

R102 – TP4 - Mise en page

Objectif

Le but de ce TP est de vous familiariser avec la méthode standard de placement des éléments dans un document HTML.

Ordre du jour

1. Préliminaires	2
1.1. Coin culture : Les moteurs de rendus HTML	2
1.2. Les modèles des boîtes	2
1.3. Comportement visuels des éléments	3
1.4. Mise en page	4
2. C'est parti	5
2.1. Phase 1 : Positionner le menu	5
2.2. Phase 2 : Mise en page des fiches métiers	6
2.3. Phase 3 : Finir le menu	7
2.4. Bonus	8
2.4.1. Disposition multi-colonnes	8
2.4.2. Amélioration du menu	8

1. Préliminaires

Certaines des informations données ici sont des rappels, d'autres non. Toutes sont à connaître.

En HTML5, les éléments sont réparties en plusieurs catégories selon leur caractéristique (https://developer.mozilla.org/fr/docs/Web/Guide/HTML/Content_categories).

1.1. Coin culture : Les moteurs de rendus HTML

On appel **moteur de rendu HTML**, le système (souvent sous forme de bibliothèques de composants logiciels) qui permet d'afficher une page web. C'est donc le cœur d'un navigateur web.

Les 3 principaux^[1] moteurs de rendu actuels sont :

- | | |
|---------------|---|
| Gecko | <ul style="list-style-type: none">• https://fr.wikipedia.org/wiki/Gecko_(moteur_de_rendu)• https://developer.mozilla.org/fr/docs/Glossary/Gecko |
| Blink | <ul style="list-style-type: none">• Basé sur Webkit• https://fr.wikipedia.org/wiki/Blink_(moteur_de_rendu)• https://www.chromium.org/blink/ |
| Webkit | <ul style="list-style-type: none">• Basé sur KHTML (venant de l'environnement graphique KDE)• Utilisé par Safari et de nombreux navigateurs sur mobiles et tablettes.• https://fr.wikipedia.org/wiki/WebKit• https://webkit.org/ |



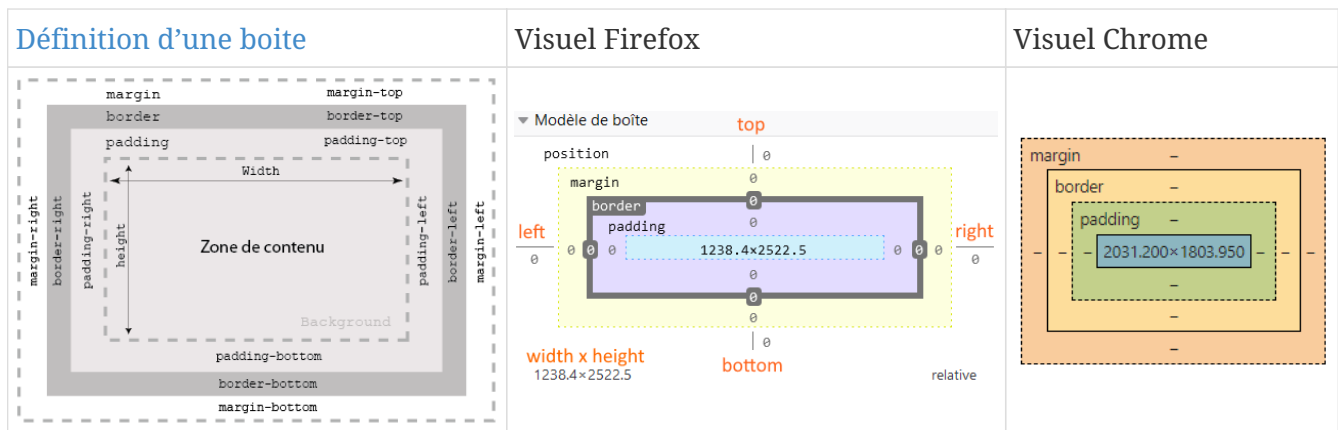
[1] : Il y a 3 moteurs de rendu principaux, mais 4 sont cités et à connaître.

1.2. Les modèles des boîtes

Tout **élément** html est vu comme une boîte rectangulaire. En voici quelques représentations (déjà vu en cours et au TP2) :

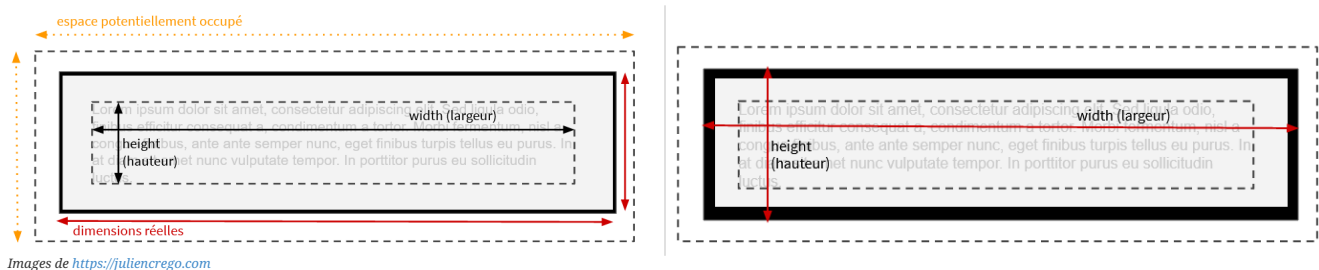
Visuellement, un élément est une boîte rectangulaire possédant ces caractéristiques :

- Un contenu ayant une *largeur* (**width**) et une *hauteur* (**height**). Ces propriétés ne sont pas toujours modifiables.
- une *marge intérieure* (ou *remplissage*, **padding**) : espacement entre le contenu de la bordure
- une *bordure* : modifiable en épaisseur (**border-width**), style (**border-style** type de trait), et couleur (**border-color**). En CSS3, s'est ajoutée l'arrondi des coins (**border-radius**), et d'autres plus génériques encore en développement dans certains navigateurs (*block-start/end*, *inline-start/end* ...).
- une *marge extérieure* (**margin**): espacement entre la bordure et les boîtes voisines.



Il existe un autre modèle de boîte où la *hauteur* et la *largeur* incluent la marge intérieure et la bordure (cf images ci-dessous). Certains développeurs préfèrent ce modèle mais ce n'est pas le modèle utilisé par défaut. Le changement de modèle se fait avec la propriété **box-sizing**.

Propriété box-sizing



1.3. Comportement visuels des éléments

Certaines propriétés tels que **Display**, **float** ou **position** permette de définir le comportement d'un élément dans son environnement.

En CSS3, la propriété **display** permet de définir simultanément le comportement externe (par rapport aux éléments voisins) et interne (comportement des éléments contenus). Et ceci peut se faire avec une syntaxe en une (valeurs avec au moins un tiret **-**), ou deux valeurs (séparées par un espace). Pour les détails, voir ici : https://developer.mozilla.org/en-US/docs/Web/CSS/display/multi-keyword_syntax_of_display et ici : <https://drafts.csswg.org/css-display/#display-value-summary>

Déjà vu : Comportements externes

- Éléments de type *bloc*** : Ils ajoutent un retour à la ligne avant ET après l'élément. Un tel élément est donc seul sur sa ligne, ce qui lui permet, sauf indication contraire, de s'étendre sur toute la largeur de son parent. Exemple : les balises **<h1>**, **<p>**, **<table>**, **<div>**...
- Éléments de type *en ligne*** : Ils correspondent à la catégorie HTML5 de *contenu phrasé* : Un élément en ligne ne commence pas sur une nouvelle ligne et prend uniquement la largeur qui lui est nécessaire. Dans un flux normal, si l'espace le permet, le prochain élément sera sur la même ligne. Les propriétés **width** et **height** n'ont pas d'effet sur ces éléments car ils s'adaptent à la ligne sur laquelle ils sont placés. Exemple : les balises ****, ****, ****, ****, **
**...

Il existe un comportement intermédiaire nommé **inline-block** permettant de prendre en compte les marges de l'élément concerné en évitant leur fusion verticale avec le contenant. Cela permet par exemple d'aligner proprement des éléments **a** ou **li** pour réaliser un menu horizontal cohérent.



En général, il n'est pas possible de placer des éléments de type *bloc* à l'intérieur d'éléments de type *en ligne*.



- Si on souhaite qu'un élément ne soit pas affiché, on peut utiliser `display:none`. Le document est alors affiché comme si l'élément n'existait pas.
- Si l'on souhaite cacher un élément tout en conservant son espace dans le document, il faudra alors utiliser la propriété `visibility`.

1.4. Mise en page

La mise en page est le placement des éléments les uns par rapport aux autres. L'ordre d'écriture des éléments dans la page HTML donne un placement initial dans le document final.

Comme vous l'avez vu avec le site d'exemple [css zen garden](#), le placement par défaut peut être complètement modifié par des feuilles de styles.

Ceci peut résulter de la combinaison de plusieurs méthodes de placements des éléments :

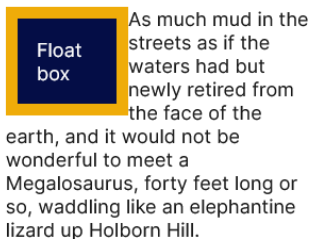


Figure 1. Image flottante

- les flottants (propriété `float`) qui va placer une boîte hors du flux, sur la droite ou la gauche tout en permettant aux éléments voisins de s'adapter autour de lui.
- La propriété `position` qui permet de contrôler avec précision le placement de boîtes par rapport aux autres. La valeur `static` est le placement par défaut dans le flux normal, mais vous pouvez manipuler les éléments pour qu'ils se comportent différemment.

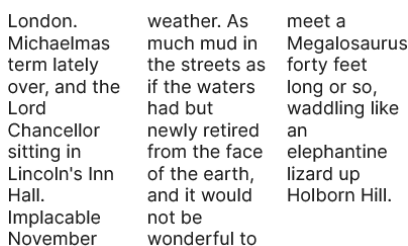


Figure 2. Multicolonne

- Mise en page en **multi-colonne** (https://developer.mozilla.org/fr/docs/Web/CSS/CSS_Columns/Using_multi-column_layouts) permet d'utiliser une disposition des boîtes en colonne(s), comme dans un article de journal par exemple.
- Et bien sûr, les méthodes de mise en page "*autonomes*", par exemple les boîtes flexibles (<https://www.w3.org/TR/css-flexbox-1/>) ou les grilles (<https://www.w3.org/TR/css-grid-2/>).

2. C'est parti

Dans ce TP, nous allons utiliser les dispositions standards. Les dispositions flexibles et en grilles seront pour d'autres TP.

Il faut tout d'abord (vous êtes maintenant habitué) :

1. Créer les dossiers associés à ce TP :

Une seule commande suffit

```
mkdir -p ~/public_html/r102/tp04/{css,img}
```

2. Et comme d'habitude, récupérer sur Moodle les ressources et placer les correctement dans les dossiers de ce TP.




- Les contenus des fichiers existants ne doivent pas être modifiés. Pour permettre au serveur http de transmettre vos ressources, pensez à vérifier les droits d'accès (fichiers lisibles et dossiers pour les atteindre traversables)
- Le but étant de vous apprendre comment placer les éléments avec les propriétés css basiques, toutes les règles à écrire ici **NE DOIVENT PAS** utiliser les méthodes de positionnement flex ou grid.

2.1. Phase 1 : Positionner le menu

Placement du menu à l'horizontal sans afficher les sous-menus (des captures d'écran se trouvent un peu plus bas).

Réalisez les actions suivantes :

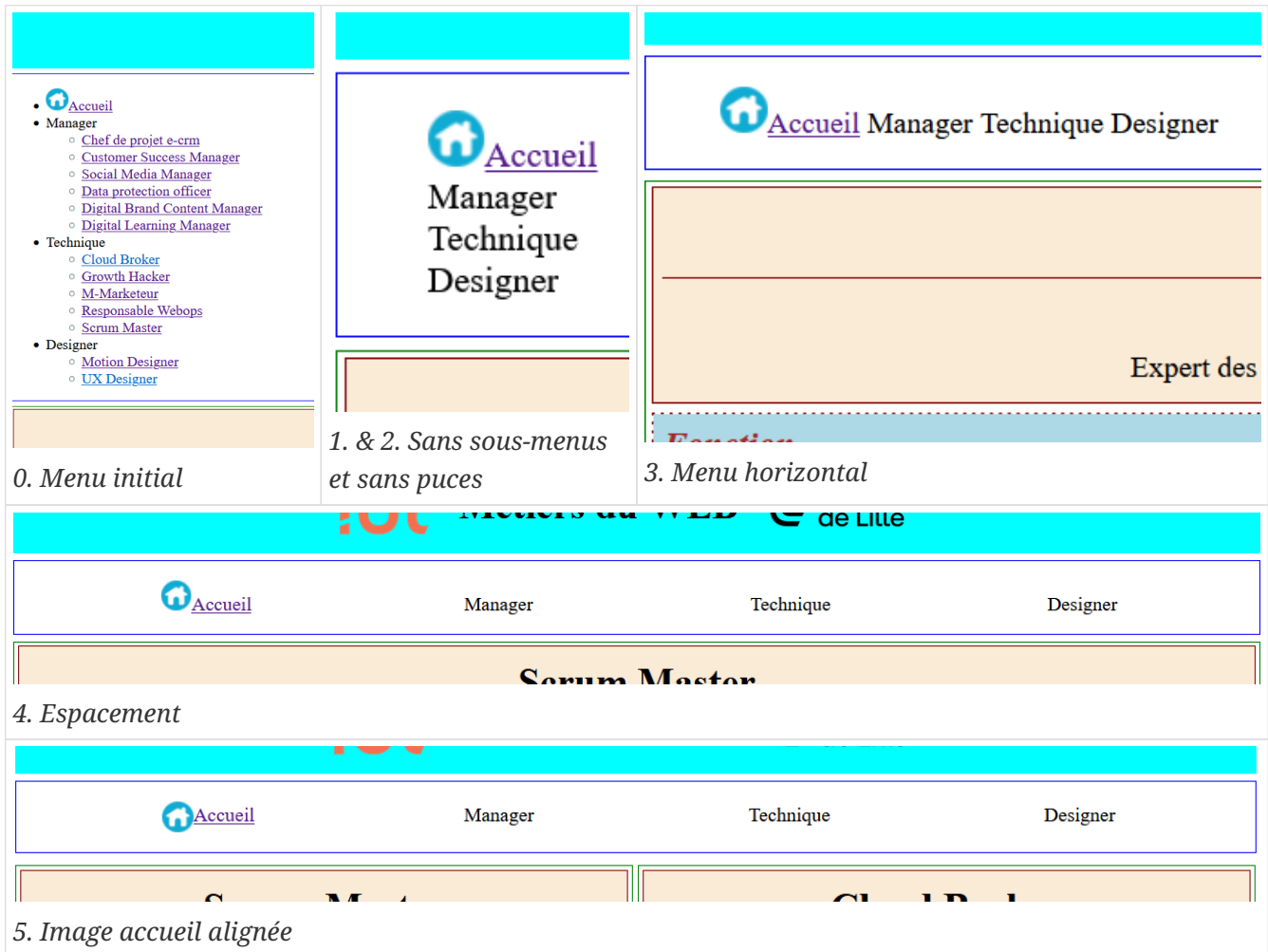
- ☐ 1. Enlevez toutes les puces : 1 règle css avec une seule déclaration (Rappel vu en cours : 1 déclaration est un couple **propriété: valeur**)
- ☐ 2. Faire disparaître les sous-menus : 1 règle css avec une seule déclaration
- ☐ 3. Mettre le menu à l'horizontal : 1 règle css avec une seule déclaration
- ☐ 4. Centrer et espacer les menus (doivent tenir sur une seule ligne pour une fenêtre pas trop petite). Ici, il vous faudra utiliser ajouter des déclarations dans le bloc du sélecteur précédent. Astuce : agrandir la largeur des 4 items au maximum possible et centrer les textes de ces items pour équilibrer visuellement leur position.
- ☐ 5. Centrer verticalement l'image  du menu Accueil et le mot *Accueil* ? : 1 nouvelle règle css avec 2 déclarations



Indice point 5

1. Il faut modifier le comportement de l'image pour pouvoir la centrer verticalement.
2. Cette méthode existe déjà dans la feuille de styles fournie.

Aide visuelle : Images des étapes de la mise en place du menu en phase 1



Le menu n'est pas terminé, il reste l'affichage correct des sous-menus au survol par la souris. Mais cette partie se fera un peu plus tard. Vous allez dans la phase suivante placer les fiches métiers sur deux colonnes.

2.2. Phase 2 : Mise en page des fiches métiers

Réalisez les actions suivantes dans l'ordre :

- ☐ Réduire la largeur des fiches métiers à 49% de la largeur d'une page. Pour le moment elles sont encore l'une sous l'autre.
- ☐ Modifier le comportement des boîtes contenant une fiche métier afin qu'elle se comporte comme une boîte **en ligne** tout en conservant la largeur que vous venez de définir.
- ☐ Aligner sur le haut les fiches métiers voisines.

C'est fini pour les fiches métiers, retournons sur le menu.

2.3. Phase 3 : Finir le menu

Dans son état actuel, le menu ne sert pas à grand chose. Nous allons le compléter pour utiliser les sous-menus.

Réalisez les actions suivantes (toujours sans utiliser les dispositions flexibles ou en grilles qui seront vu plus tard)

- ☐ Afin de mieux le repérer, mettez un fond bleu sur le menu où se trouve la souris : Une règle avec une seule déclaration
- ☐ Rendez visible le sous-menu associé : Une règle avec une seule déclaration

Le sélecteur précédent est une partie de celui à utiliser ici

- ☐ Pourquoi le sous menu possède un fond bleu et est décalé sur la droite alors que le menu est centré (réalisé en phase 1) ?
- ☐ Pour mieux visualiser et corriger ces soucis de visualisation et de positionnement des sous-menus, nous allons ajouter les déclarations suivantes UNE PAR UNE dans votre règle précédente (l'affichage d'un sous-menu au survol de la souris). Expliquez l'effet de chacune de ces déclarations.

```
border: thin dashed blue;  
background-color: white;  
padding: 0;
```

Il reste un problème : Vous avez remarqué que les menus se déplacent lorsqu'un sous-menu apparaît. Sauriez-vous expliquer ce qui se passe (et surtout pourquoi) ? Pas d'inquiétude, nous allons voir comment corriger ce problème.

- ☐ Observez où se placent les titres des menus sans sous menus affichés.

Ces menus se placent en bas. Nous voudrions que ces menus restent "en haut". Il y a donc un comportement par défaut qu'il faut corriger. C'est donc un problème d'alignement vertical; Vous avez déjà vu une méthode pour corriger un alignement vertical;

- ☐ Corriger l'alignement en utilisant la méthode déjà vu précédemment; Dans ce cas précis, la règle ciblant les bons éléments existe déjà. Donc il vous suffit d'ajouter une seule nouvelle déclaration dans le bloc de déclarations de cette règle.

Afin de repérer mieux la ligne du sous-menu sur lequel la souris se trouve,

- ☐ Ajouter un fond "coral" à la ligne du sous-menu où se trouve la souris : une nouvelle règle avec une seule déclaration suffit

2.4. Bonus

2.4.1. Disposition multi-colonnes

- ☐ Lisez cette page : [Concepts de base pour la disposition multi-colonnes](#)
- ☐ Mettez votre code de positionnement des fiches métier en double colonne (phase 2) entièrement en commentaire.
- ☐ Utilisez la méthode de disposition multi-colonnes pour placer vos fiches sur 2 colonnes.

Cette méthode de placement est plus généralement utilisé sur des textes (avec ou sans images incluses) que sur d'autres types d'éléments.

2.4.2. Amélioration du menu

Utiliser des techniques de [transition css](#) ou d'[animation css](#) pour rendre plus souple la mise en avant des élément du menu au passage de la souris (aussi bien sur le titre du menu, que sur une ligne d'un sous-menu).