HTML & CSS

La Création de pages Webs

Antoine Dieudonné

UTOPIOS
TRANSFORMATION DIGITALE INTELLIGENTE

Introduction à l'HTML

Qu'est-ce que l'HTML ?

L'HTML (**HyperText Markup Language**) peut être vu comme étant les fondations d'une page web. C'est un langage servant avant tout à la structure de la page et qui sera interprété par le navigateur dans le but de nous fournir un socle sur lequel on peut naviguer.

Par exemple, c'est grâce à l'HTML que l'on peut choisir d'insérer une image dans notre page, de mettre des sections en gras ou de modifier le titre de notre fenêtre / onglet avec l'intitulé de la page en cours de visionnage. Un site web peut donc être construit facilement en un ou plusieurs fichiers portant l'extension **.html** et qui pourront être navigués via l'utilisation de balises de type « **ancres** » nous servant à aller d'une partie du site à l'autre plus facilement. Cette navigation entre dans le contexte de ce que l'on appelle le « routing ».

A côté de l'HTML, il est fréquent de trouver du CSS et du Javascript, ces derniers servant respectivement à la stylisation et à l'interactivité de nos pages internet.

Il est cependant intéressant de noter qu'il est possible d'avoir un site web avec uniquement de l'HTML, mais qu'il n'est pas possible d'avoir un site web s'en passant (composé uniquement de fichiers CSS et / ou JS), car ces derniers se basent sur la structure créée par l'enchainement des balises HTML pour fonctionner.

Un langage de Balises

Un site internet se compose de multiples balises qui contiennent elles-mêmes d'autres balises. Une balise s'écrit sous la forme d'un mot (et d'attributs) entouré de chevrons. Il existe deux grands types de balises : les balises classiques et les balises auto-fermantes.

```
<h1>Accueil</h1>
```

Une balise classique dotée d'une balise ouvrante et d'une balise fermante.

```
<img src="Source" alt="Description">
```

Une balise auto-fermante.

Dans une balise, il est également possible de trouver des attributs, qui fonctionnent selon un principe de clé et de valeur, pour par exemple indiquer la source d'une image, le lien de sortie d'un lien hypertexte ou le nombre de pixel que prendra un tableau en hauteur et / ou largeur.

Les Commentaires

Il est également possible d'ajouter des commentaires à notre site web via l'utilisation d'une notation de la sorte :

```
<!-- Commentaire sur une ligne -->


<!-- Commentaire sur plusieurs
lignes -->
```

La Structure de Base

Dans une page HTML 5, il est nécessaire d'avoir un template de base dans lequel travailler.

Celui-ci peut être généré automatiquement par la majorité des éditeurs de code, et ressemble à ceci :

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>

</body>
</html>
```

On peut ainsi observer qu'il se compose tout d'abord d'une balise spécifiant le type de fichier (le langage balise n'étant pas réservé à l'HTML) ainsi qu'une balise **« html »** qui englobe toute la page.

Dans la balise html se trouvent deux sections :

- La première, la balise **« head »** sert à décrire et à constituer le ciment de la page web (c'est ici que se trouve la configuration et/ou les liens pour l'import de feuille de style)
- La seconde, la balise **« body »**, est l'endroit où va se trouver la majorité de notre code, car c'est ce qui apparaitra à l'écran lors de notre ouverture de la page

La balise « body » est souvent constituée à son tour de trois grandes sections, qui sont le **« header »**, le **« main »**, et le **« footer »,** permettant de séparer la page de sorte à avoir par exemple une zone de navigation ainsi qu'un emplacement pour y placer les liens externes (vers par exemple des réseaux sociaux).

Structurer le texte

Dans le but de structurer le texte, plusieurs balises existent :

- **<h1>:** Cette balise attribue à notre texte un titre « principal », qui ne doit par convention n'être qu'au nombre de un dans une page HTML.
  - De **<h2>**à **<h6>** pour ajouter des titres de moins en moins importants
- **
 :** Cette balise auto fermante sert à ajouter de l'espace entre deux parties de notre site
- **<hr> :** Pour ajouter une ligne horizontale entre deux sections de notre page
- **<p>**: Sert à créer un nouveau paragraphe pour grouper du texte ensemble
- ** :** Sert à mettre en italique le texte en plus de lui donner un niveau d'importance particulier (Semblable à **<i>**)
- **** : Sert donner de l'importance au texte en plus de le mettre en gras (Semblable à **<b>**)

```
<body>
    <h1>Bonjour à tous</h1>
    <hr width="80%" size="5">
    <h2>Section A</h2>
    <br>
    <p>Voici la <em>première</em> partie du site. <strong>Attention !</strong></p>
</body>
```

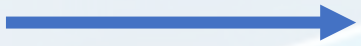
Les Listes

Il existe deux types de listes que l'on peut utiliser en HTML : Les listes à puces et les listes numérotées.

- Pour réaliser une liste à puce, il suffit de créer une section dans notre site qui contiendra les balises ** (unordered list)** dans lesquelles se trouveront de nombreuses ** (list item)** qui serviront à lister les différents éléments que l'on veut afficher.
- Pour faire une liste numérotée, il suffit de mettre les éléments **<li>** dans un élément de type ** (ordered list)**

Il est également possible de mettre des listes dans des listes, comme dans l'exemple ci-dessous :

```
<ul>
    <li>Item A</li>
    <li>Item B</li>
    <ol>
        <li>Sous-item A</li>
        <li>Sous-item B</li>
        <li>Sous-item C</li>
    </ol>
    <li>Item C</li>
    <ul>
        <li>Sous-item A</li>
        <li>Sous-item B</li>
    </ul>
</ul>

<ol>
    <li>A</li>
    <li>B</li>
    <li>C</li>
</ol>
```



- Item A
- Item B
    1. Sous-item A
    2. Sous-item B
    3. Sous-item C
- Item C
    - Sous-item A
    - Sous-item B

1. A
2. B
3. C

Les Images

Un site internet est sans aucun doute plus intéressant lorsque l'on peut y ajouter des images. Pour ce faire, il va falloir utiliser une balise auto fermante qui se compose de la sorte :

```
<img src="Source de l'image" alt="Texte descriptif de l'image">
```

La source de l'image doit être une adresse à laquelle notre navigateur va accéder dans le but de charger l'image pour l'afficher. Si jamais le lien est mort et qu'il n'est pas possible d'accéder à l'image, alors il sera possible de voir son descriptif via le texte que l'on aura placé dans l'attribut « **alt ».** Cet attribut est également essentiel pour la navigation des personnes ayant des problèmes de vue (les navigateurs se chargeront alors de lire les attributs de nos balises pour eux, et le descriptif leur permet ainsi d'avoir une présentation adaptée à leurs besoins).

Dans le cas où l'on aurait stocké l'image sur un serveur depuis lequel des utilisateurs accèdent à notre site, ou tout simplement dans un but de développement, il est possible d'accéder à l'image via un chemin relatif par rapport au fichier html actuellement visionné, comme ci-après :

```
<img src="../img/profile.png" alt="Image de profil">
```

Les Ancres

Dans la majorité des sites webs, il existe des ancres et des liens hypertext (d'où les deux premières lettres de l'acronyme HTML). Ces liens hypertextes servent à se déplacer dans un site, que ce soit vers un emplacement se trouvant dans la même page que celle actuellement visionnée, vers une page externe, ou vers un emplacement spécifique d'une page externe.

Pour réaliser un lien hypertexte, il faut se servir de la balise **<a>**, qui possèdera presque tout le temps l'attribut **href**, tel que :

```
<a href="http://google.com">Aller sur Google</a>
```

On peut ainsi voir que la balise d'ancre est une balise demandant une balise de fermeture. Le texte se trouvant entre les deux balises sera celui affiché à l'utilisateur, alors que le lien placé dans l'attribut href sera l'endroit visé par ce lien. Le fonctionnement du lien est le même que lorsque l'on se servait de l'attribut src de l'image, et l'on peut ainsi placer un lien amenant l'utilisateur à une autre page de notre site web en pointant vers son emplacement dans la hiérarchie des fichiers.

Par convention, un lien non visité se trouvera stylisé en bleu, alors qu'un lien visité sera en violet. Ce style est bien entendu modifiable en se servant du CSS.

Les Tableaux

Si l'on veut afficher plus facilement une série de données, ou si l'on veut placer des éléments les uns par rapport aux autres, on peut se servir de tableaux.

Pour créer un tableau, il va falloir se servir de plusieurs balises, qui servent à représenter différentes parties du tableau. Il nous faudra ainsi utiliser des balises telles que **<table>**, **<tr>**, **<td>** ou **<th>** pour structurer l'ensemble. Ces balises sont toutes des balises demandant d'avoir recours à une balise fermante, comme dans l'exemple ci-contre :

ID	Nom	Prénom	Age
1	MARTIN	Jacques	47
2	SMITH	John	23
3	GERTZ	Michel	62

- **<table> :** Marque le début d'un tableau
- **<thead> / <tbody> / <tfoot> :** Servent à spécifier des parties dans le tableau
- **<tr> :** Sert à créer une nouvelle ligne dans le tableau
- **<th> :** Sert à créer de nouvelles cellules stylisées (pour les titres de lignes ou de colonnes)
- **<td> :** Sert à créer de nouvelles cellules

Lorsque l'on créé un tableau, il est fréquent que l'on ait besoin d'ajuster la dimension des cellules les unes par rapport aux autres. Pour faire en sorte de résoudre un problème récurent, on peut se servir de l'attribut **colspan** disponible sur l'élément **<td>** pour lui faire prendre la largeur de plus d'une cellule.

Nom Complet		Adresse
John	Martin	404, Avenue des Roses

Les Formulaires

L'un des éléments le plus souvent rencontré dans la navigation d'un site web est sans doute le formulaire. Qu'il soit présent dans la barre de recherche, dans un formulaire de login utilisateur ou pour poster un élément dans une base de données, il est nécessaire pour cela d'avoir recours à un formulaire. Pour réaliser un formulaire, il faut tout d'abord se servir de la balise **<form>**, qui possèdera avant sa balise fermante l'ensemble du formulaire. Toute seule, cette balise ne sert pratiquement à rien, et c'est pourquoi on a pour habitude d'ajouter en son sein des balises de type **<label>** dans le but d'énoncer les endroits où l'on saisira les informations demandées. Ces balises de label serviront pour décrire les balises servant réellement à la récupération des données : les balises de type **<input>**.

Les balises input possèdent plusieurs types, qui sont spécifié via l'ajout de l'attribut type auquel on affecte un type (par exemple le type « **text** » servant à la récupération d'un ou plusieurs mots). A côté de cela, on peut aussi voir couramment les types **email**, **password**, ou encore **checkbox**.
Le déclenchement du formulaire dans le but d'envoyer les données saisies se fait quant à lui via un input de type **submit**, qui va cibler l'attribut **action** de notre formulaire. Malheureusement, à notre stade et sans utilisation de Javascript, il n'est pas possible de faire grand-chose de notre formulaire. Il est cependant possible de s'en servir pour envoyer un mail, via cette syntaxe :

```
<form action="mailto:mail@example.com" method="post">
  <input type="text" name="mail-from">
  <br>
```

Introduction au CSS

Qu’est-ce que le CSS ?

Les feuilles de style en cascade (Cascading Style Sheet – CSS) servent à donner le design d’un site web en se basant sur sa structure. Il ne peut y avoir de CSS pur car le propre de ce langage est qu’il se base sur les éléments présents dans le fichier .html, les balises, pour y fixer des propriétés de style.

Pour créer du CSS, il faut respecter une syntaxe assez simpliste, qui consiste à donner le ou les éléments que l’on souhaite cibler suivit d’un objet (délimité par deux accolades) qui contient un ensemble de clés et de valeurs.

Les clés possibles d’utiliser vont dépendre de l’élément que l’on veut cibler. Par exemple, on peut modifier la bordure d’un tableau pour qu’elle soit d’une certaine épaisseur et qu’elle fasse un dégradé de couleurs, ou modifier l’image de fond d’une cellule en plus de changer la police du texte qu’elle contient.

Le CSS est un langage très utilisé, et il est désormais possible d’en trouver des variantes, comme le SCSS ou le Sass, mais qui finiront par devenir du CSS au bout du compte.

Pour faire intervenir le CSS dans notre page HTML, il faut donc créer une jonction entre nos deux grandes parties. Cette jonction peut passer par trois fonctionnement majoritaires, qui reflètent également l’évolution des techniques de création de pages webs depuis les débuts de l’internet.

Le Style Inline, Interne et Externe

La première façon de faire intervenir le CSS dans l'HTML est sans doute d'ajouter un attribut « **style** » à une balise HTML. Cet attribut se verra ensuite être remplit d'une chaine d'instructions plus ou moins longues, ces instructions se terminant chacune par un point-virgule.

```
<body style="background-color: #fdd;">
  <header>
```

La seconde méthode pour ajouter du CSS à notre page est de l'injecter dans le **head** de notre page via l'utilisation de la balise **<style>**, qui requiert l'utilisation d'une balise fermante. Entre les deux balises se trouveront le code CSS se servant des sélecteurs et des attributs d'éléments comme dans l'exemple ci-contre :

```
  <title>Accueil</title>
  <style>

      body {
          background-color: #fff;
      }

  </style>
head>
```

Enfin, la troisième, et sans doute la plus utilisée, des méthodes pour l'injection de CSS dans une page web est l'utilisation d'un lien vers un fichier **.css** dans le header de notre page. Pour réaliser cette injection, il est nécessaire d'écrire toutes nos instructions de CSS dans un fichier dédié et de l'ajouter à notre page comme dans l'exemple ci-dessous :

```
 <title>Accueil</title>
 <link rel="stylesheet" href="mes-styles.css">
ead>
```

Les Sélecteurs CSS

Pour fonctionner, le CSS se base sur ce que l'on appelle les sélecteurs. Les sélecteurs CSS les plus utilisés sont les éléments de l'HTML, auquel on se réfère via l'utilisation des noms de balise, les classes, et les ids.

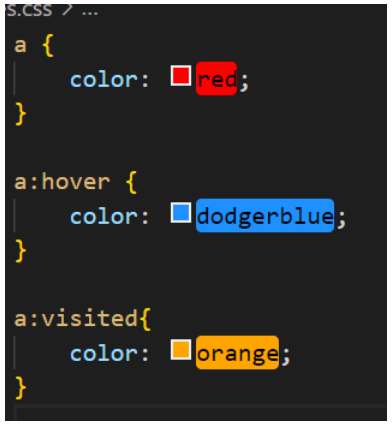
La sélection basée sur les éléments HTML va concerner tous les éléments de ce type présent dans la page liée à la feuille de style. Il faut donc faire attention, car en modifiant par exemple la hauteur et largeur d'une image, toutes les images de notre site vont se retrouver avec la même taille ! Une sélection basée sur le nom de balise se réalise via la syntaxe **element {}**

Une sélection basée sur la classe va concerner quant à elle tous les éléments possédant la classe dans leurs attributs (il est possible d'affecter une classe à un élément html via l'utilisation de **class="nom de classe"**). Plusieurs éléments HTML peuvent posséder la même classe, ce qui permet de constituer des groupes d'éléments partageant le même style mais n'ayant pas le même que les autres groupes. Pour sélectionner les éléments ayant une classe, il faut passer par une notation de type **.elementClass {}**

Une sélection basée sur l'id ne va concerner qu'un seul élément HTML. En effet, contrairement aux classes, les ids servent à rendre les élément uniques. Une selection basée sur l'id se défini par un sélecteur de type **#elementId {}**. Il est possible d'ajouter un id à un élément via la syntaxe **id="nom de l'id"**

Si l'on veut, on peut sélectionner plusieurs options en séparant les sélecteurs d'une virgule(**.thumbnail,.avatar**), ou les additionner pour augmenter la précision en les faisant se suivre (**.thumbnail.rounded.white-picture**)

Les Pseudo-classes



En plus de la sélection par le nom de l'élément et par son Id, il est également possible d'ajouter des états pour nos sélecteurs, comme par exemple les liens hypertextes n'ayant pas étés visités, ou ceux qui sont actuellement survolés par la souris. Pour ce faire, il faut utiliser une notation de type **selector:etat,** comme par exemple ci-contre :

Il est également possible de cibler spécifiquement les éléments enfants d'un autre élément, via une notation de type parent>enfant, comme par exemple toutes les images faisant partie d'une cellule de tableau via la notation **td>img.**

Dans le cas où l'on souhaite cibler les éléments étant le 4ème enfant d'un autre élément, on peut avoir recourt à une notation de type **:nth-child(4)**. De même, tous les éléments enfants pairs d'un parent seraient ciblés par une notation de type **:nth-child(even)**

Enfin, dans le cas où l'on aurait suivit par exemple une convention de nommage pour nos classes, il est possible de sélectionner toutes les divs dont le nom de leur classe commencerait par exemple par « info- » via un sélecteur de type **div[class^="info-"]**

De nombreux autres sélecteurs CSS existent, et une liste est disponible au lien ci-dessous :
https://www.w3schools.com/cssref/css_selectors.asp

Les divs et les spans

Lorsque l'on cherche à styliser son site web, il est fréquent d'avoir recourt à deux éléments HTML : les ** ** et les **.** Ces deux éléments HTML servent majoritairement à avoir un fonctionnement de type « inline » ou « block » en plus de nous fournir un contenant que l'on peut nommé en CSS via l'ajout d'une classe ou d'une id dans ses attributs.

Il est alors fréquent de parcourir des sites webs comportant de nombreuses div qui s'enchainent dans le but de fabriquer des interfaces plus ou moins poussées en arrangeant les parties d'un site les unes par rapport aux autres.

Par défaut, un élément de type span possèdera comme mode de fonctionnement un display de type « **inline** », ce qui veut dire que tous les éléments vont s'enchainer à la manière du texte d'un paragraphe avant d'effectuer un retour à la ligne lorsque l'un des ces span dépassera la largeur autorisée par son contenant (qui peut être la page elle^-même).

A contrario, les éléments de type « **block** », comme les div, vont s'enchainer verticalement en prenant toute la hauteur qui leur est attribuée car ils auront comme propriété de prendre toute la largeur de la page, quant bien même leur contenu n'en aurait pas besoin. Par exemple, un « heading » (**<h1>** par exemple) est de type block, ce qui fait que le texte qui va suivre se trouvera en dessous du titre et non à côté.

Il existe également des display de type « **inline-block** », qui permet à des éléments de type « block » de se positionner les uns à côté des autres tout en occupant verticalement la hauteur qui leur a été attribuée.

Il existe bien sur d'autres types de display sont disponibles, notamment les display « **flex** » et « **grid** ».

Le Margin, la Border et le Padding

Dans la création d'espacement entre des éléments HTML, les deux propriétés qui sont sans doute le plus souvent modifiées sont sans doute les propriétés **margin, border** et **padding**. Ces propriétés gèrent respectivement l'espacement extérieur de l'élément, l'épaisseur de son contour et son espacement intérieur. Ainsi, en combinant les éléments tels qu'un **** ou une ** ** avec les propriétés CSS de margin, de border et de padding, il est possible de créer des blocs regroupant des éléments ensemble de façon harmonieuse.

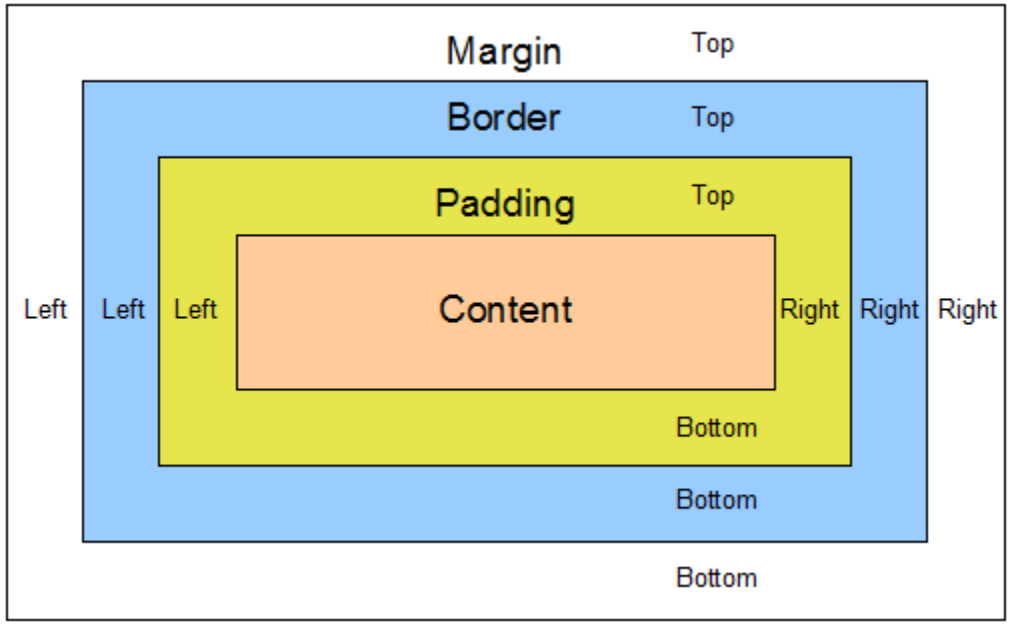
Pour alimenter les propriétés de margin et de padding, il existe plusieurs syntaxes, qui sont plus ou moins générales. Par exemple, dans le cas où l'on souhaiterai un espacement uniforme dans les 4 directions, alors le passage d'une valeur à la propriété **margin** ou **padding** suffit. Si l'on veut différencier les deux axes vertical et horizontal, alors il faut y indiquer deux valeurs. Enfin, si l'on veut repartir spécifiquement chaque axe, alors le passage de 4 valeurs est nécessaire. L'ordre des valeur va dans le sens des aiguilles d'une montre en commençant par la gauche.

```css
.ma-div1 {
    margin: 10px;
}

.ma-div2 {
    margin: 10px 20px;
}

.ma-div3 {
    margin: 0 10px 0 25px;
}
```

Un espacement uniforme de 10px autour de la div

Un espacement de 10px horizontalement et de 20px verticalement

Aucun espacement sauf de 10px en haut et de 25px en bas



Le Positionnement

Lorsque l'on cherche à positionner les éléments les uns à côtés des autres, il n'est pas obligatoire de se servir de tableaux. Il existe un certain nombres de règles en HTML / CSS qui régissent les bases du positionnement et de la place que vont prendre les éléments dans le flux de la page :

- **Le contenu détermine la taille** : Lorsque l'on cherche à placer des éléments, leur taille en largeur et en hauteur est déterminée par le contenu (un paragraphe prendra en hauteur la place requise pour afficher tous les mots qu'il contient, et une image la taille en hauteur et largeur du fichier source)
- **L'ordre des éléments suit l'ordre du code** : Si l'on veut qu'un élément soit positionné après un autre élément, alors il vaut mieux écrire son code après le premier.
- **Les enfants sont dans les parents** : Si un élément se trouve imbriqué entre deux balises parents, alors il sera positionné dans cet élément, par-dessus ce dernier.

A côté de cela, il existe plusieurs styles de positionnement :

- Le positionnement **statique** : C'est le positionnement de base des éléments en HTML, basé sur le flux du code
- Le positionnement **relatif** : Il est possible de positionner un élément relativement à sa position statique (qui restera en « fantôme » dans le flux de la page)
- Le positionnement **absolu** : On positionne l'élément relativement par rapport à son parent. En utilisant ce type de position, l'élément ne possède pas de « fantôme » (Il n'apparaît plus dans le flux naturel de la page).
- Le positionnement **fixe** : On positionne l'élément relativement à la page, ce qui permet de garder sa position malgré le défilement du site web.

La Gestion de la Taille

En CSS, on peut spécifier les taille et les distances de plusieurs façon :

- On peut se servir d'un chiffrement en **pixels**, ce qui permet d'avoir une certaine précision mais peut comporter des risques de dimensionnement ou d'espacement lorsque l'on change de résolution ou que l'on cherche à faire un site responsive (dont le positionnement réagit au changement de format du navigateur par exemple)
- On peut se servir de **pourcentages**, qui seront relatif au parent et / ou à des valeurs par défaut pour les polices.
- On peut se servir de valeurs de type **em** (dont le nom provient de l'histoire de la typographie et de l'utilisation de la lettre M pour la taille des caractères). La valeur de 1 est alors la valeur de base de la taille du texte, et on peut indiquer un rapport à cette taille par la valeur se trouvant avant « em », comme par exemple la moitié : **0,5em**
    - Il existe également une variante, les tailles en **rem** qui va se baser non pas sur les valeurs de l'élément parent, mais sur les valeurs de la racine (le « root »). Grâce à l'utilisation de « rem » au lieu de « em », on peut éviter les problèmes de bogues courants suite à l'enchainement de valeurs différentes sans prendre en compte le flux de la page.

Float et Clear

@ Utopios Consulting

Si l'on veut facilement faire des section compotant par exemple une image à côté d'un texte sans avoir recourt à des tableaux ou à des valeurs de display ou de positionnement difficiles à paramétrer, on peut se servir de ce que l'on appelle le « float » et le « clear ».

Ces deux propriétés CSS servant à permettre aux éléments de « flotter » à côté des autres en prenant en compte leur parents, ou à contrario de ne pas permettre le flottement d'élément à côté d'eux.

Ainsi, en spécifiant une valeur à la propriété « float », on peut dire que l'on souhaite mettre du texte entourant une image.

Vie de Chien

Lorem ipsum dolor sit, amet consectetur adipisicing elit. Fugiat, ex impedit. Voluptas necessitatibus, molestiae saepe ab quia pariatur. Ut totam neque molestiae laudantium atque mollitia illum vero itaque suscipit assumenda. Dolorem cumque obcaecati quibusdam quod error explicabo corporis eius alias non eveniet. Nostrum blanditiis veniam itaque, autem sequi quia aliquid animi necessitatibus distinctio inventore dolor earum, aperiam mollitia natus labore. Commodi perspiciatis sequi pariatur dolores libero quisquam illo repellat cum. Necessitatibus nulla, eaque iure reprehenderit ad, neque itaque veniam tempore totam perspiciatis quibusdam pariatur repellat assumenda consequuntur ratione magnam voluptatum! Repellendus veritatis doloremque nostrum quos et accusantium? Quas, dolor officiis magnam, aspernatur reiciendis deleniti excepturi laudantium eligendi ab nesciunt temporibus provident accusantium. Sint voluptatibus, itaque odio cumque voluptas modi in. Modi consectetur laudantium sed eligendi commodi amet quae fugit error obcaecati atque soluta adipisci, nihil illo unde quasi quam culpa omnis nulla reiciendis qui eum aspernatur consequatur perferendis quia? Voluptatem.

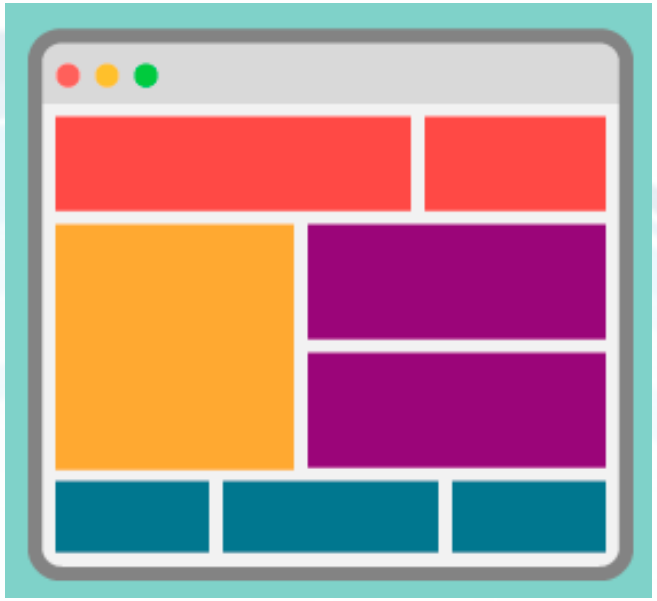
Les Flexboxes @ Utopios Consulting

Dans l'arrangement des éléments via le CSS existe deux grands candidats, le premier étant le display de type « flex » et basé sur l'arrangement de boites, d'où le nom de **flexbox.** Son objectif est la réalisation d'un layout simplifié, dans lequel on pourrait distribuer les éléments au sein d'un conteneur et qui les arrangerait « au mieux » (pour qu'ils puissent prendre la taille qu'il leur faudrait) selon un axe en fonction de la taille qu'ils sont censé prendre, en faisant en sorte de les mettre à la ligne automatiquement s'il n'y a plus la place nécessaire.

Pour réaliser un arrangement de flexboxes, il faut tout d'abord donner à un conteneur parent (souvent une ** **) le display de type « **flex** ». Une fois fait, on peut spécifier la direction que prendront les enfants, qui de base est horizontale et de gauche à droite. Pour l'en changer, on se sert de l'attribut « **flex-direction** ». Ensuite, on utilise un attribut gérant le déplacement des éléments vers un axe parallèle sous le premier en cas de dépassement ou s'il n'y a pas assez d'espace disponible via l'attribut « **flex-wrap** ».

Suite à cela, on peut définir l'alignement des éléments dans la direction de l'axe via l'attribut « **justify-content** » ou perpendiculairement à l'axe via les attributs « **align-items** » et « **align-content** ». Pour créer de l'espace entre les éléments, on peut spécifier un gap via l'attribut « **gap** », « **row-gap** » ou « **column-gap** ».

Dans les éléments enfants, on peut modifier l'ordre pour éviter le respect du flux HTML via l'attribut « **order** », augmenter la taille via le « **flex-grow** », la réduire en cas de besoin via le « **flex-shrink** » ou lui donner une taille par défaut via « **flex-basis** »



Les Grids

L'autre grande méthode pour arranger des éléments les uns par rapport aux autres est sans doute l'utilisation du display de type « grid », qui se base sur l'utilisation de lignes et de colonnes pour placer les enfants les uns après les autres en leur allouant de l'espace dés le départ. Ce type de display est moins dynamique que le display de type flex, mais permet d'être très précis lorsque l'on veut placer les bases de la structure de notre page web.

Pour commencer, il faudra encore une fois un conteneur, auquel on alouera la propriété « **display: grid;** », puis dont on spécifiera le nombre de lignes et de colonnes via l'utilisation de « **grid-template-column** » et de « **grid-template-row** ». Il est également possible de spécifier la grille via l'utilisation de « **grid-template-areas** ».

Une fois la grille créée, on peut spécifier l'alignement vertical et horizontal des éléments dans leurs cellules via l'utilisation de « justify-items » pour l'axe X et de « align-items » pour l'axe Y. Des espacements entre les cellules sont possible via le « grid-gap ».

Cuaque élément enfant devra à son tour spécifier son point de départ dans la grille via « **grid-column-start** » et « **grid-row-start** », et ses dimensions via l'utilisation de « **grid-column-end** » et de « **grid-row-end** ». Un raccourcis est possible dans ces paramétrage via simplement « **grid-column** » et « **grid-row** », par exemple :

```
.item-c {
  grid-column: 3 / span 2;
  grid-row: third-line / 4;
}
```

Dans le cas où le template de grille est l'utilisation des « areas » , alors on positionnera les éléments via « **grid-area** ».

Les Medias Queries

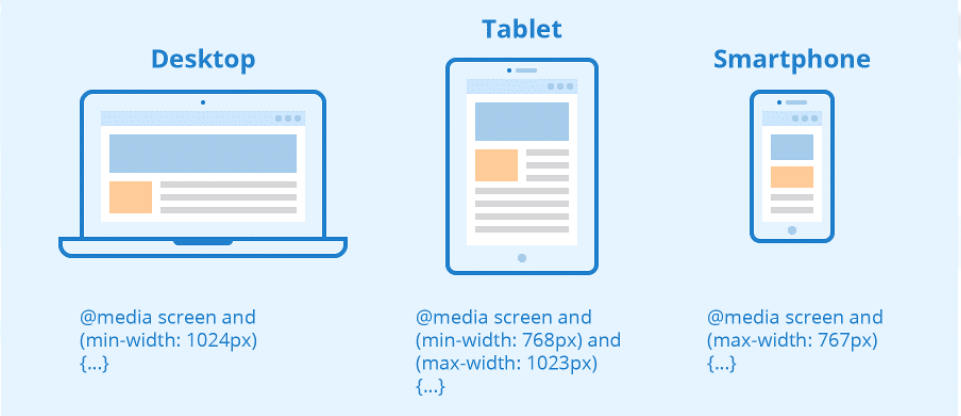
Les média queries servent à modifier les propriétés CSS de nos éléments en fonction de type d'appareil sur lequel l'utilisateur est en train de visionner notre site. Elles sont l'une des façon les plus connues de rendre notre site visionnable facilement sur mobile tout comme sur ordinateur sans avoir à faire deux sites différents.

Pour se servir des médias queries, on se sert d'une syntaxe de ce type :

```
@media screen and (max-width: 640px) {
```

Suite à cela, on va mettre entre accolades toutes les propriétés CSS qui devront être appliquées à notre HTML pour la configuration que l'on aura spécifié. Il est alors possible d'enchainer plusieurs configuration de média queries les unes après les autres, et de mettre également en commun ce qui ne doit pas être affecté par ces médias queries simplement en écrivant le CSS en dehors d'une série d'accolades.

Les configurations les plus utilisées de médias queries sont trouvable facilement sur internet, par exemple au lien ci-dessous :
https://css-tricks.com/snippets/css/media-queries-for-standard-devices/



Les Variables en CSS

Lorsque l'on travaille sur le style de notre site web, il est fréquent que l'on cherche à utiliser des gammes de couleurs qui seront les mêmes pour plusieurs éléments dans le but de respecter une charte graphique plus ou moins restrictive. Pour cela, il n'est pas forcément obligatoire de connaitre par cœur les valeurs hexadécimales de nos couleurs ou de devoir les rechercher dans le but de les copier / coller à la chaine.

On peut utiliser ce qui s'appelle les variables CSS, permettant de stocker sous la forme de mots-clé des valeurs de couleursou de taille. Pour ce faire, il est nécessaire de les écrire sous la forme de « **--nom-de-variable** » puis d'y accéder via l'utilisation de la syntaxe ci-dessous :

```
background-color: var(--main-bg-color);
```

Les variables sont disponibles en fonction du flux de l'HTML, en respectant le principe des enfants héritant de valeurs de leurs parents. Pour éviter de ne pas avoir accès aux variables à certains endroits, il est possible de déclarer les variables CSS dans la pseudo-classe **:root**, comme ci-dessous :

```
:root {
  --main-bg-color: brown;
}
```