Mission 3 HTML / CSS avancé

dans le cadre du Projet 3

display et mise-en-page

Ordre d'application des sélecteurs CSS

Comme vous vous en souvenez, les sélecteurs servent à sélectionner un ensemble de balises sur lesquels on applique une règle CSS.

Plusieurs règles CSS peuvent porter sur un même élément HTML. Si ces règles peuvent coexister, elles sont toutes appliquées. Par exemple, si vous avez le code CSS suivant :

```
div    { background-color:blue; }
.skill { color:red; }
```

alors un aura les deux propriétés background-color:blue et color:red.

Mais il peut aussi arriver que ces dernières soient contradictoires. Par exemple, si vous avez le code CSS suivant, **de quelle couleur** sera le texte d'un de classe skill ?

```
.skill { color:red; }
div    { color:blue; }
```

Pour régler ces conflits, le CSS définit une notion de priorité basée sur l'emplacement des règles CSS puis sur la spécificité des sélecteurs CSS. Dans l'exemple précédent, c'est la première règle qui prévaut comme nous le verrons dans la suite.

Différents emplacements

Il y a plusieurs emplacements possibles pour déclarer du style CSS :

1. **Style externe** : (Conseillé) On peut utiliser un fichier de style externe et le lier au document HTML avec la balise <link> dans le <head> :

```
2.    <html>
```

```
<head>
  <link rel="stylesheet" type="text/css" href="css/styles.css">
</head>
<body> ... </body>
</html>
```

8. **Style interne** : (Déconseillé) On peut inclure des règles CSS directement dans le <head> à l'aide de la balise <style> :

```
<html>
  <head>
    <style type="text/css">
      p {font-size: 12pt; color: pink}
    </style>
  </head>
  <body> ... </body>
</html>
```

17. **Style inline** : (Fortement déconseillé) On peut inclure du style CSS directement dans une balise avec l'attribut style :

```
<p style="font-size: 12pt; color: fuchsia">
  Aren't style sheets wonderful?
</p>
```

Attention à ne pas confondre le style inline avec le futur display:inline.

Enfin les navigateurs appliquent un style par défaut sur les éléments. Cela permet de ne pas avoir à définir à chaque fois les styles les plus classiques. On peut observer le style par défaut dans les outils de développement de Chrome : ce sont les règles de styles associées à *user agent stylesheet*.

Pour prendre de bonnes habitudes, on préférera les styles externes comme styles.css qui permet une séparation plus claire entre les rôles du HTML (contenu avec des balises pour donner du sens) et du CSS (présentation / mise en page).

Cette séparation est indispensable et très puissante :

- Elle permet de réutiliser une présentation d'une page à l'autre. Par exemple quand *lemonde.fr* publie un nouvel article, il ne refait pas le style expressément pour ce dernier: il s'agit d'un nouveau document HTML partageant le même CSS que les articles précédents ;
- Elle permet de refaire un site Web en se concentrant sur les CSS sans (trop) toucher au HTML ;
- Elle permet de changer la présentation d'un document suivant s'il est destiné à l'impression ou à être visualisé avec un navigateur. Dans ce cas, on appliquera un style CSS différent selon le média.

Priorité des sélecteurs

Reprenons l'exemple précédent :

```
.skill { color:red; }
div    { color:blue; }
```

Afin de savoir la couleur qui sera appliquée sur les éléments , des priorités sont définies sur les sélecteurs CSS. Dans l'idée, comme le sélecteur .skill est plus spécifique que le sélecteur div, on va appliquer en priorité la règle color:red;.

La *priorité d'un sélecteur CSS* est une valeur *(a,b,c,d)* définie comme suit :

- la valeur *a* code la priorité basée sur l'emplacement de la règle. Par ordre de priorité décroissante, nous avons
    - *a*=2 pour les styles inline,
    - *a*=1 pour les styles externes et internes,

- a=0 pour le style par défaut du navigateur.

- b compte le nombre de sélecteurs d'identifiants (e.g. #id),
- c compte le nombre de sélecteurs de classes (e.g. .skill) ou pseudo-classes (:hover,:visited,…)
- d compte le nombre de sélecteurs de balises (e.g. div, span) ou de pseudo-élements (e.g. ::first-letter, ::after)
- les opérateurs de combinaison et le sélecteur universel * ne contribuent pas à la priorité.

Pour revenir à l'exemple précédent, les règles ont donc comme priorité (en supposant qu'elles sont écrites dans un fichier de style externe) :

```
div   /* -> (1,0,0,1) (un sélecteur de balise div)   */
.skill /* -> (1,0,1,0) (un sélecteur de classe skill) */
```

L'ordre de priorité

Il reste maintenant à expliquer comment on classe les priorités (a,b,c,d). L'ordre sur les priorités est défini comme l'ordre du dictionnaire (ordre*lexicographique*) :

- on regarde d'abord la "première lettre" a. Si a est strictement plus grand alors la règle CSS est plus prioritaire. En cas d'égalité,
- on regarde la "deuxième lettre" b pour déterminer la priorité. Si b est strictement plus grand (et donc a égal) alors la règle CSS est plus prioritaire. En cas d'égalité sur a et b,
- on regarde la "troisième lettre" c. En cas d'égalité,
- on regarde la "quatrième lettre" d.
- **en cas d'égalité absolue**, la règle écrite en dernier est prioritaire.

Ce mécanisme de priorité s'appelle la cascade et correspond au C de CSS (Cascading Style Sheet).

Exemple : div.skill (priorité (1,0,1,1)) est plus prioritaire que div (priorité (1,0,0,1)) car on a égalité sur *a* et *b* mais *c* est plus grand pour div.skill.

Exercice 1

Dans un fichier texte ou sur papier, écrivez les priorités des sélecteurs suivants et classez les du plus prioritaire au moins prioritaire. On supppose que toutes ces règles sont définies dans un fichier externe, donc *a*=1, et les valeurs recherchées commencent toujours par (1,...).

```
.titi span
div span
nav.titi .tata div div div div div
ul li div.skill
#id
div > a
div + a
```

Exercice 2

Quelle est la couleur du texte “Priorité CSS” dans les deux cas suivant ? Quelle règle de priorité CSS explique votre réponse ?

1. Le fichier styles.css contient p {color:blue;}, et le fichier index.html contient

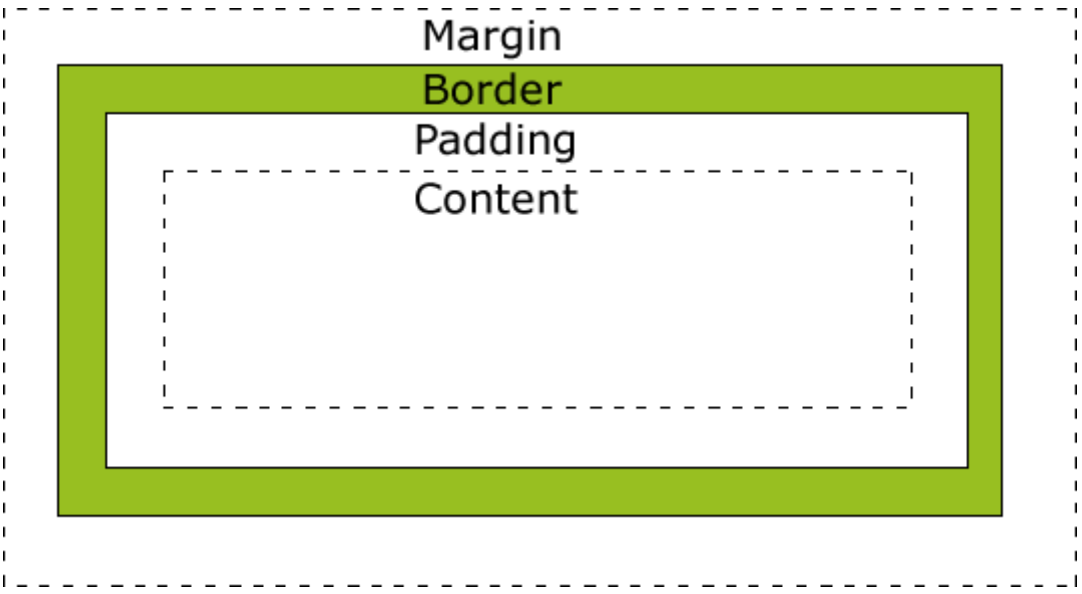
```
<html>
  <head>
    <style type="text/css">
      p {color: pink}
    </style>
    <link rel="stylesheet" type="text/css" href="css/styles.css">
  </head>
  <body><p style="color:red">Priorité CSS</p></body>
</html>
```

11. Le même fichier styles.css et le fichier index.html sans la règle inline

```
<html>
  <head>
    <style type="text/css">
      p {color: pink}
    </style>
    <link rel="stylesheet" type="text/css" href="css/styles.css">
  </head>
  <body><p>Priorité CSS</p></body>
</html>
```

La propriété display

Comme nous l'avons vu au TD précédent, à chaque balise correspond quatre boîtes (*content*, *padding*, *border* et *margin*, voir mission précédente).



La façon dont ces boîtes vont occuper l'espace est gérée par l'attribut display. Nous allons voir dans cette partie les trois valeurs principales de la propriété display.

display:block

Les éléments block sont des éléments :

- qui par défaut occupent toute la largeur,
- qui provoquent un saut de ligne avant et après son affichage (que l'on ait diminué sa largeur ou pas)
- dont on peut définir la taille en CSS via les propriétés height et width.

En pratique, on utilise des éléments de display block :

- dès que l'on veut expliciter l'agencement (layout) de certains éléments HTML d'une page. Par exemple nous voulons que l'en-tête (header) ait une hauteur de 200px, qu'un paragraphe prenne 50% de la largeur,… .
- dès qu'il est naturel de prendre toute la place par défaut (exemple un titre <h2>).

Le display:block comme style par défaut dans le navigateur explique que les blocs s'empilent verticalement comme on l'avait expliqué.

display:inline

Les éléments inline sont des éléments :

- qui prennent leur taille en fonction de leur contenu,
- qui ne provoquent pas un saut de ligne, ils sont écrits comme du texte les uns à la suite des autres
- dont on ne peut pas définir la taille en css via les propriétés height et width,

En pratique, on utilise des éléments de display inline :

1. dans du texte, pour ajouter de la sémantique sans interrompre la lecture du lecteur (mettre en exposant un nombre par <sup>, préciser l'importance d'une partie du texte par , associer un lien avec <a>),
2. lorsque l'on veut positionner des éléments à la suite.

Puisqu'associés au texte (, <a>, …), on trouve en majorité les éléments inline comme feuilles de l'arborescence du HTML.

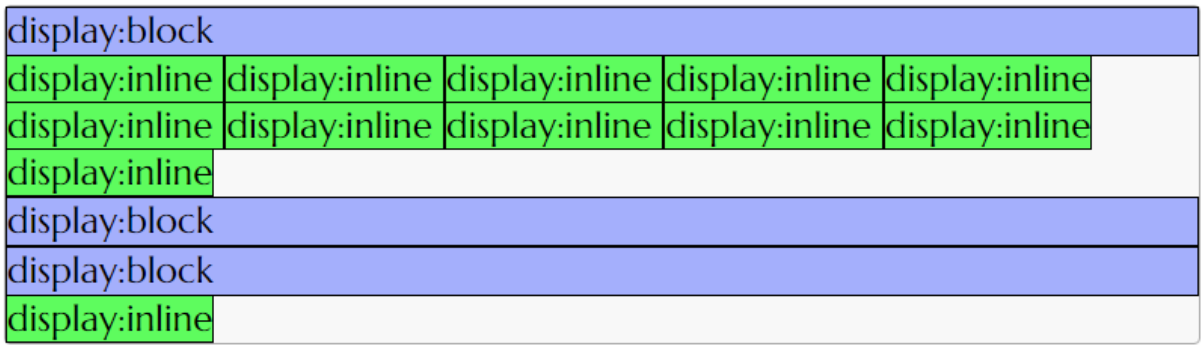
Le display:inline comme style par défaut dans le navigateur explique que les blocs se comportent comme du texte.

Exemple

Le code HTML suivant

```
<p style="display:block;">display:block</p>
<p style="display:inline;">display:inline</p>
...
<p style="display:inline;">display:inline</p>
<p style="display:block;">display:block</p>
<p style="display:block;">display:block</p>
<p style="display:inline;">display:inline</p>
```

s'affiche comme suit :



On remarque bien que les display:block prennent toute la largeur, avec un saut de ligne avant et après. Tandis que les display:inline s'affichent les uns à la suite des autres comme le texte d'un paragraphe.

Note (optionnelle) – Règle d'inclusion des éléments inline et block du point de vue du HTML et du CSS :

Inclure des éléments block dans des éléments inline n'est pas conforme en HTML, mais cela l'est du point de vue du CSS. En modifiant la propriété display d'un élément, nous pouvons

donc inclure des éléments block dans des éléments inline. Mais modifier sempiternellement le display naturel du HTML signifie que l'on n'a pas utilisé la bonne méthode (et que le code risque d'être incompréhensible). Nous nous imposons donc de respecter la règle HTML.

2. 4 -Exercices

Nous allons mettre en page le menu de navigation de notre site en mode block puis en mode inline.

Exercice 3

Dans ce premier exercice, nous allons créer un menu dans un style block.

1. Changez le menu de votre site par le suivant

```
<nav>
   <div><a href="./index.html">Accueil</a></div>
   <div><a href="./facts.html">Facts</a></div>
   <div><a href="./news.html">Actualités</a></div>
   <div><a href="./contact.html">Contact</a></div>
</nav>
```

2. Puisque <nav> est display:block par défaut (le vérifier sur Chrome si possible), il doit prendre toute la largeur. **Inspectez** donc votre <nav>pour voir si c'est le cas. Que constatez-vous ?

   Explication : En fait, un élément display:block ne prend pas toute la largeur de <body> mais de son plus proche block parent. Ici, le père de <nav> est le bloc <header> et donc <nav> prend toute la largeur de <header>.

   Le plus proche block parent s'appelle le *containing block*.

3. Puisque <nav> est de type block, nous pouvons fixer ses dimensions. Donnez-lui la largeur 75%. Que constatez-vous ?

Explication : La largeur d'une balise en display:block est relative à celle de son *containing block* (<header> ici). **Inspectez** <header> puis<nav> pour connaître leur largeurs. **Vérifiez** qu'on a bien un rapport de 75%.

4. Pour centrer le <nav> dans son parent <header>, on va lui donner des marges horizontales auto (gardez les marges verticales à 0).

   Rappel : Pour centrer un display:block dont le *containing block* est plus large, il faut mettre les marges horizontales en auto : elles se règlent alors automatiquement pour compléter la largeur manquante entre le block courant et le *containing block*. Ceci n'est pas valable pour les marges verticales.

5. Rajoutez des règles CSS pour que les fils **enfants de** <nav> aient une couleur de fond #5BBDBF,
6. Ajoutez une règle CSS pour que les éléments <a> **descendants de** <nav> aient la couleur de fond #c0d5c2.

Exercice 4

Nouvelle mise en page du menu en display:inline cette fois-ci.

1. Donnez aux enfants de <nav> le display inline.
2. Vous constatez des espaces entre les entrées du menu, ces derniers sont dû aux espaces dans le HTML, qui sont affichés lorsque les éléments sont inline.

   Une solution temporaire (en attendant display:flex) : pour supprimer les espaces, changez le code des enfant de la balise <nav> en mettant des commentaires :

```
<nav>
  <div><a href="./index.html">Accueil</a></div><!--
--><div><a href="./facts.html">Facts</a></div><!--
--><div><a href="./news.html">Actualités</a></div><!--
```

```
 --><div><a href="./contact.html">Contact</a></div>
</nav>
```

3. Donnez au <nav> une hauteur de 50px (<nav> est block donc on peut lui donner une hauteur),
4. (Optionnel) Ajoutez du padding horizontal de 10px sur les éléments <a>.
5. (Optionnel) Ajoutez à ces mêmes éléments <a> une bordure sur la gauche de 2px de style solid et de couleur noire.
6. (Optionnel) Enlevez la bordure sur le premier de ces éléments. **Astuce :** Il faut utiliser une pseudo-classe.

3 - display:none

La valeur display:none enlève complètement un élément du rendu, sans laisser d'espace à l'endroit où il aurait dû être. Nous allons coder en exercice un menu déroulant.

3-1 Menu déroulant : Partie 1 – positionnement

Dans un premier temps, nous allons juste positionner les sous-menus en dessous de leur titre. Cette exercice met en pratique les position que nous avons vu au TD précédent et que nous vous rappelons.

Position

La propriété CSS position offre de nouvelles possibilités pour le positionnement des éléments. Ses valeurs sont :

- static : comportement normal (par défaut), l'élément est inséré normalement.
- relative: le reste de la page fait comme si l'élément était positionné "normalement". De son côté, l'élément est positionné *relativement* à la position où il aurait dû être. On voit donc un espace où l'élément aurait dû être en position:static.

- absolute : le reste de la page fait comme si l'élément n'existait pas. L'élément se positionne relativement à son plus proche ancêtre **positionné**(voir ci-dessous) ou sinon à <body> (si aucun ancêtre n'est positionné).
- fixed : le reste de la page fait comme si l'élément n'existait pas. L'élément se positionne relativement à la fenêtre d'affichage ; il paraît donc *fixé*lors d'un défilement de la page.

Un élément est dit **positionné** s'il a une position autre que static (qui est la valeur par défaut). Pour indiquer le décalage de position, on utilise les propriétés top, left, right et bottom. Par exemple, les propriétés

```
position:relative;
top:20px;
left:20px;
```

vont positionner un élément 20px plus à droite et en bas qu'il n'aurait dû l'être.

Référence : Mozilla Developer Network (MDN)

Exercice 5

1. Ajoutez les sous-menus suivants aux éléments de la navigation. Ces sous-menus doivent être les petits frères des liens <a>, *c-à-d* qu'ils se placent juste après <a> tout en ayant le même père .
    - pour l'ancre "Accueil"

```
<div class="submenu">
  <div><a href="./one.html">one</a></div>
  <div><a href="./two.html">two</a></div>
  <div><a href="./three.html">three</a></div>
</div>
```

    - pour l'ancre "Contact"

```
<div class="submenu">
  <div><a href="./other.html">other</a></div>
  <div><a href="./another.html">another</a></div>
</div>
```

2. Positionnons bien ces sous-menus : nous souhaitons que l'affichage du reste de la page fasse comme si ces sous-menus n'existaient pas. De plus, nous souhaitons que les sous-menus se placent sous leur titre de menu (le parent "Accueil" ou "Contact"). Nous allons procéder en plusieurs étapes :
    i. Quelle valeur de position correspond à ce comportement des sous-menus ?
    ii. Créez la règle CSS qui affecte cette valeur de position aux balises de classe submenu avec un décalage de 50px par rapport au haut et de 0px par rapport à la gauche.
    iii. Les sous-menus ne sont pas encore bien placés car ils se positionnent par rapport à la mauvaise balise. **Quelle est cette balise** par rapport à laquelle ils se sont positionnés ? Relisez la section sur les éléments dits positionnés pour confirmer votre impression.
    iv. Nous souhaitons que nos sous-menus se placent par rapport à leur parent. Il va donc falloir rendre ce *positionné*, sans que cela ne le déplace ni que le reste de la page ne bouge. **Quelle valeur de position** donnée aux parent correspond à la description précédente ? Créez la règle CSS et le menu doit être enfin bien placé.

3-2- Menu déroulant : Partie 2 – affichage lors du survol

Nous allons maintenant nous servir de display:none pour afficher le sous-menu juste quand la souris est au dessus du titre correspondant.

Exercice 6

1. Donnez aux sous-menus la couleur de fond #aca. Puis masquez-les par défaut en CSS.

2. Réaffichez le premier sous-menu (resp. le deuxième) avec display:block lorsque la souris passe au-dessus de “accueil” (resp. “contact”) en CSS.

   Aide : En pratique, nous allons vérifier si le père du sous-menu est survolé (:hover). Nous souhaitons donc un sélecteur CSS qui sélectionne les éléments de classe submenu qui sont fils d’un survolé qui sont fils d’un <nav>.

À ce stade les sous-menus apparaissent bien lorsque l’on survole les éléments “Accueil” et “Contact”. Par contre il n’est pas possible d’entrer dans ces sous-menus. Nous consacrons toute la prochaine section au comportement display:flex qui va permettre d’améliorer l’apparence de notre menu et de remédier à ce problème d’accessibilité.

4- display:flex

La valeur de display flex correspond au layout appellé FlexBox. Appliquée à un élément, la valeur de display flex va permettre de modifier la disposition **de ses enfants**. C’est donc une différence fondamentale avec les valeurs block et inline, qui eux avaient un impact directement **sur l’élément lui-même**.

Lorsque le display est flex, on peut par exemple :

- fixer le sens de rendu des enfants,
- l’espace entre ces derniers,
- leurs centrages dans les différentes directions,
- modifier leurs ordres d’apparition, …

Nous précisons par la suite quelques-unes de ces contraintes/propriétés.

La propriété flex-direction

La propriété flex-direction permet de préciser si les enfants vont se mettre en ligne ou en colonne et dans quel ordre. Ses valeurs sont :

- row : mise en ligne dans l’ordre classique de gauche à droite ;

- row-reverse : mise en ligne dans l'ordre inversé ;
- column : mise en colonne dans l'ordre classique de haut en bas ;
- column-reverse : mise en colonne dans l'ordre inversé.

L'attribut flex-direction s'appliquant aux enfants, il rentre en conflit avec les valeurs de display block ou inline de ces derniers. L'exercice suivant va nous permettre entre autres de savoir qui surcharge l'autre en cas de conflit.

Exercice 7

1. Donnez à l'élément <nav> la valeur de display flex.
2. Changez la valeur du display des directement enfants de <nav> en block puis en inline. Que cela change-t-il ?
3. Donnez à l'élément <nav> le bloc de déclaration flex-direction:column, que cela change-t-il ?
4. Remettez la direction dans sa valeur par défaut, row.
5. Donnez une largeur fixe de 100px à tous les **descendants** de <nav> pour que les menus et les sous-menus soient de la même largeur.

La propriété align-items

La propriété align-items permet de préciser comment les enfants vont venir occuper l'espace perpendiculairement à la direction donnée parflex-direction. Dans notre cas (flex-direction:row), align-items va donc nous permettre de préciser comment occuper l'espace vertical des contenus dans le <nav>.

Ses valeurs sont :

- stretch : étirer pour prendre tout l'espace (valeur par défaut);
- center : centrer sans étirer ;
- baseline : aligne les lignes de base des éléments ;
- flex-start : alignement au début de l'axe perpendiculaire au sens de flex-direction ;
- flex-end : alignement à la fin de l'axe perpendiculaire.

Référence : Mozilla Developper Network

Exercice 8

Dans cet exercice, nous souhaitons que les titres des menus soient centrés verticalement mais qu'ils prennent toute la hauteur.

1. Pour mieux voir le comportement, nous allons donner jusqu'à la fin de cet exercice une hauteur de 200px et la couleur de fond #FF00FF à<nav> et un décalage de 200px vers le bas de sous-menu.
2. Centrez les éléments enfants de <nav> à l'aide de la propriété align-items de <nav>. Que constatez-vous sur la hauteur des et l'accessibilité des sous-menus avec la souris ? Est-ce que le est centré verticalement ?

   Note : Si rien de ne se passe, utiliser l'inspecteur du navigateur pour comprendre ce qui est centré (la marge automatique placée sur le<nav> par exemple peut expliquer des choses).

3. Utilisez maintenant la valeur stretch. Que constatez-vous sur la hauteur des et l'accessibilité des sous-menus avec la souris ? Est-ce que le est centré verticalement ?
4. Faites en sorte que les sous-menus restent accessibles et que les textes "Accueil" et "Contact" soient centrés verticalement dans la barre de navigation. Pour ceci :
   i. Faites en sorte que les prennent toute la hauteur à l'aide des 2 question précédentes.
   ii. Centrez verticalement les liens <a> au sein des : il faut pour cela que les enfants du <nav> soient flex, ce qui va permettre de centrer verticalement ses enfants <a> à l'aide de l'un des 2 comportements vues précédemment.
5. Annulez (commentez) les propriétés temporaires de la question 1.

4-3- La propriété justify-content

Cette propriété permet de préciser la façon dont les enfants vont se disposer dans la direction donnée par flex-direction.

Les valeurs possibles sont :

- flex-start : au début de l'axe donné par flex-direction ;
- flex-end : à la fin de cet axe ;
- center : centré sur cet axe ;
- space-between : les élements vont du début à la fin de l'axe en rajoutant des espaces entre les éléments ;
- space-around : les élements vont du début à la fin de l'axe en rajoutant des espaces autour des éléments (donc entre les éléments et avant le premier et après le dernier) ;

Exercice 9

1. Justifiez les blocs de titres de menus non à gauche mais à droite de <nav>, avec votre display favori. Colorez temporairement le fond de<nav> pour pouvoir vérifier que les titres se sont bien déplacés sur la droite de <nav>.

4-4-Flexbox, une valeur relativement récente

Si ces dernières possibilités offertes par flex semblent triviales voire naturelles pour le néophyte, elles représentent en pratique une avancée majeure dans le monde du CSS. Avant flex, certaines propriétés relevaient d'une expertise véritable de l'intégrateur (exemple : le centrage vertical), ou étaient même confinées dans le domaine du fantasme (les justifications, le comportement des éléments sur l'espace restant, etc.).

Aujourd'hui flexbox est bien implémenté dans les différents navigateurs. Nous ne vous présenterons donc pas d'autres valeurs de display, car elles sont devenues inutiles (display:inline-block, display:table), ni encore moins des techniques d'alignement avec des float, qui ont toujours été techniquement merdiques.

Il y a d'autres propriétés intéressantes autour de flexbox, la référence suivante est très instructive : https://css-tricks.com/snippets/css/a-guide-to-flexbox/

5-Mise en page globale

5-1- En-tête

Exercice 10

1. Placez la citation et le menu de navigation sur la même ligne (la citation sera à gauche et le menu à sa droite) toujours avec votre display favori.
2. Repositionnez le menu pour qu'il soit en bas de la balise <header> (oui, encore avec flexBox).
3. Mettez un fond blanc au <header>.
4. Enlevez la couleur de fond des liens dans le menu.

Deux colonnes

Il est temps d'avoir un layout (aménagement de l'espace) pour notre site.

Exercice 11

1. Donnez au body la width de 900px.
2. Déplacez dans le HTML la section contenant la <table> dans <aside> si cela n'est pas déjà fait.
3. Utilisez la valeur de display flex sur la balise <main> pour que ses enfants <article> et <aside> s'affichent comme deux colonnes côte à côte.
4. Fixez la largeur de <article> à 60%, et celle de <aside> à 30%. Ce dernier élément aura une marge gauche de 10%.
5. Donnez à <aside> et à <article> la couleur de fond #CCC.

Cacher ou enlever un élément du rendu

Il existe plusieurs façons de faire disparaitre de l'écran un élément HTML avec un bloc de déclaration :

- display:none
- visibility:hidden

Nous avons déjà utilisé le bloc de déclaration display:none dans le menu pour cacher un sous-menu sans que ce dernier marque la page par son abscence. Le bloc de déclaration visibility:hidden quant à lui laisse l'espace innocupé à la place de l'élément.

Voyons un usage de visibility:hidden :

Exercice 12

Nous voulons marquer visuellement le menu sous la souris par une petite puce dans les <a> du menu.

```
<span class="puce">■</span>
```

Elle se positionne à gauche du texte, elle n'est visible que lorsque la souris survole le contenant. Dans le cas contraire l'espace reste occupé (pour ne pas faire un effet de flicker/tremblement au survol du menu)

1. Ajoutez cette puce devant le texte des liens <a>,
2. Ajoutez visibility:hidden à l'élement de classe puce et la valeur visible sur le survole de la souris sur le parent.
3. Supprimez la couleur de fond sur les balises <a> pour plus de lisibilité.
4. (Optionnel) Styliser les sous-menus.

Fini !