Lorsque vous ou vos visiteurs cliquerez sur « page 2 », « page 3 » ou « page 4 », vous serez redirigés vers la page en question.

Pour créer des liens externes, maintenant, vous allez voir que c'est beaucoup plus simple : il suffit d'indiquer l'URL complète de la page vers laquelle on veut faire un lien en valeur de l'attribut href. En pratique, pour faire un lien vers la page d'accueil de Wikipédia par exemple, on écrira :



A noter qu'il existe pour les liens internes et externes des attributs facultatifs qui peuvent modifier le comportement par défaut de ces liens. C'est par exemple le cas de l'attribut target : l'attribut target permet de choisir si vous voulez que la cible de votre lien s'ouvre dans une nouvelle fenêtre / nouvel onglet ou pas.

Pour que la cible de votre lien s'ouvre dans une nouvelle fenêtre ou un nouvel onglet, on attribuera la valeur _blank à l'attribut target. Un exemple immédiatement en image :

Attribut à retenir donc, car celui-ci peut s'avérer très utile dans de nombreux cas (lorsque vous ne voulez pas que vos visiteurs quittent votre site par exemple). Notez en revanche que vous ne pouvez pas choisir si le lien va s'ouvrir dans un nouvel onglet ou dans une nouvelle fenêtre.

2.1.11. Les autres types courants de liens

Les liens internes et externes sont très certainement les types de liens les plus courants, mais c'est loin d'être les seuls ! En effet, on peut utiliser les liens pour faire bien d'autres choses.

Commençons avec les liens de type ancre. Les liens de type ancre sont des liens menant à un autre endroit d'une même page web. Ils peuvent être utile dans le cas d'une page web très longue pour donner à vos visiteurs un accès rapide à une section en particulier par exemple.

Vous comprendrez qu'il va donc tout d'abord nous falloir rajouter quelques lignes de textes dans notre page HTML pour pouvoir tester les ancres (sinon, on n'en verra pas l'effet).

Pour créer une ancre, on commence par rajouter un attribut id à une balise ouvrante HTML à l'endroit où l'on veut envoyer le visiteur. On peut attribuer n'importe quelle valeur à cet attribut, le mieux étant de choisir une valeur qui fasse sens.

Ensuite, on crée le lien cliquable en soi qui va amener à notre id. Pour cela, on utilise toujours notre élément a avec son attribut href (on ne réinvente pas la roue à chaque fois), mais cette fois ci on va devoir placer un dièse avant d'écrire la valeur de l'attribut href.

La valeur inscrite pour l'attribut href doit être strictement la même que celle donnée à notre id.

```
<!DOCTYPE html>
    <html>
        <head>
            <title>Les ancres</title>
            <meta charset="utf-8"/>
        </head>
        <body>
            <!--Imaginez un article bien documenté ou une dissertation-->
            <p>Sommaire (cliquez sur une partie pour y accéder directement)</p>
            <ol>
                <li><a href="#partie1">Partie I : nous allons parler du HTML</a></li>
                <li><a href="#partie2">Partie II : voyons ensemble le CSS</a></li>
                <li><a href="#partie3">Partie III : il est temps de s'attaquer au PhP !</a></li>
            </ol>

            <h2 id="partie1">Partie I : nous allons parler du HTML</h2>
            <p>Beaucoup de texte.Beaucoup de texte.Beaucoup de texte.</p>
            <p>Beaucoup de texte.Beaucoup de texte.Beaucoup de texte.</p>
            <p>Beaucoup de texte.Beaucoup de texte.Beaucoup de texte.</p>
            <p>Beaucoup de texte.Beaucoup de texte.Beaucoup de texte.</p>
            <p>Beaucoup de texte.Beaucoup de texte.Beaucoup de texte.</p>
            <h2 id="partie2">Partie II : voyons ensemble le CSS</h2>
            <p>Beaucoup de texte.Beaucoup de texte.Beaucoup de texte.</p>
            <p>Beaucoup de texte.Beaucoup de texte.Beaucoup de texte.</p>
            <p>Beaucoup de texte.Beaucoup de texte.Beaucoup de texte.</p>
            <p>Beaucoup de texte.Beaucoup de texte.Beaucoup de texte.</p>
            <p>Beaucoup de texte.Beaucoup de texte.Beaucoup de texte.</p>
            <h2 id="partie3">Partie III : il est temps de s'attaquer au PhP !</h2>
            <p>Beaucoup de texte.Beaucoup de texte.Beaucoup de texte.</p>
            <p>Beaucoup de texte.Beaucoup de texte.Beaucoup de texte.</p>
            <p>Beaucoup de texte.Beaucoup de texte.Beaucoup de texte.</p>
            <p>Beaucoup de texte.Beaucoup de texte.Beaucoup de texte.</p>
            <p>Beaucoup de texte.Beaucoup de texte.Beaucoup de texte.</p>

        </body>
    </html>
```

Faites attention à bien choisir des valeurs différentes pour chaque id sinon votre lien ne saura pas où ramener !

Pour envoyez un mail, maintenant, on donne tout simplement une valeur de type « mailto : » à notre attribut href comme ceci :

```html
<!DOCTYPE html>
<html>
    <head>
        <title>Pour envoyer un mail...</title>
        <meta charset="utf-8"/>
    </head>
    <body>
        <p>Envoyez moi <a href="mailto:monmail@gmail.com">un mail !</a></p>
    </body>
</html>
```

Enfin, voyons les liens permettant à vos visiteurs de télécharger un fichier. Pour cela, il va tout d'abord falloir nous armer d'un fichier (au format zip, pdf, ods ou autre) que l'on va placer dans le même dossier que la page web à partir de laquelle on crée le lien.

Ensuite, il ne reste plus qu'à créer un lien comme on en a l'habitude en utilisant un chemin relatif. Comme notre fichier et notre page web sont dans le même dossier, nous n'avons donc que le nom du fichier à renseigner en valeur de l'attribut href.

Par exemple, si mon fichier est un pdf qui s'appelle « fichier », on aura :

```html
<!DOCTYPE html>
<html>
    <head>
        <title>Télécharger un fichier</title>
        <meta charset="utf-8"/>
    </head>
    <body>
        <p>Téléchargez mon <a href="fichier.pdf">super fichier</a></p>
    </body>
</html>
```

2.1.12. Validation et compatibilité entre navigateurs

Un même code interprété par différents navigateurs peut produire différents résultats. Autrement dit, il est possible que votre code s'affiche différemment selon le navigateur utilisé par vos visiteurs.

Cela est de moins en moins vrai aujourd'hui avec les dernières versions des grands navigateurs qui sont toutes standardisées. De plus, nous verrons comment contourner ce problème par la suite avec les reset CSS entre autres.

Deuxième mise en garde ou plutôt bonne pratique, toujours dans cette idée d'avoir un code propre et qui fonctionne bien : pensez systématiquement à vérifier sa validité.

Pour cela, le W3C (World Wide Web Consortium – l'autorité en matière de codification des langages informatiques) met à notre disposition différents outils, et notamment des validateurs HTML et CSS.

Pour pouvez trouver ces validateurs ici :

- Pour le validateur HTML : http://validator.w3.org
- Pour le validateur CSS : http://jigsaw.w3.org/css-validator

2.2. Les bases du CSS

2.2.1. Sélecteurs, propriétés et valeurs

Pour rappel, le CSS sert à modifier l'apparence de nos pages web en appliquant des styles au contenu en HTML.

Un sélecteur, tout d'abord, va servir à déterminer à quel(s) élément(s) HTML ou à quel type d'éléments on souhaite appliquer un style particulier. Si l'on souhaite appliquer un style particulier à tous nos paragraphes, par exemple, on utilisera le sélecteur « p ».

Une propriété va nous servir à modifier le style d'un élément en ciblant un critère bien particulier comme la taille d'un texte, sa police ou sa couleur par exemple.

Une valeur, enfin, va venir compléter une propriété et va en déterminer le comportement. Pour la propriété servant à changer la couleur d'un texte par exemple, la valeur va être la nouvelle couleur à appliquer.

Voici ci-dessous une illustration concrète de ce que l'on vient de dire :

```
p{
    color : blue;
    font-size : 16px;
}
```

Dans cet exemple, nous utilisons le sélecteur simple « p », ce qui signifie que nous souhaitons appliquer un style particulier à tous les paragraphes de nos pages.

Nous utilisons les propriétés « color » (qui sert à modifier la couleur d'un texte) et « font-size » (pour changer la taille d'un texte). Cela signifie donc que nous travaillerons sur la couleur et la taille de nos paragraphes.

Enfin, nous indiquons que nous voulons que tous nos paragraphes s'affichent en bleu grâce à la valeur « blue » et que notre texte ait une taille de 16px avec la valeur « 16px ».

Notez d'ores-et-déjà la syntaxe de notre première déclaration en CSS. On entoure les propriétés et les valeurs avec des accolades et on place un point virgule après avoir spécifié une valeur pour chacune de nos propriétés. Chaque propriété est séparée de sa valeur par un deux-points.

2.2.2. Où écrire le CSS ?

Nous avons trois possibilités pour écrire notre CSS. L'une d'elles est préférable aux deux autres et nous allons immédiatement voir pourquoi.

Nous pouvons écrire le CSS :

- A l'intérieur de l'élément head de notre document HTML ; • Dans la balise ouvrante des éléments de notre fichier HTML ;
- Dans un fichier portant l'extension .css séparé.

Pour des raisons de performances du code, de clarté et d'économie de temps, je vous recommande vivement d'utiliser la dernière méthode dès que cela est possible.

Voyons ensemble comment cela se passe en pratique pour chacune de ces trois méthodes.

Commençons avec la première façon : écrire son code CSS dans l'élément head de notre page HTML. Pour faire cela, il suffit d'insérer un élément style dans notre élément head et de placer nos déclarations CSS à l'intérieur de cet élément style comme ceci :

```
<!DOCTYPE html>
    <html>
        <head>
            <title>Illustrations CSS</title>
            <meta charset="utf-8"/>
            <style>
                p{
                    color:blue;
                    font-size:16px;
                }
            </style>
        </head>

        <body>

        </body>
    </html>
```

Deuxième méthode maintenant : écrire du CSS dans la balise ouvrante d'un élément HTML. Pour faire cela, nous allons devoir utiliser un attribut style et lui affecter en valeur nos propriétés CSS :

```
<!DOCTYPE html>
    <html>
        <head>
            <title>Illustrations CSS</title>
            <meta charset="utf-8"/>
        </head>

        <body>
            <p style="color:blue;font-size:16px;">Un paragraphe</p>
        </body>
    </html>
```

Vous remarquerez que l'on respecte la syntaxe du CSS à l'intérieur de l'attribut style en utilisant les deux-points et les points virgules.

Attention cependant : dans notre premier cas, on utilisait l'élément style tandis que dans le cas présent, style est un attribut.

Troisième et dernière méthode enfin (la méthode recommandée) : écrire le code CSS dans un fichier séparé. Pour faire cela, nous allons déjà devoir ouvrir un nouveau fichier dans notre éditeur de texte et l'enregistrer au format « .css ». Vous pouvez le nommer « style.css ».

Pensez bien à enregistrer ce fichier dans le même dossier que votre fichier HTML dont vous souhaitez modifier le style, sinon vous risquez d'avoir des problèmes.

Une fois que vous avez fait cela, retournez sur votre page HTML. Nous allons maintenant devoir lier nos deux fichiers HTML et CSS. On va faire cela à l'aide d'un élément link que nous allons placer dans l'élément head de cette manière :

```
<head>
    <title>Illustrations CSS</title>
    <meta charset="utf-8"/>
    <link rel="stylesheet" href="style.css"/>
</head>
```

L'élément link est représenté sous forme de balise orpheline et doit être accompagné de ses deux attributs « rel » et « href ».

L'attribut rel sert à préciser le style du fichier lié (dans notre cas c'est une feuille de style, donc « stylesheet » en anglais). L'attribut href, que vous connaissez déjà, sert à faire le lien en soi.

Si le fichier avait été placé dans un dossier parent ou dans un sous-dossier par rapport à notre fichier HTML, il aurait fallu refléter cela dans la valeur de notre attribut href . Finalement, nous n'avons plus qu'à écrire notre code CSS dans le fichier style.css :

```
p{
    color : blue;
    font-size : 16px;
}
```

2.2.3. Les commentaires en CSS

On a déjà vu que l'on pouvait commenter en HTML. Cela est également possible en CSS.

Les fichiers CSS deviennent rapidement très long (beaucoup plus que les fichiers HTML par exemple) donc si vous ne commentez pas efficacement vous risquez d'être très vite perdu.

De plus, si vous distribuez votre code, je pense que la personne sera contente d'avoir quelques lignes de commentaires pour l'aider à trouver ce qu'elle cherche au milieu de 2000 lignes de code !

En CSS, on écrit les commentaires de cette manière :

```
/* Un commentaire en CSS*/
p{
    color : blue;
    font-size : 16px;
}
```

2.2.4. Sélecteurs simples et limitations

Jusqu'à présent, nous n'avons manipulé que des sélecteurs que l'on appelle « simple », car ils correspondent à des éléments HTML seuls et sans attributs (par exemple le sélecteur p).

Ce type de sélecteur doit être préféré tant que possible pour des raisons d'optimisation et de performance du code.

En effet, ils requièrent moins de code et sont donc moins gourmands en énergie que des sélecteurs plus complexes. Votre page mettra ainsi moins de temps à charger.

Le problème reste qu'on est quand même très limité avec des sélecteurs simples : comment faire pour appliquer un style différent à deux éléments de même type, deux paragraphe par exemple ?

Pour cela que l'on a créé les attributs class et id.

Class et Id sont deux attributs HTML qui ont été créés pour pouvoir appliquer différents styles à des éléments de même type. Class permet également de faire l'inverse et d'appliquer le même style à différents éléments choisis.

Premièrement on se place dans la balise ouvrante d'un élément HTML, on écrit le nom de notre attribut (class ou id), et on lui donne une valeur cohérente.

Cette valeur ne devrait contenir ni de caractères spéciaux (accents et autres) ni d'espace. Par exemple :

Ensuite, on retourne sur le fichier CSS. On va devoir commencer notre déclaration par un point là où on a utilisé un attribut class et par un dièse si l'on a utilisé l'attribut id.

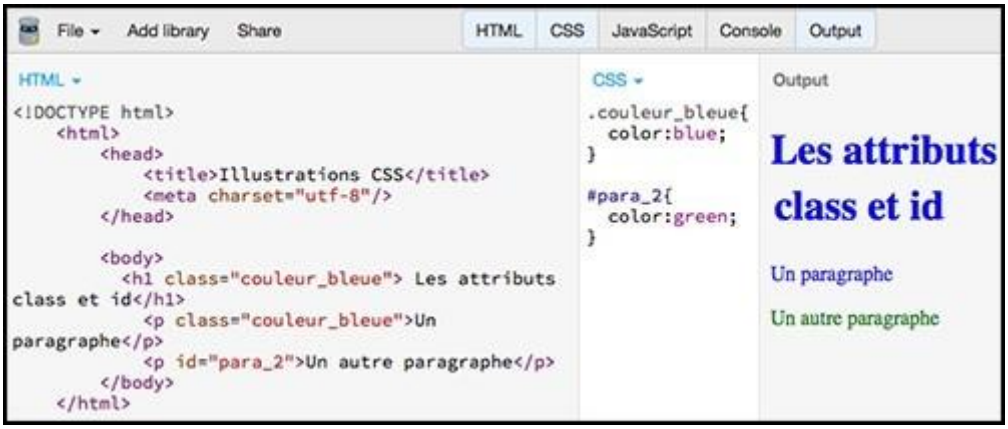
Après le point ou le dièse, on écrit la valeur de l'attribut en question pour former notre sélecteur. Enfin, on écrit le code CSS voulu. Voilà ce que ça donne en pratique :

Nous pouvons maintenant appliquer un style différent à chaque élément HTML grâce aux attributs class et id.

Pourquoi avoir créé deux attributs pour faire la même chose ? En fait, il existe une différence notable entre class et id : un attribut id avec une valeur précise ne peut être utilisé qu'une fois dans une page, au contraire de class.

Id sera donc utilisé pour des éléments uniques dans une page web, comme le logo de votre site par exemple.

En revanche, on peut utiliser plusieurs attributs class identiques (c'est à dire ayant la même valeur) par page. C'est d'ailleurs une des méthodes que nous utiliserons pour appliquer un même style à différents éléments.



Notez que, dans l'exemple précédent, utiliser deux attributs class n'est pas la meilleure solution.

Nous voilà déjà un peu moins limités. Cependant, nous ne pouvons pour le moment appliquer un style qu'à un contenu entre balises. Effectivement, on ne pourrait pas appliquer de style particulier au mot « attributs » de notre titre dans l'exemple précédent.

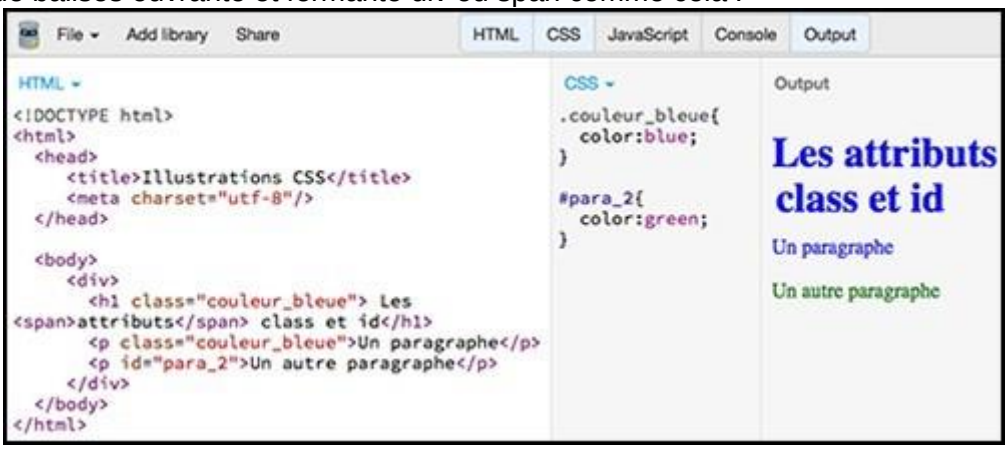
Pour remédier à cela, on a inventé les deux éléments HTML div et span.

2.2.5. Les éléments div et span

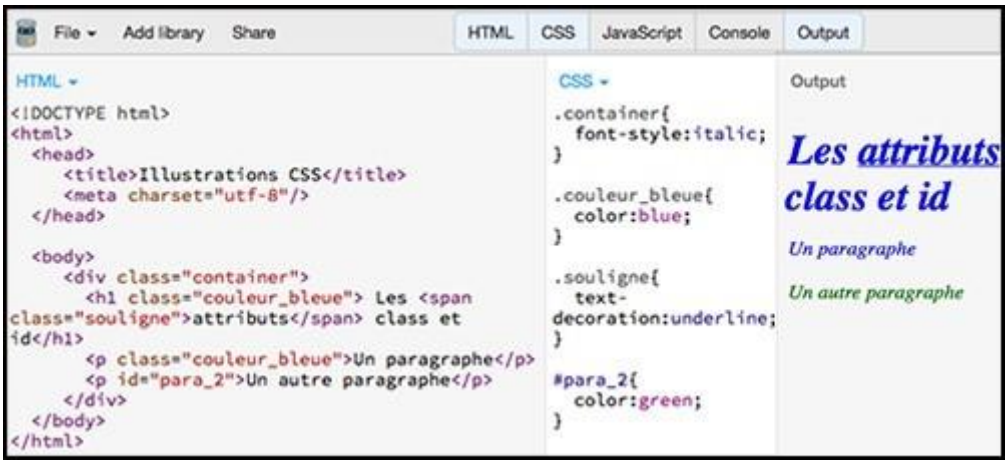
Les éléments div et span ne possèdent aucune valeur sémantique, ce qui va à l'encontre même du rôle du HTML. Ainsi, vous ne devez les utiliser que lorsque vous n'avez pas d'autre choix.

Les éléments div et span vont nous servir de containers. Nous allons nous en servir pour entourer des blocs de code et ainsi pouvoir attribuer des styles particuliers à ces blocs.

L'utilisation des éléments div et span est très simple : il suffit d'entourer le bloc de code voulu avec une paire de balises ouvrante et fermante div ou span comme cela :



Généralement, on attribuera une class ou un id à div et span afin de pouvoir différencier nos différents div et span dans notre page. Ainsi, on peut désormais appliquer un style particulier à n'importe quel bout de code dans notre page HTML.



Tout comme pour class et id, il existe une différence entre div et span : div est un élément de type block tandis que span est un élément de type inline.

2.2.6. Les éléments de type block et inline

En HTML, tout élément est soit de type block, soit de type inline. Par exemple, div est un élément de type block tandis que span est un élément de type inline.

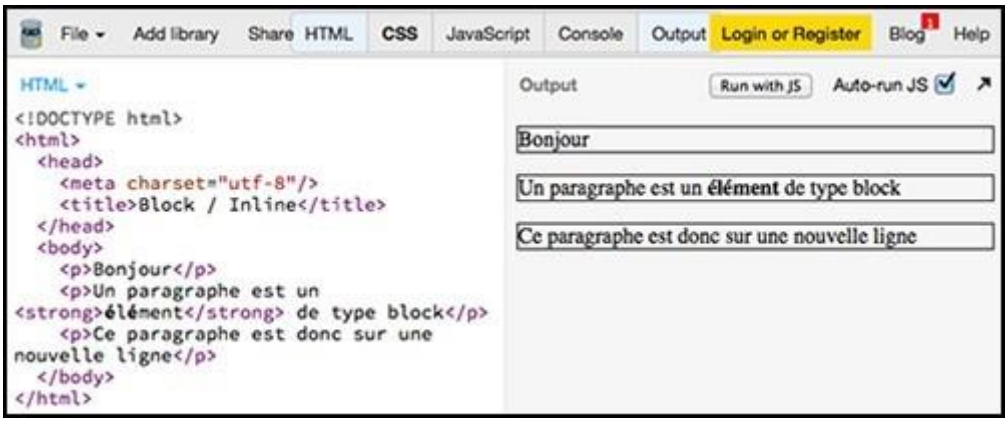
Les éléments de type block sont fondamentalement différents des éléments de type inline en HTML et il est essentiel de bien comprendre les différences entre ces deux types si vous voulez un jour créer un site Internet, ne serait-ce que pour des raisons de mise en page.

Les éléments de type block…	Les éléments de type inline…
Commencent sur une nouvelle ligne	S'insèrent dans une ligne
Occupent toute la largeur disponible	Occupent seulement la largeur nécessaire
Peuvent être imbriqués les uns dans les autres et contenir des éléments de type inline	Peuvent être imbriqués les uns dans les autres mais ne peuvent pas contenir d'éléments de type block

Afin que vous compreniez bien la différence entre les deux types d'éléments, voyons ensemble quelques exemples d'éléments de type inline ou block pour que vous puissiez observer leur comportement.

Eléments de type block	Eléments de type inline
p	em
h1, h2, h3, h4, h5, h6	strong
ol, ul,dl	mark
li	a
table	img

Illustrons ce que nous venons de dire en regardant par exemple de plus près le comportement des éléments p et em :



On voit bien dans l'exemple ci-dessus les différences citées dans cette partie entre les éléments de type block et inline. On a mis des bordures autour des paragraphes afin que vous soyez bien convaincu qu'un élément de type block occupe toujours toute la largeur disponible.

2.2.7. Les sélecteurs avancés

Les sélecteurs avancés sont l'une des grandes forces du CSS. En effet, grâce à eux, nous allons pouvoir cibler du contenu très précisément et assez simplement.

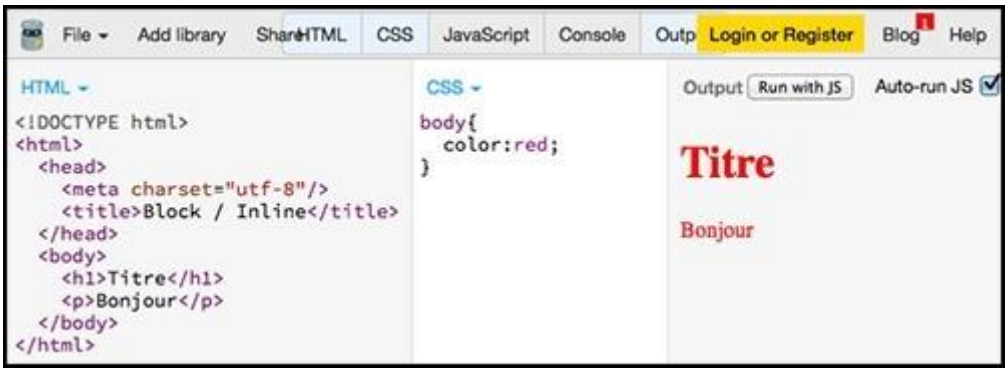
Il faut savoir qu'il existe de très nombreux sélecteurs avancés en CSS et on ne présentera que les plus utiles :

Ce sélecteur sert à…	Et il s'écrit comme cela…
Sélectionner tous les éléments (sélecteur universel)	*
Sélectionner deux éléments A et B	A, B
Sélectionner un élément B contenu dans un élément A	A B
Sélectionner le premier élément B suivant un élément A	A + B
Sélectionner tous les éléments A possédant un attribut particulier	A[nom de l'attribut]
Sélectionner tous les éléments A possédant un attribut particulier avec une valeur	A[nom de l'attribut* = « valeur »]
Sélectionner tous les éléments A possédant un attribut particulier avec une valeur précise	A[nom de l'attribut = « valeur »]

2.2.8. La notion d'héritage

L'héritage est une notion centrale et fondamentale du CSS. L'héritage signifie que tout élément HTML va hériter des styles de ses parents (c'est le fameux « cascading »).

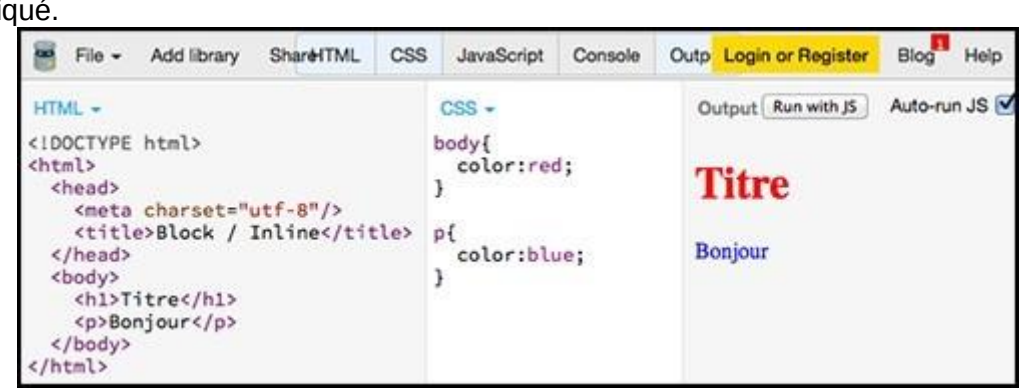
En HTML, si un élément A est inclus dans un élément B ; l'élément A s'appellera l'enfant et l'élément B sera le parent de l'élément A. Ainsi, si l'on applique un style à l'élément B, l'élément A en héritera automatiquement.



Ici, l'élément body est le parent des éléments h1 et p, puisque les éléments h1 et p sont bien contenus dans l'élément body. Ainsi, lorsqu'on applique un style à l'élément body (ici, mettre le texte en rouge), les éléments p et h1 héritent automatiquement de ce style.

Que se passe-t-il lorsque l'on donne deux ordres contradictoires à un élément parent et à son enfant en CSS (par exemple, donner une couleur rouge au parent et bleue à l'enfant) ?

Le CSS possède ici sa logique et le style appliqué sera celui le plus proche de l'élément en question. Cela signifie que si on applique un style à un élément enfant, c'est bien ce style qui lui sera appliqué.



3. Formater du Texte & Positionner des Éléments

3.1. Les Propriétés de Type « Font- »

3.1.1. La propriété font-size

Nous utiliserons la propriété font-size lorsque nous voudrons modifier la taille d'un texte.

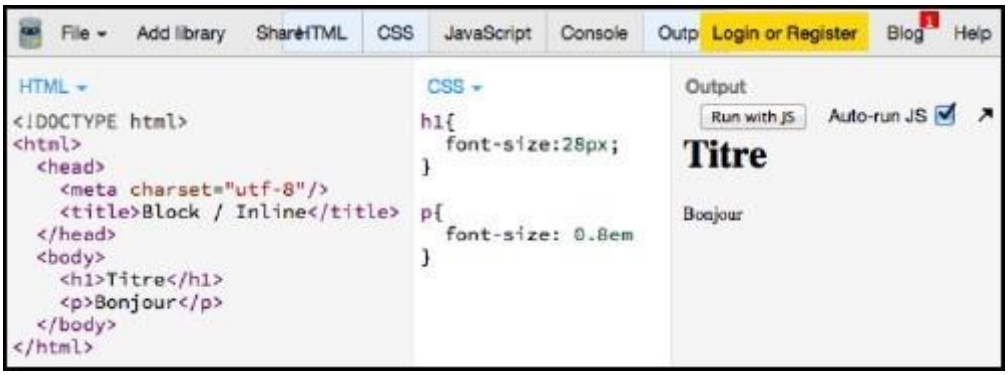
Cette propriété accepte deux types de valeurs : des valeurs de type absolu (en pixel ou en point), ou relatif (en em, ex ou en pourcentage).

Les valeurs de type « relatives » sont appelées de la sorte car elles permettent au texte de s'adapter relativement aux préférences de vos visiteurs. En clair, si vous fixez la taille d'un texte à 100%, ce texte pourra avoir des tailles différentes selon les réglages faits par vos visiteurs sur leurs navigateurs.

Ce type de valeur présente des avantages indéniables, et notamment le fait que tous vos visiteurs devraient être capables de lire vos écrits sans difficulté. De plus, le texte peut également s'adapter relativement aux autres éléments de votre page web.

Une valeur de type absolu, en revanche, va fixer la taille d'un texte définitivement. Le grand avantage de ce type de valeur est que vous pouvez contrôler précisément donc le rendu de votre texte et de votre page web.

Voici un exemple d'utilisation de la propriété font-size, avec des valeurs relative et absolue. Notez qu'on utilisera les notations px pour pixel, pt pour point et % pour pourcentage.



3.1.2. La propriété font-style

La propriété font-style permet de fixer l'inclinaison d'un texte.

La propriété font-style accepte 4 valeurs différentes :

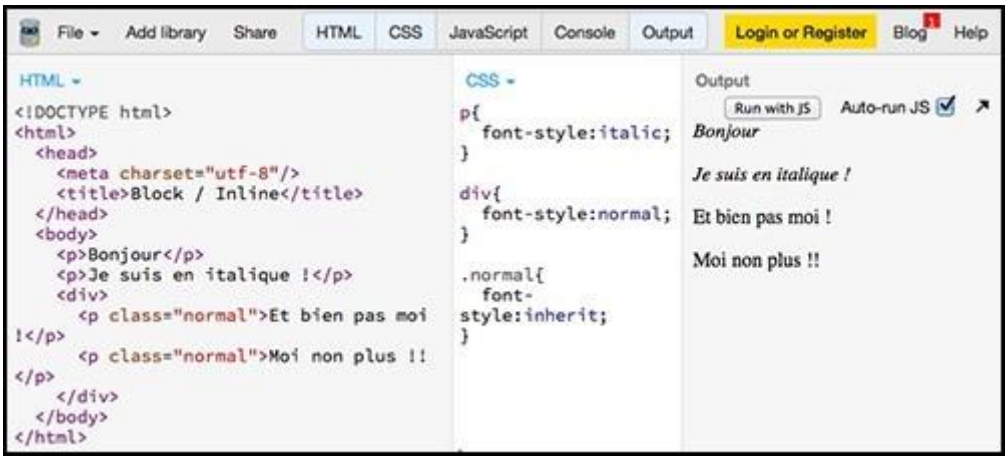
- Normal (valeur par défaut) ;
- Italic (change le texte en italique) ;
- Oblique (penche le texte) ;
- Inherit (hérite des propriétés de l'élément parent).

Nous avons dit plus haut que tout élément enfant héritait par défaut des styles de ses parents. A quoi sert donc la valeur inherit ? En fait, cette dernière sert à annuler un style dans un cas bien particulier.

Imaginez par exemple que nous ayons défini un font-style : italic pour tous les paragraphes de notre page et un font-style : normal à un div en particulier.

Par définition, les paragraphes à l'intérieur de ce div seront en italique puisque la notion d'héritage nous dit d'appliquer la règle la plus proche de l'élément en question (et pour un élément de type p, on applique donc les règles portant sur p en premier).

Si l'on souhaite que les paragraphes héritent du même style que notre div, l'utilisation de la valeur inherit est alors une excellente solution.



Certains d'entre vous vont certainement dire « très bien, mais il était aussi rapide dans ce cas d'indiquer un font-style = normal à notre attribut class normal ». C'est tout à fait vrai.

Cependant, imaginez maintenant que vous connaissiez le PHP ou le JavaScript et que vous laissiez la possibilité à vos visiteurs de choisir le font-style du div, mais que vous voulez absolument que les paragraphes à l'intérieur de ce div aient le même font-style que leur parent.

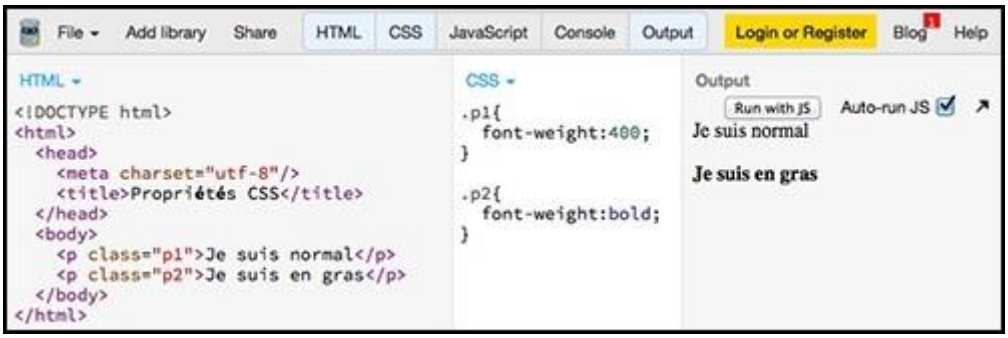
Pour faire cela, le plus simple est d'utiliser inherit.

Concernant les valeurs italic et oblique, vous pourrez constater par vous mêmes qu'elles rendent un résultat visuellement quasi-identique. Cependant, l'italique est un état spécial d'une police, et qui n'est pas supporté par toutes les polices tandis que l'oblique correspond à une inclinaison forcée de la version normale d'une police, et peut donc être appliqué à toutes les polices.

3.1.3. La propriété font-weight

La propriété font-weight permet de fixer le poids d'un texte. Cette propriété accepte 6 valeurs différentes :

- Normal (la valeur par défaut) ;
- Lighter (version allégée de la police) ;
- Bold (la police est en gras) ;
- Bolder (la police est encore plus en gras) ;
- Une centaine compris entre 100 et 900 (du plus léger au plus gras) ;
- Inherit (hérite des styles de ses parents).



A noter que certaines polices et certains navigateurs ne supportent pas les valeurs inférieures à « normal » et supérieures à « bold ». Si votre valeur n'est pas supportée, la police s'affichera dans la valeur la plus proche supportée (pour bolder, votre police s'affichera en bold par exemple).

3.1.4. La propriété line-height

Cette propriété n'est pas une propriété de type « font- » à proprement parler mais concerne également la mise en forme du texte, donc elle a tout à fait sa place dans cette partie.

La propriété line-height sert à fixer l'écartement, c'est-à-dire la distance entre deux lignes de texte. Une pratique communément admise est d'indiquer une valeur pour notre line-height équivalent à 1,5 fois la taille en pixel de notre texte.

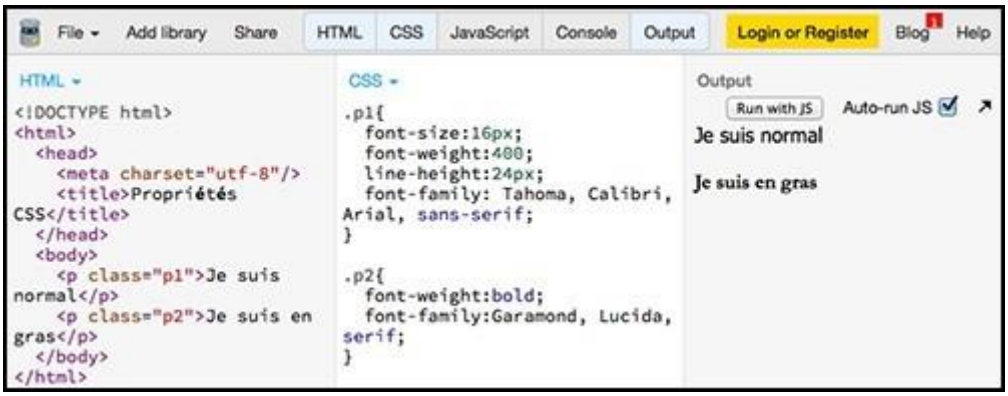
3.1.5. La propriété font-family

La propriété font-family permet de choisir la police du texte.

Dans tous les cas, nous déclarerons plusieurs polices (on parle de « famille » de polices, d'où le nom de cette propriété) afin de s'assurer qu'au moins une des polices mentionnées soit supportée par vos visiteurs.

En effet, il existe toujours des versions de navigateurs et des ordinateurs ne supportant pas certaines polices, d'où tout l'intérêt d'en déclarer plusieurs.

La première police déclarée sera le choix par défaut. Si elle n'est pas lue par votre visiteur, alors on utilisera la seconde et etc.



3.1.6. Les « web safe fonts »

Les Web Safe Fonts correspondent à un ensemble de polices, un « pack » qui est préinstallé sur toutes les machines permettant d'aller sur Internet. Ce sont donc des polices lues par tout le monde, d'où le mot « safe » qui veut dire en français « sûr », « hors de danger ».

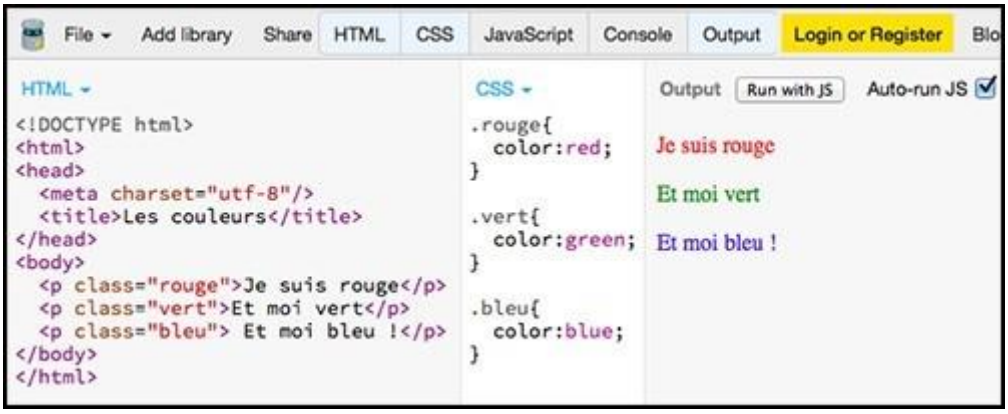
En général, on mentionnera au moins une police Web Safe comme dernière valeur de la propriété font-family.

Parmi les Web Safe Fonts, nous avons les polices Arial, Times New Roman, Courier New, Verdana, Georgia, Lucida, Tahoma, Trebuchet et Garamond entre autres.

3.1.7. La propriété color

Il existe plusieurs façons de gérer la couleur d'un texte.

Première façon de changer la couleur d'un texte : en attribuant un nom de couleur (en anglais) en valeur de la propriété color.



Cette méthode reste très limitée car nous ne pouvons choisir que parmi seize noms de couleurs qui sont les suivants :

Les valeurs que vous voyez sous le nom des couleurs sont des valeurs qu'on appelle hexadécimales. Ce type de valeur va correspondre à notre deuxième méthode pour fixer la couleur d'un texte.

« Hexadécimal » signifie « qui fonctionne en base 16 ». Pour le dire simplement, un système hexadécimal est un système qui utilise 16 symboles pour compter, à savoir dans notre cas les nombres de 0 à 9 et les lettres de A à F. La lettre A correspondra ainsi à 10, B à 11, C à 12, D à 13, E à 14 et F à 15. Cela nous fait donc bien 16 symboles en comptant le zéro.

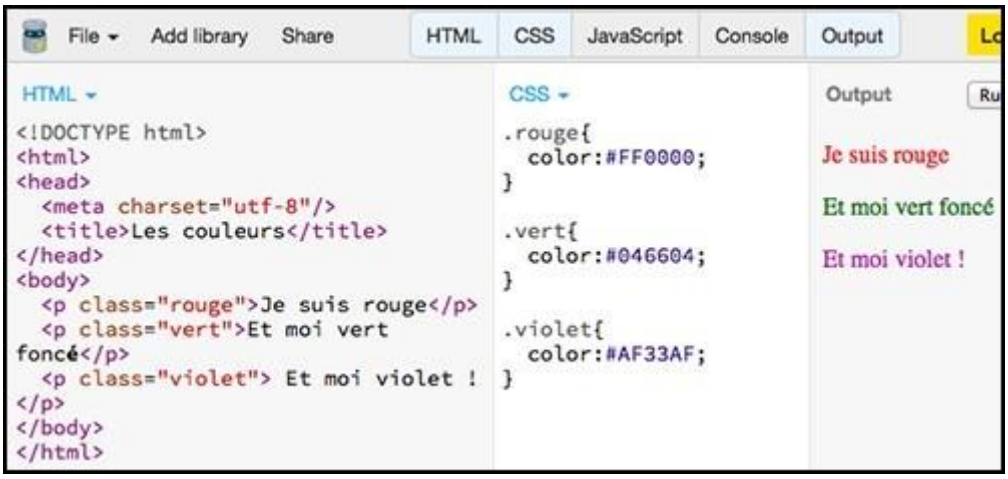
Ce système de comptage peut paraître complexe et assez étrange pour ceux qui n'ont pas fait d'études de mathématiques, je veux bien le comprendre. Cependant, je vous demande de ne pas en avoir peur et de le voir tel qu'il est : un système pour compter de 0 à 15 en soi très simple.

Mais quel rapport avec nos couleurs ? En fait, on va pouvoir donner une valeur hexadécimale en valeur de la propriété color, afin de choisir très précisément la couleur voulue. Cette valeur devra commencer par un dièse, suivie de 6 symboles choisis entre 0 et F.

Les deux premiers symboles vont définir l'intensité de rouge de notre couleur, les deux suivants l'intensité de vert et les deux derniers l'intensité de bleu. Vous pouvez imaginer que cela se passe comme lorsque vous mélangez de la peinture afin d'obtenir une couleur précise.

L'intérêt de ce type de valeur est de pouvoir choisir parmi plus de 16 millions de nuances différentes. Si vous avez bien suivi, l'échelle d'intensité va de 0 à 255 pour chaque binôme de valeurs, 00 étant l'intensité la plus faible et FF l'intensité la plus forte.

Pourquoi de 0 à 255 ? Considérez que 00 correspond à 0 sur notre échelle d'intensité. 01 correspond à 1, 02 à 2, 0A à 10 et 0F à 15. Après 0F, on a 10 (lisez « un, zéro » pour comprendre plus clairement) qui correspond à 16 ; 11 qui correspond à 17 ; 1A qui correspond à 26 et etc. jusqu'à FF qui correspond à 255. Cela fonctionne finalement comme pour nos dizaines, à part qu'ici on compte en seizaines.



Le plus important est encore une fois de retenir que les deux premiers symboles correspondent à l'intensité de rouge, les deux suivants à l'intensité de vert et les deux derniers à l'intensité de bleu.

Enfin, le dernier type de valeurs que l'on peut utiliser est le type RGB (pour Red Green Blue). Nous allons voir que ce type de valeurs et les valeurs hexadécimales reposent sur la même base.

Cette fois-ci, nous allons devoir indiquer trois nombres compris entre 0 t 255 en valeur. Le premier nombre correspond une nouvelle fois à l'intensité de rouge, le second à l'intensité de vert et le troisième à l'intensité de bleu que l'on veut utiliser pour former notre couleur finale.

Il existe de nombreuses similitudes entre les notations en hexadécimal et en RGB. On peut d'ailleurs tout à fait convertir l'hexadécimal en RGB, en partant du principe que 00 en hexa correspond à 000 en RGB et que FF en hexa correspond à 255 en RGB.

3.1.8. L'opacité d'un texte

Le CSS nous offre deux méthodes pour fixer le niveau d'opacité des textes.

Si on utilise une valeur de type nom de couleur ou hexadécimale avec la propriété color, on doit utiliser la propriété opacity pour fixer l'opacité des textes.

Cette propriété prend une valeur entre 0 (totalement transparent) et 1 (totalement opaque). Attention : n'oubliez pas qu'on utilise des points et non pas des virgules pour les nombres à virgule étant donné qu'on utilise le système d'écriture anglo-saxon lorsque l'on code.

Si nous avons utilisé une valeur de type RGB avec la propriété color, nous pouvons également maîtriser l'opacité de notre texte d'une façon plus simple, en utilisant une valeur de type RGBa. Dans ce cas, il suffit de rajouter une valeur pour l'opacité de notre texte après les trois valeurs de notre propriété RGB.

3.2. Les Propriétés de Type « Text- »

3.2.1. L'alignement d'un texte

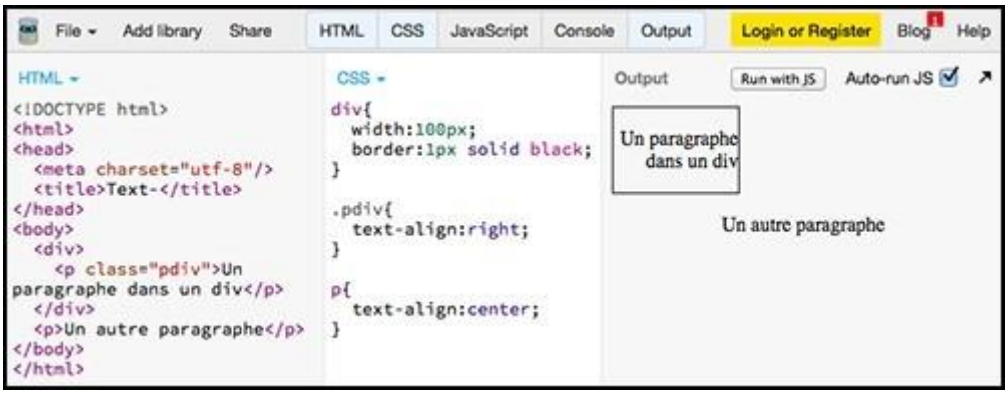
Pour modifier l'alignement d'un texte, nous allons utiliser la propriété text-align.

Cette propriété peut prendre cinq valeurs différentes :

- Left : le texte sera aligné sur la gauche ; valeur par défaut ;
- Center : le texte sera centré ;
- Right : le texte sera aligné sur la droite ;
- Justify : le texte sera justifié ;
- Inherit : hérite des propriétés de l'élément parent.

Le centrage ou l'alignement se fait toujours par rapport à l'élément parent le plus proche du texte.

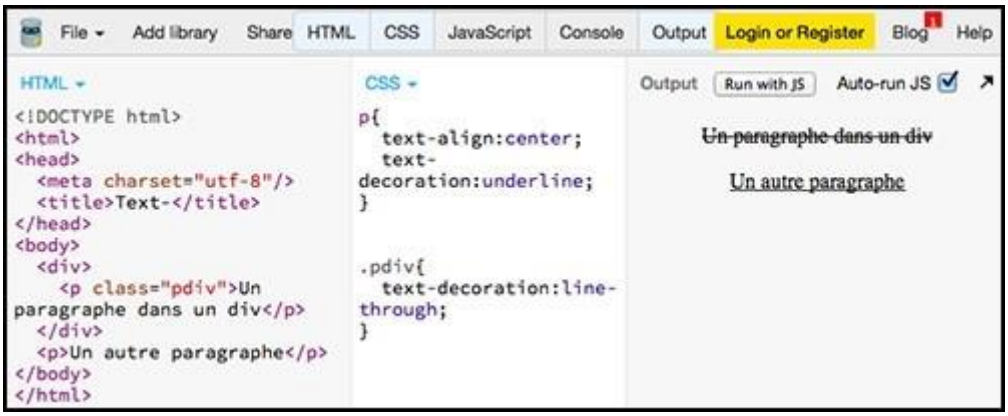
Dans l'exemple suivant, on voit bien que mon paragraphe « pdiv » est aligné à droite de son élément parent (c'est-à-dire le div qui fait lui-même 100px de large) et non pas de la page. Le second paragraphe, n'ayant pour parent que l'élément body, est donc bien lui centré sur la page.



3.2.2. La propriété text-decoration

On peut modifier la décoration d'un texte grâce à la propriété text-decoration parmi six valeurs pour cette propriété :

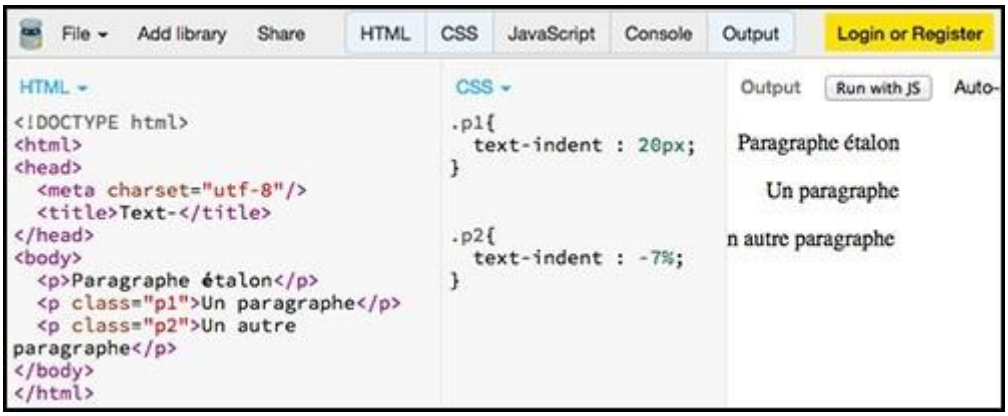
- Underline : le texte sera souligné ;
- Overline : une ligne apparaîtra au dessus du texte ;
- Line-through : le texte sera barré ;
- Blink : le texte clignotera (attention, ne fonctionne pas sur tous les navigateurs) ;
- Inherit ;
- None : pas de décoration, comportement par défaut.



3.2.3. La propriété text-indent

La propriété text-indent sert à gérer l'indentation d'un texte. L'indentation, pour ceux qui l'auraient oublié, est le décalage d'un texte sur la droite généralement.

Cette propriété accepte des valeurs de type absolu (px, pt) et relatif (em, ex, %). Vous pouvez également lui attribuer des valeurs négatives afin de décaler votre texte sur la gauche.

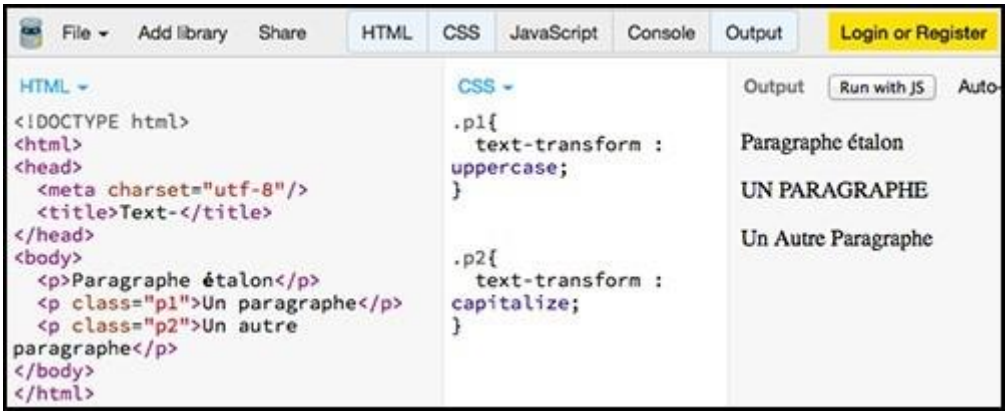


3.2.4. La propriété text-transform

On utilise la propriété text-transform pour modifier l'aspect des caractères d'un texte (majuscules ou minuscules).

Nous pouvons choisir parmi cinq valeurs :

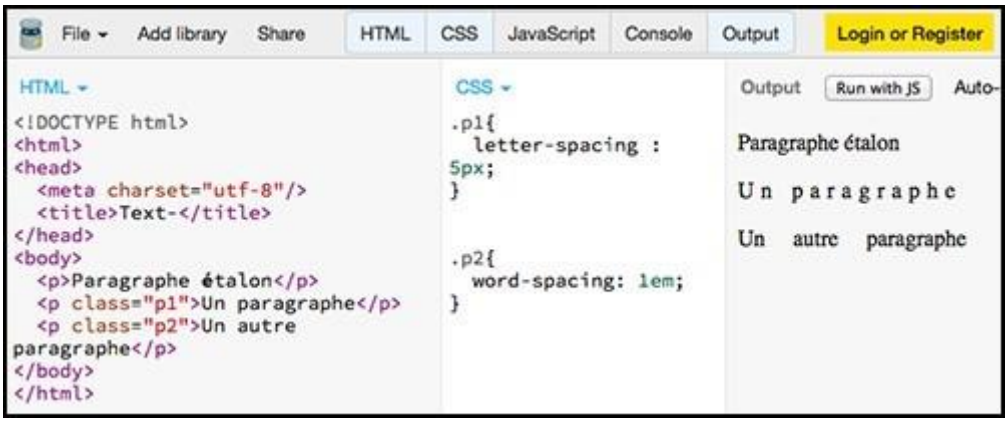
- Uppercase : transforme tout le texte en majuscules ;
- Lowercase : met tout le texte en minuscules ;
- Capitalize : met uniquement la première lettre de chaque mot en majuscule ;
- None : pas de transformation ;
- Inherit : hérite des styles de l'élément parent.



3.2.5. Les propriétés letter-spacing et word-spacing

Les propriétés letter-spacing et word-spacing permettent respectivement d'ajuster l'espace entre les lettres et entre les mots.

Ces propriétés se comportent de manière similaire et acceptent des valeurs absolues (px, pt) et relatives (ex, em, %).



3.2.6. Les ombres des textes

On peut ajouter des effets d'ombre à un texte en utilisant la propriété text-shadow.

Cette propriété nécessite au minimum deux valeurs pour fonctionner. Cependant, dans la grande majorité des cas, nous en utiliserons quatre, dans l'ordre vu ci-dessous.

Les trois premières valeurs correspondent à des longueurs et la dernière est la couleur de l'ombre :

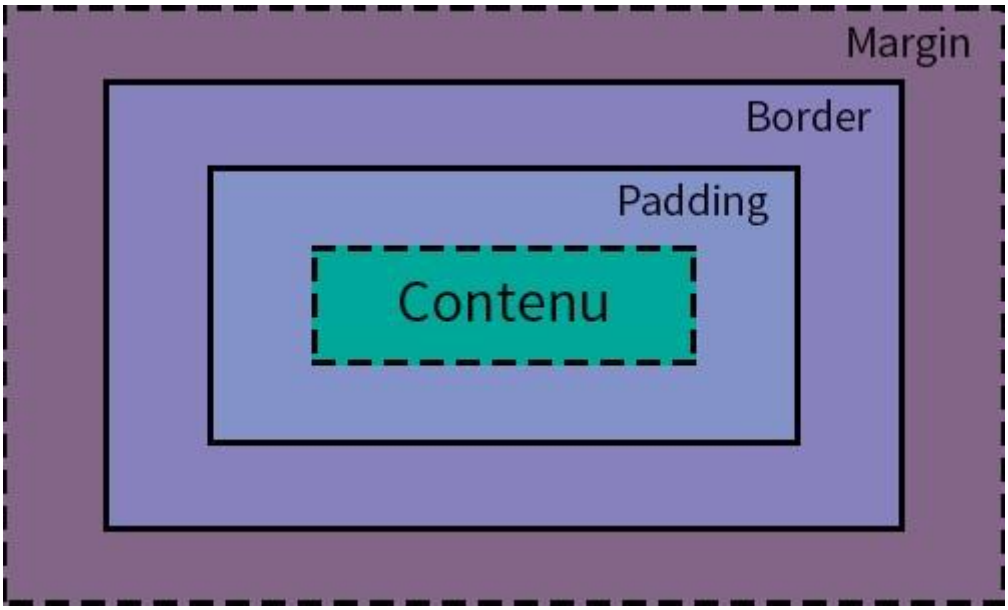
- 1ère valeur : déplacement horizontal de l'ombre ;
- 2ème valeur : déplacement vertical de l'ombre ;
- 3ème valeur : rayon de propagation (flou gaussien) de l'ombre ;
- 4ème valeur : couleur de l'ombre (accepte les mêmes valeurs que la propriété « color »).

3.3. Le Modèle des Boîtes

Le modèle des boîtes est un concept essentiel : « tout élément d'une page est une boîte rectangulaire et peut avoir un padding, une marge et des bordures ».

Cela mérite d'être répété : tout élément, qu'il soit un élément de type block ou de type inline, est une boîte rectangulaire.

Les propriétés permettant d'indiquer la longueur, la largeur, la marge, le padding et les bordures d'un élément forment ce qu'on appelle le modèle des boîtes.



La première boîte est définie par la longueur et la largeur d'un élément. La padding, ou marge intérieure, forme ensuite la seconde boîte. Puis viennent les bordures qui constituent la troisième boîte. Enfin, la marge extérieure vient former la quatrième et dernière boîte.

3.3.1. Hauteur et largeur d'un élément

Tout élément possède une hauteur et une largeur par défaut.

La hauteur d'un élément est déterminée par son contenu. Par exemple, des paragraphes d'une ligne ou de deux lignes n'occuperont pas la même hauteur.

La largeur par défaut d'un élément est avant tout déterminée par son type (block ou inline) puis par son contenu si l'élément est de type inline. En effet, rappelez vous que les éléments de type block occupent automatiquement toute la largeur disponible.

Pour modifier la hauteur d'un élément, on utilise la propriété height à laquelle on attribue une valeur en px, % ou égale à auto dans la grande majorité des cas.

En utilisant la valeur auto, on laisse le navigateur de nos visiteurs décider de la hauteur que doit prendre l'élément visé. Cela est très utile dans le cas où l'on veut conserver les proportions d'une image tout en l'adaptant à la taille de l'écran de nos visiteurs.

Pour modifier la largeur d'un élément, on utilise cette fois la propriété width. Cette propriété prend les mêmes types de valeurs que height.

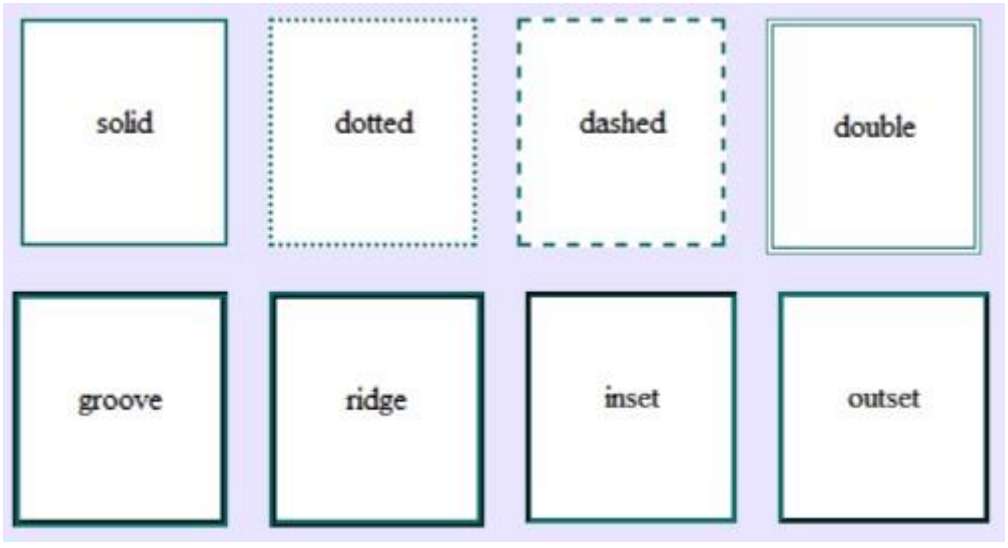
3.3.2. Les bordures et les bordures arrondies

Il existe de nombreuses sortes de bordures dont certaines sont plus ou moins bien supportées par certains navigateurs.

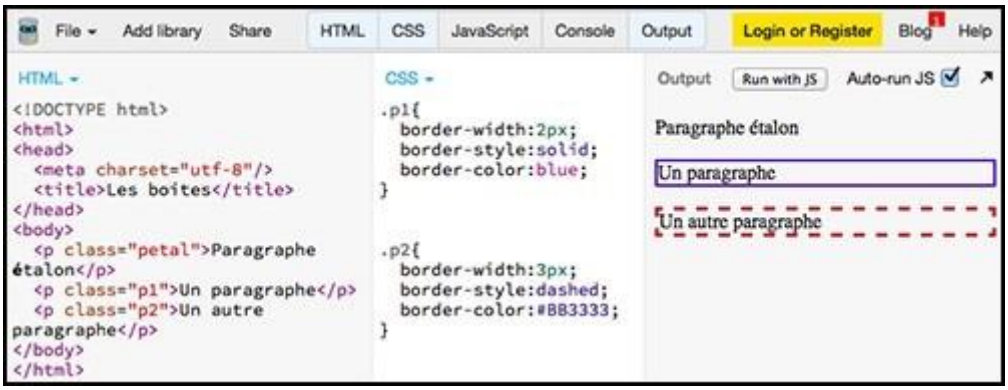
Pour créer des bordures et les personnaliser, nous allons avoir besoin de trois propriétés :

- Border-width, qui va définir l'épaisseur de la bordure (valeur en px) ;
- Border-style, qui va définir le style de la bordure ;
- Border-color, qui va définir la couleur de la bordure (accepte les mêmes valeurs que la propriété « color »).

La propriété border-style peut prendre de nombreuses valeurs différentes. Les valeurs les plus utilisées sont solid, dotted (pointillé) et dashed (tiret). Vous pouvez voir ci-dessous le résultat pour chaque style de bordure :



Et voici un exemple d'utilisation des propriétés de type border :



Une fonctionnalité longtemps attendue par les webmasters et les développeurs et qui a vu le jour avec le CSS3 est la possibilité de créer des bordures arrondies.

Pour faire cela, nous allons utiliser la propriété border-radius. Cette propriété va prendre une valeur : la taille de l'angle, en px.

Notez que l'on peut définir des angles différents pour chaque côté de nos bordures en utilisant les mots clefs top-left, top-right, bottom-left et bottom-right.

Notez également que ces mots clefs sont souvent employés en CSS, ensemble ou séparément (que top, ou que left par exemple). N'hésitez donc pas à les tester avec certaines propriétés déjà vues ou que l'on va voir !

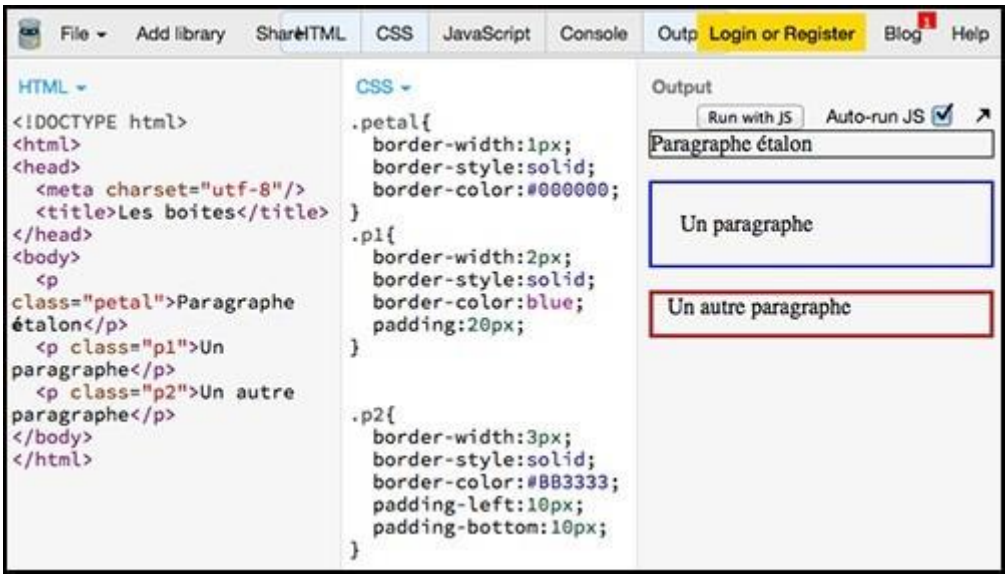
3.3.3. Les marges intérieures

Pour définir les marges intérieures d'un élément, nous utiliserons la propriété padding.

On peut considérer qu'un élément HTML possède toujours une bordure. Celle-ci peut être explicite, c'est-à-dire matérialisée à l'aide des propriétés CSS vues précédemment ou implicite (invisible).

La propriété padding va définir l'espace entre l'élément en soi et sa bordure. Cette propriété doit être utilisée uniquement dans ce but, et jamais pour positionner des éléments dans une page ou les uns par rapport aux autres.

On donnera généralement une valeur en px à padding. Notez que l'on peut définir des espacements différents pour chaque marge intérieure de nos éléments en utilisant les propriétés padding-right, padding-bottom, padding-left et padding-top.



3.3.4. Les marges extérieures

Pour définir la taille des marges extérieures, c'est-à-dire de l'espace à l'extérieur des bordures d'un élément, nous allons utiliser la propriété margin.

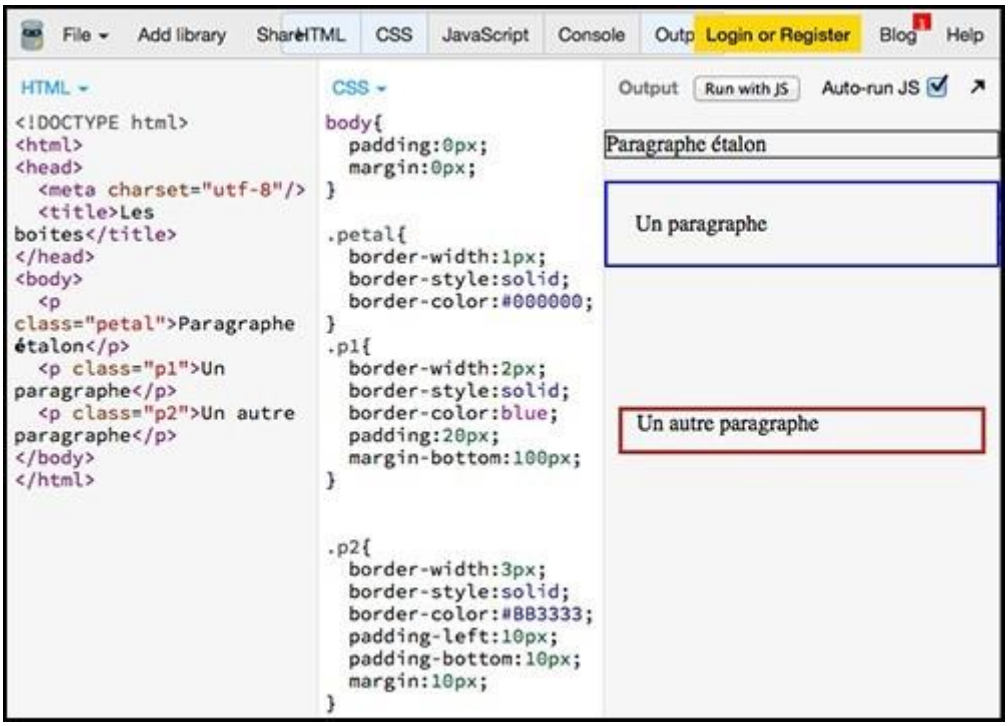
Contrairement à la propriété padding, la propriété margin peut tout-à-fait être utilisée pour positionner des éléments dans une page ou les uns par rapport aux autres.

Nous attribuerons généralement des valeurs en px ou en % à cette propriété. Tout comme la propriété padding, nous allons pouvoir des marges différentes de chaque côté de nos éléments avec les propriétés margin-right, margin-bottom, margin-left et margin-top.

Notez que les valeurs par défaut des marges intérieures et extérieures peuvent légèrement différer d'un navigateur à un autre. Cela peut impacter le design général de votre site pour certains de vos visiteurs.

Afin de s'assurer que chaque visiteur verra un résultat conforme à nos attentes, nous pouvons utiliser un « reset CSS » pour notre padding et notre margin.

Dans ce cas là, c'est très simple, il suffit par exemple d'appliquer un padding et une margin avec des valeurs égales à zéro à notre élément body. Ensuite, on précisera les différentes marges souhaitées à nos éléments enfants.



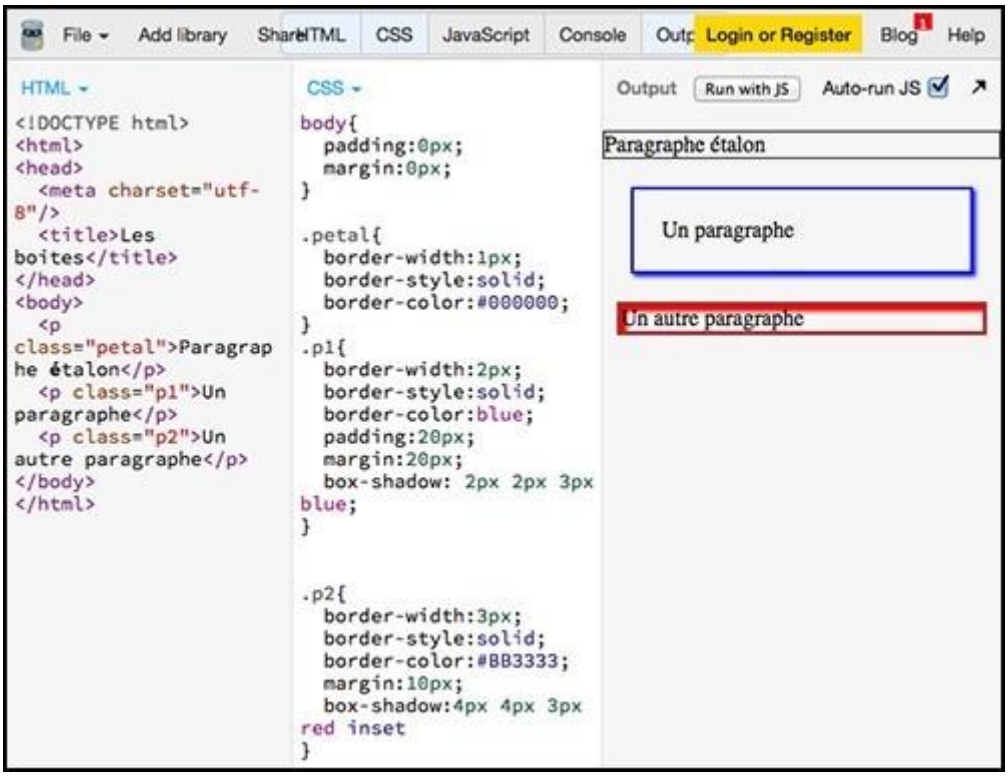
3.3.5. Les ombres des boîtes

On peut créer des ombres autour des boîtes, tout comme on l'avait fait pour nos textes précédemment.

Nous allons cette fois-ci utiliser la propriété box-shadow. Cette dernière fonctionne exactement comme text-shadow, avec deux valeurs obligatoires et quatre recommandées pour marcher :

- 1ère valeur : déplacement horizontal de l'ombre ;
- 2ème valeur : déplacement vertical de l'ombre ;
- 3ème valeur : rayon de propagation (flou gaussien) de l'ombre ;
- 4ème valeur : couleur de l'ombre (accepte les mêmes valeurs que la propriété « color »).

Notez que dans le cas des ombres des boîtes, il peut être intéressant de rajouter à la fin une dernière valeur, l'inset, si l'on souhaite créer une ombre intérieure.

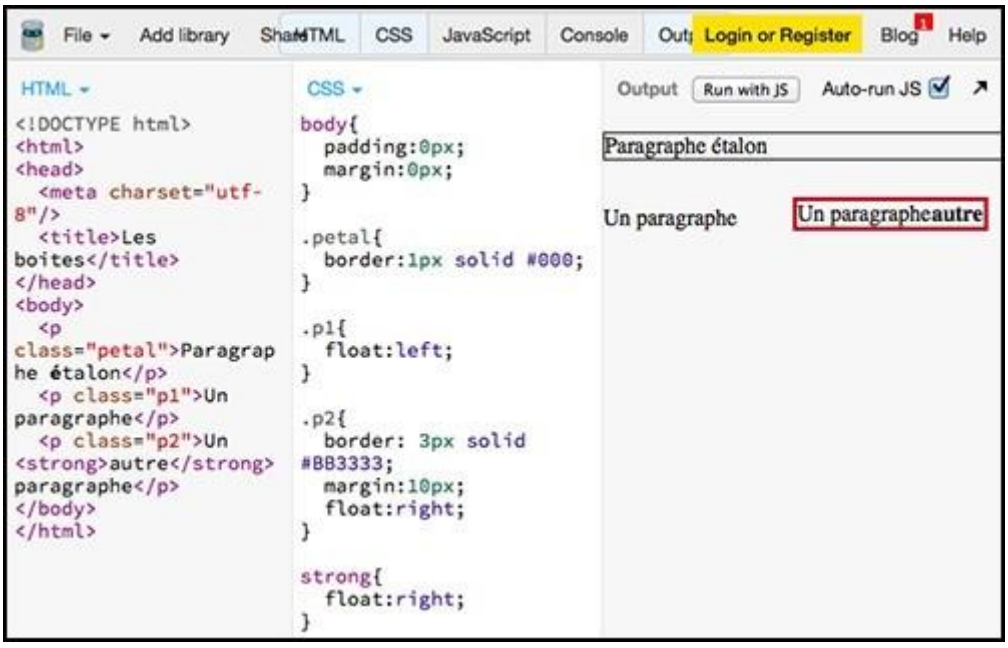


3.3.6. Faire flotter un élément

Pour aligner des éléments les uns par rapport aux autres, on peut les faire « flotter ». Pour faire flotter un élément, nous utiliserons la propriété float avec les valeurs suivantes : left, right, none ou inherit.

Un élément flottant va sortir du schéma naturel (du « flow ») d'une page web pour venir se placer contre le bord gauche ou droit de l'élément qui le contient ou contre le bord de la page.

Lorsque l'on fait flotter un élément, les éléments après l'élément flottant vont venir se positionner à côté de celui-ci.



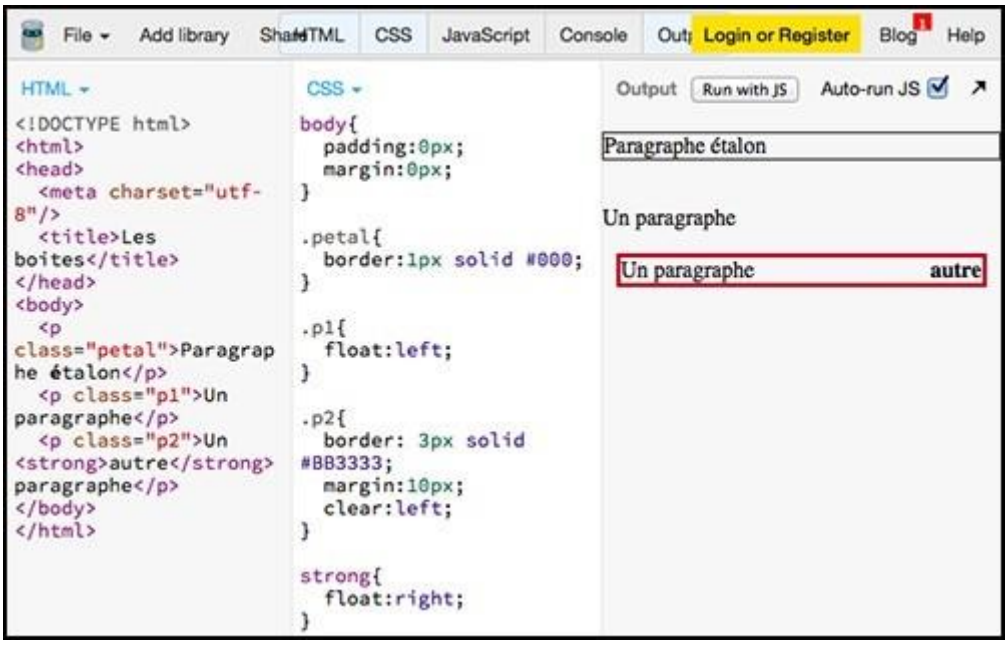
Dans l'exemple ci-dessus, on remarque que l'élément strong, contenu dans l'élément p2, va venir se placer dans le coin à droite de son élément parent (l'élément p2 donc).

L'élément p1 va lui se placer à gauche dans la page tandis que l'élément p2 va se placer à droite ; à côté de l'élément p1.

Généralement, on utilisera plutôt la propriété float sur des éléments de type inline comme des images par exemple. En effet, cette propriété peut être la cause de problème d'affichages lorsqu'elle est mal utilisée sur des éléments de type block.

Si l'on veut qu'un élément suivant un élément flottant vienne se placer sous cet élément flottant, il faudra utiliser la propriété clear. Celle-ci accepte trois valeurs : left, right ou both :

- left : un élément va se placer en dessous après un float left ;
- right : un élément va se placer en dessous après un float right ;
- both : un élément va se placer en dessous après un float left ou un float right.



3.3.7. La propriété display

La propriété display est une propriété extrêmement puissante : elle permet de changer le type d'un élément de block à inline ou d'inline à block.

Cette propriété supporte quatre valeurs différentes qui correspondent aux différents types d'éléments possibles : inline, block, inline-block et none.

La nouveauté ici est le type inline-block. Ce nouveau type ne peut être donné à un élément que grâce à la propriété display. Il va être un mix des types inline et block.

Un élément de type inline-block se comporte de cette façon : l'élément en soi (contenu et boîtes) se comporte comme un type block tandis que le contenu seulement se comporte comme un type inline.

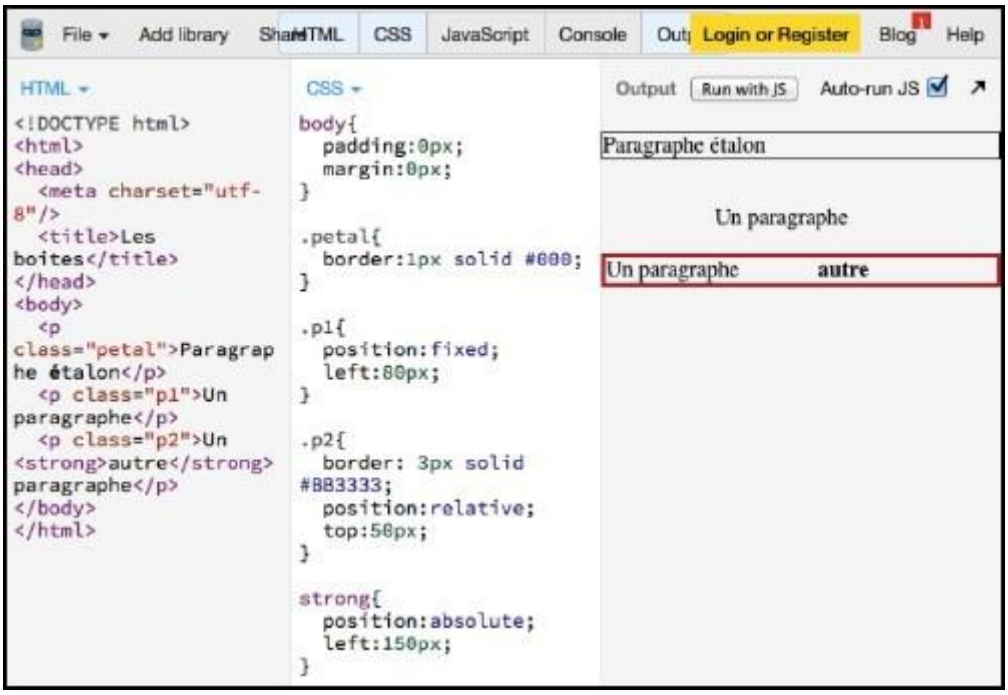
Pour le dire plus simplement, un élément de type inline-block se comportera comme un élément de type inline excepté que l'on va pouvoir contrôler précisément sa hauteur et sa largeur.

3.3.8. La propriété position

La propriété position est une autre propriété très puissante nous permettant de définir l'emplacement d'éléments HTML dans une page. Cette propriété peut prendre cinq valeurs :

- Static : valeur par défaut, ne change pas la position de base d'un élément ;
- Absolute : permet de positionner un élément n'importe où dans la page, par rapport à son élément parent direct ;
- Fixed : permet de positionner un élément n'importe où dans la page. De plus, l'élément reste visible si vous descendez ou remontez le long d'une page ;
- Relative : permet de replacer un élément relativement par rapport à son positionnement par défaut ;
- Inherit : l'élément hérite des propriétés de son parent.

Pour ensuite jouer sur la position des éléments, on utilise les mots clefs right, bottom, left et top.



Dans cet exemple, notre paragraphe p1 a un positionnement fixed et a été décalé de 80px sur la gauche par rapport au bord de la page.

Le paragraphe p2 possède lui une position relative. Il a été décalé de 50px par rapport à son positionnement de base (par défaut, il aurait été placé au dessus de notre p1 à cause de son positionnement fixed).

L'élément strong possède un positionnement absolute. Sans notre left :150px, il aurait été collé sur le bord gauche de son élément parent, l'élément p2.

Notez qu'un positionnement absolute sur un élément annule la propriété float si on lui en a appliqué une.

3.3.9. Le z-index

Parfois, lorsque l'on crée le design de pages web, il arrive que deux éléments se chevauchent selon certaines circonstances.

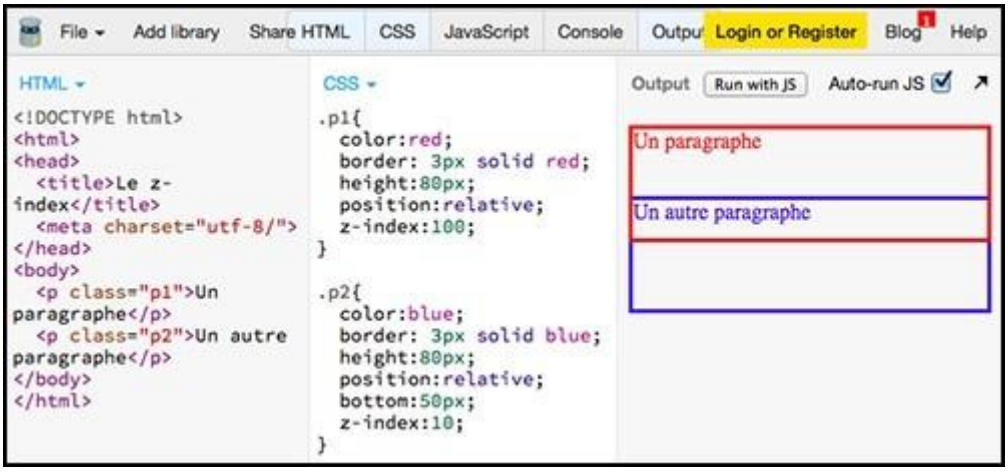
Il peut alors être utile de savoir comment indiquer quel élément doit apparaître au dessus de quel autre en cas de chevauchement.

Pour cela, nous utiliserons la propriété z-index. Celle-ci fonctionne avec tous les éléments positionnés et permet d'indiquer quel élément doit être au dessus de quel autre en cas de conflit.

Un élément positionné est un élément auquel on a appliqué la propriété position avec une valeur soit absolute, soit relative, soit fixed.

Le z-index ne fonctionnera donc pas sur des éléments positionnés en static (qui est la valeur par défaut).

Un élément avec un z-index possédant une valeur plus élevée qu'un autre sera au dessus de cet autre élément. Cette propriété va donc s'utiliser de cette manière :



Dans l'exemple précédent, par défaut, l'élément p2 devrait être par dessus l'élément p1.

Or, en appliquant un z-index avec une valeur plus élevée à p1, on arrive à refaire passer p1 par dessus p2 (on peut le voir au niveau des bordures).

3.3.10. Notations long-hand et short-hand

On peut écrire les trois propriétés relatives aux bordures sous la forme condensée d'une seule propriété border.

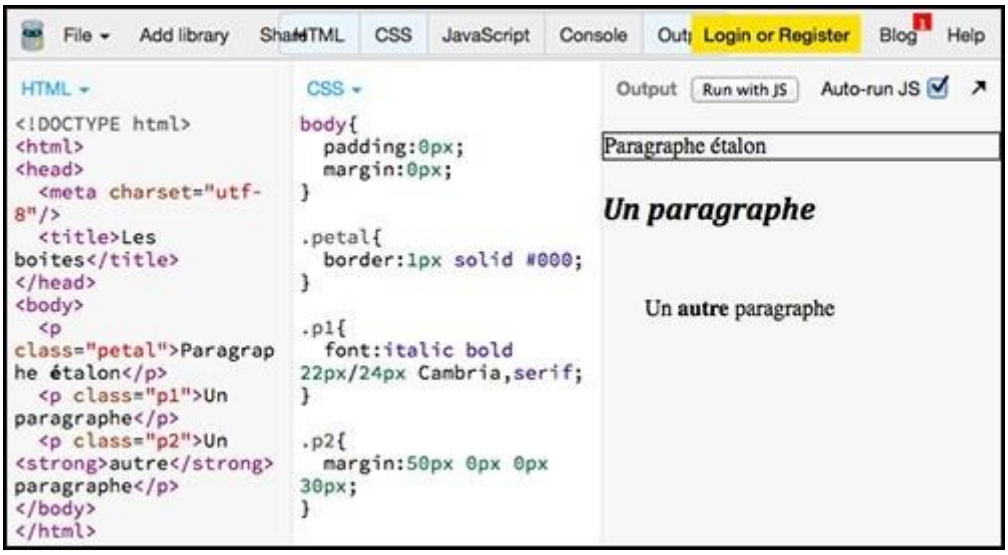
La première écriture, avec les trois propriétés de type border, est ce que l'on appelle une notation long-hand, tandis que la forme condensée est une notation short-hand.

Jusqu'à présent, on a fait écrire que des propriétés sous leur forme long-hand afin que vous compreniez bien les bases et la logique du CSS et afin de ne pas compliquer inutilement les choses.

Dorénavant, vous avez tout à fait le niveau pour utiliser des notations short-hand et je vous encourage à les utiliser dès que possible (celles-ci sont plus rapides à écrire et donc moins gourmandes en code et en temps d'exécution).

Voici ci-dessous une liste de quelques propriétés acceptant une écriture short-hand. Encore une fois, écrire une propriété sous sa forme long-hand ou short-hand produira exactement le même résultat.

Long Hand	Short Hand
[font-style] [font-weight] [font-size]/[line-height] [font-family]	font
[border-width] [border-style] [border-color]	border
[margin-top] [margin-right] [margin-bottom] [margin-left]	margin
[background-color] [background-image] [background-repeat] [background-attachment] [background-position]	background



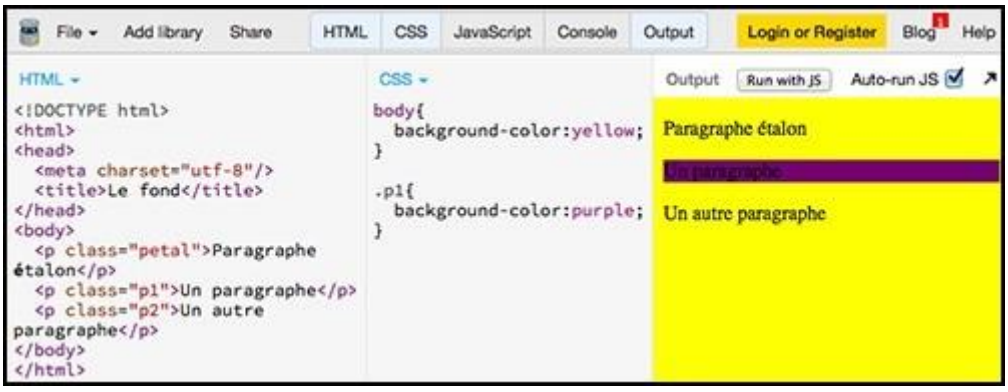
4. Fonctionnalités Avancées

4.1. Gestion du Background

4.1.1. Ajouter de la couleur ou une image pour le fond

Pour ajouter une couleur de fond, nous allons utiliser la propriété background-color.

Cette propriété accepte les mêmes valeurs que la propriété color que nous avons vu précédemment, à savoir des valeurs de type nom de couleur, hexadécimale ou RGB.



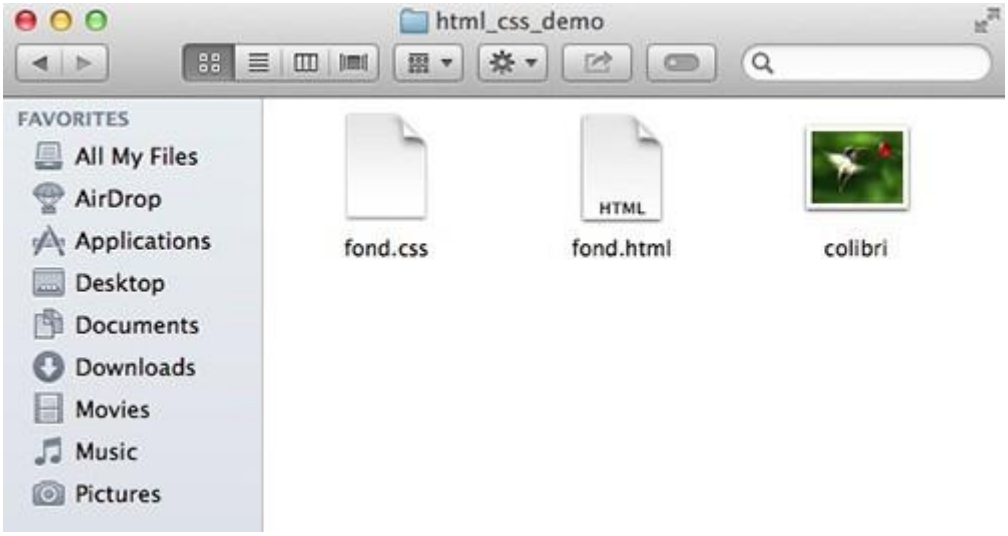
On peut également ajouter une image de fond. Pour ce faire, on va cette fois utiliser la propriété background-image.

On lui donne en valeur l'url de l'image qui, en l'occurrence, prend la forme d'un chemin relatif comme nous avions vu pour les liens.

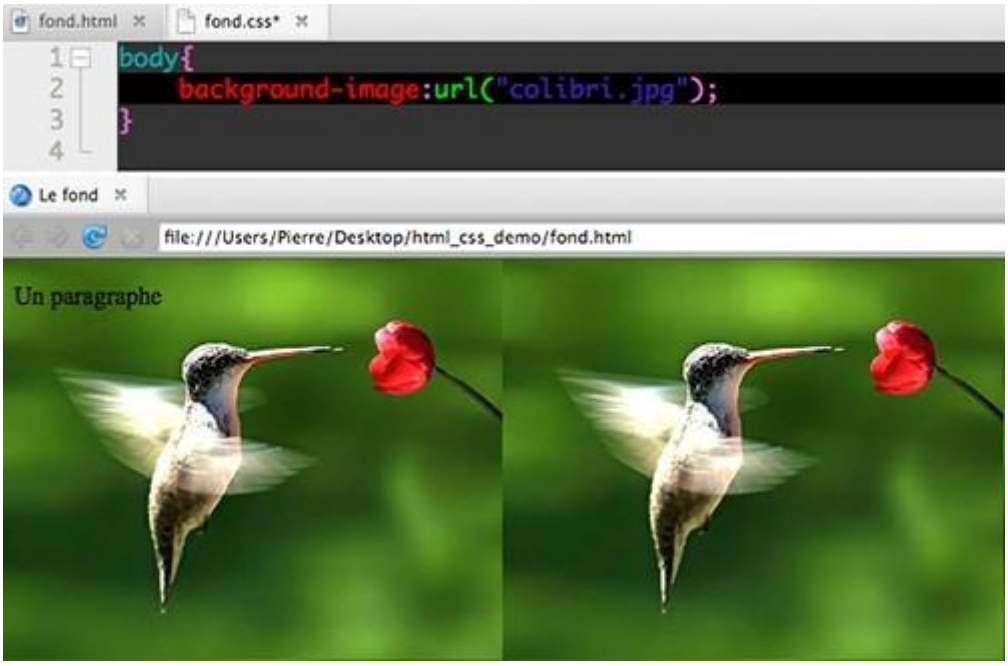
Tout d'abord, il va nous falloir une page HTML et une page CSS que nous allons lier entre elles et enregistrer dans le même dossier pour plus de simplicité.

```
1  <!DOCTYPE html>
2      <html>
3          <head>
4              <title>Le fond</title>
5              <meta charset="utf-8"/>
6              <link rel="stylesheet" href="fond.css"/>
7          </head>
8          <body>
9              <p>Un paragraphe</p>
10         </body>
11     </html>
12
```

Ensuite, il va nous falloir une image. On l'enregistre également dans le même dossier.



Retour finalement sur notre page CSS et utilisons maintenant la propriété background-image sur notre élément body (on peut utiliser cette propriété sur n'importe quel élément mais en général ce sera sur l'élément body).



Comme vous pouvez le constater, par défaut, l'image est répétée horizontalement et verticalement pour occuper tout l'espace de notre page web puisque nous l'avons insérée dans notre élément body.

Si votre image occupe d'origine plus de place que l'élément dans laquelle vous l'insérez, elle ne sera donc pas répétée mais au contraire rognée.

C'est ce qu'il se passe dans l'exemple suivant où nous avons insérée notre image en fond de notre élément p auquel nous avons donné des dimensions de 200px * 100px :

```
p{
    height:100px;
    width:200px;
    background-image:url("colibri.jpg");
}
```

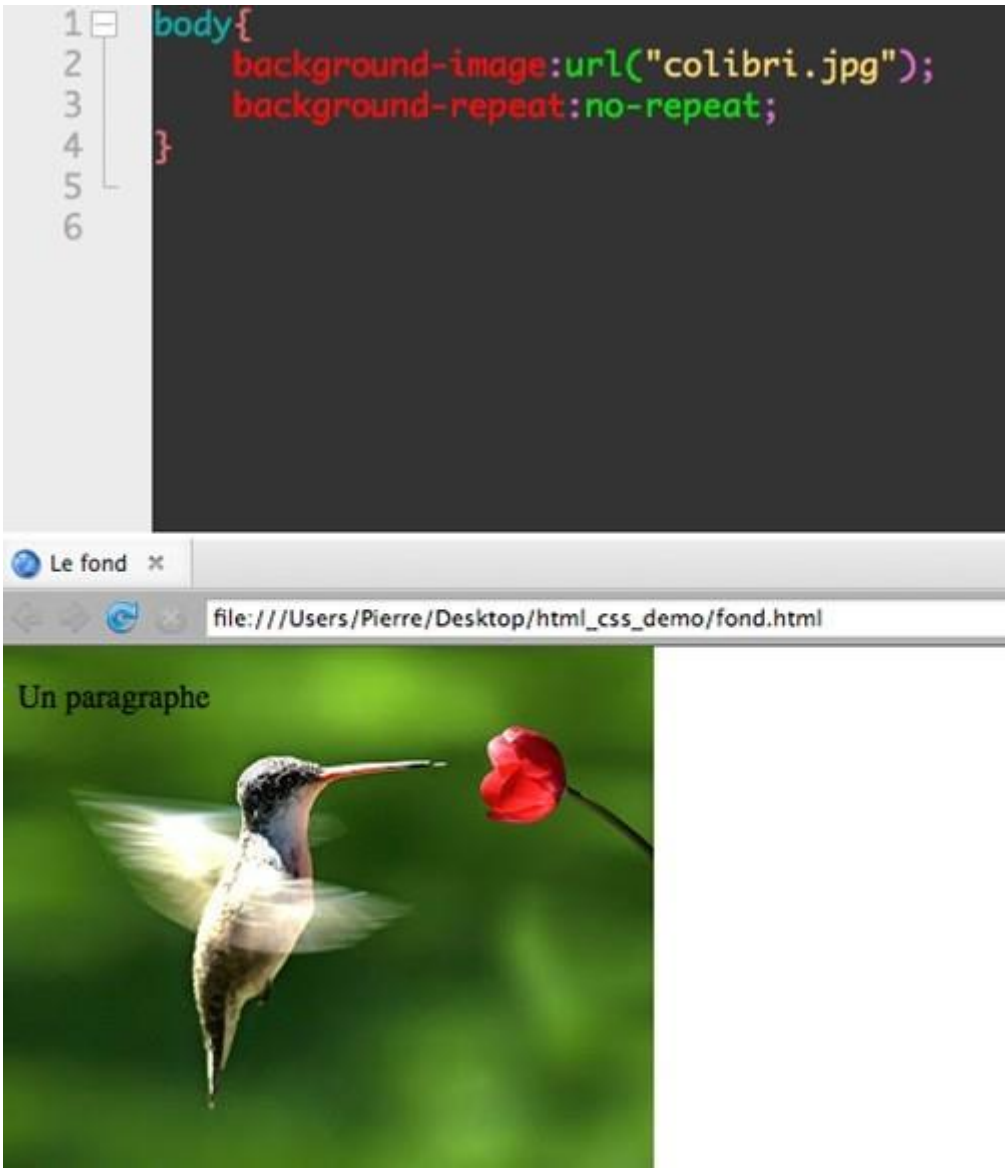
Notez que pour insérer une image de fond nous pouvons également utiliser la propriété short-hand «background » plutôt que la propriété background-image.

Cette propriété gère également d'autres aspects de la mise en forme du fond que nous allons voir ensuite.

4.1.2. Position et répétition du fond

La propriété background-repeat nous permet de gérer la répétition de notre fond. Cette propriété accepte quatre valeurs :

- Repeat : le fond se répète horizontalement et verticalement, c'est le comportement par défaut ;
- Repeat-x : le fond ne se répète qu'horizontalement ;
- Repeat-y : le fond ne se répète que verticalement ;
- No-repeat : le fond ne se répète pas.



Pour contrôler la position de notre image de fond, nous allons utiliser la propriété backgroundposition.

Cette propriété a besoin de deux valeurs pour fonctionner : une coordonnée horizontale et une coordonnée verticale. Le fond sera décalé par rapport au bord en haut à gauche de son élément parent.

```
body{
    background-image:url("colibri.jpg");
    background-repeat:no-repeat;
    background-position : 100px 10px;
}
```

Dans cet exemple, on a décalé l'image de 100px vers la droite et de 10px vers le bas par rapport au coin en haut à gauche de l'élément body, son parent. Rappelez vous que l'élément body occupe toute la page.

4.1.3. Fixer le fond ou le faire défiler avec la page

Pour fixer le fond et ainsi donner un effet intéressant, on utilise la propriété background-attachment à laquelle on donne une valeur égale à fixed.

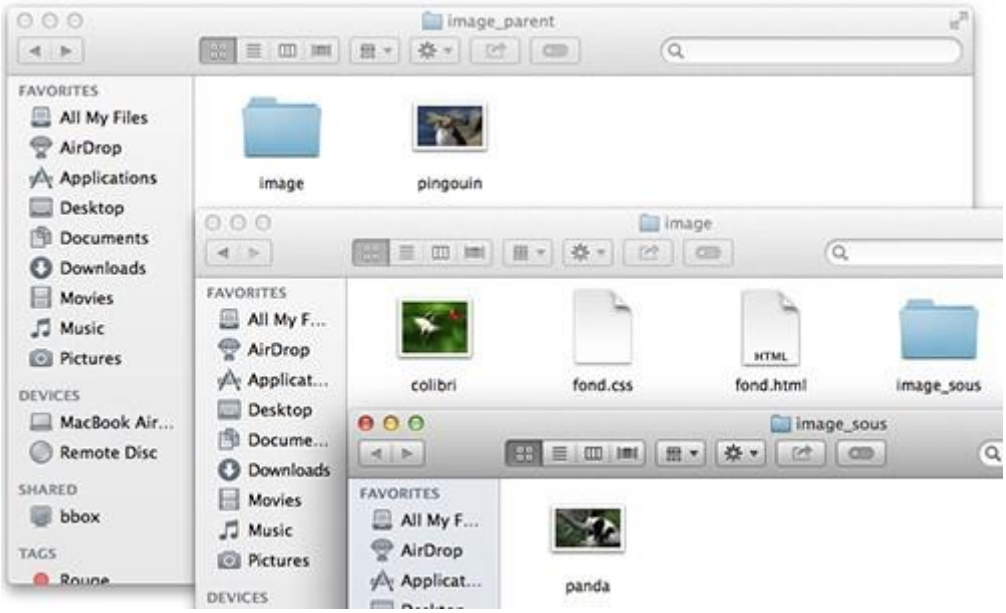
Cette propriété accepte deux valeurs : fixed donc ou scroll, qui est la valeur par défaut et qui va faire défiler le fond en même temps que la page.

4.1.4. Insérer plusieurs images de fond

Il est possible, grâce au CSS3, d'insérer plusieurs images de fond. Pour ce faire, on va devoir utiliser la notation short-hand de nos propriétés relatives au fond et donc la propriété background. On va ensuite tout simplement séparer les déclarations par une virgule.

Pour cela, on va prendre deux nouvelles images qu'on va placer dans un sous-dossier et dans un dossier parent pour changer un peu et voir tous les cas de figure.

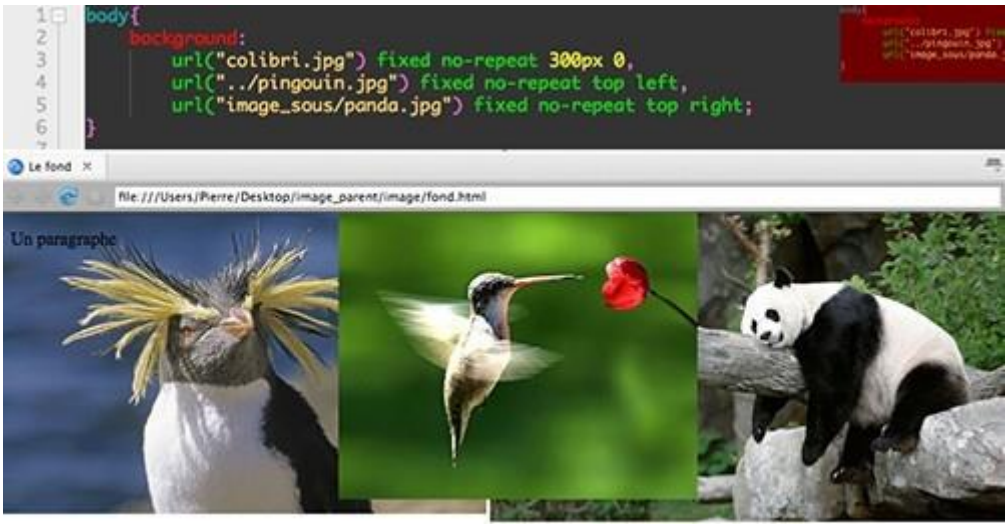
En pratique, si votre site est bien construit et pas trop compliqué, vous devriez avoir toutes vos images dans un dossier nommé « images » ou « img ».



On a trois images dans trois dossiers différents :

- Mon image de colibri qui est dans le même dossier que mes fichiers HTML et CSS ;
- Mon image de pingouin qui est dans un dossier parent ;
- Mon image de panda qui est dans un sous-dossier.

Il ne nous reste donc plus qu'à utiliser notre propriété background de la façon suivante :



Ici, on a inséré les trois images dans l'élément body, en leur demandant d'être fixe et de ne pas se répéter. J'ai décalé mon image colibri de 300px sur la droite tandis que j'ai collé mon image pingouin en haut à gauche grâce à top left et mon image panda en haut à droite grâce à top right.

Notez que deux images peuvent tout à fait se chevaucher. Par défaut, la première image déclarée sera au dessus. Attention donc à l'ordre de déclaration !

4.1.5. Créer un fond en dégradé

Nous avons deux types de dégradé à notre disposition : des dégradés linéaire ou radial.

Les fonds en dégradé sont considérés comme des images et donc, à ce titre, nous allons à nouveau utiliser la propriété background-image pour les créer.

Il y a quelques années de cela, tous les navigateurs ne reconnaissaient pas les mêmes propriétés et ne les implémentaient pas de la même manière. En l'occurrence, chaque navigateur avait une méthode bien à lui d'implémenter un fond en dégradé.

Les préfixes vendeurs ont alors été créés afin que chaque navigateur affiche une propriété de la même manière que les autres. Ce sont des petits mots clefs qui vont être placés avant la déclaration du type de dégradé souhaité.

Même si aujourd'hui la majorité des navigateurs suivent des standards qui ont été établis et implémentent les propriétés de la même façon, cela peut toujours être utile (et ça ne coûte rien) de mentionner des préfixes vendeurs dans le cas où certains de vos visiteurs utiliseraient des vieilles versions de navigateurs.

Les préfixes vendeurs sont les suivants :

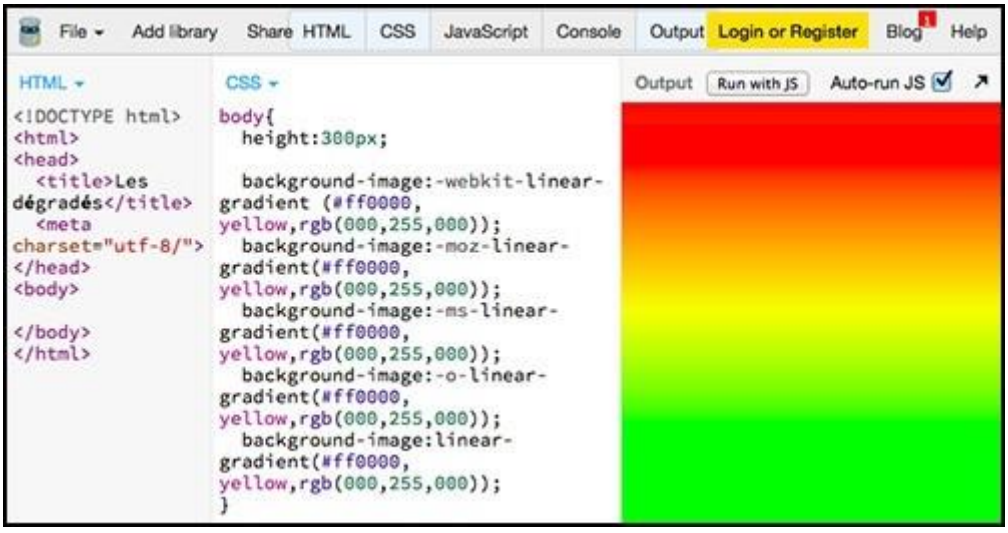
- Pour Chrome et Safari : -webkit-
- Pour Mozilla : -moz-

- Pour Internet Explorer : -ms-
- Pour Opera : -o-

4.1.6. Les dégradés de type linéaire

Pour créer un dégradé linéaire, on utilise les mots clefs linear-gradient. On peut ensuite spécifier autant de valeurs que souhaitées, chaque valeur correspondant à une couleur de notre dégradé.

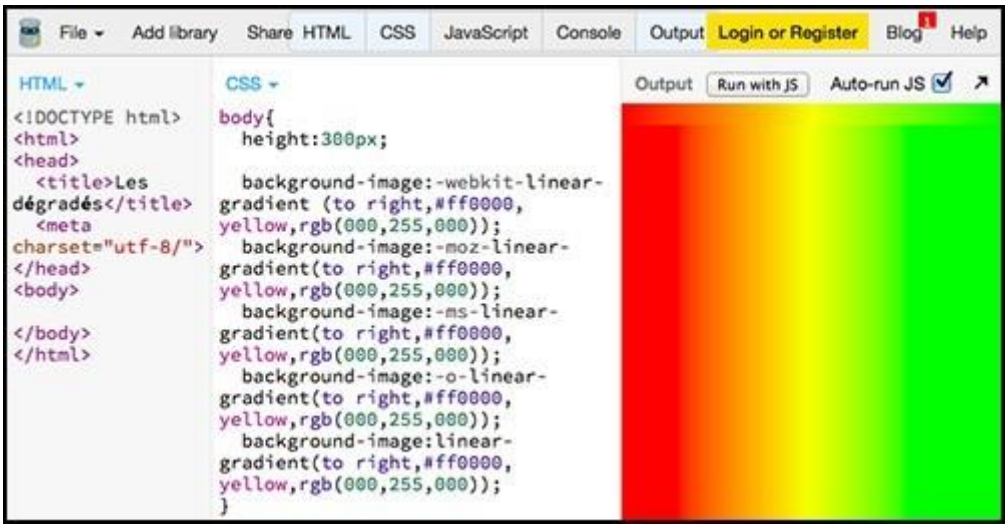
Afin que le tout fonctionne correctement, vous devez mentionner chaque préfixe vendeur et il est recommandé de finir avec le linear-gradient sans aucun préfixe (ce sera la valeur par défaut) comme ceci :

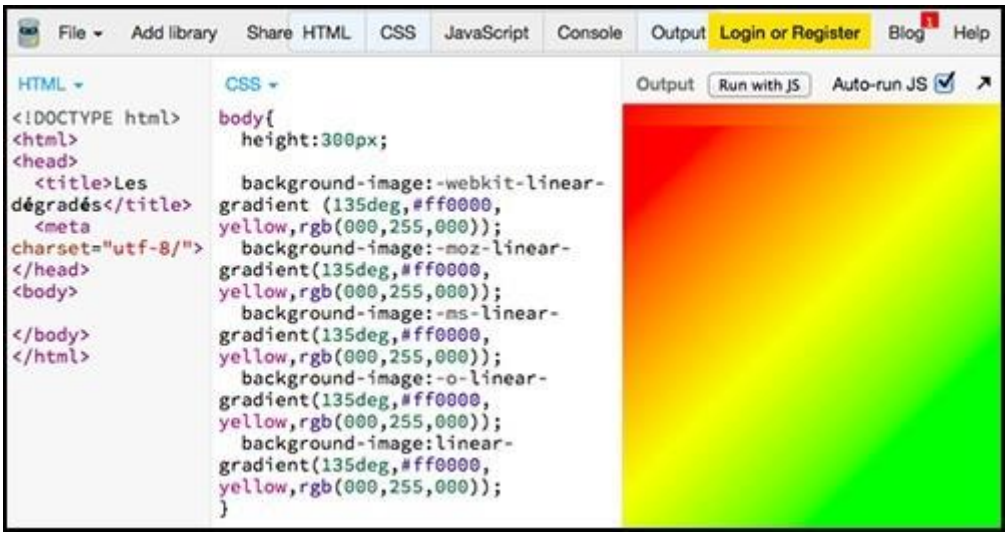


Les dégradés linéaires peuvent aller du haut vers le bas, de bas en haut, de gauche à droite, de droite à gauche en encore en diagonale.

Par défaut, cependant, les dégradés linéaires partent du haut vers le bas d'un élément. On peut choisir l'inclinaison de nos dégradés en utilisant des mots clefs ou en spécifiant la valeur en degré de l'angle voulu avant d'indiquer les couleurs du dégradé.

Pour que notre dégradé aille vers la droite, on utilisera la valeur to right, pour qu'il aille à gauche to left et pour qu'il aille de bas en haut to top :

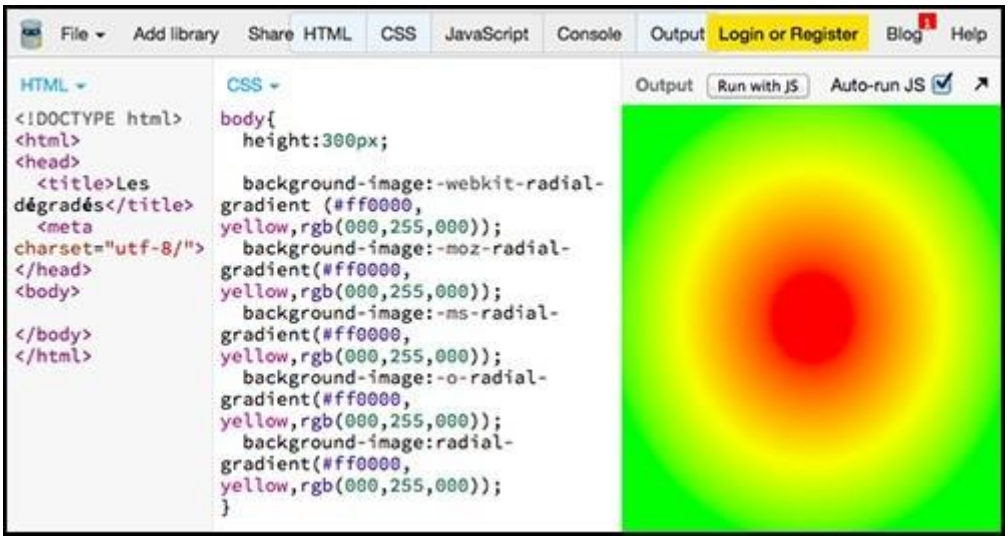




4.1.7. Les dégradés de type radial

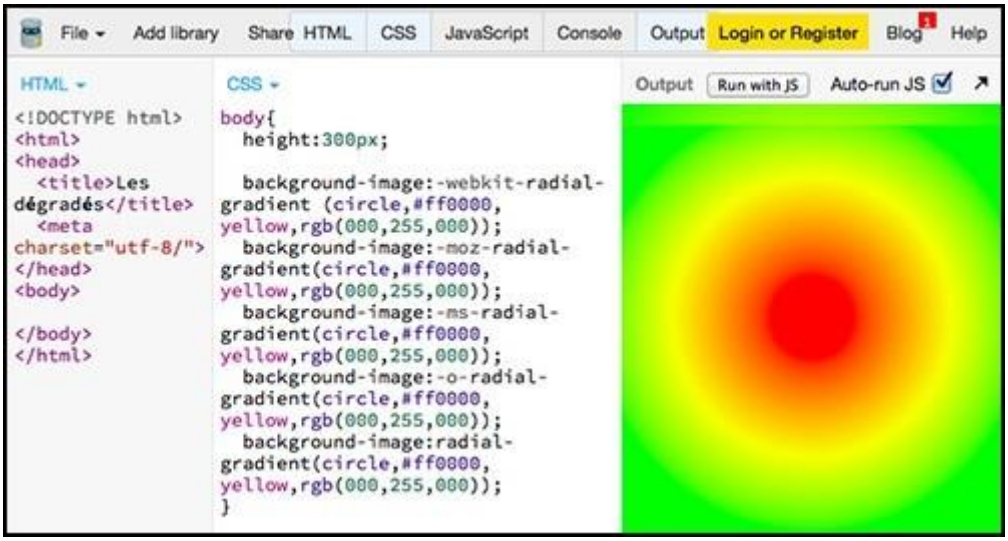
Un dégradé de type radial est un dégradé créé à partir d'un point central.

Les dégradés de type radial fonctionnent sur le même schéma que ceux de type linéaire. Cette foisci, nous allons utiliser les mots clefs radial-gradient comme ceci :

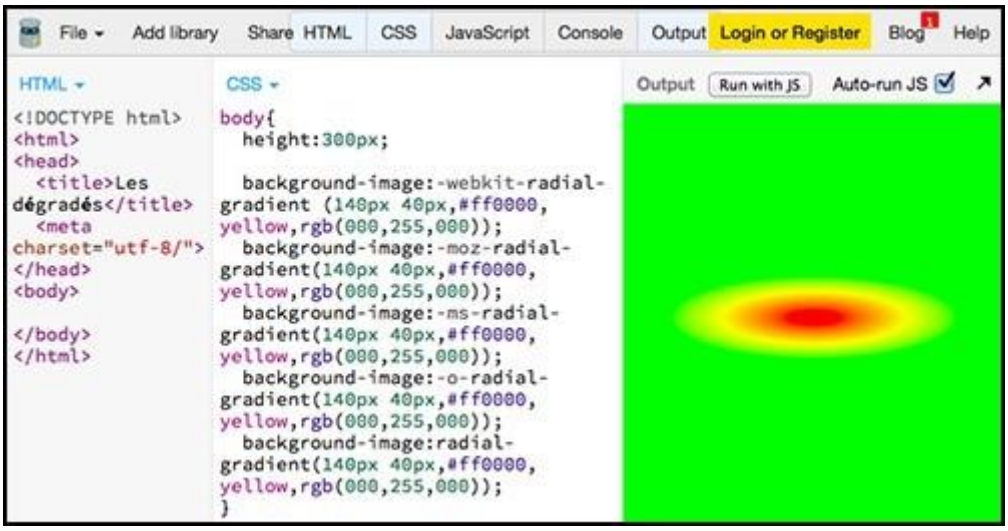


Évidemment, il n'est plus question ici d'indiquer une quelconque direction (cela n'aurait pas de sens puisque le dégradé part du centre dans toutes les directions.

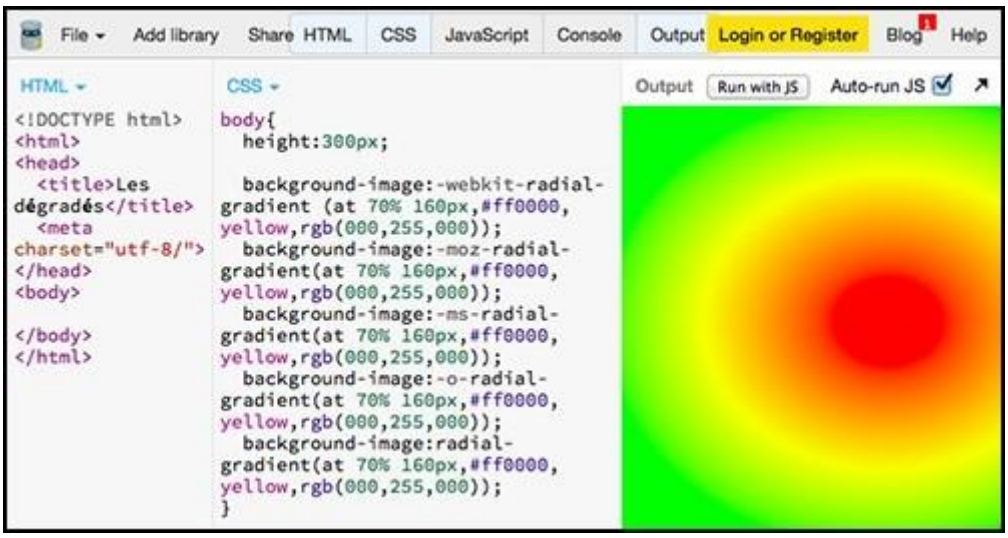
Cependant, on peut choisir si l'on souhaite que notre dégradé ait la forme d'un cercle ou d'une ellipse en précisant les mots clefs circle et ellipse juste avant nos couleurs.



On peut également choisir avec précision les dimensions et la forme de notre dégradé en précisant deux chiffres (généralement en px) correspondant respectivement à la largeur et à la hauteur de notre dégradé.



On peut également utiliser une combinaison de mots clefs et de valeurs pour déplacer le centre de notre dégradé radial :



Dans l'exemple ci-dessus, nous avons déplacé le centre du dégradé vers la droite (il est à 70% sur la droite par rapport à son élément parent) et l'avons fixé à 160px du bord supérieur de son élément parent.

Nous aurions également pu combiner mots clefs et valeurs en écrivant « at right 160px » ou encore « at 40% bottom ».

La première valeur correspond au déplacement horizontal du centre de notre dégradé tandis que la seconde correspond au déplacement vertical de ce centre. Par défaut, le dégradé radial sera centré.

4.2. Intégrer des Images, de l'Audio et de la Vidéo

4.2.1. Insérer une image

Lorsque vous enregistrez une image sur votre ordinateur, vous pouvez en théorie choisir entre différents formats avec entre autre le png, le jpg ou jpeg, bitmap ou gif pour ne citer que les plus connus.

Comme vous vous en doutez, ces formats ne sont pas strictement équivalents et chaque format à été crée pour un type précis d'images.

Si votre image est une photographie, je vous recommande d'utiliser le format jpg. Si, en revanche, votre image est un dessin ou tout autre type d'image statique, préférez le png qui est un type qui gère également la transparence. Enfin, si votre image est animée, vous devrez choisir le format gif. Oubliez le bitmap qui est un format lourd et sans réel avantage.

Pensez à choisir avec attention le nom de vos images et évitez les caractères spéciaux et les espaces (préférez les underscores ou les tirets).

Pour insérer une image, nous allons utiliser l'élément HTML img. L'élément img doit être placé au sein d'un élément de type block, comme l'élément p ou un div par exemple.

Pour que votre code soit valide, vous devez ajouter deux attributs à l'élément img : l'attribut source, abrégé src et l'attribut alternative, qu'on note alt.

L'attribut src indique le chemin de l'image. Vous pouvez lui donner un chemin relatif ou absolu. L'attribut alt sert à donner un texte descriptif à a photo. Ce texte est utile notamment dans les cas où votre image ne s'affiche pas ou pour les robots (de Google, entre autres).

Enfin, vous pouvez également ajouter l'attribut facultatif title, ce qui aura pour effet d'afficher un texte lorsque vos visiteurs passeront la souris sur votre image.

Comme vous pouvez le voir, on a indiqué une adresse absolue en source de l'image (sur Google Image). Notez que l'élément img est représenté par une balise orpheline.

4.2.2. Ajuster et positionner une image

Pour ajuster la taille d'une image en CSS, on utilise les propriétés height (hauteur) et width (largeur) que l'on connaît déjà sur notre élément img.

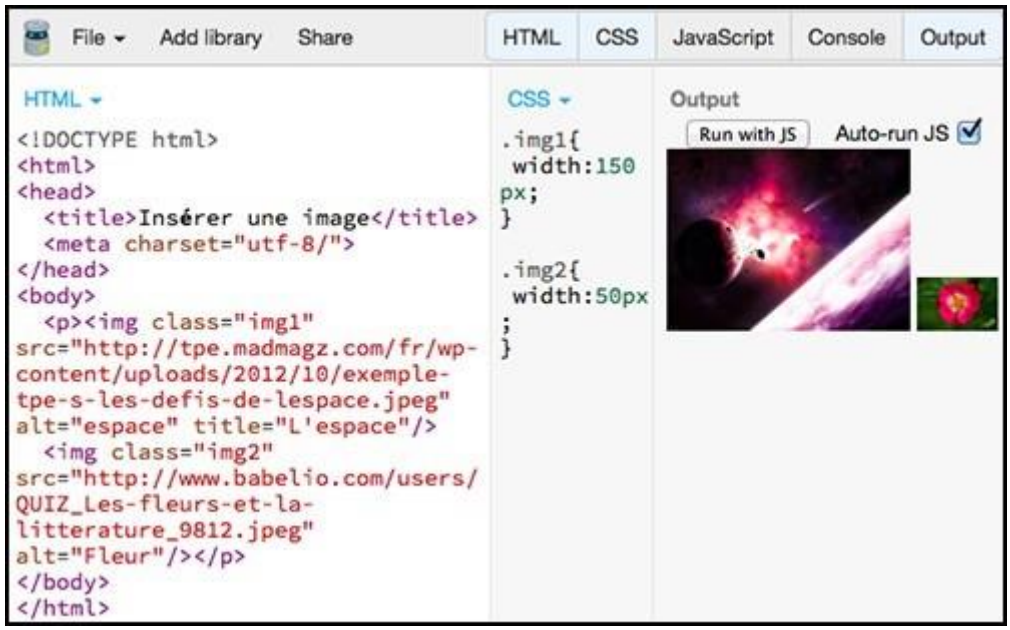
Si l'on ne spécifie qu'une hauteur ou qu'une largeur, la deuxième dimension va s'ajuster automatiquement. Attention, lorsque l'on spécifie une hauteur et une largeur, à ne pas trop casser les proportions initiales de l'image pour un meilleur rendu !

Une astuce performance et référencement : essayez tant que possible d'ajuster la taille de vos images avant de les mettre sur votre serveur (avec Paint ou PhotoShop par exemple).

En effet, si vous envoyez une image de 2000px de large sur votre serveur et que vous la redimensionnez à 400px par exemple par la suite avec le CSS, l'affichage peut être ralenti pour vos visiteurs.

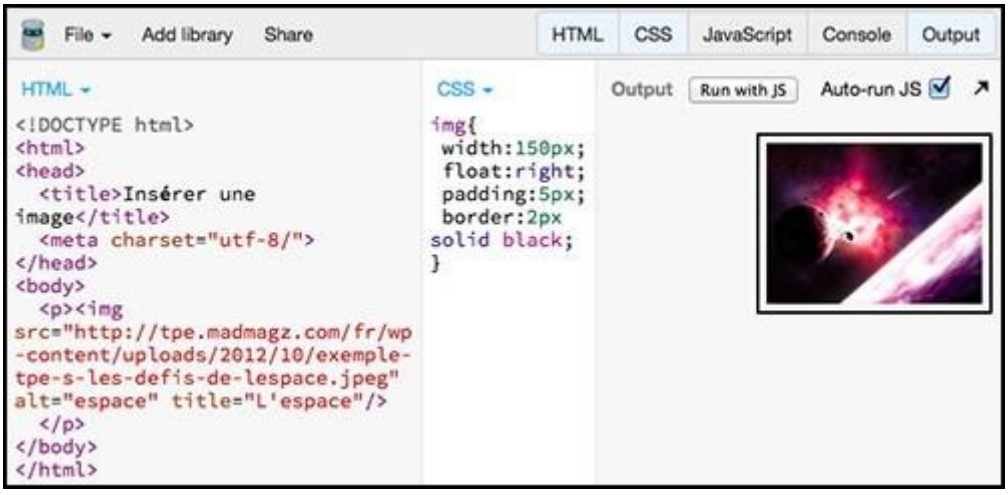
A chaque fois, notre serveur va envoyer l'image au format original et l'image devra être redimensionnée au format souhaité par le navigateur des visiteurs. Or, cela use évidemment beaucoup plus de mémoire et va donc influer sur le temps d'affichage de notre page web.

L'élément img est un élément de type inline par défaut. Cela sous-entend qu'une image va se positionner sur la même ligne qu'un autre contenu de type inline.



Souvent, lors de la création d'un site web, nous voudrons qu'une image occupe sa propre ligne. Dans ce cas là, il nous faudra utiliser la propriété display en changer le type de l'élément img.

On peut également utiliser la propriété float pour positionner une image, en utilisant les valeurs right ou left. Pour ajuster l'espace entre une image et le contenu qui l'entoure, on va utiliser la propriété margin. On peut même créer un cadre en utilisant également les propriétés padding et border.



4.2.3. Insérer de l'Audio

Tout comme pour les images, il existe différents formats audio. Cependant, ça se complique avec l'audio car il n'existe pas de format supporté par tous les navigateurs. Cela signifie qu'il va falloir indiquer plusieurs formats lorsque l'on va insérer de l'audio en HTML.

Browser	MP3	Wav	Ogg
Internet Explorer	YES	NO	NO
Chrome	YES	YES	YES
Firefox	NO **Update:** Firefox 21 running on Windows 7, Windows 8, Windows Vista, and Android now supports MP3	YES	YES
Safari	YES	YES	NO
Opera	NO	YES	YES

MIME Types for Audio Formats

Format	MIME-type
MP3	audio/mpeg
Ogg	audio/ogg
Wav	audio/wav

Les formats mp3 et ogg qui sont les deux formats les mieux supportés.

Pour ajouter de l'audio sur une page web, on va utiliser l'élément audio. Tout comme pour l'élément img, l'élément audio va demander un attribut src pour fonctionner. Cet attribut prendre en valeur l'URL du fichier audio.

Toutefois, si vous n'écrivez que cela, vous ne verrez rien à l'écran. En effet, par défaut, l'élément audio n'est pas affiché sur les pages web. Il va donc nous falloir utiliser d'autres attributs.

L'attribut controls, tout d'abord, va servir à afficher les boutons de contrôle tels que les boutons lecture, pause et volume entre autres. Cet attribut est donc bien évidemment obligatoire.

L'attribut autoplay va nous permettre de lancer automatiquement le fichier audio lors du chargement de la page. L'attribut loop nous permet de faire répéter la musique en boucle.

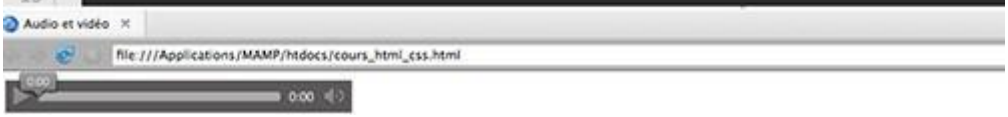
L'attribut width est utile pour modifier la largeur par défaut de la barre de l'élément audio.

Enfin, l'attribut preload nous permet de sauvegarder de la bande passante et d'accroître les performances de notre site. Cet attribut peut prendre trois valeurs différentes : metadata, auto et none. En pratique, on utilisera généralement la valeur auto qui va préloader toutes les informations et les données.

Entre les deux balises de l'élément audio, on peut ajouter un texte qui sera affiché dans le cas où le navigateur ne supporterait pas le format audio choisi comme par exemple : « Mettez à jour votre navigateur, on vit au 21è siècle ! ».

En théorie, il faudrait donc écrire ceci pour insérer de l'audio dans une page web :

```html
<html>
  <head>
      <title>Audio et vidéo</title>
    <meta charset="utf-8"/>
  </head>
  <body>
      <audio src="demo.mp3" controls="controls" autoplay="autoplay" preload="auto">
        Pour lire cette piste, vous devez mettre à jour votre navigateur !
      </audio>
  </body>
</html>
```



Cependant, en écrivant notre élément audio comme cela, on ne peut pas insérer différents formats audio pour les différents navigateurs, ce qui est très gênant.

En pratique, nous utiliserons donc un deuxième élément source et écrirons cela pour insérer de l'audio qui sera lu par tous les navigateurs :

```html
<!DOCTYPE html>
<html>
  <head>
      <title>Audio et vidéo</title>
    <meta charset="utf-8"/>
  </head>
  <body>
    <audio controls="controls" autoplay="autoplay" preload="auto">
      <source src="demo.mp3"></source>
      <source src="demo.ogg"></source>
    </audio>
  </body>
</html>
```

Ainsi, le navigateur des visiteurs lira la version qu'il reconnaît et ignorera les autres.

Notez que l'on a donné des valeurs égales aux noms des attributs pour controls et autoplay. En théorie, il n'est pas obligatoire de préciser de valeur pour ces attributs.

4.2.4. Insérer de la Vidéo

Pour créer et afficher une vidéo, il nous faut obligatoirement trois choses : un codec audio (format mp3, ogg, etc.), un codec video (h.264, ogg thoera, webM) et un format contener dans lequel on va stocker ces deux éléments (généralement avi, mp4 ou mkv).

Là encore, il n'existe pas de format supporté par tous les navigateurs et il est donc recommandé d'en utiliser plusieurs pour être certain que la vidéo soit correctement affichée.

Browser	MP4	WebM	Ogg
Internet Explorer	YES	NO	NO
Chrome	YES	YES	YES
Firefox	YES **Update 1:** Firefox 21 on Windows and Android now supports MP4 **Update 2:** Firefox 30 on Linux now supports MP4	YES	YES
Safari	YES	NO	NO
Opera	NO	YES	YES

- MP4 = MPEG 4 files with H264 video codec and AAC audio codec
- WebM = WebM files with VP8 video codec and Vorbis audio codec
- Ogg = Ogg files with Theora video codec and Vorbis audio codec

MIME Types for Video Formats

Format	MIME-type
MP4	video/mp4
WebM	video/webm
Ogg	video/ogg

Pour ajouter une vidéo, on utilise l'élément video. Les attributs pris par cet élément sont exactement les mêmes que ceux pris par l'élément audio, à savoir : src, autoplay, controls, loop, preload et width.

Contrairement aux éléments audio, une vidéo sera par défaut affichée sur une page web.

Il est également possible d'ajouter un attribut poster à votre vidéo. L'attribut poster permet de télécharger et d'afficher une image qui sera affichée avant le lancement de la vidéo.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Audio et vidéo</title>
    <meta charset="utf-8"/>
  </head>
  <body>
    <video controls="controls" preload="auto" poster="pixar.jpg" width="380px">
      <source src="pixar.mp4" />
      <source src="pixar.webm" />
      <source src="pixar.ogg" />
    </video>
  </body>
</html>
```

Une autre alternative, beaucoup plus simple, pour ajouter de la vidéo ou de l'audio sur son site est de stocker ces contenus sur des sites comme YouTube, Dailymotion ou Vimeo puis de les intégrer à votre site grâce au code d'intégration fourni.

De cette manière, vous n'aurez plus aucun problème d'affichage quelque soit le navigateur de vos visiteurs puisque la transcription dans différents formats est effectuée par les sites hébergeurs.

4.2.5. Les Éléments Figure et Figcaption

Les éléments figure et figcaption ont été créés pour marquer sémantiquement du contenu comme des images, de l'audio ou de la vidéo.

L'élément figure est un élément de type block. On l'utilise autour d'images, de contenus audio ou vidéo ou de blocs de code.

L'idée est d'utiliser l'élément figure pour envelopper du contenu lorsque celui-ci n'est pas strictement décoratif. A l'intérieur de l'élément figure, on peut utiliser l'élément HTML figcaption qui va nous permettre d'accoler une légende à notre contenu.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Audio et vidéo</title>
    <meta charset="utf-8"/>
  </head>
  <body>
    <figure>
      <img src="pixar.jpg" alt="C'est pas bien d'être méchant"/>
      <figcaption>For the birds par Pixar</figcaption>
    </figure>
  </body>
</html>
```

4.3. Les Tableaux

4.3.1. Création d'un tableau simple

Les tableaux en HTML ne servent et ne doivent être utilisés que pour organiser des données.

Pour créer un tableau, il va nous falloir utiliser au minimum trois éléments : les éléments table, tr et td.

L'élément table définit le tableau en soi. A l'intérieur de cet élément table, on va utiliser l'élément tr (table raw) pour ajouter des lignes à notre tableau.

Tout tableau en HTML sera construit ligne par ligne.

Enfin, on utilise l'élément td (table data) pour ajouter des cellules dans nos lignes.

Si l'on insert plusieurs éléments td à l'intérieur d'un élément tr, des colonnes vont se créer automatiquement à l'intérieur de notre ligne.

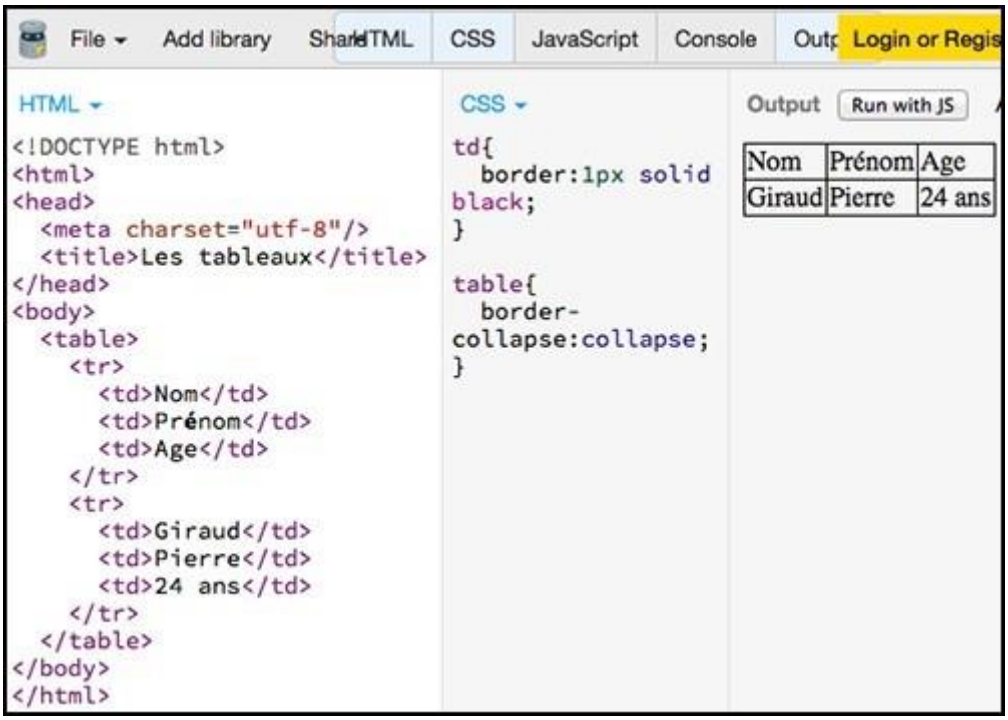
Nous venons de créer notre premier tableau. Cependant, celui-ci ne ressemble pas vraiment à l'image d'un tableau car nous ne l'avons pas encore mis en forme.

4.3.2. Mise en forme d'un tableau

Nous allons commencer par ajouter des bordures à notre tableau. Pour cela, nous allons utiliser les propriété CSS border, et border-collapse.

On connaît déjà la propriété border qui va nous permettre de créer des bordures autour de chaque cellule de notre tableau. Ensuite, pour « coller » ces différentes bordures entre elles, nous allons utiliser la propriété border-collapse.

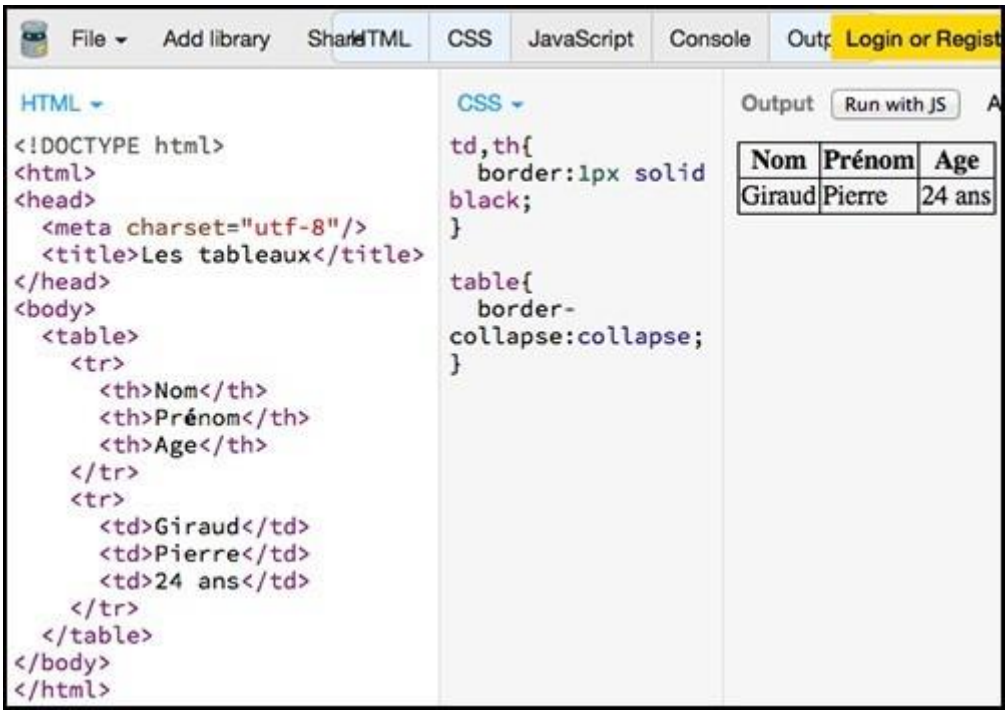
Cette propriété accepte trois valeurs : collapse (les bordures des cellules se collent), separate (valeur par défaut) et inherit.



Faites bien attention à appliquer la propriété border-collape à l'élément table et la propriété border à vos éléments td, sinon ça ne marchera pas !

Généralement, les tableaux possèdent une ligne d'en-tête. Pour créer cette ligne, on va cette fois-ci utiliser l'élément th (table head) à la place du ou des éléments td de notre première ligne. D'un point de vue sémantique, l'élément th est à td ce que l'élément h1 est à p.

Si l'on veut créer une ligne d'en-tête en colonne, il suffit de remplacer le premier élément td de chaque élément tr par des éléments th.



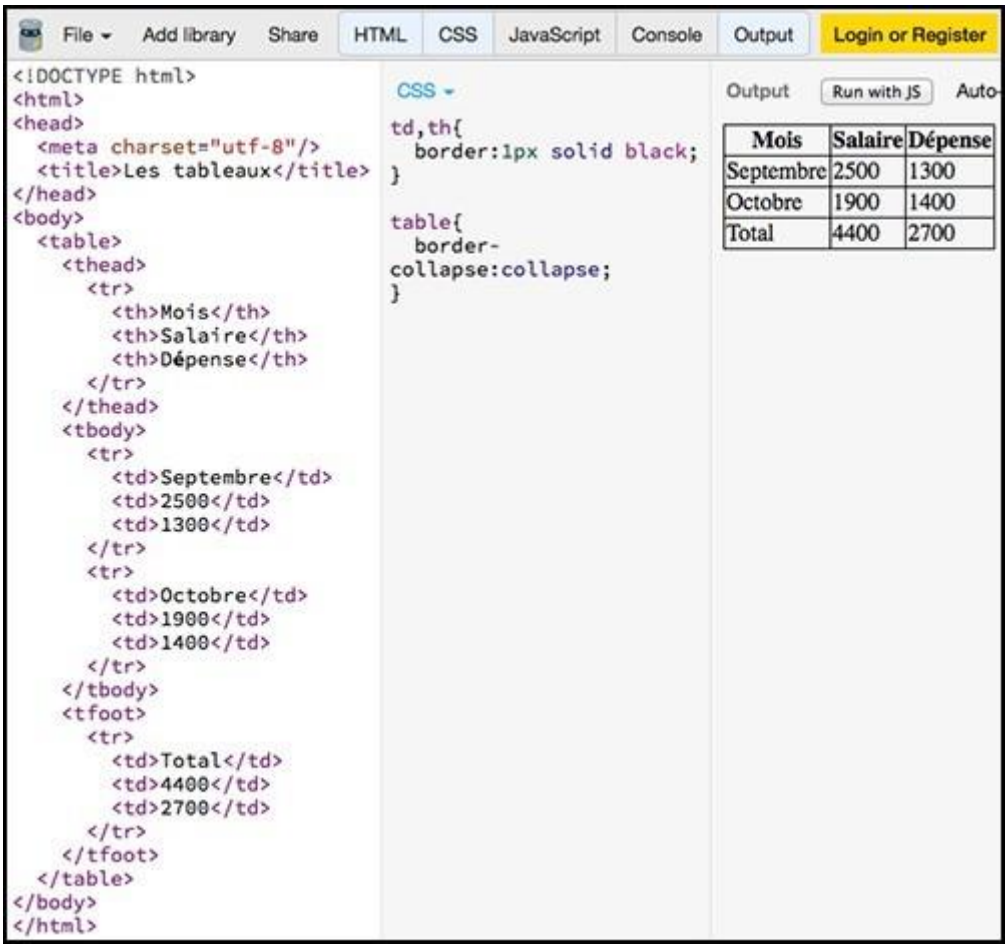
Évidemment, on n'oubliera pas d'appliquer notre propriété border à nos éléments th également.

4.3.3. Construire un tableau structuré

Si vous créez un tableau long, il sera certainement préférable de commencer à l'organiser en le divisant en plusieurs sous-parties.

On peut diviser un tableau en trois sous-parties : une partie d'en-tête (header), un corps de tableau (body) et un pied (footer). Ces trois parties sont matérialisées en HTML par les éléments thead, tbody et tfoot.

L'élément thead va entourer la ou les lignes d'en-tête. L'élément tfoot va comporter des données récapitulatives de notre tableau, comme des totaux par exemple. Enfin, l'élément tbody va contenir notre tableau à proprement parler.

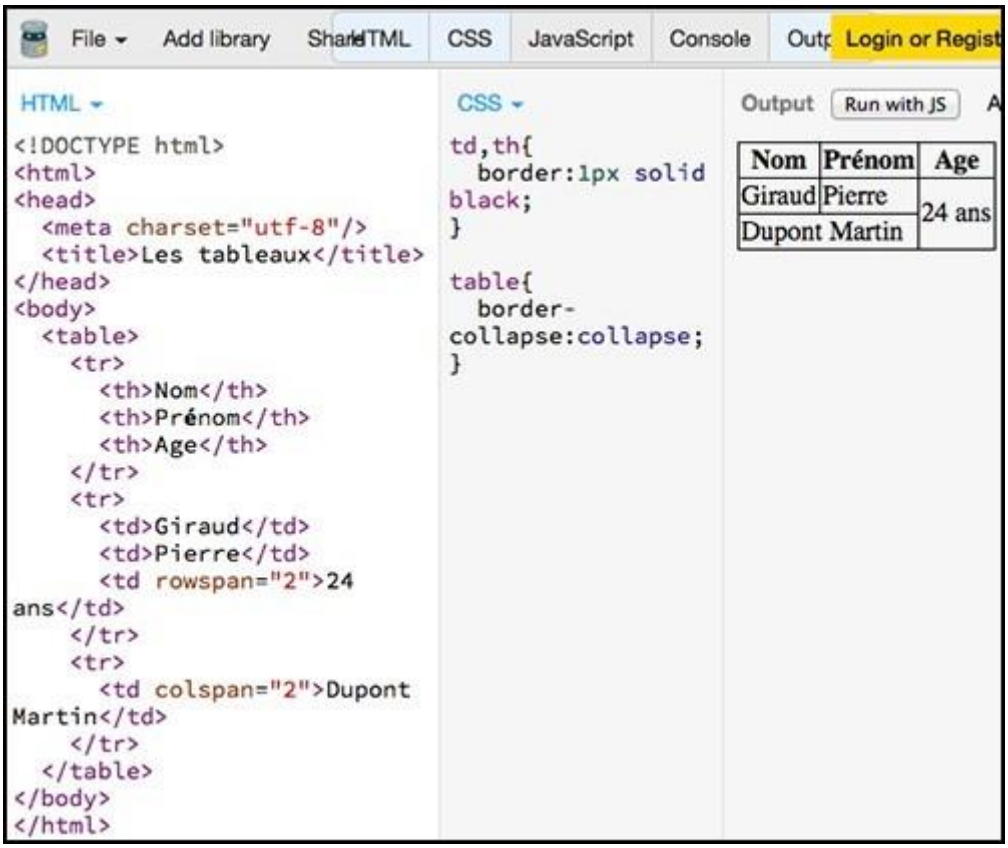


4.3.4. Combiner des cellules

Pour combiner des cellules, on va utiliser les attributs HTML colspan et rowspan.

L'attribut colspan va nous permettre de combiner des cellules appartenant à différentes colonnes dans une même colonne tandis que l'attribut rowspan va nous permettre de combiner des cellules provenant de différentes lignes.

Chacun de ces deux attributs accepte un nombre entier en valeur qui indique le nombre de cellules qui doivent être collées entre elles.



Ici, nous avons fusionné les cellules des deux colonnes « Nom » et « Prénom » en une pour « Dupont Martin » et les cellules de deux lignes pour « 24 ans ».

4.4. Les Formulaires

4.4.1. Introduction aux formulaires

Les formulaires sont certainement le moyen le plus simple et le plus utilisé pour recueillir des données à propos de vos utilisateurs. En cela, ils sont essentiels et incontournables.

Cependant, nous touchons là aux limites du HTLM. En effet, si l'on peut créer des formulaires en HTML, en ne peut pas en revanche stocker ni utiliser les données du formulaire avec ce langage. Pour cela, nous devrons utiliser d'autres langages comme le PHP et le MySQL par exemple.

4.4.2. Créer le squelette d'un formulaire

Pour créer notre formulaire, nous allons devoir tout d'abord utiliser l'élément form avec deux attributs : method et action.

L'attribut action, tout d'abord, va servir à indiquer où les informations recueillies dans le formulaire doivent être envoyées pour être traitées. Ce sera généralement vers une page PHP. Comme nous ne savons pas encore coder en PHP, il faudra l'imaginer pour le moment.

L'attribut method va lui spécifier de quelle manière on va envoyer ces données. On peut choisir entre deux valeurs : GET et POST. En utilisant la valeur get, les données vont transiter via l'URL de la page ce qui ne sera pas le cas si l'on utilise la valeur post.

Il faut savoir que la valeur get est assez limitée par rapport à post. En effet, avec get, on est limité dans le nombre d'informations que l'on peut envoyer et surtout les informations sont visibles lors de l'envoi, ce qui est problématique si l'on envoie un mot de passe par exemple.

C'est pourquoi il vaut mieux utiliser la valeur post, qui ne possède pas ces inconvénients.

```HTML
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8"/>
  <title>Les formulaires</title>
</head>
<body>
  <form method="POST" action="traitement.php">
  </form>
</body>
</html>
```

4.4.3. Créer un formulaire simple

Nous allons commencer par créer un formulaire très simple, demandant simplement un pseudo et un mot de passe à l'utilisateur.

Pour capturer des données textuelles simples comme un pseudo par exemple, on utilise soit l'élément input (pour des textes courts), soit l'élément textarea (pour des textes longs).

Dans notre cas, nous allons utiliser l'élément input. Cet élément prend forcément un attribut « type ». La valeur de l'attribut type correspond au type de données demandées. On a le choix entre text, password, date, email, tel, number, time, color et url.

Toutes ces valeurs ont été créées pour des raisons de sémantique. A noter également que l'affichage par défaut de chaque champ de votre formulaire pourra être légèrement différent selon la valeur choisie pour l'attribut type.

Il faut également préciser un deuxième attribut à l'élément input qui est l'attribut name. On lui donnera la valeur que l'on souhaite en essayant de rester cohérent. Cet attribut va nous être très utile pour traiter les données de notre formulaire.

A noter que l'élément input est représenté sous forme d'une balise orpheline.

Il va maintenant falloir indiquer à votre visiteur ce qu'il doit renseigner comme information dans chaque champ. C'est le rôle du label, qu'on va donc ajouter juste avant l'input.

A noter que pour afficher du texte dans un formulaire, il faut entourer ce texte avec un élément de type block. Généralement, on utilisera donc un élément p.

Ensuite, toujours pour des raisons de sémantique, il est bon et conseillé de lier le label à l'input qui lui correspond. Pour cela, on va ajouter un attribut for au label et un attribut id à l'input, et leur attribuer exactement la même valeur. Notez que cette valeur peut être la même que celle donnée à l'attribut name de l'élément input.

Voilà donc pour notre champ « pseudo ». Maintenant, on va faire exactement la même opération pour notre champ mot de passe, à la différence que cette fois on va utiliser un input de type « password ». Cela aura pour effet que les caractères inscrits ne s'affichent pas à l'écran.

4.4.4. Saisie d'un champ email ou url

Pour demander à l'utilisateur de saisir son adresse mail, il vous faudra utiliser un input de type « email ». Cela permet à votre navigateur de savoir qu'il doit normalement recevoir une adresse mail.

Si vous souhaitez donner la possibilité à vos utilisateurs de laisser un lien vers leur site Internet, vous devrez utiliser un input de type « url ».

Si, enfin, vous voulez que l'utilisateur renseigne son numéro de téléphone, il vous faudra utiliser un input de type « tel ».


```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8"/>
  <title>Les formulaires</title>
</head>
<body>
  <form method="POST" action="traitement.php">
    <p>
      <label for="mail">Entrez votre mail :
</label>
      <input type="email" name="mail" id="mail"/>
      <br/>
      <label for="site">Entrez l'adresse de votre
site :</label>
      <input type="url" name="site" id="site"/>
      <br/>
      <label for="num">Entrez votre téléphone :
</label>
      <input type="tel" name="num" id="num"/>
    </p>
  </form>
</body>
</html>
```

4.4.5. Créer une zone de saisie multi-lignes

Imaginons que vous vouliez créer un livre d'or pour votre site web, ou tout simplement que vous souhaitiez laisser la possibilité à vos visiteurs de laisser des commentaires, alors vous aurez certainement besoin de créer une zone de texte multi-lignes.

Pour faire cela, on va cette fois-ci utiliser l'élément textarea plutôt que l'élément input. Tout comme pour l'élément input, on va lui attribuer un attribut name et utiliser un label.

Si vous souhaitez modifier la taille de votre champ, vos pouvez évidemment utiliser les propriétés width et height en CSS. Notez que cela fonctionne également pour un élément de type input.

4.4.6. Cases à cocher, zones d'options et listes

On peut encore agrémenter notre formulaire en proposant à nos visiteurs de choisir une réponse parmi différentes possibilités.

Nous allons voir trois façons de faire cela avec les types checkbox (cases à cocher), radio (zones d'options) et l'élément select (listes déroulantes).

Pour créer une zone de formulaire avec des cases à cocher, il suffit d'utiliser l'élément input avec cette fois-ci un attribut de type checkbox.

Cette fois-ci, on écrira le label après l'input afin de bien avoir la case à cocher avant le texte.

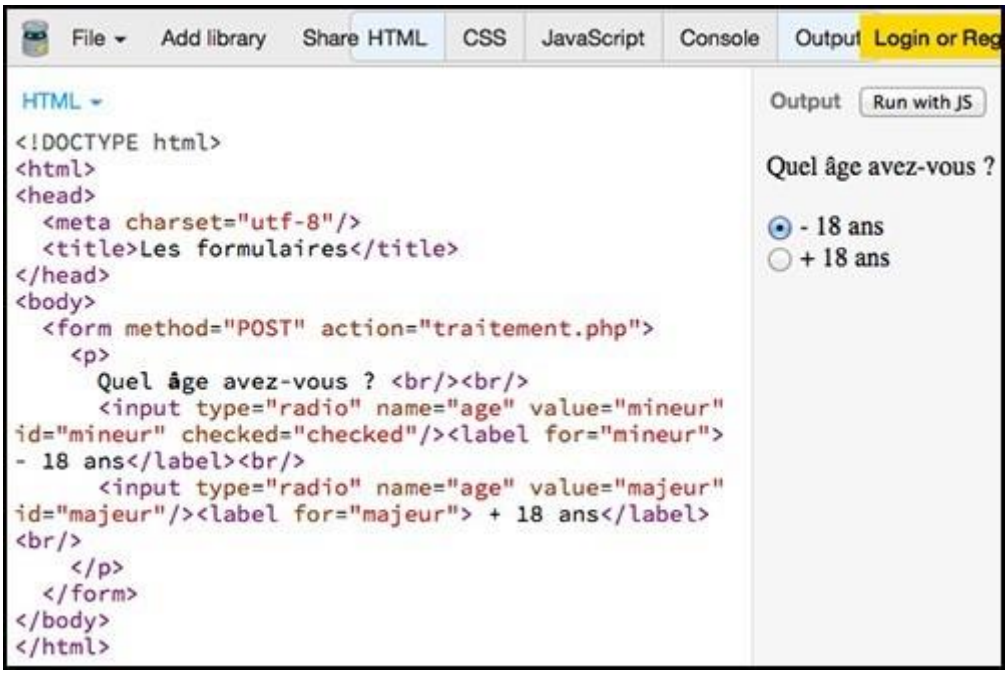
Pensez bien également à mettre une valeur différente au name de chaque input afin de pouvoir par la suite identifier quelle case a été cochée.

Si vous le désirez, vous pouvez pré-cocher une case par défaut en ajoutant l'attribut checked à un input.

Pour les zones d'options, on va cette fois-ci utiliser un élément input de type radio.

Notez que cette fois-ci, et dans ce cas uniquement, vous devez donner la même valeur à l'attribut name pour toutes les options à l'intérieur d'une même question.

On utilisera donc en plus un attribut value afin de bien savoir quelle case a été cochée par le visiteur.



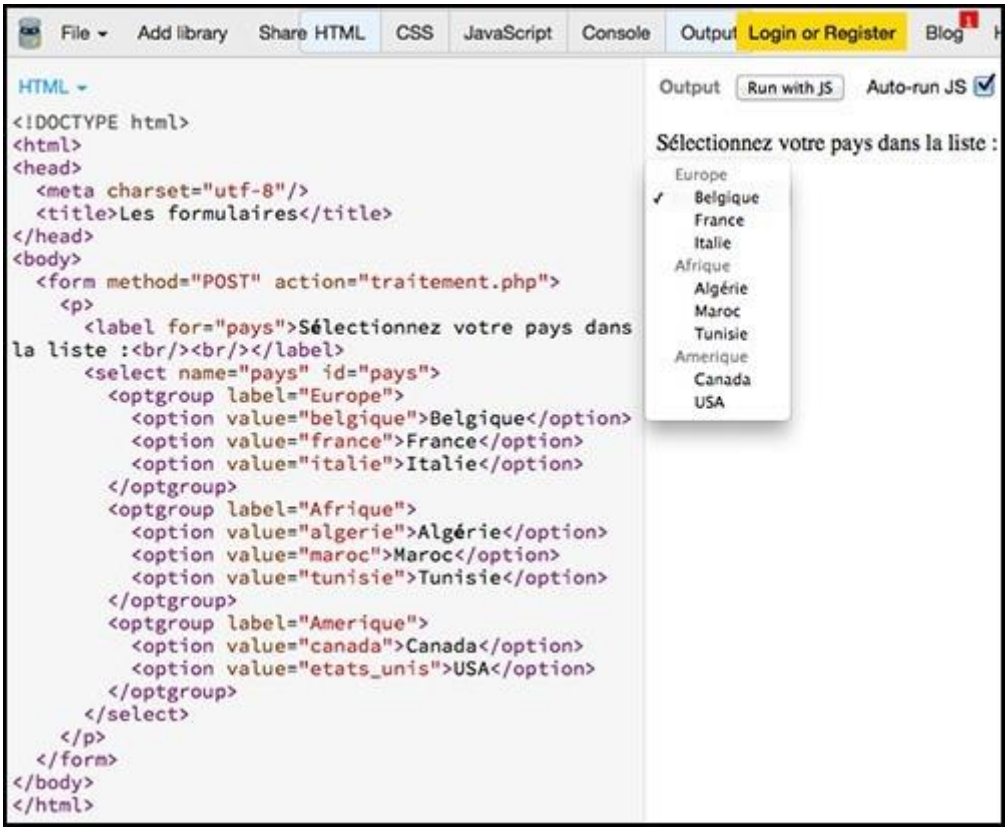
On peut encore une fois utiliser l'attribut checked afin de pré-cocher une case.

La différence entre les cases à cocher et les zones d'options est que le visiteur ne peut choisir qu'une réponse dans le cas des zones d'options, au contraire des cases à cocher où il peut cocher autant de cases qu'il souhaite.

Les listes déroulantes, enfin, s'utilisent généralement lorsqu'il faut faire un choix parmi une longue liste. On ne va cette fois-ci pas utiliser l'élément input mais l'élément select à la place avec ses attributs name et id.

A l'intérieur de cet élément select, nous allons utiliser un élément option pour chaque option de la liste accompagné d'un attribut value.

Enfin, sachez également que l'on peut grouper les options des listes déroulantes sous un dénominateur commun grâce à l'élément optgroup et son attribut label.

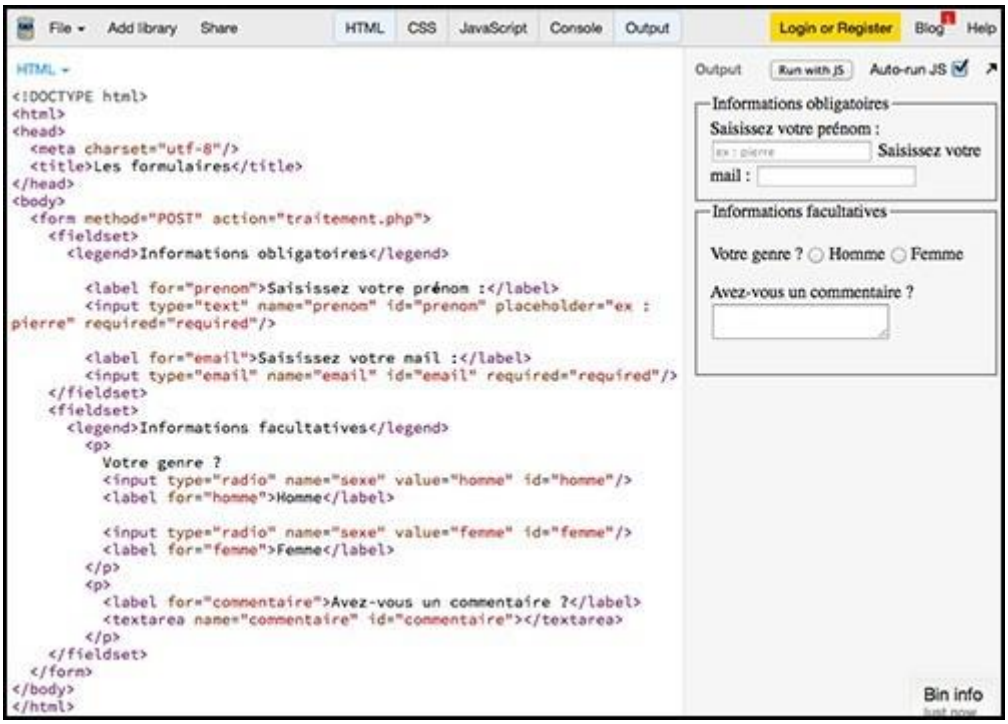


4.4.7. Finaliser et envoyer un formulaire

L'attribut placeholder sert à donner davantage d'indications sur vos champs à vos visiteurs. Il vous sert à donner un exemple de remplissage de champ.

Pour rendre une question de votre formulaire obligatoire, vous devrez utiliser l'attribut required.

Enfin, l'élément fieldset sert à organiser votre formulaire en différentes rubriques. Cela peut être utile si celui-ci devient long. Vous pouvez ensuite donner une légende à chaque rubrique grâce à l'élément legend.



Pour créer le bouton d'envoi, on va tout simplement utiliser un input de type submit avec un attribut value.

Il ne reste « plus » qu'à apprendre à traiter les données grâce au langage PHP...

5. Aller plus loin

5.1. Le responsive design

Lorsque l'on parle de responsive design, on parle généralement de l'ensemble des techniques nous permettant de créer un site qui pourra s'adapter en fonction de la taille de l'écran de vos visiteurs.

Aujourd'hui, nous disposons de trois moyens pour créer un site pouvant s'adapter à différents terminaux :

1. Créer un site dédié pour chaque terminal différent (un site pour mobile, un site pour tablettes, un pour ordinateur, etc.)
2. Créer des applications mobiles natives (pour Android, iPhone, etc.)
3. Créer une version responsive de votre site.

Chaque méthode possède des avantages et des inconvénients, ainsi que des prix divers. Dans notre cas, nous allons nous intéresser à la dernière : la création d'une version responsive.

Pour créer la version responsive de notre site, nous allons utiliser ce qu'on appelle des media queries, qui ne sont in plus ni moins que des règles qui seront appliquées selon certaines conditions (par exemple, lorsque la taille d'un écran est comprise entre X et Y pixels).

On va ainsi, grâce au media queries, pouvoir modifier le style de chaque élément de notre site web afin de l'adapter à l'écran de vos visiteurs. La détection de la taille de l'écran se fait évidemment automatiquement.

Deux solutions s'offrent à nous pour appliquer des media queries : soit on crée un nouveau fichier CSS (généralement, son nom sera responsive.css), soit on rajoute ces règles dans notre fichier CSS principal.

Si l'on utilise la première solution, il nous faudra charger une feuille de style différente selon la taille de l'écran de vos visiteurs. Pour cela, nous devrons modifier l'élément link contenu dans l'élément head de notre page HTML.

Pour l'instant, nous allons nous contenter de les ajouter à un fichier CSS déjà existant pour plus de simplicité.

Écrivons immédiatement notre première media query :

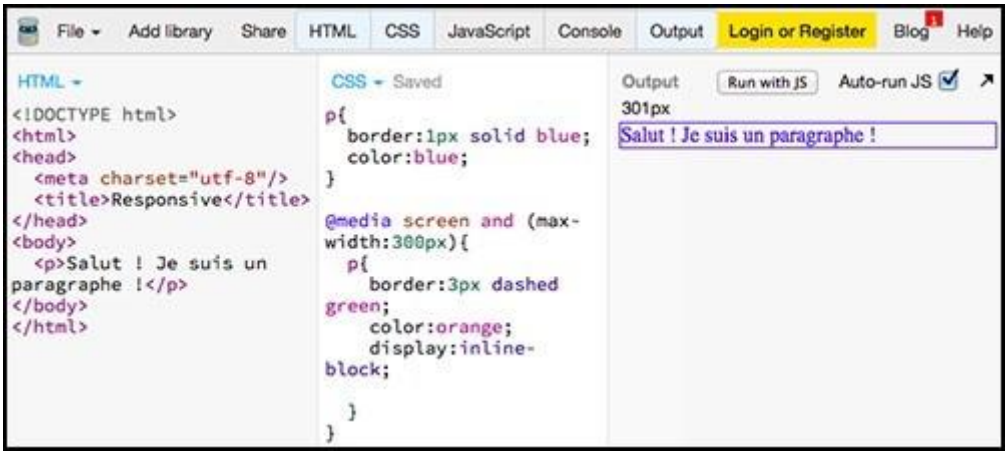
```css
p{
    color:red;
}

@media all and (max-width: 1280px){
    p{
        color:blue;
    }
}
```

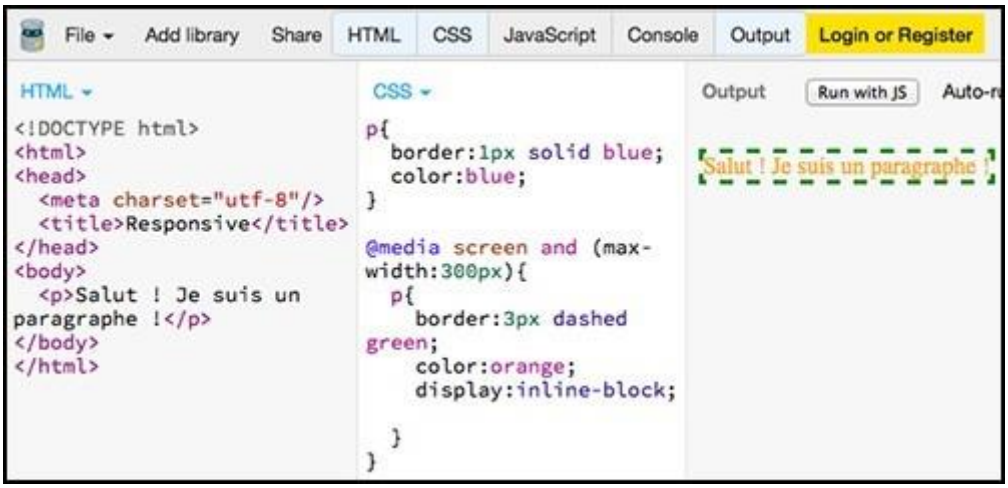
Notre media query ici correspond à la ligne avec le arobase. Comme vous le voyez, nous allons réécrire des propriétés CSS à l'intérieur de nos media queries.

Dans cet exemple, nous affichons par défaut tous les paragraphes de notre page HTML en rouge. Cependant, notre media query vient changer cela. Pour faire court, les paragraphes s'afficheront en bleu pour tous les écrans dont la taille sera inférieure à 1280px.

Lorsque l'on veut créer une version responsive d'un site, on utilisera très souvent la media query @media. Cependant, vous devez savoir qu'il en existe bien d'autres, comme par exemple @width, @height, @color, @orientation, etc.

De même, nous utiliserons majoritairement @media all (la règle s'appliquera à tous les types d'écrans), mais nous pouvons également utiliser @media screen (la règle ne s'applique qu'aux écrans classiques, @media handheld (mobile) ou encore @media tv (télévision).





Ici donc on demande aux paragraphes de s'afficher en bleu avec une bordure d'1px solide et bleue pas défaut.

Cependant, on pose une règle supplémentaire : si la taille de l'écran (ici, la taille de notre fenêtre « output ») est inférieure à 300px pour un écran classique, alors les paragraphes s'affichent désormais en orange, ont une bordure verte en tirets de 3px d'épaisseur et on applique un type inline-block à nos paragraphes.

Cet exemple devrait vous faire comprendre tout ce qu'il est possible de faire grâce aux media queries : vous pouvez changer le type d'un élément, adapter la taille d'une boîte, ajouter des propriétés clear pour un meilleur rendu selon la taille de l'écran ou encore changer le positionnement de vos éléments.

5.2. Les pseudo-classes en css

Le CSS permet de changer le style de nos éléments selon l'état de ceux-ci, c'est-à-dire de façon dynamique. Pour faire cela, nous allons utiliser ce qu'on appelle les pseudo-classes (ou pseudoformats).

Les pseudo-classes peuvent par exemple être utilisées pour modifier l'apparence d'un élément lorsque la souris de vos visiteurs est dessus, ou lorsque l'élément a déjà été cliqué.

Pour modifier l'apparence d'un élément au survol de la souris, nous allons utiliser la pseudo classe :hover de cette façon :

Dans le premier état, ma souris n'est pas sur l'élément p, donc rien ne se passe. Dans le second cas, en revanche, j'ai passé ma souris sur l'élément p. Celui-ci se color donc en rouge.

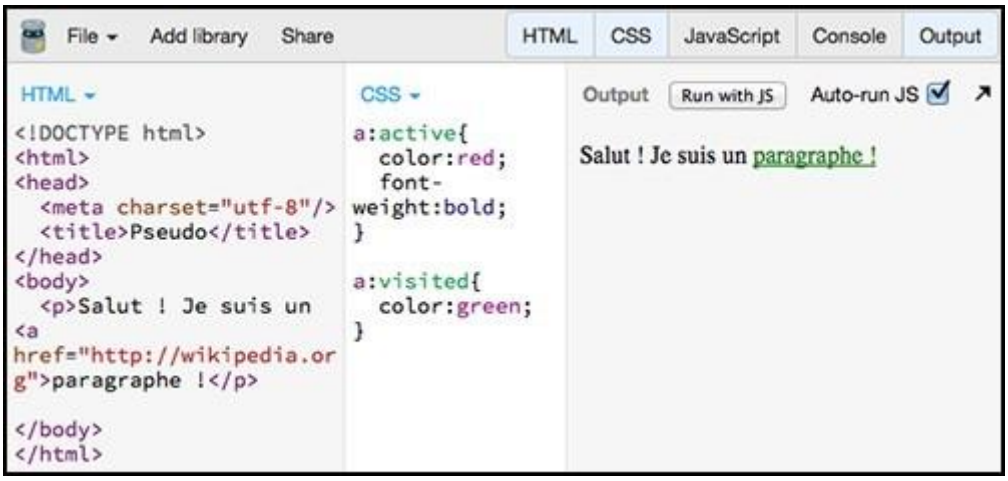
On peut également appliquer différents styles à des éléments selon qu'ils aient été cliqués ou non grâce au pseudo-classes :active (lors de la sélection) et :visited (une fois cliqué).

On appliquera souvent ces pseudo-classes à des éléments de type lien.

Ci-dessous, on applique une couleur rouge ainsi qu'une mise en gras au lien lors du clic, puis ensuite une couleur verte après qu'on l'ait cliqué.

Voici le résultat au moment du clic :

Et une fois que le lien a déjà été cliqué :



Les pseudo-classes peuvent nous mener beaucoup plus loin et nous permettre d'appliquer des styles à des éléments très précis ou selon un état précis des éléments.

Voici ci-dessous une liste de toutes les pseudo-classes utilisables :

Selector	Example	Example description
:link	a:link	Selects all unvisited links
:visited	a:visited	Selects all visited links
:active	a:active	Selects the active link
:hover	a:hover	Selects links on mouse over
:focus	input:focus	Selects the input element which has focus

5.3. Les pesudo-elements en css

Les pseudo-éléments sont utilisés en CSS pour modifier l'apparence de certaines parties spécifiques d'un élément.

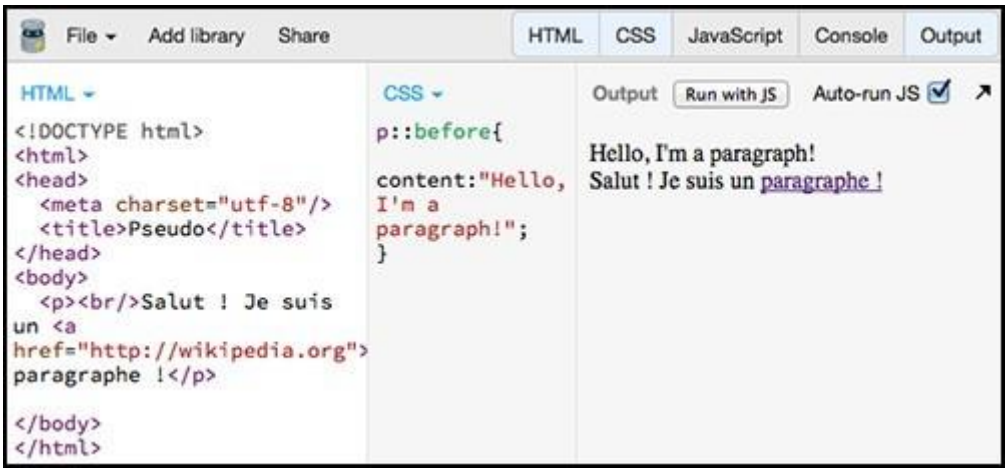
Les pseudo-éléments peuvent être utilisés, entre autres, pour modifier l'apparence de la première lettre d'un élément ou pour insérer du contenu avant ou après un élément.

L'utilisation des pseudo-éléments est très proche de celle des pseudo-classes, je ne m'étendrais donc pas autant dessus. Attention cependant : on va utiliser cette fois « :: » et non pas « : ».

Voici la liste des pseudo-éléments disponibles en CSS :

Selector	Example	Example description
::first-letter	p::first-letter	Selects the first letter of every <p> element
::first-line	p::first-line	Selects the first line of every <p> element
:first-child	p:first-child	Selects every <p> elements that is the first child of its parent
::before	p::before	Insert content before every <p> element
::after	p::after	Insert content after every <p> element
:lang(*language*)	p:lang(it)	Selects every <p> element with a lang attribute value starting with "it"

Voyons un exemple avec le pseudo-élément ::before par exemple.



5.4. Les éléments structurants du HTML5

Un des grandes nouveautés du HTML5 a été d'introduire des éléments structurants.

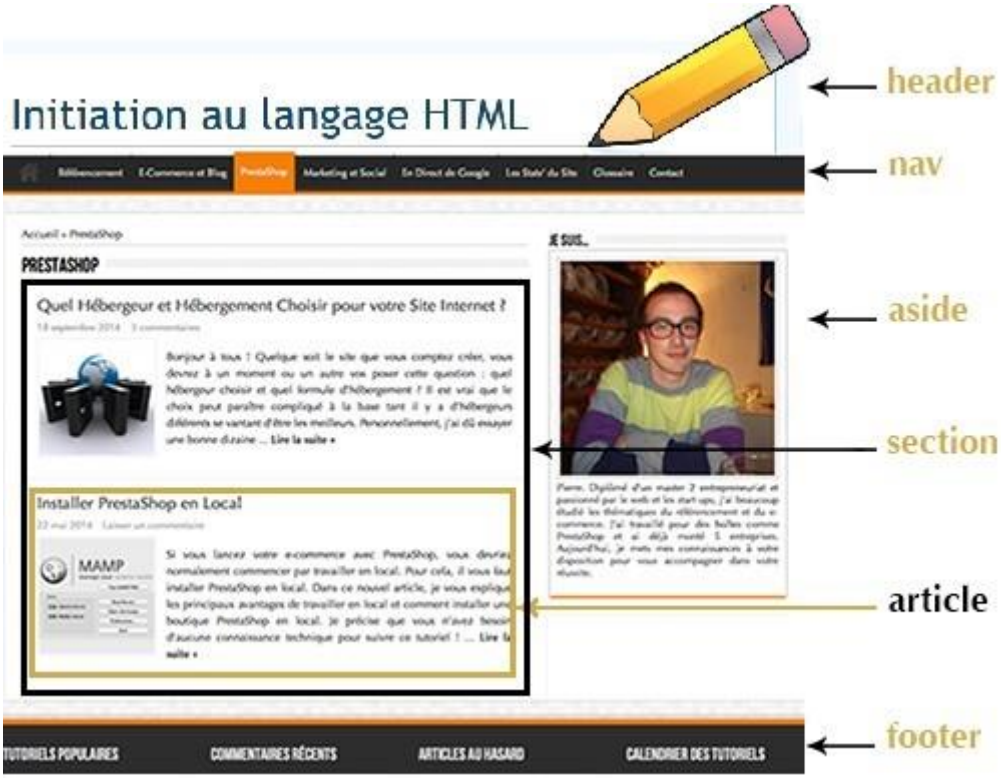
L'idée était une nouvelle fois d'améliorer la sémantique et de nous permettre également de faciliter nos mises en page.

Ces éléments vont nous permettre de dire aux moteurs de recherche « ceci est l'en-tête de mon site », « ceci est un article », « ceci est une section » ou encore « ceci est mon pied de page ».

Les éléments structurants introduits sont les suivants :

- Header (l'en-tête de votre site) ;
- Nav (votre menu principal de navigation) ;
- Section (pour grouper des éléments en fonction d'une thématique commune) ;
- Article (un article indépendant) ;
- Aside (informations complémentaires) ;
- Footer (le pied de page).

En pratique, voilà à quoi correspondent les différents éléments :



Ainsi, idéalement, tout le contenu du header devrait être placé entre les balises <header>, tout le contenu du footer entre les balises <footer>, etc.

Il y a fort à parier que ces éléments vont être de plus en plus pris en compte par les moteurs de recherche et vont compter de plus en plus pour le référencement des sites, ne les négligez donc pas !