

SUPPORT DE COURS B1-DEV1

HTML, CSS et BootStrap

CSS



CSS3

| date | révision |
|--------------|--------------------|
| Janvier 2019 | Amélioration (v67) |
| | |
| | |
| | |



TABLE DES MATIÈRES

| | | |
|---------|---|----|
| 1 | Introduction..... | 4 |
| 2 | Fonctionnement..... | 5 |
| 2.1.1 | Définition CSS..... | 5 |
| 2.1.2 | hiérarchie..... | 5 |
| 2.1.3 | Les sélecteurs..... | 6 |
| 3 | Mise en forme CSS..... | 7 |
| 3.1 | Inclure du CSS..... | 7 |
| 3.2 | Standardiser le CSS..... | 8 |
| 3.3 | Les couleurs et arrière-plans..... | 8 |
| 3.3.1 | Couleurs..... | 8 |
| 3.3.2 | Images..... | 9 |
| 3.4 | Les polices et le texte..... | 10 |
| 3.4.1 | Familles de polices..... | 10 |
| 3.4.2 | Taille de police..... | 11 |
| 3.4.3 | Alignement de texte..... | 11 |
| 3.4.4 | Utiliser des polices spécifiques..... | 12 |
| 3.5 | Les effets de survol..... | 13 |
| 3.5.1 | Effets sur les liens..... | 13 |
| 3.5.2 | Effets pour un texte..... | 13 |
| 3.6 | Les groupements d'éléments..... | 14 |
| 3.6.1 | Groupements logiques : "Class" et "Id"..... | 14 |
| 3.6.2 | Groupements physiques : div et span..... | 14 |
| 3.6.2.1 | <DIV> structure conteneur de bloc (%block)..... | 14 |
| 3.6.2.2 | structure en ligne (%inline)..... | 14 |
| 3.6.2.3 | Le modificateur CSS 'float'..... | 15 |
| 3.6.2.4 | Le modificateur CSS 'display'..... | 15 |
| 3.7 | Le modèle de boîte..... | 16 |
| 3.7.1 | Marges et espaces..... | 16 |
| 3.7.1.1 | Marges..... | 16 |
| 3.7.1.2 | Espaces..... | 17 |
| 3.7.1.3 | Bordures..... | 18 |
| 3.8 | Les divisions et la mise en page..... | 19 |
| 3.8.1 | Structure classique d'une page web..... | 19 |
| 3.8.1.1 | Tableau : obsolète..... | 19 |
| 3.8.1.2 | Frameset : obsolète..... | 19 |
| 3.8.1.3 | Division : la référence..... | 20 |
| 3.8.2 | Structure moderne d'une page web..... | 21 |
| 3.8.2.1 | Création de blocs..... | 21 |
| 3.8.2.2 | Block, inline et inline-block..... | 23 |
| 3.8.3 | Les grilles (grid)..... | 25 |
| 3.8.4 | Fixed layout versus Fluid layout..... | 26 |
| 3.8.5 | Les boîtes flexibles (flexbox)..... | 27 |
| 3.9 | Transformations et animations..... | 28 |
| 3.9.1 | Transformations..... | 28 |
| 3.9.2 | Transitions..... | 29 |



| | | |
|-------|--|----|
| 4 | Les Media Queries..... | 30 |
| 4.1 | syntaxe..... | 30 |
| 4.2 | Fonctionnalités..... | 31 |
| 4.2.1 | Orientation et ratio..... | 31 |
| 4.2.2 | autres propriétés..... | 31 |
| 4.2.3 | sources media queries..... | 31 |
| 5 | Débogage de page..... | 32 |
| 5.1 | Source de la page..... | 32 |
| 5.2 | Inspecteur de code intégré..... | 33 |
| 6 | En résumé..... | 35 |
| 7 | Annexes..... | 36 |
| 7.1 | Les étapes de réalisation d'un site..... | 36 |
| 7.1.1 | Cahier des charges..... | 36 |
| 7.1.2 | Design..... | 36 |
| 7.1.3 | Intégration..... | 36 |
| 7.1.4 | Création d'un modèle..... | 36 |
| 7.1.5 | Mise en ligne..... | 37 |
| 7.2 | Outil de création de grille..... | 37 |



1 INTRODUCTION

C'est en mai 1995 que le CERN développe CSS 1.0.

CSS : ¹ signifie Cascading Style Sheets : ce sont des feuilles de styles en cascade. Pourquoi en cascade ? Car il est possible d'utiliser plusieurs fichiers contenant les définitions de plus en plus précise des éléments de présentation.

CSS permet de décrire la forme que prendra une balise HTML : le style CSS est donc lié aux balises utilisées dans le document.

L'intérêt majeur, est de définir dans une section ou un fichier, la mise en forme des balises, puis de n'utiliser que les balises dans la page web.

Vous allez donc découvrir les possibilités de CSS : c'est l'état de l'art car plus personne ne code la mise en forme avec HTML.

Note : le CSS reprend la syntaxe du PHP, pour les commentaires, à savoir :

```
/* ceci est commenté */
```

1 CSS : https://fr.wikipedia.org/wiki/Feuilles_de_style_en_cascade

2 FONCTIONNEMENT

L'idée est de séparer la description de la page web (faite en HTML) et la présentation de ces données (en CSS).

2.1.1 Définition CSS

Ainsi, les balises HTML sont associées à un code CSS de mise en forme.

| Fichier HTML | Fichier CSS |
|---|---|
| <code><h1>Titre du paragraphe</h1></code> | <pre>h1 { font-size: 32px; font-family:Arial, Helvetica, sans-serif; color: #090; }</pre> |

Tableau 2.1 : association d'une mise en forme CSS à une balise HTML

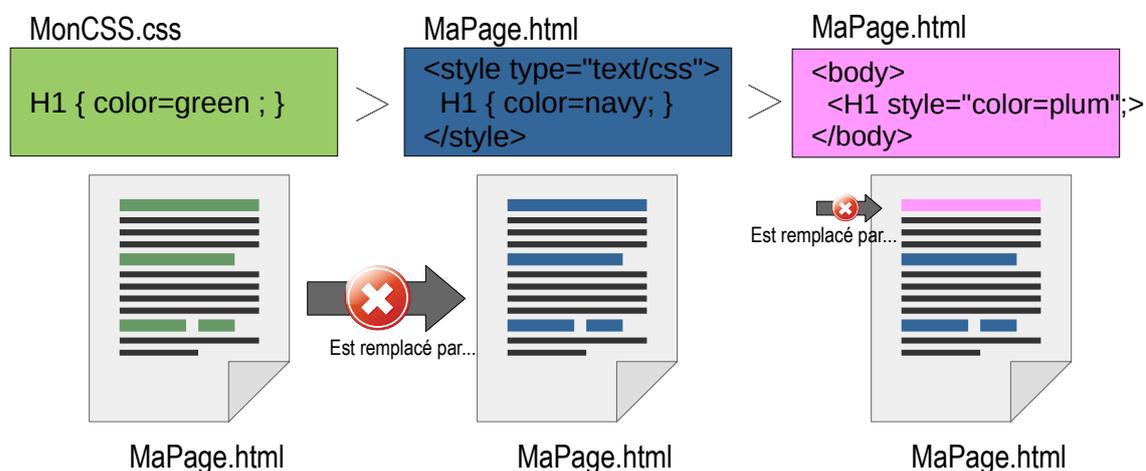
L'intérêt est que le code CSS peut-être placé dans la page web, sera lu par le navigateur mais ne sera pas montré à l'utilisateur.

En revanche, lorsque le navigateur rencontre une balise ayant une description CSS, il affiche le texte avec la mise en forme demandée.

2.1.2 hiérarchie

CSS accepte une certaine forme de hiérarchie, par cascade :

La définition d'un style CSS pour une balise, dans un fichier séparé sera moins **prioritaire** que la définition de cette même balise dans le fichier HTML courant, qui sera elle-même moins prioritaire qu'un style à l'intérieur de la balise.



Ce principe de "cascade" (d'où tire son nom CSS) est lié à la notion de surcharge : il est possible de changer le résultat pour une seule balise au milieu d'un document. Toutes les autres balises utiliseront le style par défaut, sauf la balise surchargée :

| CSS dans une balise |
|---|
| <pre><h1 style="background-image:url(images/background-h1-city.jpg)" > Démonstration de Balise H1 </h1></pre> |

Tableau 2.2 : exemple de surcharge

En rédigeant ainsi, le style ne s'appliquera qu'au texte contenu dans cette balise HTML. Les autres balises <h1> conserveront le style CSS défini ailleurs.

2.1.3 Les sélecteurs

Enfin, CSS est également capable de travailler sur des identifiants de **classe** : cela signifie qu'un objet qui n'a pas une balise associée à un style (H1, H2, H3..., P, BR, etc.) peut quand même recevoir une mise en forme.

On appelle ces éléments les **sélecteurs**.

pour appliquer un style à un sélecteur de type "class", le style sera précédé de **.**,

pour appliquer un style à un sélecteur de type "id", le style sera précédé de **#**.

| HTML | CSS |
|---|--|
| <pre><div>Ceci est un bloc standard</div> <div class="animal">un article sur le chien</div></pre> | <pre>div {background:white; } .animal {background:red; }</pre> |
| <pre><div>Ceci est un bloc standard</div> <div id="sport">un article de sport</div></pre> | <pre>div {background:white; } #sport {background:red; }</pre> |

Tableau 2.3 : exemple de sélecteur

La balise <div> crée un bloc : tout ce qui est écrit entre <div> et </div> sera contenu dans au même endroit sur la page web.

Les pages suivantes contiennent donc les informations nécessaires pour éditer du code CSS et pouvoir créer des sites web évolutifs (changement rapide de charte graphique), propres et conformes.

La balise <div> sera décrite plus en détails dans la suite du cours.

3 MISE EN FORME CSS

Le chapitre Erreur : source de la référence non trouvée, Erreur : source de la référence non trouvée, permet déjà de comprendre combien il est difficile d'écrire une page web et en même temps, son contenu !

Le texte sera émaillé de nombreuses références HTML que le lecteur ne verra pas mais que le rédacteur doit inclure pour formater son texte.

Alors pour éviter de compliquer les choses, il faut – dès que possible – utiliser le langage CSS, et de préférence, dans un fichier séparé.

3.1 INCLURE DU CSS

Il y a en effet deux possibilités : mettre le code CSS dans le fichier HTML, ou bien mettre le code CSS dans un fichier séparé.

Pour les exemples simples, la première solution est idéale, car le codeur n'a qu'un fichier à modifier.

Pour les sites comportant plusieurs pages HTML, il devient intéressant de n'utiliser qu'un seul fichier CSS, pour mettre en forme toutes les pages avec le même style.

| Dans la page HTML | Dans un fichier séparé |
|--|---|
| <pre><!DOCTYPE html> <html> <head> <title>Le "Hello World"</title> <style> body { background:white; color:blue; } </style> </head> <body> <p>Hello world!</p> </body> </html></pre> | <pre><!DOCTYPE html> <html> <head> <title>Le fameux "Hello World "</title> <link rel="stylesheet" href="/css/main.css"> </head> <body> <p>Hello world!</p> </body> </html></pre> |
| | <pre>body { background:white; color:blue; }</pre> |

Tableau 3.1 : les deux possibilités d'inclure du CSS

Il est évident qu'il vaut mieux prendre rapidement le réflexe de créer ses fichiers CSS séparés.

Cependant, l'emploi des balises <style> reste possible est sera prioritaire sur l'emploi de <link>, ce qui signifie que le fichier CSS s'applique en premier mais que les styles seront modifiés par ceux contenus dans les balises <style>.

3.2 STANDARDISER LE CSS

Oui, cela peut paraître bête, mais même en 2018, tous les navigateurs ne comprennent pas toutes les instructions de la même manière.

C'est même la raison d'être du site <http://nicolasgallagher.com/about-normalize-css/>

Ce site propose un fichier CSS à télécharger qui ne définit pas de couleurs ou d'effets mais qui harmonise les présentations des différents navigateurs.

Comme il est possible de faire référence à plusieurs fichiers CSS, votre code HTML devra contenir la ligne suivante :

```
<link rel="stylesheet" href="normalize.css">
```

Il faut noter que vous devez tenir compte du chemin d'accès au fichier :

- Le mode absolu nécessite de commencer par / et de décrire tous les répertoires contenant le fichier (par exemple /css/normalize.css)
- Le mode relatif qui implique de parcourir les dossiers à partir du dossier de la page courante.

3.3 LES COULEURS ET ARRIÈRE-PLANS

3.3.1 Couleurs

La première approche la plus simple et logique, concerne le changement de couleur sur une page web :

- color : #FF0000 ;
- background-color : black ;

Comme il s'agit d'une propriété qui s'applique à une balise HTML (le sélecteur), le code CSS (pour le titre type h1) sera

```
h1 {  
    color : #7FFF33 ;  
    background-color:#303030 ;  
}
```

Pour modifier le fond de toute la page web, il suffira d'appliquer les attributs à la balise HTML <body>.

```
Body { background-color:dimgray ; }
```

Pour les couleurs, il est possible d'utiliser le code hexadécimal, la fonction rgb(vRouge, vVert, vBleu) ou encore le code de la couleur (exemple : gray, red, black). Voir : <http://stylecss.free.fr/couleurs.php>

3.3.2 Images

CSS permet également la mise en œuvre d'images. Ainsi, il applique un fond graphique plutôt qu'une couleur, par l'utilisation de la propriété `background-image` :

```
background-image : url("image.png") ;
```

N'oubliez jamais que l'image peut ne pas s'afficher, il convient donc de choisir un fond de couleur compatible avec le texte. Si votre image représente un ciel noir étoilé, la couleur de fond devrait être noire, afin que l'écriture reste visible en cas de problème pendant le chargement de l'image.

```
Background-color : #000000;  
background-image : url("image.png") ;
```

Si l'image n'est pas assez grande, ou bien si l'image ne doit pas scroller avec la page, il existe d'autres attributs de contrôle :

- **background-attachment** : **scroll** permet à l'image de suivre le mouvement de la page
- **background-attachment** : **fixed** bloque l'image (le texte défile au-dessus).
- **Background-repeat** : permet de répéter l'image...
 - horizontalement : **repeat-x**
 - verticalement : **repeat-y**
 - sur tous les axes : **repeat**
 - jamais de répétition : **no-repeat**

Exemple de code :

```
body {  
  background-color : dimgray ;  
  background-image : url("../fondsombre.gif") ;  
  background-repeat : no-repeat ;  
  background-attachment : fixed ;  
}
```

il existe notamment un raccourci pour toutes les propriétés `background` qui permet de réduire les 4 lignes précédentes à :

```
background : dimgray url("../fondsombre.gif") no-repeat fixed ;
```

Pour modifier la taille de l'image, on utilise `background-size : XXXpx YYYpx;` et pour couvrir le fond de l'écran automatiquement, on utilise `background-size : cover;` (mais pas les deux en même temps !).

```
background-size : 1024px 768px ;  
background-size : cover ;
```

3.4 LES POLICES ET LE TEXTE

Le style d'un site web est fortement influencé par les polices de caractères utilisées. La plupart du temps, 2 ou 3 polices maximum sont nécessaires : une police principale et une ou deux pour afficher du code ou des définitions.

3.4.1 Familles de polices

Le choix d'une police doit cependant obéir à des critères précis. En effet, il existe de nombreuses polices mais seulement 3 familles majeures :

| Famille | Explication | Exemple |
|------------|---|---|
| serif | l'empatement de ces polices est important. <i>Sont utilisées dans les livres et l'empatement est censé améliorer la lecture. Aspect plus classique.</i> | Times new roman Garamond Georgia |
| sans serif | Ces polices n'ont pas d'empatement <i>Sont plutôt utilisées sur les sites web et en informatique. Aspect plus moderne.</i> | Arial Verdana Calibri |
| monospaced | Les lettres de ces polices occupent le même espaces, quelle que soit leur largeur visible. Peut-être avec ou sans serif ! <i>Sont utilisées pour les lignes de code des programmes car, elles gardent les alignements.</i> | Courier courier new consolas Deja vu sans mono |

Le choix d'une police doit se faire en se rappelant que l'affichage se fait sur l'ordinateur du lecteur : si celui-ci n'a pas la police choisie, votre site s'affichera différemment.

Voici comment choisir une famille de police pour un titre :

```
h1 { font-family: arial, verdana, sans-serif ; }
```

Cette ligne indique au navigateur que les titres <H1> doivent utiliser Arial, sinon Verdana et enfin, si aucune police n'est disponible, la police sans serif par défaut.

Si la police contient des espaces (comme Time new roman) il faut utiliser des guillemets.

Pour définir le style <H2> à être en serif, italique, voici le code CSS :

```
h2 {font-family: "Times New Roman", serif; font-style: italic; }
```

La propriété **font-style** n'a que 3 valeurs possibles : **normal**, **italic** et **oblic**.

Pour obtenir un texte en gras, il faut utiliser la propriété **font-weight** avec les valeurs **normal** ou **bold**.

3.4.2 Taille de police

Enfin, il est possible de changer la taille d'une police avec **font-size**...

```
h1 {font-size: 30px;}
h2 {font-size: 12pt;}
h3 {font-size: 120%;}
h4 {font-size: 10vw;}
p {font-size: 1em;}
```

Ce code nécessite quelques explications :

- **px** (pixel) et **pt** (point) sont des valeurs de taille absolue. Le pixel est l'unité sur les écrans alors que le point est l'unité des imprimantes. Si la maîtrise de l'affichage sur un ordinateur fixe reste un objectif intéressant, l'adaptation sur les terminaux mobiles sera plus difficile.
- **%**, **vw** et **em** (et **rem** pour 'root em') sont des unités relatives. 100 % signifie généralement 12pt mais dépend des paramètres du navigateur. Pour une personne visuellement déficiente, la taille par défaut sera plus élevée et 100 % signifiera alors peut-être 18pt ! Em signifie "ième", dont 1em = 12pt, 2em = 24pt et .5em sera 6pt. Enfin, vw est une unité en centième de la largeur de la page.

Pour une meilleure compatibilité, il faut donc privilégier les unités relatives (% ou em sont les plus reconnues).

Vous trouverez des explications et des exemples plus précis sur <https://kyleschaeffer.com/development/css-font-size-em-vs-px-vs-pt-vs/>

Il est possible de simplifier l'écriture des propriétés sur les polices (font) en n'utilisant qu'une seule ligne pour plusieurs propriétés :

```
h1 { font: italic bold 30px arial, sans-serif; }
```

3.4.3 Alignement de texte

La propriété **text-align** correspond au choix d'aligner le texte à gauche (**left**), à droite (**right**), au centre (**center**) ou de justifier (**justify**).

```
h1 { text-align: center; }
```

La propriété **vertical-align** permet de positionner un texte verticalement dans un tableau (et des cellules), avec les valeurs **middle**, **top**, **bottom** pour respectivement le milieu, le haut ou le bas de la cellule.

```
td { vertical-align: middle; }
```

plus d'informations sur <http://stylecss.free.fr/vertical-align.php>

3.4.4 Utiliser des polices spécifiques

Il existe une balise spécifique qui permet de charger une police ou un ensemble de polices dans une page web.

```
@font-face
```

Dans le code CSS, il faut intégrer la police et éventuellement, ses variations (comme light, bold, italic, regular, etc.)

```
@font-face {  
    font-family: "Police personnelle";  
    src: url('MaPolice.ttf');  
}  
@font-face {  
    font-family: "Police personnelle";  
    font-style: bold;  
    src: url('MaPolice-Bold.ttf');  
}  
@font-face {  
    font-family: "Police personnelle";  
    font-weight: italic;  
    src: url('MaPolice-Italic.ttf');  
}
```

il faut ensuite définir quels styles utiliseront cette police :

```
{  
    font-family: "Police personnelle", Arial, sans-serif;  
}
```

Il ne faut pas oublier de définir un style de police par défaut (si la police choisie ne se charge pas).

A noter : la police ci-dessus n'accepte que les attributs bold et italic.

Vous pouvez découvrir un article plus détaillé sur les polices dans les pages web :

<https://www.alsacreation.com/astuce/lire/630-fonte-personnalisee-site-web.html>

3.5 LES EFFETS DE SURVOL

Les effets utilisent des pseudo-classes, dont les plus connues sont liées aux liens.

3.5.1 Effets sur les liens

Les liens sont définis avec les balises HTML `<a>` et ``. Voici les pseudo-classes associées :

- **link** pour l'état d'un lien non visité
- **visited** pour un lien visité
- **hover** pour un lien survolé
- **active** pour un lien sélectionné (clic actif)

Le code CSS ressemblera à la section suivante :

```
a:link { color: orange; text-decoration:none; }
a:hover { background-color: yellow; }
a:visited { color: pink; }
```

Les effets sont très pratiques pour rendre l'affichage d'un lien ou d'une information dynamique. Il faut cependant que ce soit utilisé à bon escient : si c'est animé au survol de la souris, le lecteur s'attend à une action possible avec un clic de souris !

3.5.2 Effets pour un texte

Dans cette catégorie, il devient possible d'afficher une "aide contextuelle" ou un "tooltip" lors du survol d'un texte ou d'une image. Par exemple, Vous pouvez appliquer cet effet sur les textes en italique. Dans le fichier CSS :

```
i.tooltip span {display:none; padding:2px 3px; margin-left:10px; width:150px;}
i.tooltip:hover span{display:inline; position:absolute; border:1px solid #cccccc;
background:#ffffff; color: #000;}
```

et dans le code HTML :

```
les <i class="tooltip">balises <span>instructions dans HTML</span></i> sont lues par
le navigateur<br/>
```

il existe même des générateurs de "tooltip" pour des effets plus puissants.

<http://www.menucool.com/tooltip/css-tooltip>

<http://www.cssportal.com/css-tooltip-generator/>

3.6 LES GROUPEMENTS D'ÉLÉMENTS

3.6.1 Groupements logiques : "Class" et "Id"

Il est possible d'appliquer un style à un ensemble de balises ou à une seule et unique balise :

- **class** : le style s'appliquera à toutes les balises comportant cette classe
- **id** : le style s'appliquera à l'élément unique

Les sélecteurs "class" et "id" sont donc à intégrer dans la balise HTML de l'élément dont on veut définir le style.

```
<div id="intro">
  <h1>Revue du sport</h1>
  <p class="football">Les joueurs de l'Olympique Breignolles ont gagné</p>
  <p class="tennis">Wilfrid Williams aurait déclaré forfait lors du tournoi
  Rolland Wimbledon</p>
</div>
```

Pour définir le style dans la feuille CSS, il faut

- précéder d'un point le nom de la classe : `.nom`
- précéder d'un dièse le nom de l'ID : `#nom` (*astuce en inversant 'ID' cela donne le début de 'Dièse'*)

Ce qui correspond au code suivant :

```
#intro { background-color: DarkSlateBlue; color: white; padding: 20px;}
.football { background-color: red; color : white; }
.tennis { background-color: DarkGreen; color: white; }
```

On peut donc associer un style selon une catégorie d'événement ou pour créer des sections.

3.6.2 Groupements physiques : div et span

Ces deux balises n'affichent rien dans une page HTML... elles sont neutres tant que l'on ne leur associe pas du code CSS. Ce sont des structures HTML (%block et %inline) : elles sont délimitées par un cadre invisible et peuvent contenir d'autres structures. Ne pas confondre %block et l'attribut 'block' de CSS.

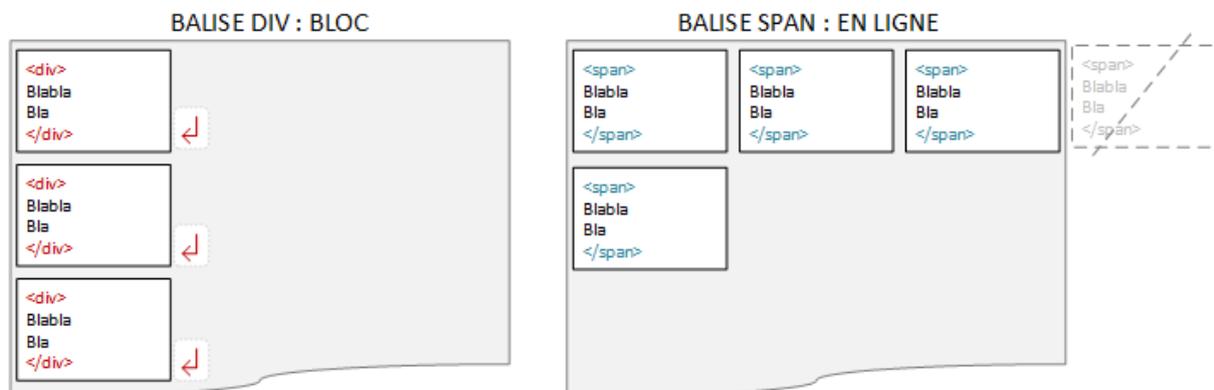
3.6.2.1 <DIV> structure conteneur de bloc (%block)

La balise HTML <div> et </div> divise un document en section. Chaque section peut ainsi avoir son propre style et contenir d'autres blocs. Par défaut, après chaque <div></div> le navigateur passe à la ligne.

3.6.2.2 structure en ligne (%inline)

La balise HTML et ne passe pas à la ligne : l'intérêt réside dans la possibilité de l'intégrer partout : par exemple, au milieu d'un texte ou plusieurs sections sur une même ligne.

Voici le comportement des deux balises, sur une page web d'une largeur suffisante pour contenir plusieurs sections :

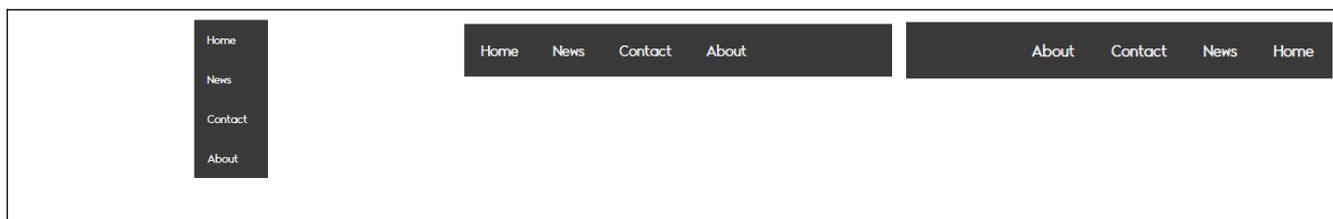


En associant une classe ou un identifiant (ID) à un groupement DIV ou SPAN, il devient possible de lui affecter des propriétés propres :

- Pour le groupement DIV, on définira souvent sa taille et sa position
- Pour le groupement SPAN, il s'intègre facilement à la suite d'un texte, permettant de modifier cette partie sans modifier le reste du texte. Par exemple, mettre en rouge et italique les mots difficiles d'un texte.

3.6.2.3 Le modificateur CSS 'float'

Son usage est déprécié et est remplacé par le modificateur 'display'. Les valeurs possibles sont 'left', 'right', 'none', 'initial' et 'inherit'. Voici un exemple de menu avec none, left, puis right :



3.6.2.4 Le modificateur CSS 'display'

La balise CSS 'display'² vient modifier ce fonctionnement³ :

- display:block ► l'élément se comporte comme un bloc (dimensionnement possible)
- display:inline ► l'élément se comporte comme une ligne
- display:inline-block ► l'élément est un bloc (dimensionnement possible) qui se comporte comme sur une ligne
- display:grid ► l'élément est un bloc (pré dimensionné au format de la grille) au sein d'une grille
- display:flex ► l'élément bloc contient d'éléments flexibles

2 https://www.w3schools.com/CSSref/pr_class_display.asp

3 <https://www.alsacreation.com/tuto/lire/530-structure-balises-css-display-bloc-block-ligne-inline.html>

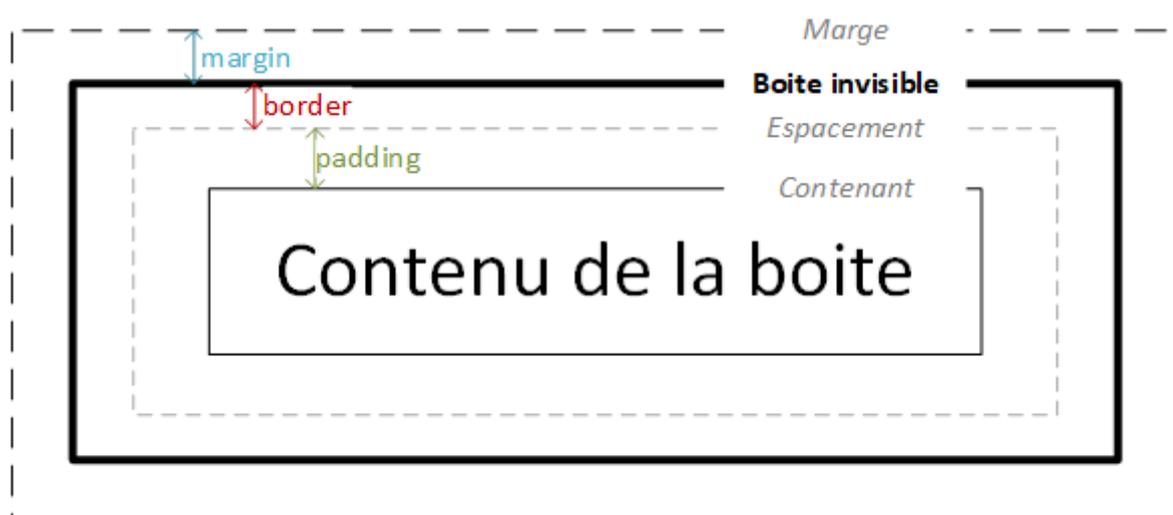
3.7 LE MODÈLE DE BOITE

Jusqu'à maintenant, nous avons considéré que les balises HTML étaient modifiables en style de couleurs, polices, alignement, etc.

La puissance de CSS est en réalité de considérer que chaque élément est contenu dans une boîte invisible par défaut : de nouvelles propriétés vont permettre de modifier ces boîtes, pour transformer n'importe quelle balise en objet "graphique" rudimentaire.

3.7.1 Marges et espaces

Voici une boîte et les différents espaces autour de celle-ci :



Les propriétés `margin`, `border`, `padding` définissent les espaces entre les éléments.

Prenons l'exemple de la balise `<body>` qui détermine comment s'afficheront les éléments de la page web dans l'espace du navigateur : il est extrêmement rare que le texte commence contre la bordure du navigateur.

3.7.1.1 Marges

Voici un code CSS pour définir les marges :

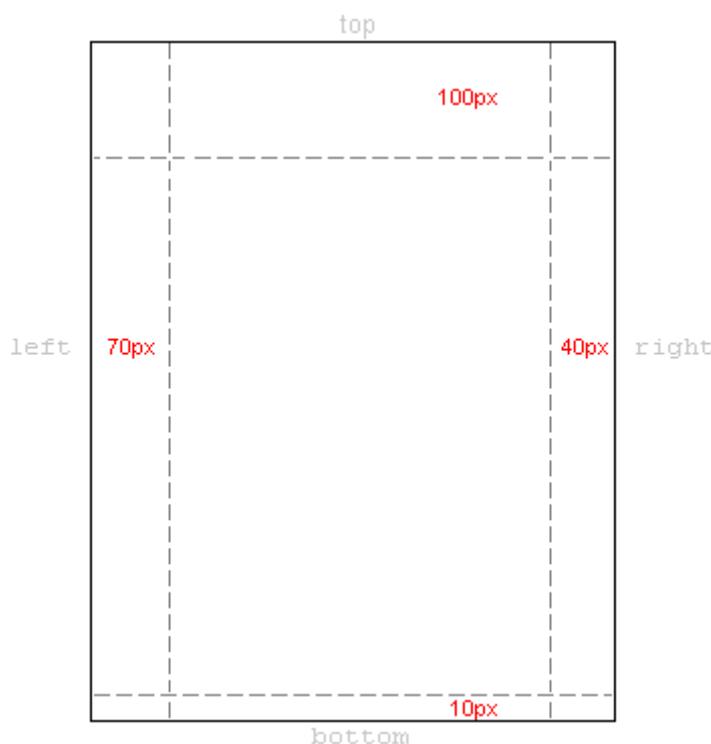
```
body {  
  margin-top: 100px;  
  margin-right: 40px;  
  margin-bottom: 10px;  
  margin-left: 70px;  
}
```

ou bien, en forme concise :

```
body { margin: 100px 40px 10px 70px; }
```

(on tourne dans le sens horaire, à partir de midi)

Cela donne la mise en page suivante :



Désormais, les éléments présents entre les balises `<body>` et `</body>` s'afficheront à 70 pixels du bord gauche... mais ces éléments ont leurs propres marges :

```
p { margin: 5px 10px 5px 20px; }
```

Si la marge gauche du paragraphe n'est que de 10 pixels, la marge totale entre le bord du navigateur et le paragraphe sera de 80 pixels (10 pixels + 70 pixels).

3.7.1.2 Espaces

Désormais, nous pouvons décider de l'espace entre la bordure et le contenu : c'est le padding. Le padding élargit la hauteur et/ou la largeur de la boîte "autour" du contenu.

Prenons une balise de titre H1 :

```
h1 {
  background: gray;
  padding: 5px 20px 5px 80px;
}
```

Le titre s'affichera à 70 pixels du bord du navigateur... mais le texte sera à 150 pixels du bord !

En effet, la boîte qui était invisible est remplie en gris (propriété `background`) et cette dernière commence bien à la marge de la balise `<body>`. C'est le padding qui décale le contenu.

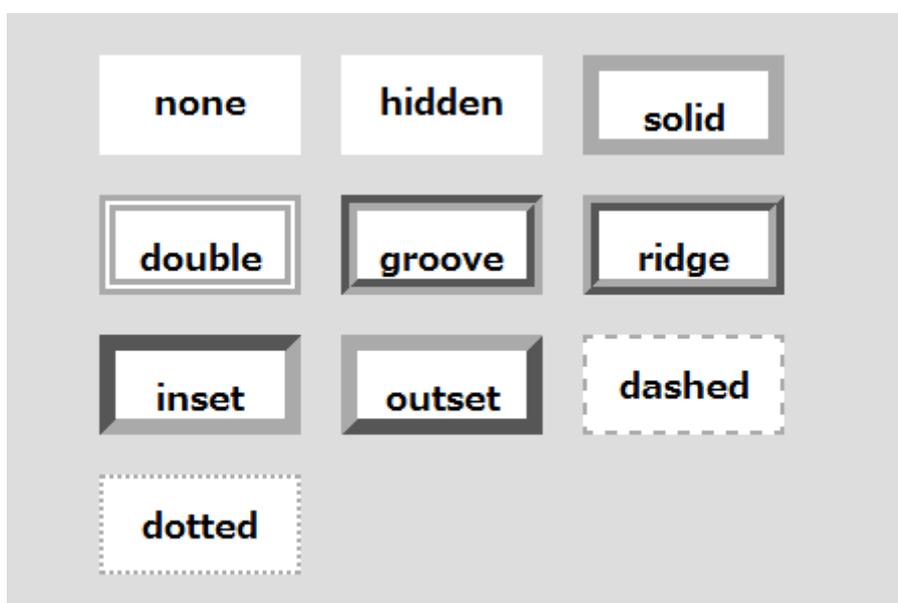
3.7.1.3 Bordures

Il ne faut pas oublier l'épaisseur de la bordure : actuellement nulle, la modification de cet attribut décale d'autant le contenu de la boîte.

```
h1 { border-width : 2px ; border-color: black; }
```

Avec cet exemple, le contenu utilise le padding par défaut mais sera quand même décalé de 2 pixels par l'épaisseur de la bordure.

Concernant les bordures, il est également possible de choisir parmi plusieurs styles avec la propriété **border-style** : none, dotted, dashed, solid, double, groove, ridge, inset, outset, hidden.



et également définir une taille d'arrondi (en pixel) : **border-radius**

voici un exemple :

```
h1 { border-width : 2px ; border-style: dotted; border-radius: 20px; }
```

3.8 LES DIVISIONS ET LA MISE EN PAGE

Finalement, nous pouvons désormais utiliser toute la puissance des divisions (balises <div>), voyons comment.

3.8.1 Structure classique d'une page web

Si les premières pages ne contenaient pas de menus ou d'entêtes, c'est rapidement devenu nécessaire ! Les sites de plus en plus complexes ont eu besoin de se structurer physiquement (les dossiers et fichiers) mais aussi logiquement (menus).

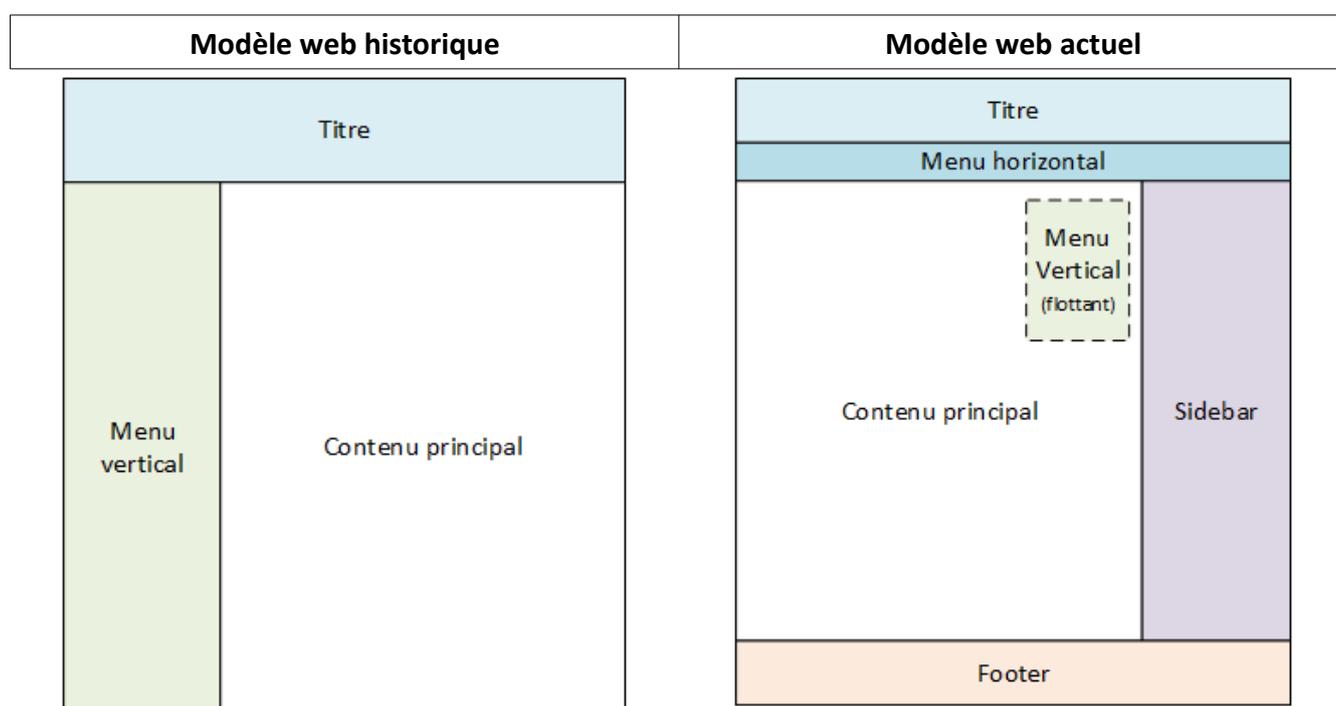


Tableau 3.2 : différence entre les modèles de page en 2000 et actuellement

3.8.1.1 Tableau : obsolète

Les premiers sites ont géré l'affichage avec des tables. L'aspect dynamique reste mineur, il s'agit en fait de créer des pages identiques, dont seule la cellule contenant l'article change.

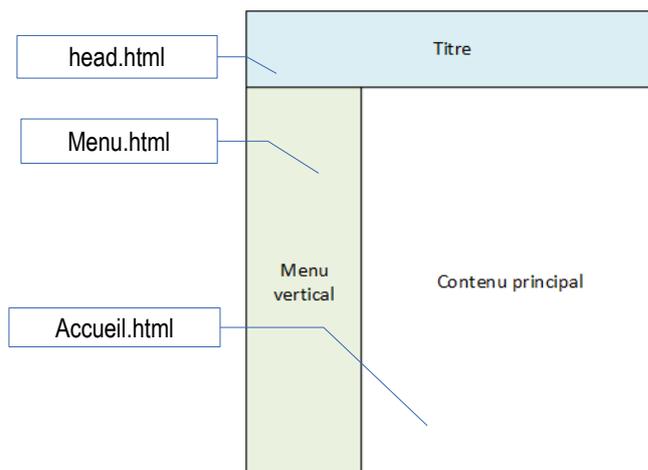
Modifier le style demandait de modifier chaque page (avec CSS, ce serait maintenant plus simple) mais surtout, **le navigateur charge tout le tableau (donc toute la page) à chaque fois !** Sur une connexion à faible débit, cela rend la navigation poussive.

3.8.1.2 Frameset : obsolète

Ce modèle semble très pratique : la page index.html ne contient que le code de traçage des parties de la page et le lien vers les fichiers HTML correspondant :

L'intérêt est que chaque partie est totalement indépendante ! Chaque partie peut avoir sa propre barre de scrolling, et un clic sur un lien dans le menu (avec la propriété 'target') permet d'afficher une page dans la partie souhaitée.

Malheureusement, le revers de la médaille est le référencement : **les moteurs ne trouvent pas forcément les liens internes du site, imprimer le contenu principal n'est pas possible directement** (il faut faire un clic droit sur la partie concernée), etc.



Ce modèle est obsolète et vous devrez le remplacer si vous travaillez à la remise à jour d'un site.

A noter : il existe une nouvelle variante `<iframe>`⁴ dans HTML5 pour faciliter la migration vers les bonnes pratiques !

3.8.1.3 Division : la référence

Les balises divisions `<div>` ou `` ne sont, en HTML, que des balisages sans effet. Cependant, CSS permet d'y associer le modèle de boîte pour diviser et placer ces boîtes.

Les propriétés des boîtes permettent de créer :

- des boîtes de tailles fixes
- des boîtes de tailles variables
- des boîtes en positions fixes
- des boîtes en positions variables

Mieux : l'arrivée de la version 5 de HTML a vu l'ajout de divisions particulières : `<section>`, `<article>`, `<nav>`, `<aside>`, `<header>` et `<footer>` qui remplacent certains styles fréquemment employés.

⁴ <http://www.peachpit.com/blogs/blog.aspx?uk=Frames-are-Dead-Long-Live-Iframes>

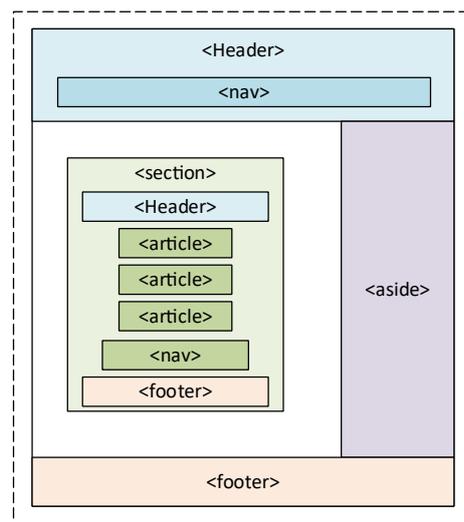
3.8.2 Structure moderne d'une page web

La définition d'éléments flottants ou fixes, à tailles variables ou fixes permet de créer un site adaptable, flexible, souple. Les nouvelles divisions reprises dans HTML5 apporte un confort de mise en forme. Un site ainsi créé sera lisible sur de nombreux navigateurs et s'adaptera à la fenêtre de l'utilisateur.

Ce n'est pas complètement nouveau, mais c'est désormais la norme de fait pour HTML 5 permet de :

- remplacer `<div id="header">` par `<header>`
- remplacer `<div id="menu">` par `<nav>`
- remplacer `<div id="footer">` par `<footer>`
- etc.

De plus, cette norme formalise également certains éléments qui peuvent être utilisés à plusieurs endroits (header et footer peuvent apparaître dans la page comme dans une section).



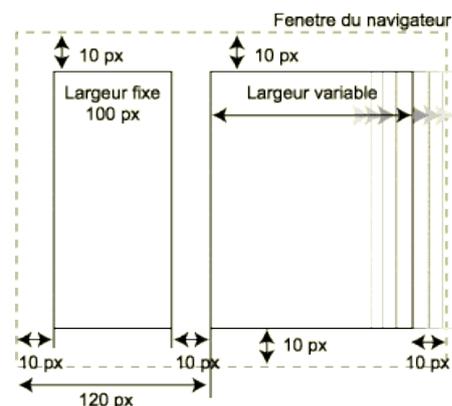
3.8.2.1 Création de blocs

Vous pouvez créer deux blocs côte à côte, avec deux fonctionnements différents :

Le premier bloc ayant une taille et une position fixe ne bougera jamais sur la page.

Le deuxième bloc gardera également sa position mais adaptera son contenu en fonction de la largeur de la fenêtre du navigateur.

Attention : en utilisant des positions fixes, les blocs peuvent se chevaucher si les coordonnées de position sont mal calculées.



La définition d'un bloc utilise les propriétés suivantes sur une balise `<div>` :

- **border** (-top, -left, -bottom, -right), border-style, border-width
- **position** (fixed, float, absolute), top, left, right, bottom
- **width, height**
- **overflow** (auto, scroll, visible, hidden)
- **z-index** (NDLR : le niveau d'affichage, comme un calque. Permet de placer une boîte au-dessus ou au-dessous d'autres boîtes)

Voici un exemple (correspondant à l'image précédente).

| Partie CSS | Partie HTML |
|--|---|
| <pre>.bloc-fixe { position:fixed; border:solid 1px; width:100px; top:10px; left:10px; bottom:10px; } .bloc-variable { position:fixed; border:solid 1px; top:10px; left:120px; bottom:10px; right:10px; }</pre> | <pre><!doctype html> <html lang="fr"> <head> <meta charset="UTF-8"> <title>Document</title> <link rel="stylesheet" type="text/css" href="style.css"> </head> <body> <div class="bloc-fixe"> Texte du bloc fixe. </div> <div class="bloc-variable"> Texte du bloc variable. La largeur est variable, le texte s'adapte. </div> </body> </html></pre> |

Tableau 3.3 : différence de bloc (largeur variable)

Voici deux résultats : Le premier bloc est à 10 pixels du bord et a une largeur de 100 pixels. Le deuxième bloc est placé à 120 pixels du bord, mais sa largeur n'est pas spécifiée. Par défaut un bloc occupe alors toute la largeur restante.

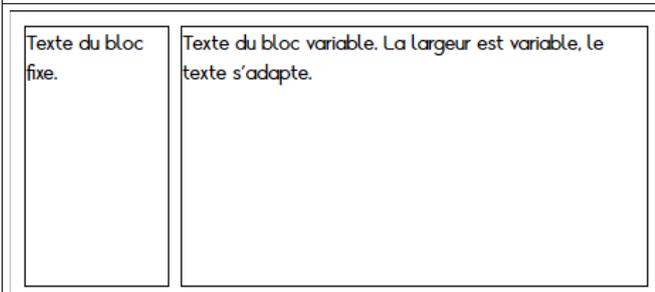
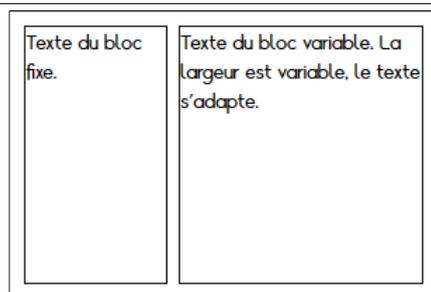
| large | étroit |
|---|--|
|  |  |

Tableau 3.4 : variation de la largeur des blocs

Pour créer une sorte de grille de boîtes, la position fixe devient complexe, car il faut calculer toutes les largeurs et ajouter tous les espaces entre les boîtes... mais en plus, ce site présenterait l'inconvénient de ne pas s'adapter à la largeur de la fenêtre !

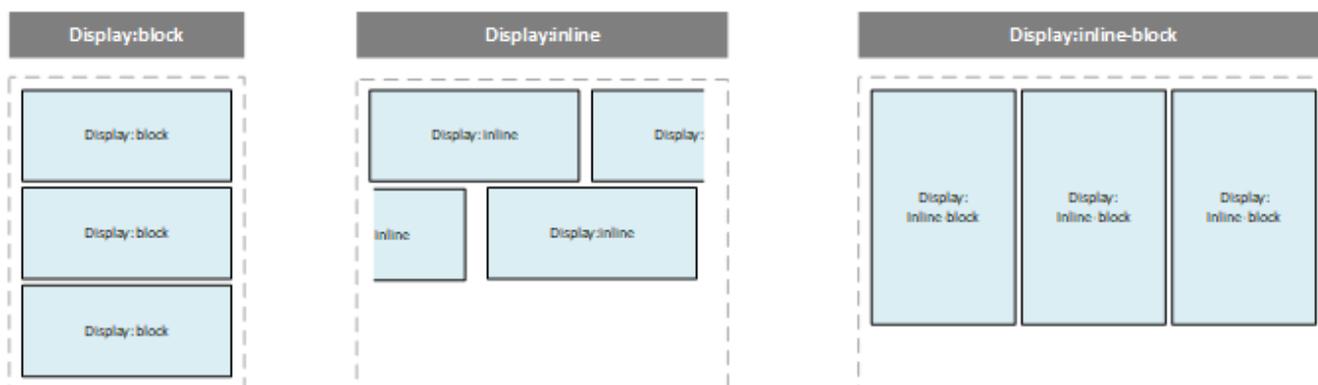
La solution actuellement dépréciée serait d'utiliser les positions de type "float". Désormais, il est préférable d'utiliser les propriétés de "display" : block, inline, et inline-block.

3.8.2.2 Block, inline et inline-block

Les blocs et inlines sont une représentation des balises utilisées en HTML.

Les notions sont simples :

- Les **blocs** sont des éléments que le navigateur affiche les uns après les autres. La création de blocs fixes ou variables permet de définir la mise en page du site.
- Les **inlines** sont des éléments que le navigateur affiche sans passer à la ligne. Ils ne cassent pas la continuité du texte.
- Les **inline-blocks** sont des éléments qui se comportent comme des inlines (au milieu d'un texte) mais la hauteur est celle d'un bloc.



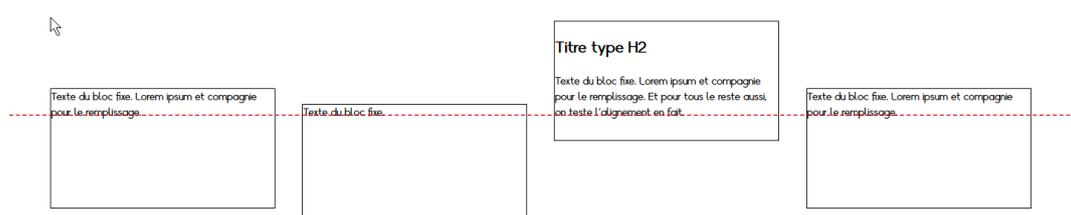
Attention, les éléments "block" et "inline" ne portent pas sur les mêmes balises :

| Block / inline-block | inline |
|--|--|
| <p>, <div>, <form>, <header>, <nav>, , , <h1>... | <a>, , , <i>, , <cite>, <mark>, <code>, |

Tableau 3.5 : les balises propres au type bloc et type enligne

L'intérêt des inline-block, est de permettre l'adaptation du site aux différents écrans : en effet, si la largeur est trop faible pour afficher tous les inline-block côte-à-côte, ils passeront à la ligne suivante, comme les mots dans une boîte.

En revanche, il faut conserver l'alignement des blocs les uns par rapport aux autres en ajoutant une propriété "**vertical-align : top**", sinon l'affichage tiendra compte du texte dans les blocs et ne sera pas esthétique, car le navigateur alignement chaque fois la dernière ligne de texte.



Voici un exemple (correspondant à l'image précédente).

| Partie CSS | Partie HTML |
|---|--|
| <pre>.bloc-fixe { display: inline-block; border:solid 1px; width:300px; height:160px; margin:1em; vertical-align:top; }</pre> | <pre><!doctype html> <html lang="fr"> <head> <meta charset="UTF-8"> <title>Document</title> <link rel="stylesheet" type="text/css" href="new_inline-block_style.css"> </head> <body> <div class="bloc-fixe"> Texte du bloc fixe. Lorem ipsum et compagnie. </div> <div class="bloc-fixe"> Texte du bloc fixe. Lorem ipsum et compagnie. </div> <div class="bloc-fixe"> Texte du bloc fixe. Lorem ipsum et compagnie. </div> <div class="bloc-fixe"> Texte du bloc fixe. Lorem ipsum et compagnie. </div> <div class="bloc-fixe"> Texte du bloc fixe. Lorem ipsum et compagnie. </div> <div class="bloc-fixe"> Texte du bloc fixe. Lorem ipsum et compagnie. </div> <p>Le paragraphe commence après les blocs</p> </body> </html></pre> |

Tableau 3.6 : répartition automatique des blocs, avec alignement par le haut

Astuce : si vous rencontrez un site web utilisant la propriété "float", recherchez un autre style contenant la propriété "clear". C'est le seul moyen de mettre fin à l'enchaînement des blocs.

Si vous devez utiliser la propriété 'float', voici un exemple avec float :

```
.box {
float: left;
width: 200px;
height: 100px;
margin: 1em;
}
.after-box {
clear: left;
}
```

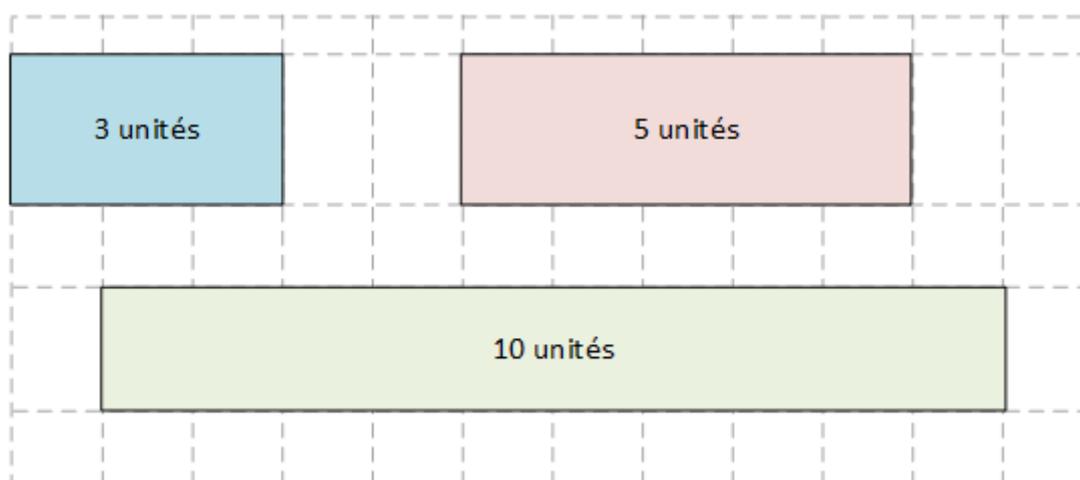
3.8.3 Les grilles (grid)

Au printemps 2017, cette spécification majeure devient disponible dans la plupart des navigateurs.

Les grilles permettent de découper la largeur de la page en plusieurs rectangles de largeurs identiques. On utilise fréquemment une grille de 12 colonnes car c'est le nombre qui accepte le plus grand nombre de diviseurs : 1, 2, 3, 4, 6.

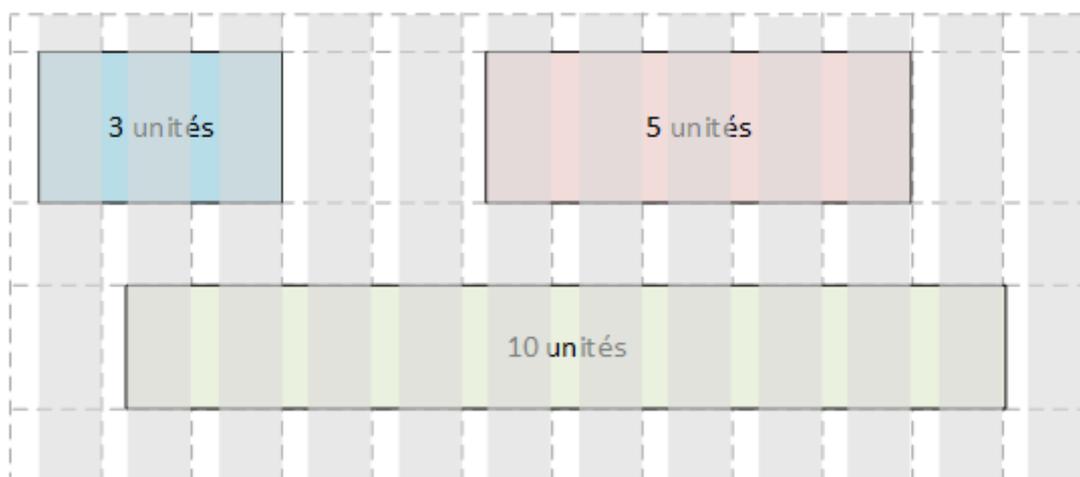
Ainsi, une boîte pourra prendre une largeur de 3 unités et il restera 9 unités pour d'autres éléments : c'est plus simple que de donner des coordonnées et des tailles en pixels et cela présente l'avantage d'être "arrangeable" selon le périphérique d'affichage (PC, tablette ou smartphone).

Il faut toutefois se rappeler qu'en HTML, la hauteur n'est pas aussi facile à gérer. Cela signifie que les hauteurs entre les lignes ne seront pas constantes mais dépendront du contenu des boîtes.



Le fonctionnement de certains frameworks comme Bootstrap sont basés sur ce mode de fonctionnement.

Il faut faire attention, car la plupart des modèles de grilles prévoient un espacement entre les blocs (marge). C'est donc plutôt un modèle similaire à celui-ci qui est utilisé :

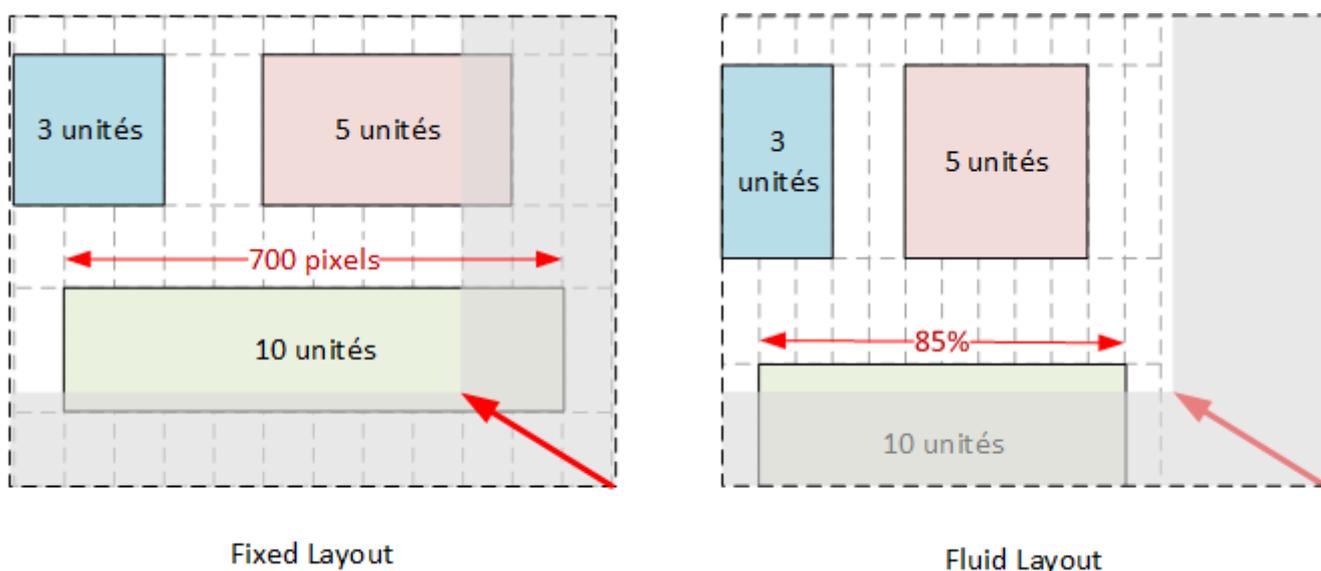


3.8.4 Fixed layout versus Fluid layout

La principale difficulté d'un site qui doit s'adapter aux différents terminaux, est de ne pas fixer les dimensions en pixels, mais plutôt en unités souples comme le pourcentage.

L'intérêt est de permettre un affichage qui conserve les proportions tout en ayant la capacité de s'adapter à la taille d'affichage de n'importe quel terminal.

L'image ci-dessous donne une idée de ce qui se passe lorsqu'un site utilise une grille fixe (en pixel) ou une grille fluide (en pourcentage).



Il reste évident que pour afficher le même contenu, chaque boîte devra s'allonger vers le bas, en fonction de son contenu.

Cependant, si la page devient trop étroite, il devient intéressant de ne plus réduire les blocs car un bloc ne contenant que quelques lettres par ligne devient illisible. C'est ici que les balises CSS `min-width`, `max-width`, `min-height`, `max-height` peuvent imposer des limites de tailles minimum et maximum.

```
ul {  
  list-style-type: none;  
  margin: 0;  
  padding: 0;  
  width: 50%;  
  max-width: 140px;  
  min-width: 50px;  
  background-color: #f1f1f1;  
}
```

Ici, les blocs du menu s'adapte sur une certaine plage de pixels.

Les captures d'écran faite avec différentes largeurs montrent bien ce fonctionnement

| | | |
|--|--|--|
|  <p><i>Bien que le menu puisse prendre 50 % de la largeur, il est limité à 140 pixels.</i></p> |  <p><i>Le menu prend environ 50 % de la largeur.</i></p> |  <p><i>Le menu prend plus de 50 % pour conserver un aspect lisible (50 px).</i></p> |
|--|--|--|

3.8.5 Les boîtes flexibles (flexbox)

Il s'agit d'une manière plus souple encore de créer des pages adaptables, mais dont l'utilisation est recommandée pour de petites échelles de composants. Pour les pages complexes et à grande échelle, les grilles restent plus adaptées.

Voici un site proposant une démonstration des deux solutions (n'hésitez pas à modifier la taille de la fenêtre).

<https://demo.tutorialzine.com/2017/03/css-grid-vs-flexbox/#flexbox>

Les "flexbox" ont leurs propres propriétés et notamment des

```
display: flex ;
display : flex-inline ;
```

3.9 TRANSFORMATIONS ET ANIMATIONS

Les possibilités de CSS sont nombreuses, mais voici deux paragraphes qui pourront faciliter l'intégration d'articles et d'effets pour rendre votre site plus graphique.

3.9.1 Transformations

CSS propose également quelques effets graphiques de transformations :

translation (translate), rotation (rotate), échelle (scale), perspective/déviation (skew)

Ces transformations s'appliquent sur un bloc. Il faut noter que l'affichage de l'objet transformé pourra sortir de son cadre d'origine.

Pour permettre des effets réussis (et des animations en utilisant la propriété hover), il est possible de déplacer le point d'origine du bloc avec la propriété transform-origin.

Il est possible d'utiliser des coordonnées en pixels mais aussi en pourcentage.

Exemple :

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>Document</title>
    <style>
      .cadre {display: block; border: 1px dotted black; }
      .carreExemple {
        display: inline-block;
        border: 1px solid gray;
        background: orange;
        color:white;
        width:300px;
        height:200px;
      }
      .carreExemple:hover {
        display: inline-block;
        border: 5px solid gray;
        background: orange;
        color:white; width:300px;
        height:200px;
        transform: rotate(-20deg);
      }
    </style>
  </head>
  <body>
    <h1>Les transformations</h1>
    <h2>Points d'origines</h2>
    <div><span class="cadre">hello<br>
      <div class="carreExemple">Boite 1</div>
      <div class="carreExemple" style="transform: rotate(-30deg)">Boite 2</div>
      <div class="carreExemple" style="transform: translateY(30px)">Boite 3</div>
      <div class="carreExemple" style="transform: scale(0.75)">Boite 4</div>
      <div class="carreExemple" style="transform-origin:0 100%">Boite 5 normale</div>
    </span></div>
  </body>
</html>
```

Le code affichera l'image ci-après (n'hésitez pas à passer la souris sur les boites).

Les transformations

Points d'origines



Notez les sorties du cadre d'origine.

Si vous voulez en découvrir plus sur les transformations, suivez ce lien :

<http://www.the-art-of-web.com/css/css-animation/>

3.9.2 Transitions

C'est la dernière propriété de ce chapitre : elle permet d'effectuer les transitions de manière douce. Pour cela, il suffit de rajouter cette propriété au style souhaité.

Dans l'exemple précédent, ajouter la ligne rouge au même endroit que ci-dessous.

```
.carreExemple {
    display: inline-block;
    border: 1px solid gray;
    background: orange;
    color:white;
    width:300px;
    height:200px;
    transition: width 2s, transform 2s;
```

Vous constaterez que si le passage de la souris sur un objet ne change rien, lorsque la souris sort de l'objet, il revient lentement à sa place initiale.

Si les transitions vous intéressent, vous pouvez consulter le lien suivant :

https://developer.mozilla.org/fr/docs/Web/CSS/CSS_Transitions/Utiliser_transitions_CSS

4 LES MEDIA QUERIES

Le nombre de format d'affichage ayant explosé, CSS a ajouté la capacité à gérer le design spécifiquement en fonction de la résolution.

On utilise les media queries qui sont des blocs de styles CSS.

Elles permettent de prendre en compte les écrans (width, height), les projecteurs, téléviseurs ou imprimantes (device-width, device-height), la gestion des couleurs (color, en bit/pixels), etc.

4.1 SYNTAXE

Les media queries sont des expressions conditionnelles qui ne seront exécutées seulement si la dimension d'un périphérique y correspond.

La syntaxe est la suivante :

```
@media all (max-width : 800px) {  
    ...code CSS  
}
```

Le code CSS à l'intérieur sera exécuté si la résolution de l'appareil ne dépasse pas 800 pixels de large.

Le fichier CSS peut donc contenir plusieurs sections qui ne seront pas toutes lues. Par contre, plusieurs sections peuvent être lues.

Il existe aussi une balise media qui permet de faire des choses en fonction des capacités du média :

```
<link rel="stylesheet" media="screen" href="screen.css" type="text/css" />  
<link rel="stylesheet" media="print" href="print.css" type="text/css" />
```

media accepte alors des arguments moins classiques que la résolution mais plus intéressants :

- media="all" ► il s'agit d'une partie de code pour tous les périphériques
- media="screen" ► il s'agit d'un écran
- media="tv" ► il s'agit d'un écran (pouvant avoir un ratio différent d'un écran classique)
- media="handheld" ► il s'agit d'un petit périphérique, probablement vertical
- media="aural" ► il s'agit d'un système de synthèse vocal (CSS 2.1, media="speech")
- media="braille" ► il s'agit d'un périphérique en braille (pour les aveugles)
- media="projection" ► il s'agit d'un projecteur (ou système par page)
- media="tty" ► il s'agit d'un terminal utilisant une police monospace

Depuis CSS3, il est possible d'ajouter des opérateurs logiques (**and**, **only** et **not**).

```
@media screen and (min-width: 200px) and (max-width: 640px) { ... }
```

4.2 FONCTIONNALITÉS

L'intérêt des média-queries est d'obtenir des renseignements sur le mode d'affichage réel.

4.2.1 Orientation et ratio

Par exemple, un téléphone peut être tenu verticalement ou posé horizontalement. Les propriétés orientation et aspect-ratio permettent de déterminer cela :

```
<link rel="stylesheet" media="(orientation:portrait)" href="portrait.css">  
<link rel="stylesheet" media="(orientation:landscape)" href="paysage.css">
```

On peut également déterminer le ratio d'une télévision :

```
@media tv, (device-aspect-ratio: 16/9), (device-aspect-ratio: 4/3)
```

4.2.2 autres propriétés

La plupart des périphériques ont des spécificités qu'il est possible de découvrir pour appliquer le meilleur style. Souvenez-vous que les objets connectés, les écrans de voitures, les panneaux d'affichages en ville peuvent contribuer à l'affichage de page web...

- color, monochrome ► périphérique supportant la couleur ou monochrome
- device-height, device-width ► dimensions en hauteur/largeur d'un périphérique
- grid ► périphérique bitmap ou grille comme un écran LCD
- height, width ► dimension de la hauteur/largeur de la zone d'affichage
- scan ► périphérique à balayage progressif (écran LCD) ou interlacé (TV écran cathodique)

La plupart des propriétés supportent d'être préfixées par **min-** ou **max-** quand elles acceptent des valeurs numériques.

4.2.3 sources media queries

Le cours d'OpenClassroom est très bien fait, je vous recommande de regarder la courte vidéo :

<https://openclassrooms.com/courses/apprenez-a-creer-votre-site-web-avec-html5-et-css3/mise-en-page-adaptative-avec-les-media-queries>

Les informations d'Alsacreation suivantes sont aussi très utiles :

<https://www.alsacreations.com/article/lire/930-css3-media-queries.html>

5 DÉBOGAGE DE PAGE

Une fois la mise en page écrite et les feuilles de styles rédigées, il n'est pas toujours évident de comprendre pourquoi l'affichage ne correspond pas.

Il existe plusieurs méthodes pour vérifier le code, valider le CSS que nous allons voir ici.

5.1 SOURCE DE LA PAGE

Dans une certaine mesure, il est possible d'afficher le code de la page, tel que le navigateur la voit.

Il suffit de faire un clic droit ► Code-source de la page, ou bien d'utiliser la séquence de touches [CTRL]+[SHIFT]+[U].

Le résultat est le suivant :

```

1
2
3 <!DOCTYPE html PUBLIC "-//W3C/DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
4
5
6 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr" dir="ltr">
7 <head>
8 <meta http-equiv="Refresh" content="60">
9 <title>Site de l'académie de Grenoble - Accueil</title>
10
11 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
12
13 <!--<meta http-equiv="refresh" content="60;URL=http://www.ac-grenoble.fr/admin/spip/spip.php"-->
14
15 <link href="squelettes/css/styles.css" rel="stylesheet" type="text/css" />
16 <script language="JavaScript" type="text/JavaScript">
17 <!--
18 function MM_swapImgRestore() { //v3.0
19   var i,x,a=document.MM_sr; for(i=0;a&&i<a.length&&(x=a[i])&&x.oSrc;i++) x.src=x.oSrc;
20 }
21
22 function MM_preloadImages() { //v3.0
23   var d=document; if(d.images){ if(!d.MM_p) d.MM_p=new Array();
24     var i,j=d.MM_p.length,a=MM_preloadImages.arguments; for(i=0; i<a.length; i++)
25       if (a[i].indexOf("#")!=0){ d.MM_p[j]=new Image; d.MM_p[j++].src=a[i];}
26 }
27
28 function MM_findObj(n, d) { //v4.01
29   var p,i,x;  if(!d) d=document; if((p=n.indexOf("?"))>0&&parent.frames.length) {
30     d=parent.frames[n.substring(p+1)].document; n=n.substring(0,p);}
31   if(!(x=d[n])&&d.all) x=d.all[n]; for (i=0;!x&&i<d.forms.length;i++) x=d.forms[i][n];
32   for(i=0;!x&&d.layers&&i<d.layers.length;i++) x=MM_findObj(n,d.layers[i].document);
33   if(!x && d.getElementById) x=d.getElementById(n); return x;
34 }
35

```

Les deux navigateurs fournissent le même résultat, cependant, Firefox est capable de détecter et afficher en rouge lorsqu'un élément est faux ou que l'ensemble des balises ne ferment pas.

Dans l'exemple ci-contre, il s'agit en réalité d'un <div> auquel il manque un / dans les premières lignes.

La présence d'une marque en rouge reste une aide précieuse.

```

58
59
60
61
62
63

```

</div>
 </article>
 </div>
 </body>
 </html>

5.2 INSPECTEUR DE CODE INTÉGRÉ

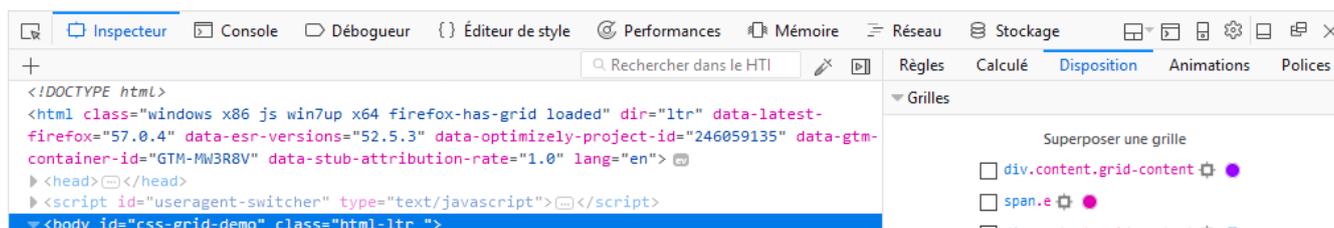
Google Chrome et Mozilla Firefox proposent par défaut des outils d'analyse de page. Il suffit de faire un clic droit dans la page ou bien d'appuyer sur les touches [CTRL]+[Shift]+[I].

| Accéder à l'inspecteur de Chrome | Accéder à l'inspecteur de Firefox |
|----------------------------------|-----------------------------------|
| | |

Tableau 5.1 : accéder aux inspecteurs de codes avec les touches CTRL+SHIFT+I

Le menu de l'inspecteur de code de Mozilla Firefox comporte plusieurs parties :

- La barre de menu
- Le panneau HTML
- Le panneau CSS



La puissance des outils fournis permet aux développeurs d'analyser les pages des sites web qu'ils jugent intéressants, mais aussi de rechercher les causes de ralentissement ou de problèmes d'affichage.

Ci-dessous, la liste des principaux outils et leur utilité.

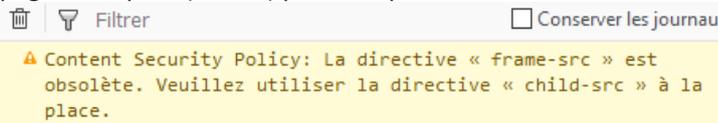
| symbole | explication |
|--|---|
|  | Lorsque vous activez cette fonction, le survol avec la souris, de la page web à analyser affiche la section de code et un cadre bleu sur la zone correspondante. Elle est active/inactive pour l'ensemble des fonctions qui suivent. |
|  Inspecteur | C'est l'outil actif par défaut. Grâce à lui, vous pouvez éditer le code <u>local</u> de la page web pour modifier les champs. Cet outil est utilisé pour certains challenges sur les sites de sécurité. <pre><form> <input type="password"/> </form></pre> |
|  Console | La console affiche les avertissements et les erreurs contenues dans la page. On y trouvera notamment les erreurs de chargement de scripts. Il est également possible d'accéder aux variables de la page, en tapant (en bas) par exemple : <code>window.location.href</code>  |
|  Débugueur | C'est un outil particulier qui permet de stopper, mettre en pas-à-pas des scripts Javascript. |
|  Éditeur de style | Cet outil permet de modifier en direct les feuilles de styles présentes dans le document. Il permet également d'ajuster en temps réel les couleurs et positions des éléments. Les checkbox devant chaque ligne permettent de visualiser les modifications faites. <pre>body { pebbles- font-size: 18px; font-size: 1rem; color: #00FFFF;</pre> |
|  Performances | Cet outil (lorsqu'il est activé) affiche les événements temporels rencontrés sur le site. C'est une sorte d'analyseur en temps réel. |
|  Mémoire | Cet outil affiche la répartition de l'occupation mémoire des objets présents dans la page ; |
|  Réseau | Cet outil analyse les connexions réseaux établies par les contenus de la page. On y retrouve des informations qui sont visibles dans des outils comme Wireshark. |
|  Stockage | Cet outil indique les éléments de la page stockés sur l'ordinateur (notamment les cookies). |

Tableau 5.2 : les fonctions des outils de développement

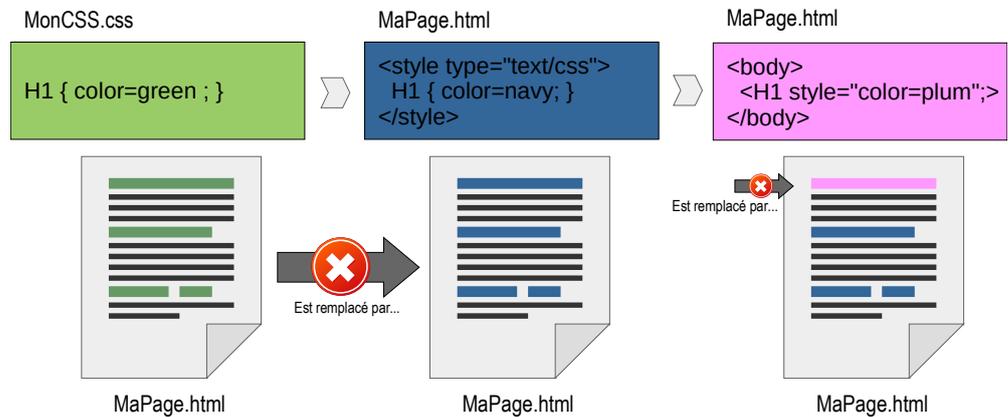
Vous trouverez toutes les informations complémentaires sur le site de Mozilla⁵.

5 <https://developer.mozilla.org/fr/docs/Outils>

6 EN RÉSUMÉ

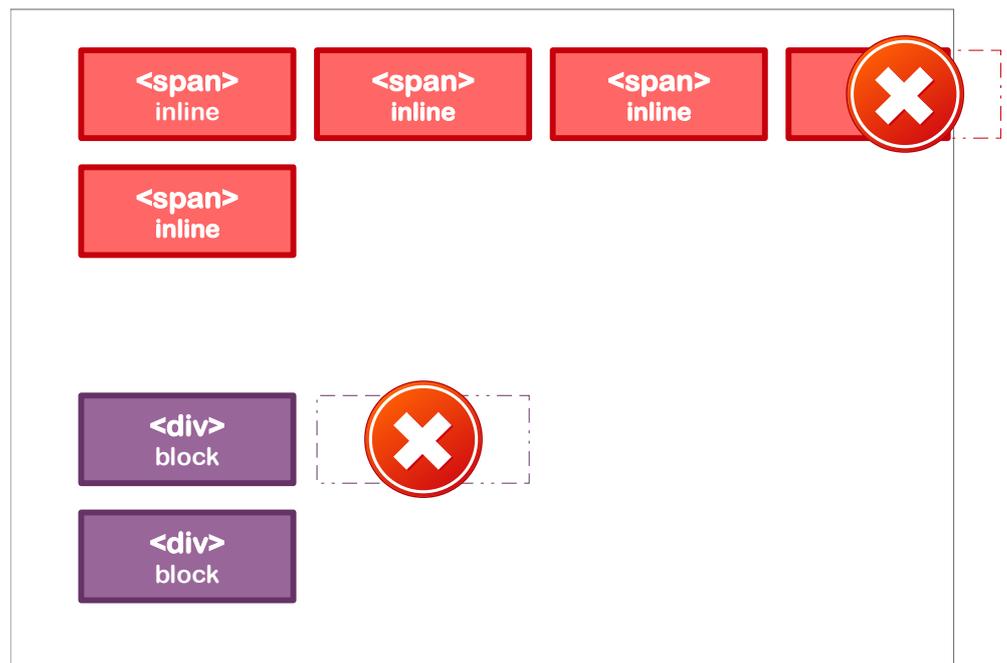
Le code CSS permet de mettre en forme les informations (le fond).

Plus un style est proche d'une balise, et plus il est prioritaire (notion de cascade, le style s'applique à tous les éléments d'une balise).



Les sélecteurs sont les principaux éléments pour positionner les éléments entre-eux.

On distingue les "**block**" (toujours les uns **sous** les autres) et les "**inline**" (les uns **après** les autres).



Les blocs sont le résultat de plusieurs "boites" : **padding** (intérieur), **border** (ligne) et **margin** (ext.).

Cela signifie que le contenu est entouré d'un cadre dont l'épaisseur est la somme de padding, border et margin.



7 ANNEXES

7.1 LES ÉTAPES DE RÉALISATION D'UN SITE

La plupart des débutants pensent que la création d'un nouveau site passe par... l'outil de graphisme.

En réalité, les professionnels utilisent un mockup (zoning, graphique fil de fer, etc.)

En effet, les étapes de création pour ne pas perdre de temps sont les suivantes.

7.1.1 Cahier des charges

C'est l'étape la plus importante et pourtant... la plus négligée !

Il s'agit de réunir le client et le créateur pour définir le projet. C'est dans cette phase que le designer doit poser les bonnes questions. Plus le besoin est clairement défini, moins le risque d'un site erroné est grand. Les questions peuvent porter sur l'aspect du site (nombre de page, arborescence, charte de couleur, images/photos...) mais aussi les fonctionnalités (plugins, modules externes, localisation, météo, formulaire de contact, ...)



7.1.2 Design

Le design est très subjectif : ce que vous pensez être magnifique peut ne pas plaire au client. En général, il est préférable de présenter deux maquettes. En effet, chaque modification sur la maquette n'est pas coûteuse en temps. Lorsque l'étape d'intégration commence, les changements peuvent être désastreux sur la durée du projet.



7.1.3 Intégration

C'est la partie codage et préparation. Il faut travailler sur une version indépendante, qui n'écrase pas le site actuel du client. Il est même préférable d'effectuer une sauvegarde préalable pour éviter tout risque.

Votre savoir fera toute la différence : HTML, CSS, Javascript, SQL... et tous les outils d'aide à la conception (frameworks, IDA, etc.)

7.1.4 Création d'un modèle

Il s'agit d'une phase appelée "habillage". Les données et ressources présents sur le site ne sont pas les documents finaux. Cela permet en revanche de voir le site quasiment tel qu'il sera définitivement. On ne regarde pas les textes et photos (contenus) mais l'aspect dynamique du site (contenant).

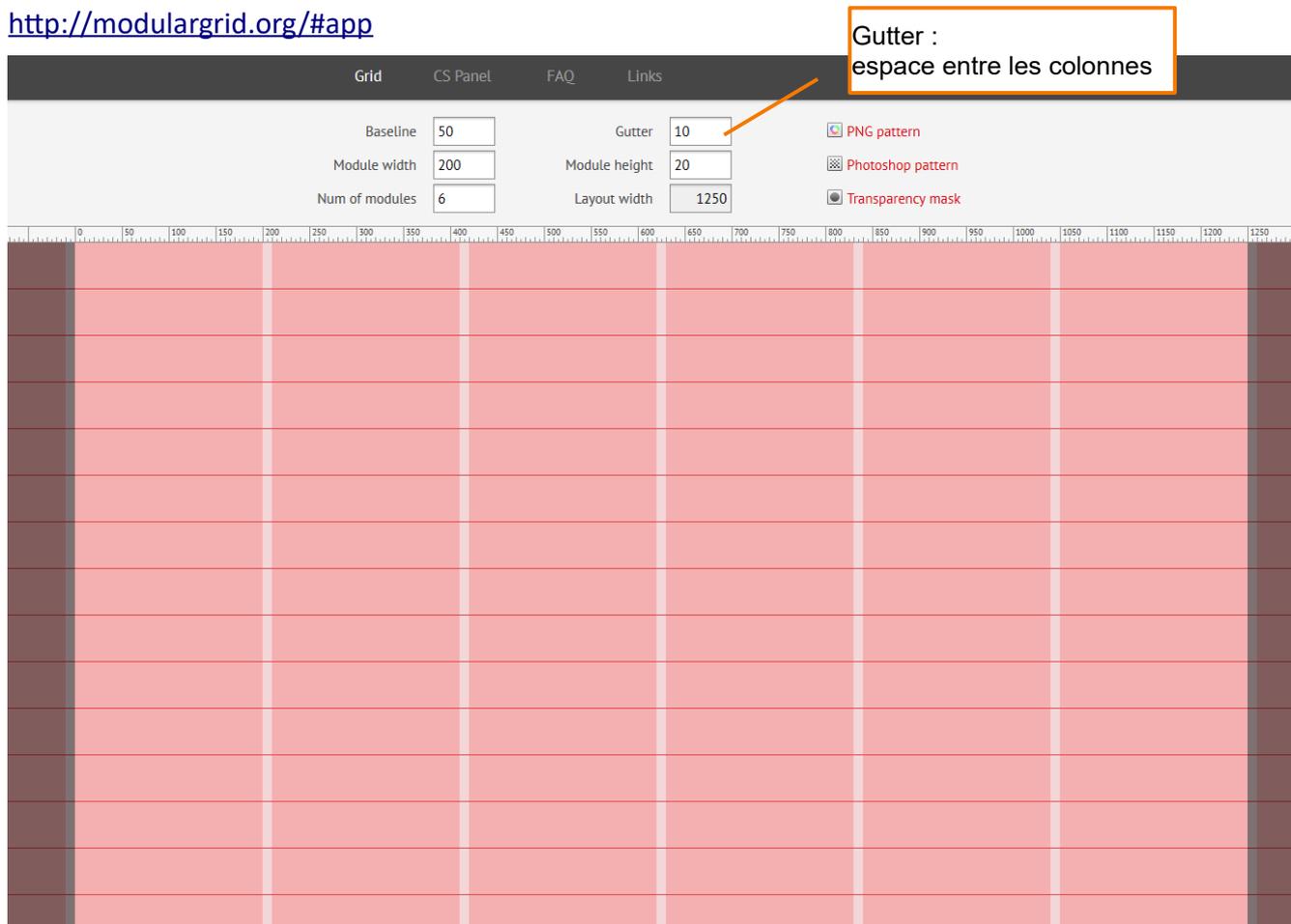
Il faut noter qu'il est de plus en plus rare de proposer des sites sans gestionnaire de contenus (CMS). Soit vous optez pour un gestionnaire existant (Wordpress, Joomla, Drupal, etc.) soit vous le créez.

7.1.5 Mise en ligne

7.2 OUTIL DE CRÉATION DE GRILLE

Cet outil ne crée pas la grille en CSS. C'est un outil de design : il permet de créer une grille, utilisable dans un logiciel comme Adobe Photoshop ou Gimp...

<http://modulargrid.org/#app>



The screenshot shows the Modulargrid tool interface. At the top, there are navigation links: Grid, CS Panel, FAQ, and Links. Below these are input fields for grid configuration:

- Baseline: 50
- Module width: 200
- Num of modules: 6
- Gutter: 10
- Module height: 20
- Layout width: 1250

On the right side, there are three checkboxes for pattern options: PNG pattern, Photoshop pattern, and Transparency mask. A callout box with an orange border and arrow points to the Gutter input field, containing the text: "Gutter : espace entre les colonnes". Below the input fields is a horizontal ruler with a scale from 0 to 1250. The main area of the interface displays a grid with 6 columns and 10 rows, with a light red background and white grid lines.