Henallux – DA1 – web : principes de base

Laboratoire 6

Layout

Sommaire

[1. (Exo) Flux naturel en HTML 2](#_Toc160628590)

[2. (Exo) le box model 7](#_Toc160628591)

[3. (Exo) Éléments flottants 11](#_Toc160628592)

[4. (Exo) Qu’est-ce que le jeu de rôle ? 16](#_Toc160628593)

[5. Solution partielle pour l’exo 3, étape 3 22](#_Toc160628594)

# (Exo) Flux naturel en HTML

Objectifs

* observer le flux naturel des éléments HTML
* modifier le mode d’affichage d’une balise
* observer les différences de comportement entre inline et block

Le but de cet exercice est d’observer le comportement normal (le mode d’affichage normal) de quelques balises HTML.

## Étape 1 : balises à examiner et document de départ

Voici la description courte d’une petite vingtaine de balises HTML. Vous en avez déjà rencontré certaines dans les laboratoires précédents.

|  |  |
| --- | --- |
| Balise | Utilisation |
| a | lien |
| abbr | abbréviation (comme DA, CP, TI, USA, …) |
| blockquote | citation longue |
| button | bouton |
| code | un programme (entier ou une partie de programme) |
| em | un texte en emphase |
| h1 | un grand titre |
| h3 | un titre moyen |
| h6 | un petit (sous-)titre |
| img | une image |
| kbd | un texte à entrer tel quel au clavier (kbd = keyboard) |
| mark | un texte marqué (par exemple pour marquer  un mot recherché sur une page) |
| pre | un texte à afficher tel quel,  texte déjà préformaté (espaces conservées) |
| p | un paragraphe |
| q | une citation courte |
| sub | un indice (par exemple : H20) |
| sup | un exposant (par exemple : 9 m2) |

Créez un document HTML et incorporez le code suivant dans la partie <body>. Ce code reprend un exemple pour chacune des balises citées ci-dessus. Notez que, dans le code HTML, on a placé une balise par ligne, mais ceci n’influence pas l’apparence de la page web vu que les blancs répétés sont ignorés en HTML.

<a href="http://www.google.be">Lien</a>

<abbr>Abréviation</abbr>

<blockquote>Citation plus longue</blockquote>

<button>Bouton</button>

<code>Code informatique</code>

<em>Texte en emphase</em>

<h1>Titre h1</h1>

<h3>Titre h3</h3>

<h6>Titre h6</h6>

<img src="eye.jpg" />

<kbd>Texte à entrer au keyboard</kbd>

<mark>Texte marqué</mark>

<pre>Pre formaté</pre>

<p>Paragraphe</p>

<q>Citation/quote courte</q>

<sub>Indice</sub>

<sup>Exposant</sup>

## Étape 2 : inline et block

Ouvrez le document HTML sous Firefox et observez comment le contenu des balises est présenté.

Certaines de ces balises correspondent à un affichage de type « inline » par défaut alors que d’autres sont affichées en mode « block » par défaut. Repérez lesquelles correspondent à quel type d’affichage.

|  |  |
| --- | --- |
| **Affichage « inline »** | **Affichage « block »** |
| Balises : | Balises : |

Plus tôt dans le cours, on a introduit les balises génériques <div> et <span>. La seule différence entre ces deux balises qui, a priori, n’ont aucune signification propre ni aucun style d’affichage par défaut (pas de gras ou de couleur ou d’autres artifices) est le mode d’affichage.

La balise <span> correspond par défaut à un mode d’affichage inline (donc, un bout de paragraphe) alors que la balise <div> correspond par défaut à un affichage de type block (donc un bloc qui se tient seul sur sa ligne).

## Étape 3 : espace occupé par chaque balise

Pour rendre les choses plus claires et pouvoir plus facilement repérer l’emplacement réservé pour chacun des éléments HTML, vous pouvez ajouter la règle CSS suivante à votre document.

\* {border: 1px solid blue}

Pour rappel, \* est un sélecteur universel : la règle s’appliquera donc à tous les éléments de la page. Notez que même la balise <body> reçoit un encadrement bleu.

Examinez le résultat sur Firefox et notez entre autres que **tous** les éléments, quel que soit leur mode d’affichage, occupent un emplacement rectangulaire.

## Étape 4 : changer le mode d’affichage d’un élément

Vous pouvez modifier le mode d’affichage des éléments HTML. Pour cela, utilisez par exemple la règle suivante, pour que tous les éléments à l’intérieur de <body> soient affichés en mode « inline ».

body \* {display: inline}

Tout tient-il alors sur une seule ligne ? Examinez le rendu sous Firefox en rétrécissant et en élargissant la largeur de la fenêtre de votre navigateur.

Le flux naturel pour le mode d’affichage « inline » consiste à écrire les éléments de gauche à droite, sur la même ligne. Lorsqu’il n’y a plus de place sur la ligne en question, l’élément est placé au début de la ligne suivante. (Note : il existe des options en HTML pour plutôt écrire de droite à gauche au lieu de gauche à droite, ce qui est nécessaire pour certaines langues.)

Finalement, modifiez la règle que vous venez d’ajouter pour faire en sorte que tous les descendants de <body> soient en mode d’affichage « block » et observez le résultat.

## Étape 5 : inline et block

Créez un nouveau document et placez-y le contenu suivant, qui est composé de deux parties :

* un paragraphe comportant plusieurs balises inlines (le texte provient de Wikipédia) ;
* un div comportant plusieurs balises block (il s’agit de citations de Robert C. Martin, un auteur qui a beaucoup écrit sur le clean code).

<p class="conteneur">L'<strong>Extensible Markup Language</strong>, généralement appelé <strong>XML</strong>, <em>langage de balisage extensible</em> en français, est un métalangage informatique de balisage générique qui est un sous-ensemble du <strong>Standard Generalized Markup Language</strong> (<strong>SGML</strong>). Sa syntaxe est dite <em>extensible</em> car elle permet de définir différents langages avec pour chacun son vocabulaire et sa grammaire, comme <strong>XHTML</strong>, <strong>XSLT</strong>, <strong>RSS</strong>, <strong>SVG</strong>… Elle est reconnaissable par son usage des <em>chevrons</em> encadrant les noms des balises. L'objectif initial de <strong>XML</strong> est de faciliter l'échange automatisé de <em>contenus complexes</em> (arbres, texte enrichi, etc.) entre systèmes d'informations hétérogènes (interopérabilité).</p>

<div class="conteneur">

<blockquote>It is not enough for code to work.</blockquote>

<blockquote>You should name a variable using the same care with which you name a first-born child</blockquote>

<blockquote>Clean code is not written by following a set of rules. You don’t become a software craftsman by learning a list of heuristics. Professionalism and craftsmanship come from values that drive disciplines.</blockquote>

<blockquote>Redundant comments are just places to collect lies and misinformation.</blockquote>

<blockquote>A long descriptive name is better than a long descriptive comment.</blockquote>

</div>

Ajoutez à votre document des règles CSS pour

1. placer une bordure de 2 pixels de largeur autour de chaque élément descendant de la balise <body> (choisissez une couleur bien visible) ;
2. rendre encore plus visibles les espaces occupés par les éléments à l’intérieur du <p> et du <div> en leur attribuant une couleur de fond (background-color), par exemple lemonchiffon. (Notez que le paragraphe et le div portent tous les deux la classe « conteneur », ce qui peut simplifier l’écriture de cette seconde règle.)

## Étape 6 : margin collapsing!

Pour observer le fonctionnement des marges, et plus particulièrement l’effet de « margin collapsing »,

1. ajoutez dans votre document un paragraphe *entre* les deux éléments de classe « conteneur » (par exemple <p>Séparateur</p>) ; ce paragraphe vous aidera à mieux visualiser et vérifier le fonctionnement des deux règles CSS suivantes ;
2. ajoutez une règle donnant une marge vers le haut (margin-top) de 60 pixels au <div> de classe « conteneur » ;
3. ajoutez une règle donnant une marge vers le bas (margin-bottom) de 30 pixels au paragraphe de classe « conteneur », mais uniquement quand ce paragraphe est survolé par le curseur de la souris.

Observez l’effet de ces règles et notez, entre autres, que l’ajout de la marge en-dessous du paragraphe repousse tout le contenu (y compris le paragraphe séparateur que vous avez ajouté).

Après avoir vérifié que ces règles fonctionnaient bien, supprimez le paragraphe séparateur ajouté au point 1 et examinez le comportement de votre page HTML. Notez entre autres ce qui se passe quand vous survolez le premier paragraphe avec la souris… et faites le lien avec ce qui a été dit en cours.

## Étape 7 : rembourrage et marge

La règle CSS qui cible tous les éléments internes au paragraphe et au <div> de classe conteneur affecte à la fois les éléments inline dans le texte et les citations, qui sont des éléments block. En la modifiant, vous allez pouvoir comparer l’effet de certaines propriétés sur ces deux types d’éléments. Dans cette étape, on désignera ces éléments (les inline du paragraphe et les block du div) sous le nom « d’éléments à tester ».

1. Dans un premier temps, ajoutez à cette règle des propriétés fixant la largeur (width) et la hauteur (height) des éléments à tester à 100 pixels. Observez l’effet (qui est différent sur les inline et sur les block).
2. Commentez les lignes correspondant aux deux propriétés que vous venez d’ajouter et ajoutez padding: 20px. À nouveau, observez l’effet de cette propriété.  
     
   Notez que le padding est pris en compte tant sur les inline que sur les block mais, dans le cas des inline, le contenu est repoussé uniquement horizontalement (les mots avant et après les balises inline), pas verticalement (les lignes au-dessus et en-dessous), alors que c’est bien le cas pour les block.
3. Remplacez la propriété padding par margin et observez ce que ça change.  
     
   Assurez-vous de bien comprendre la différence entre  
   • le padding/rembourrage, qui sépare le contenu de la bordure (et donc se trouve à l’intérieur du cadre, coloré en jaune) et  
   • la margin/marge, qui se situe à l’extérieur de la bordure et n’est pas coloré.  
     
   Constatez que, comme dans le cas du padding, la marge est prise en compte horizontalement pour les éléments inline mais pas verticalement.

## étape 8 : le mode inline-block

Vous avez pu observer l’effet des propriétés width, height, padding et margin sur les éléments inline et block et entre autres noté que certains effets étaient limités ou ne s’appliquaient pas du tout sur les inline.

Il reste un troisième mode d’affichage standard à examiner : inline-block. Comme il n’existe a priori aucune balise qui utilise ce mode par défaut, il va falloir le forcer.

Dans votre document HTML, dupliquez le paragraphe « conteneur » et ajoute la classe « inlineblock » à la copie. Pour rappel, un élément HTML peut porter plusieurs classes. Dans ce cas-ci, votre code devrait ressembler à ceci :

<p class="conteneur inlineblock">…

Ajoutez une règle CSS qui cible tous les éléments situés dans un paragraphe de classe « inlineblock » et les passe en mode inline-block (display: inline-block).

Reprenez ensuite les modifications de l’étape précédente (width et height dans un premier temps, puis padding, et enfin margin) pour en observer les effets sur les inline-block.

# (Exo) le box model

Objectifs

* observer le fonctionnement du box model
* utiliser l’inspecteur de Firefox pour examiner un élément
* manipuler les paramètres du box model

Tous les éléments HTML, qu’ils soient affichés au mode inline ou au mode block occupent un emplacement rectangulaire. La taille de cet emplacement et l’espace qui le sépare de ce qui l’entoure sont déterminés par le box model.

## Étape 1 : histoire de trois div

Commencez par créer un document HTML vide dans lequel vous ajouterez le code suivant.

<div id="div1">DIV1</div>

<div id="div2">DIV2</div>

<div id="div3">DIV3</div>

Le contenu est composé de trois <div> auxquels on a associé des identificateurs afin de pouvoir les cibler aisément dans les règles CSS.

Pour rendre l’espace occupé par les <div> plus facilement repérable, ajoutez des règles CSS leur imposant des couleurs de fond différentes. Notez que, par défaut, le <div> occupe toute la largeur disponible.

## Étape 2 : largeur et block centré

Pour certains sites web où la position de chacun des éléments est calculée au pixel près, il peut être dérangeant de laisser les <div> occuper toute la largeur de la fenêtre. En effet, dans ce cas, l’apparence change selon que la fenêtre est plus ou moins étirée.

Une astuce pour contourner ce problème consiste à « emballer » le contenu dans un <div> dont la largeur est imposée. Ce <div> porte souvent l’identificateur « wrapper » (littéralement : emballage).

Modifiez le contenu de votre fichier HTML en ajoutant un tel <div>, qui sera donc l’unique fils de <body>. N’oubliez pas de corriger l’indentation !

Construisez ensuite une règle CSS  portant sur le wrapper et imposant une largeur de 600 pixels. Ajoutez-y également les deux propriétés suivantes, qui permettent de centrer horizontalement le wrapper au sein du <body>.

margin-left: auto;

margin-right: auto;

Testez l’effet de l’ensemble sous Firefox en modifiant la largeur de la fenêtre. Vérifiez les cas où la largeur est plus grande que 600 pixels et où elle est plus petite que 600 pixels.

## Étape 3 : inspecter le box model sous Firefox

Sous Firefox, effectuez un clic droit dans le DIV2 puis choisissez l’option « Inspect » pour ouvrir l’inspecteur.

À droite, choisissez l’onglet « Layout ». Cela devrait dévoiler un affichage composé d’un rectangle bleu au milieu et de trois cadres (dont la couleur dépend du thème que vous avez choisi pour Firefox).

A screenshot of a computer

Description automatically generated

Le rectangle central représente le contenu (si vous laissez votre curseur sur ce rectangle, le tooltip « Content » apparaît). C’est la taille réservée pour le contenu du <div>, même si le contenu actuel (le texte « DIV2 ») n’occupe pas toute la place.

Normalement, la taille du rectangle central devrait être de 600×18 (c’est-à-dire 600 pixels horizontalement sur 18 pixels verticalement). Notez-la quelque part ; dans la suite, on examinera comment cette taille évolue.

Les trois couches en oignon (de l’intérieur vers l’extérieur) sont :

* le rembourrage / padding : l’espace entre le contenu et le cadre,
* l’encadrement / border : la bordure (plus ou moins épaisse), et
* la marge / margin : l’espacement entre ce div et ce qui l’entoure.

Si vous positionnez votre curseur sur l’une de ces couches, l’espace correspondant est mis en évidence sur la page HTML, ce qui peut être pratique pour bien comprendre comment le box model fonctionne (et pour « débugguer » des règles CSS qui n’ont pas l’effet désiré).

## Étape 4 : padding

Ajoutez une règle CSS ciblant le DIV2 (ou modifiez la règle existante) pour ajouter du rembourrage sur 6 pixels dans les 4 directions. Cela devrait « aérer » le texte en l’éloignant des bords du <div>.

Observez non seulement l’effet de cette modification sur le rendu visuel mais également dans l’inspecteur.

Que devient la taille du rectangle bleu/contenu ? Pourquoi a-t-elle diminué ?

## Étape 5 : border

Ajoutez une bordure en pointillés autour du DIV2, avec une épaisseur de 4 pixels.

À nouveau, observez l’effet d’un point de vue visuel et dans l’inspecteur. Jetez également un coup d’œil sur la taille du contenu.

## Étape 6 : margin

Finalement, ajoutez une marge de 12 pixels dans les 4 directions autour de DIV2. Et, à nouveau, observez les modifications.

C’est plutôt désirable d’éloigner le DIV2 des deux autres DIV mais c’est relativement inutile de créer des marges à gauche et à droite. À l’aide des slides du cours, modifiez la règle pour que la marge fasse 12 pixels en haut et en bas mais 0 pixels à gauche et à droite. Trouvez au moins 3 manières de le faire.

## Étape 7 : marges auto

Modifiez à nouveau la règle portant sur DIV2 pour faire en sorte que les marges à gauche et à droite aient comme valeur auto. Observez que cela ne change rien : les marges à gauche et à droite restent nulles.

La valeur auto signifie « prendre l’espace restant et le diviser aussi équitablement que possible entre toutes les marges qui ont pour valeur auto ». Cependant, comme le <div> occupe toute la largeur disponible, il ne reste aucun espace à distribuer ! Pour qu’il reste de l’espace disponible, il faudrait imposer une largeur plus petite au DIV2.

Considérez l’option suivante (ne l’implémentez pas tout de suite, réfléchissez d’abord) : si on ajoute la règle CSS width: 400px au DIV2, quelles seront les tailles des marges à gauche et à droite (si on leur donne la valeur « auto ») ?

Testez votre réponse sous Firefox !

(Si vous avez répondu 100 pixels à gauche et 100 pixels à droite, ce n’est pas correct…)

## Étape 8 : retour aux marges

Pour rendre les dimensions du wrapper plus visibles, ajoutez-lui une bordure en trait plein de 1 pixel d’épaisseur.

Notez que, contrairement à ce qui s’est passé plus haut, cette modification ne change pas la largeur disponible pour le contenu du wrapper : celle-ci reste de 600 pixels. Expliquez pourquoi c’est le cas en vous basant sur les slides du cours.

Ajoutez ensuite la règle CSS suivante dans votre fichier HTML.

div { margin: 12px }

Pour l’exercice, ajoutez tout d’abord cette règle *avant* la règle ciblant le DIV2 et observez les modifications… puis déplacez la nouvelle règle *après* la règle ciblant le DIV2 et notez que l’ordre des règles ne change rien !

C’est parce que l’interpréteur CSS détecte que la règle de sélecteur #div2 a une cible plus précise que la règle de sélecteur div. Il lui accorde donc une importance plus grande, quel que soit l’ordre dans lequel les règles sont présentées.

Revenons à l’effet de la règle ajoutée : elle est censée placer une marge de 12 pixels tout autour de chacun des <div>. C’est effectivement ce qu’elle fait… mais observez ce qui se passe entre le DIV1 et le DIV2, ou entre le DIV2 et le DIV3.

Entre les <div> adjacents, les marges se superposent : plutôt que d’avoir un espace de 24 pixels, on a seulement un espace de 12 pixels. C’est à nouveau l’effet « collapsing margin » introduit dans l’exercice précédent.

## Étape 9 : centrer

Pour terminer cet exercice, une remarque sur le centrage d’éléments en HTML.

Pour centrer horizontalement un élément en affichage « block » (comme DIV2), il faut utiliser la propriété margin et des valeurs de auto à gauche et à droite (comme on l’a fait plus haut). Ces propriétés ciblent directement l’élément qui doit être centré.

Par contre, pour centrer horizontalement des éléments en affichage « inline » (comme le texte « DIV2 » à l’intérieur du <div>), il faut établir une règle qui cible le *parent* (ici, le <div>) et utiliser la propriété text-align: center.

Modifiez les règles CSS pour que le texte contenu dans le 2e div soit centré horizontalement.

# (Exo) Éléments flottants

Objectifs

* comprendre la propriété CSS float
* utiliser des éléments flottants
* placer plusieurs éléments flottants côte à côte

Cet exercice s’appuie sur le fichier langages.html qui présente les premiers langages de programmation créés en donnant, pour chacun d’eux, son nom, son année de création et son auteur.

## Étape 1 : reset CSS

Dans un premier temps, observez la structure du document : son contenu est divisé en deux grandes parties, une balise <main> qui contient le contenu principal, lui-même divisé en plusieurs <article> et une balise <footer> qui contient le pied de page.

Ajoutez la règle CSS suivante.

\* {

margin: 0;

padding: 0;

}

Cette règle (qu’on appelle généralement un « reset CSS ») permet de gommer certaines propriétés d’affichage par défaut établies par le navigateur (ici, les marges et les rembourrages).

Certains infographistes considèrent que ce type de « reset » est une bonne chose et d’autres ont une opinion complètement opposée. Chacun se fera son avis…

## Étape 2 : premiers styles

Comme dans l’exercice précédent, commencez par « emballer » tout le contenu dans un wrapper dont le contenu pourra occuper 700 pixels de large.

Arrangez-vous pour que ce wrapper soit centré horizontalement dans la fenêtre d’affichage, ait un fond de couleur #C29D4E, un rembourrage intérieur de 10 pixels et une marge de 20 pixels à l’extérieur (pour éviter qu’il ne colle au bord de la fenêtre). Donnez également à l’arrière-plan la couleur #253E66.

Arrangez-vous également pour que le footer soit affiché sur un fond de couleur #E3DFD7, que son texte soit centré et écrit en italique (sans modifier le HTML !). Ajoutez également une marge vers le haut de 10 pixels pour séparer le footer du reste du contenu.

## Étape 3 : chouette, des calculs !

On choisit de présenter chacun des <article> sous la forme d’un cadre et de placer 4 cadres par ligne. Sachant qu’on désire donner à chacun des cadres une bordure de 1 pixel d’épaisseur, un rembourrage de 4 pixels et une marge de 10 pixels, pourriez-vous déterminer la valeur à donner à la propriété width de chacun des cadres ?

(Un petit dessin sur un bout de papier pourrait vous aider !)

Une indication : Dans ce cas-ci, les marges qui séparent deux éléments mis côte à côte s’additionnent sans se chevaucher.

A

10px de  
marge

4px de  
padding

1px de  
bordure

A

A

A

A

<main>

On désire 4 articles par ligne.

Faites de chaque article une boîte dont le contenu aura la largeur calculée, une hauteur de 100 pixels, une bordure en trait plein de 1 pixel d’épaisseur et de couleur #253E66 et des marges de 10 pixels dans toutes les directions.

Avant de poursuivre, consultez la dernière section de ce document pour vous assurer que vous avez trouvé la largeur correcte à imposer aux articles !

## Étape 4 : placer les boîtes côte à côte

Pour l’instant, les boîtes apparaissent les unes au-dessus des autres… il existe plusieurs manières de les organiser en ligne. On en considérera trois ici et une quatrième plus tard.

**1re méthode (à éviter) : un tableau.** On pourrait modifier la partie HTML pour réaliser un tableau dont les articles occuperaient les cases. Cette méthode a deux inconvénients.

Le premier inconvénient est que la balise <table> n’est pas prévue pour ce genre de mise en page (ça ne colle pas à sa sémantique) : cette balise est censée contenir des valeurs, des données (chiffrées par exemple) formant un tableau, pas servir à la mise en page.

Le second inconvénient est que les fureteurs qui explorent le web de manière automatique (pour référencer les pages selon les thèmes abordés afin de constituer les bases de données des moteurs de recherche) pourraient ignorer le contenu d’une balise <table> en estimant qu’il ne s’agit que de détails, d’informations secondaires. Le texte des articles ne serait alors pas pris en compte dans les mots-clefs associés à la page.

**2e méthode : le mode d’affichage inline-block.** On a parlé plus tôt du mode d’affichage combiné inline-block, qui permet d’afficher un élément HTML en « inline » tout en le traitant comme un bloc (ce qui permet par exemple de choisir sa taille via CSS).

*Avant de poursuivre plus en avant avec cette méthode, effectuez une copie de votre fichier, de manière à pouvoir revenir à cette version pour aborder la 3e méthode présentée plus bas.*

Ajoutez la propriété suivante à la règle CSS ciblant les articles et observez le résultat.

display: inline-block;

## Étape 5 : solution inline-block

Du côté positif, on voit que les articles forment des blocs qui sont placés en ligne. Du côté négatif, deux observations : (1) les blocs ne sont pas alignés et (2) il n’y en a que 3 par lignes au lieu de 4 !

**Problème de l’alignement**

Le problème de l’alignement des blocs est facile à résoudre. Dans un placement en inline-block, tous les éléments sont alignés pour que les dernières lignes à l’intérieur des blocs soient à la même hauteur (ainsi, COBOL, avec son auteur en 2 lignes, commence plus haut que les autres cadres).

Pour modifier cela, il suffit d’ajouter aux articles la propriété

vertical-align: top;

qui indique que les blocs doivent être alignés pour que leur sommet arrive à la même hauteur.

**Problème d’agencement**

Le problème du « 3 blocs par ligne au lieu de 4 » est un peu plus complexe, tant à comprendre qu’à résoudre.

Ouvrez l’inspecteur de Firefox. Dans la structure HTML à gauche, ciblez l’élément « whitespace » qui sépare les deux premiers articles et observez l’espace qu’il occupe sur la page web (voir captures d’écran ci-dessous).

A screen shot of a computer program

Description automatically generated A screenshot of a computer screen

Description automatically generated

Dans le code HTML, les retours à la ligne comptent comme des espaces. Entre deux éléments en mode « bloc » (entre deux paragraphes par exemple), ces espaces sont ignorés. Mais ce n’est pas le cas entre deux balises en mode « inline » ou « inline-block » : dans ce cas-là, l’espace est affiché (au même titre qu’un espace entre 2 mots).

Ici, le navigateur affiche donc, en plus des marges autour des articles, des espaces séparateurs qui n’ont pas été pris en compte dans le calcul de la largeur des articles. C’est pour cela qu’il n’y a assez de place que pour 3 articles par ligne !

Une solution consiste à enlever ces espaces dans le code HTML. Une manière de procéder serait d’effacer tous les retours à la ligne entre les balises fermante </article> et les balises ouvrantes <article> qui suivent.

Modifiez votre fichier en conséquence (pour que les </article> et les <article> qui les suivent soient chaque fois collés sans retour à la ligne) et observez le résultat.

Une autre manière d’enlever les espaces serait de les transformer en commentaires. Au final, le code HTML ressemblerait à ceci :

</article><!—

--><article>

..

</article><!—

--><article>

## Étape 6 : solution float

La méthode d’affichage inline-block fonctionne… mais on peut ne pas apprécier les modifications qu’elle nécessite au niveau du code HTML, qui devient moins lisible.

**3e méthode : la propriété CSS float.** Revenez à la version sauvegardée précédemment et ajoutez cette fois-ci la propriété suivante dans la règle CSS ciblant les balises <article>.

float: left;

Observez le résultat…

Cette fois-ci, les blocs se placent bien à 4 par lignes… mais l’affichage sort du cadre de couleur.

Que s’est-il passé et que signifie ce float exactement ?

La propriété float indique que l’élément ciblé (le « flottant ») sort du flux normal et va se placer contre le bord indiqué (ici, le bord gauche). S’il y a plusieurs éléments flottants, les suivants viennent se placer à côté du précédent, du moins tant qu’il y a de la place sur la ligne.

L’élément-clef à bien comprendre est que le flottant est complètement retiré du flux normal. Dans ce cas-ci, tous les <article> ont été retirés du flux… ce qui fait que la balise <main> est considérée comme vide !

Pour vérifier que c’est bien le cas, ouvrez l’inspecteur de Firefox, sélectionnez l’élément <main></main> dans le cadre en bas à gauche et cliquez sur l’onglet « Box model ». Firefox indique que <main> a une taille de 700 pixels de large sur 0 pixel de haut ! La balise <main> étant vide, le <footer> remonte et vient empiéter sur le territoire des <article>.

Dans ce cas-ci, la solution est toute simple : il suffit de dire au footer qu’il doit rester à l’écart de tous les flottants et ne s’afficher qu’après eux. Cela peut se faire en ajoutant dans la règle CSS ciblant le footer la propriété suivante.

clear: both;

La propriété clear indique que la cible ne peut pas se mélanger avec des flottants. On peut aussi préciser clear:left (ou clear:right) pour ne tenir compte que des flottants à gauche (ou à droite).

Notez qu’une autre manière de résoudre le problème aurait été d’imposer une hauteur à l’élément <main>. Par exemple en lui associant la propriété CSS suivante.

height: 240px;

# (Exo) Qu’est-ce que le jeu de rôle ?

Objectifs

* réviser et affiner les connaissances utilisées plus haut
* aborder quelques nouvelles techniques de layout

Dans cet exercice, vous allez partir d’un document HTML brut (layout.html) et lui ajouter progressivement diverses règles CSS pour obtenir un résultat final similaire à la capture d’écran suivante.

A screenshot of a computer

Description automatically generated

## Étape 1 : wrapper

Commencez tout d’abord par observer la structure du document HTML. Si cela peut vous être utile, dessinez l’arborescence HTML sur un bout de papier.

Comme précédemment, emballez tout le contenu dans un wrapper que vous centrerez. La largeur du contenu de ce wrapper sera de 800 pixels ; autour de ce contenu s’étendra un rembourrage de 10 pixels (dans les 4 directions) puis une bordure noire de 4 pixels d’épaisseur en trait double. La couleur de fond du wrapper sera gray.

Plutôt que de mettre une couleur de fond pour l’arrière-plan, on va utiliser une image répétée. Pour ce faire, créez une règle CSS ciblant la balise <body> et contenant la propriété suivante.

background-image: url('charcoal.jpg');

## Étape 2 : header et footer

Ciblez désormais l’en-tête <header> et le pied de page <footer>.

Donnez-leur une couleur de fond navy, un rembourrage de 4 pixels et faites en sorte que leur contenu utilise la couleur papayawhip pour le texte et que celui-ci soit centré.

Notez que vous pouvez mettre toutes ces propriétés communes dans une seule règle de sélecteur header, footer puis ajouter ensuite deux règles séparées, une pour les particularités du header et l’autre pour les particularités du footer.

Particularités de <header> (attention, toutes celles-ci ne sont pas forcément exprimées dans la règle de sélecteur header !) :

* coins supérieurs gauche et droit arrondis avec un rayon de 10 pixels ;
* titre h1 avec des caractères de taille 300% et utilisant une police de la famille « cursive » ;
* mettre les marges du titre h1 à 0

Particularités de <footer> (même remarque) :

* coins inférieurs gauche et droit arrondis avec un rayon de 10 pixels ;
* taille de caractères de 85% ;
* tout le contenu (y compris les liens, qu’ils aient été visités ou pas) en papayawhip ;
* marges à 0 sauf celle du dessus à 8 pixels ; et mettre les marges du <p> à 0

## Étape 3 : barre de navigation

Occupez-vous ensuite de la partie <nav>. Même si celle-ci a la structure d’une liste en HTML, elle devra apparaître sous la forme d’une barre de liens.

A blue and white sign with white text

Description automatically generated

Dans un premier temps, rédigez une règle CSS qui cible les liens de navigation <a> (rendez la règle suffisamment précise pour qu’elle ne cible *que* les liens qui se trouvent dans <nav> et pas d’autres liens qui pourraient se trouver ailleurs).

Dans cette règle, imposez les propriétés suivantes :

* fonte de la famille « sans-serif », texte non-souligné (text-decoration: none) ;
* texte de couleur navy sur fond de couleur lightgray et centré ;
* largeur de contenu de 150 pixels (à ajuster par la suite) ;
* bordure en trait plein de couleur navy de 2 pixels d’épaisseur, coins arrondis avec un rayon de 6 pixels ;
* rembourrage de 1 pixel en haut et en bas, de 0 à gauche et à droite.

Si Firefox ne prend pas en compte vos indications quant à la largeur/hauteur de la boîte autour de chaque lien, n’oubliez pas que ces propriétés ne sont pas complètement prises en compte pour certains modes d’affichage. Il faut donc modifier le mode d’affichage des éléments <li> !

Faites également en sorte que les liens survolés par le curseur voient leur borduer passer en couleur papayawhip et que leur texte devienne (momentanément) en gras.

Dans une seconde règle ciblant maintenant la liste <ul> (uniquement dans <nav>), imposez à cette liste une marge de 0, un rembourrage de 0, une couleur de fond « navy » et ajoutez-lui la propriété

list-style-type: none;

qui enlève les marques (•) devant chaque item.

Il reste à faire en sorte que les liens soient affichés sur une même ligne plutôt que les uns au-dessus des autres. On pourrait utiliser la technique de l’affichage inline-block (voir exercice précédent) en supprimant les espaces entre les liens. Ici, utilisez plutôt la technique des flottants. Ciblez les éléments <li> et rendez-les flottants à gauche.

Ajoutez une marge aux liens pour les espacer : de 5px au-dessus, 8px en-dessous, 15px à gauche et 0 à droite. Les liens ne tiennent plus tous dans une seule ligne… calculez la largeur à leur imposer pour qu’ils rentrent pile-poil avec 15 pixels de marge entre eux ainsi qu’à gauche du premier et à droite du dernier (la réponse se trouve en fin de document).

Finalement, en rendant les liens flottants, ils ont été (graphiquement) extraits de la balise <ul>. Pour que le fond bleu de celle-ci s’étende jusqu’en-dessous des liens, imposez-lui une hauteur de 39 pixels.

## Étape 4 : point de modification unique

Plusieurs règles utilisent les couleurs papayawhip et navy. Si on désirait passer à une autre palette de couleurs, il faudrait donc modifier chacune de ces règles individuellement.

Tout comme en programmation, il vaut mieux adopter le principe du point de modification unique en organisant le code de manière à ce que de telles modifications puissent être effectuées via un seul changement.

Créez une nouvelle règle CSS ciblant la balise html. Dans ce cas-ci, il ne s’agira pas vraiment d’une règle de style mais plutôt de déclaration et d’initialisation de variables CSS.

html {

--couleur-de-fond: navy;

--couleur-texte: papayawhip;

}

Cette règle déclare deux variables CSS nommées --couleur-de-fond et --couleur-texte (le nom des variables CSS doit commencer par deux tirets) et les initialise aux valeur navy et papayawhip respectivement.

Comme la règle est associée au sélecteur html qui englobe tout le contenu du document, ces variables sont utilisables partout. Dans les autres règles, remplacez toutes les occurrences de navy par var(--couleur-de-fond). Faites de même pour papayawhip.

Vérifiez que le résultat n’a pas changé sous Firefox.

Testez le fonctionnement des variables en remplaçant leur valeur, par exemple en choisissant #7C0B2B pour la couleur de fond et #FFCBDD pour la couleur du texte.

## Étape 5 : les articles

Il faut désormais s’occuper du contenu, et plus particulièrement des <article>. Sur la largeur disponible de 800 pixels, on désire placer 3 articles séparés par des espaces de 10 pixels. Attention : contrairement à ce qu’on a fait jusqu’ici, on ne désire pas d’espace à gauche du premier article ni à droite du dernier article.

Rédigez une règle ciblant les articles et leur imposant une couleur de fond papayawhip (utilisez la variable CSS) une marge de 0 et un rembourrage de 3 pixels. Imposez-leur également la largeur que vous aurez calculée au plus juste.

Occupez-vous également de créer des styles pour le contenu des articles, conformément aux instructions suivantes :

* Pour les titres h1 : fonte de la famille « cursive » en taille 120% avec une bordure inférieure en pointillés de couleur navy avec 2 pixels d’épaisseurs ; ajoutez-leur également une marge à droite de 8 pixels pour éviter qu’ils n’aillent jusqu’au bord droit de l’article.
* Pour les paragraphes : fonte de famille « sans-serif » de taille 85% ;
* Pour les textes mis en emphase : gras, non-italique, couleur navy ;
* Pour les textes importants (balise <strong>) : écriture avec des mini-majuscules (utilisez la propriété font-variant:small-caps) ;
* Pour la première image : centrée verticalement et occupant 95% de la largeur disponible (attention : pour pouvoir préciser la largeur, il faut que l’élément soit affiché en mode « block ») ;
* Pour la seconde image : occupant ¼ de la largeur disponible, flottante contre le bord droit, avec des marges de 6px en haut, à gauche et en-dessous (mais 0 à droite, pour qu’elle colle au bord droit).

Notez que, pour coder ces styles, vous pouvez utiliser plusieurs règles séparées telles que

article {

… // propriétés pour article

}

article h1 {

… // propriétés pour les h1 dans un article

}

mais que le CSS permet également d’imbriquer les règles de la manière suivante.

article {

… // propriétés pour article

h1 {

… // propriétés pour les h1 dans un article

}

}

Cette syntaxe n’est pas encore reconnue pas tous les navigateurs (voir par exemple <https://caniuse.com/css-nesting>) mais Firefox la gère sans aucun problème.

## Étape 6 : trois colonnes

Maintenant que le contenu des articles est mis en page, il reste à rendre ces articles flottants.

En utilisant la propriété margin, faites en sorte qu’il y ait des espaces de 10 pixels entre les articles. Attention : si vous voyez ces espaces comme des marges de gauche, cela signifie que les articles *sauf le premier* auront cette marge ; si vous les considérez comme des marges à droite, cela signifie que les articles *sauf le dernier* auront cette marge.

Pour éviter que les articles ne collent aux éléments <header> et <nav>, ajoutez également (grâce à une nouvelle règle CSS) une marge de 8px en haut de l’élément <main>.

## Étape 7 : footer casse-pied

Comme dans l’exercice précédent, il reste un problème : le <footer> s’étend dans les articles (observez jusqu’où remonte son fond bleu).

Une solution partielle serait d’ajouter la propriété clear:both au <footer>. Testez cette solution et observez que, dans ce cas-là, le <footer> est collé au pied des articles, malgré la marge de 8px demandée.

L’explication est la même que précédemment : vu que les articles sont flottants, ils sont (graphiquement) extraits de l’élément <main>, qui est désormais vide (hauteur de 0). Cet élément <main> apparaît donc sous la forme d’une « ligne invisible » située au niveau du sommet des articles. Cette « ligne invisible » étant bien distante de plus de 8 pixels du <footer>, le navigateur n’ajoute aucun espace.

On pourrait résoudre ce problème en imposant une hauteur à l’élément <main>. Cette hauteur pourrait être calculée en observant la taille réelle du plus grand article. Effectuez cette modification, en utilisant l’inspecteur de Firefox pour obtenir la taille du plus grand des trois articles (n’oubliez pas d’ajouter l’espace occupé par le rembourrage et la bordure).

Cette solution fonctionne… mais il faut recalculer la taille chaque fois que le contenu des articles change.

Une meilleure solution consiste à ajouter, à l’intérieur du <main>, après tous les <article>, un <div> dont la seule fonction sera d’étendre l’élément <main> jusqu’en bas des articles.

Au niveau HTML, cela revient à ajouter le code suivant juste avant la balise fermante </main>

<div id="endmain"></div>

ainsi que la règle CSS suivante.

#endmain { clear: both; }

Avec cette solution, l’élément <main> est obligé de s’étendre suffisamment pour contenir ce <div> vide situé en-dessous des flottants. Vous pouvez vérifier que <main> a bien été étiré en l’observant dans l’inspecteur de Firefox.

# Solution partielle pour l’exo 3, étape 3

On dispose d’une largeur totale de 700 pixels.

Horizontalement, ces 700 pixels doivent être couverts par :

* 10 px de marge à gauche du premier article
* + 10 px de marge à droite du dernier article
* + 3 × 20 px de marge entre des articles

plus les boîtes (bordure + padding + contenu) de 4 articles.

A

A

A

A

<main>

Cela fait donc un total de 80 px de marge ; il reste alors 620 px pour les 4 boîtes, ou encore 155 px pour chacune des boîtes.

De ces 155 px, il faut encore retirer 2 pixels de bordure (1 à gauche, 1 à droite) et 8 pixels de rembourrage (4 à gauche, 4 à droite).

Il reste donc finalement 145 pixels pour le contenu de l’article.

1px de  
bordure

A

4px de  
padding

10px de  
marge