

# Valider la qualité de son site

# Table des matières

<b>I. Contexte</b>	<b>3</b>
<b>II. Qualité du code HTML</b>	<b>3</b>
<b>III. Exercice : Appliquez la notion</b>	<b>6</b>
<b>IV. Qualité du code CSS</b>	<b>7</b>
<b>V. Exercice : Appliquez la notion</b>	<b>11</b>
<b>VI. Les normes de développement</b>	<b>14</b>
<b>VII. Exercice : Appliquez la notion</b>	<b>17</b>
<b>VIII. Les outils pour améliorer la qualité</b>	<b>18</b>
<b>IX. Exercice : Appliquez la notion</b>	<b>21</b>
<b>X. Auto-évaluation</b>	<b>23</b>
A. Exercice final .....	23
B. Exercice : Défi .....	25
<b>Solutions des exercices</b>	<b>28</b>

## I. Contexte

**Durée :** 1 h 15

**Environnement de travail :** VSCode

**Pré-requis :** Bases de HTML, CSS

### Contexte

Les langages du Web ont une particularité : ils sont relativement permissifs. Dans d'autres langages compilés, comme le C par exemple, la moindre erreur de syntaxe peut causer des erreurs dans la compilation et l'échec critique dans la construction de l'application.

Sur le Web, le navigateur va généralement pouvoir passer outre les erreurs qu'il va rencontrer pour afficher tout de même à l'écran un résultat, parfois en interprétant ce que le développeur a vraiment voulu indiquer. Cette permissivité, d'apparence bénéfique, est en réalité problématique.

En effet, ces langages semblent plus faciles à prendre en main dans un premier temps, mais il est possible que des erreurs dans le code soient masquées et ne ressurgissent que beaucoup plus tard. La faute à du code qui serait d'apparence fonctionnel, mais qui cacherait des erreurs de syntaxe.

Il convient, pour se prémunir de ces erreurs, de respecter les normes de développement et au besoin de se munir d'outils pour nous aider à analyser les violations de syntaxe et à nous faire respecter une nomenclature plus stricte.

## II. Qualité du code HTML

### Objectifs

- Voir un ensemble de mauvaises pratiques HTML et savoir comment les contourner
- Ne pas tomber dans le piège de la balise `table` pour gérer le layout de la page
- Comprendre que le HTML est un langage permissif et qu'il est nécessaire de s'imposer des règles pour produire du meilleur code

### Mise en situation

Il est possible d'écrire du code considéré comme non-valide, mais qui va pourtant fonctionner dans certains contextes.

Par exemple, écrire du code HTML en dehors de la balise `<html>` est possible, mais ce n'est pas considéré comme du code valide.

### Méthode

#### Respecter la structure

Il est nécessaire d'utiliser correctement les balises `<html>`, `<head>` et `<body>` pour que le navigateur puisse correctement prendre en charge et afficher le code qui lui est fourni.

Comme nous avons déjà pu le voir, certaines informations essentielles sont déclarées dans la balise `<head>`, par exemple l'encodage utilisé. Écrire du code sans passer par la structure valide du HTML ouvre le risque que le navigateur ne comprenne pas exactement ce qu'on souhaite réaliser.

De la même manière qu'il faut respecter la structure de base HTML, il est nécessaire de fournir un `<!DOCTYPE>` correct et à jour. Il existe plusieurs `<!DOCTYPE>` correspondant aux différentes versions de HTML.

```
1 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

À ce jour, le `<!DOCTYPE>` le plus souvent utilisé est HTML5. Il s'agit du plus récent et permet d'écrire du code HTML5 valide.

```
1 <!DOCTYPE html>
```

### Fondamental Fermer les balises dans le bon ordre

De la même manière, le code suivant n'est pas considéré comme valide. En effet, les balises doivent être fermées dans l'ordre dans lequel elles ont été ouvertes.

À l'écran, la première version fonctionne et affichera le texte, mais pourrait poser des problèmes de mise en page, en plus d'être invalide.

### Exemple

```
1 <div><p>Un paragraphe</div></p>
```

L'ordre à respecter est le suivant :

```
1 <div><p>Un paragraphe</p></div>
```

### Complément Le cas des listes

De la même manière, certaines balises ne peuvent contenir que des balises spécifiques, au risque, là encore, de causer des problèmes de mise en page. Par exemple, la balise `<ul>` ne peut contenir que des balises `<li></li>`.

Ici, la `<div>`, qui a été placée pour regrouper deux `<li>`, n'est pas du code valide et risque de briser la mise en page.

### Exemple

```
1 <ul>
2   <div>
3     <li>Chat</li>
4     <li>Chien</li>
5   </div>
6   <li>Lapin</li>
7   <li>Hibou</li>
8 </ul>
```

### Méthode Ne pas détourner l'utilisation des balises

Une autre forme de mauvaise pratique est d'utiliser des balises HTML de manière correcte, mais pour en faire une utilisation détournée. L'exemple qui vient tout de suite à l'esprit est la balise `<table>`.

Comme nous l'avons vu, la balise `<table>`, et ses balises enfants `<tr>` et `<td>` notamment, servent à afficher des données structurées sous la forme de tableaux.

Ce n'est plus tellement d'actualité aujourd'hui, mais les limitations et la complexité de CSS il y a une dizaine d'années a poussé certains développeurs à se tourner vers la balise `<table>` pour la mise en place du layout de leurs pages.

En effet, il est possible de mettre en place un tableau qui prend une grande partie de l'écran et, en masquant les bordures, de placer des éléments côte à côte, là où la bonne approche serait d'utiliser `float`, `flexbox` ou `CSS grid`, qui sont aujourd'hui des outils dédiés au positionnement d'éléments dans la page.

Cette manière de faire est à proscrire, car elle est considérée comme une très mauvaise pratique.

En règle générale, il convient d'utiliser la bonne balise pour son cas d'utilisation. HTML propose de très nombreuses balises sémantiques parfaitement adaptées à une situation. Détourner ces balises pour les utiliser de manière dérivée est souvent mal-perçu.

À l'inverse, utiliser des balises génériques pour produire le comportement d'une balise sémantique est, encore une fois, une mauvaise pratique.

### Exemple Utiliser la sémantique

Pour représenter un site web de type blog, il vaut mieux utiliser la structure suivante :

```
1 <body>
2   <header></header>
3   <main>
4     <article></article>
5   </main>
6   <aside>
7     <section></section>
8   </aside>
9   <footer></footer>
10 </body>
```

Plutôt que le code suivant, qui sur le papier revient au même, mais est beaucoup plus générique et moins sémantique :

```
1 <body>
2   <div></div>
3   <div>
4     <div></div>
5   </div>
6   <div>
7     <div></div>
8   </div>
9   <div></div>
10 </body>
```

Ce code est également beaucoup plus compliqué à modifier, l'usage de trop de `<div>` risque de perdre le développeur qui prendra en charge votre code.

### Complément L'utilisation des identifiants

Un dernier exemple de mauvaise pratique en HTML est l'usage majoritaire des `id` au détriment des `class`, qui permettent une plus grande réutilisabilité de votre code CSS.

De plus, multiplier les `id` augmente le risque de duplication involontaire d'un `id`, ce qui peut poser des problèmes de conflits avec le code CSS ou JavaScript qui s'appuierait sur ces identifiants.

En JavaScript, justement, deux `id` avec le même nom provoqueront un `warning` dans la console.

**Syntaxe**    **À retenir**

- Il est nécessaire de préciser un DOCTYPE, si possible HTML5.
- En HTML, il faut faire attention à bien fermer les balises dans l'ordre inverse de l'ouverture.
- Attention à ne pas écrire n'importe quelle balise dans une liste ou un tableau.
- Il ne faut pas détourner les balises de leurs usages.
- Pour le CSS, les classes sont à privilégier par rapport aux identifiants.

### III. Exercice : Appliquez la notion

#### Question

[solution n°1 p.29]

- Modifiez la structure HTML suivante afin d'utiliser les éléments sémantiques nécessaires plutôt qu'un tableau (aidez-vous des classes définies). Ne vous préoccupez pas d'y ajouter du CSS, faites simplement en sorte d'avoir une structure sémantiquement correcte.
- Ajoutez-y également le `doctype` correspondant à une déclaration HTML5.
- Veillez à ce que l'ensemble des balises soient fermées dans le bon ordre.

```

1 <html lang="fr">
2 <head>
3   <meta charset="UTF-8">
4   <meta name="viewport" content="width=device-width, initial-scale=1.0">
5   <title>Document</title>
6 </head>
7 <body>
8 <table>
9   <tr>
10    <td colspan="2"><h1>Mon site personnel</h1></td>
11  </tr>
12  <tr>
13    <td>
14      <div class="nav">
15        <h2>Menu</h2>
16        <ul>
17          <div>
18            <li><a href="#">Mes réalisations</a></li>
19            <li><a href="#">A propos</a></li>
20          </div>
21          <li><a href="#">Contact</a></li>
22          <li><a href="#">Site partenaire</a></li>
23        </ul>
24      </div>
25    </td>
26    <td class="main">
27      <div class="article">
28        <h3>Blog post 1</h3>
29        <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur nec ex
bibendum, viverra metus in, eleifend nibh. Fusce eu risus sem. Nullam lacinia aliquam elit, et
ullamcorper quam elementum eu.</p>
30      </div></td>
31      <div class="article">
32        <h3>Blog post 2</h3>
33        <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur nec ex
bibendum, viverra metus in, eleifend nibh. Fusce eu risus sem. Nullam lacinia aliquam elit, et
ullamcorper quam elementum eu.</p>
34      </div>

```

```

35         <div class="article">
36             <h3>Blog post 3</h3>
37             <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur nec ex
bibendum, viverra metus in, eleifend nibh. Fusce eu risus sem. Nullam lacinia aliquam elit, et
ullamcorper quam elementum eu.</p>
38         </div>
39     </td>
40 </tr>
41 <tr>
42     <td colspan="2" class="footer">Tous droits réservés</td>
43 </tr>
44 </table>
45 </body>
46 </html>
47

```

## IV. Qualité du code CSS

### Objectifs

- Voir les différentes erreurs de syntaxe qu'il est possible de faire en CSS
- Comprendre la notion de spécificité
- Penser ses styles pour être atomiques

### Mise en situation

De la même manière qu'il existe de nombreux moyens d'écrire du code invalide en HTML, CSS est également très permissif.

#### Méthode Vérifier les propriétés utilisées

Une erreur commune est d'utiliser une propriété CSS inexistante, ou une valeur de propriété qui n'est pas valide.

```

1 position: block; /* Code invalide, "block" n'est pas une valeur correcte pour position */
2 position: relative;
3
4 position-relative: true; /* Code invalide, la propriété "position-relative" n'existe pas */

```

Le site ne va pas forcément s'arrêter de fonctionner si on utilise des propriétés CSS inexistantes, mais ça pourrait fortement impacter le reste du code CSS, sans que vous puissiez le voir. La console de développeur de Google Chrome indique d'ailleurs un message d'avertissement quand il détecte une propriété invalide, avec des informations supplémentaires sur le survol de ces propriétés.

```

element.style {
  ⚠ position: block;
  ⚠ position-relative: true;
  min-height: 100%;
  top: 0px;
}

```

## Méthode Éviter les répétitions

Certaines pratiques ne sont pas considérées comme des erreurs au niveau du langage, mais sont en revanche considérées comme des mauvaises pratiques.

Répéter du CSS dans plusieurs sélecteurs en fait partie, tel que :

```
1 header {
2   font-size: 18px;
3   color: lightyellow;
4 }
5
6 main {
7   font-size: 18px;
8   margin: 10px;
9 }
10
11 footer {
12   font-size: 18px;
13   background: black;
14 }
```

Ici, il faudrait préférer :

```
1 body {
2   font-size: 18px;
3 }
4
5 header {
6   color: lightyellow;
7 }
8
9 main {
10   margin: 10px;
11 }
12
13 footer {
14   background: black;
15 }
```

Ou alors grouper les propriétés au sein d'un même sélecteur, pour ne pas dupliquer les règles communes à tous les éléments :

```
1 header,
2 main,
3 footer {
4   font-size: 18px;
5 }
6
7 header {
8   color: lightyellow;
9 }
10
11 main {
12   margin: 10px;
13 }
14
15 footer {
16   background: black;
17 }
```



**Remarque** Respecter la spécificité

Un autre élément qui, s'il est mal compris et géré par le développeur, peut devenir problématique dans le cadre d'une application volumineuse : la spécificité.

En CSS, quand deux règles entrent en conflit sur un élément, le langage calcule la priorité d'application des règles, en observant notamment la spécificité du sélecteur.

La spécificité est un concept assez simple à comprendre, mais assez complexe à maîtriser. En ajoutant une classe à un élément HTML et en ciblant cet élément HTML, le sélecteur sera plus spécifique qu'un sélecteur sans la classe, et la règle sera donc appliquée.

```
1 <body>
2   <main class="content">
3     <span class="contact-email">
4       test@example.com
5     </span>
6   </main>
7 </body>
```

```
1 span {
2   color: blue;
3 }
4
5 /* Ce sélecteur est plus spécifique que celui du dessus, l'adresse email sera donc en rouge */
6 span.contact-email {
7   color: red;
8 }
```

Il serait alors tentant d'être très spécifique, pour être certain que le style souhaité s'applique correctement. Pour reprendre notre exemple :

```
1 /* Ce sélecteur est très spécifique, sans doute trop... */
2 body > main.content > span.contact-email {
3   color: red;
4 }
```

Ici, il y a trop de spécificité dans ce sélecteur. Effectivement, il y a de bonnes chances pour que la couleur du texte soit rouge, mais il ne faut pas perdre de vue le fonctionnement en cascade de CSS.

Avec trop de spécificité, on tue la cascade : il devient très compliqué de changer ce style et celui-ci devient ciblé sur un seul élément de la page. L'idéal serait de pouvoir réutiliser ce style `color: red` ailleurs.

La spécificité ne fonctionne que dans un seul sens, il n'est pas possible de revenir en arrière. Pour passer par dessus, il faut être encore plus spécifique. Il faut donc l'appliquer contextuellement, là où il y en a besoin et pas trop lourdement dès le début du projet.

**Attention** Éviter le mot-clé `!important`

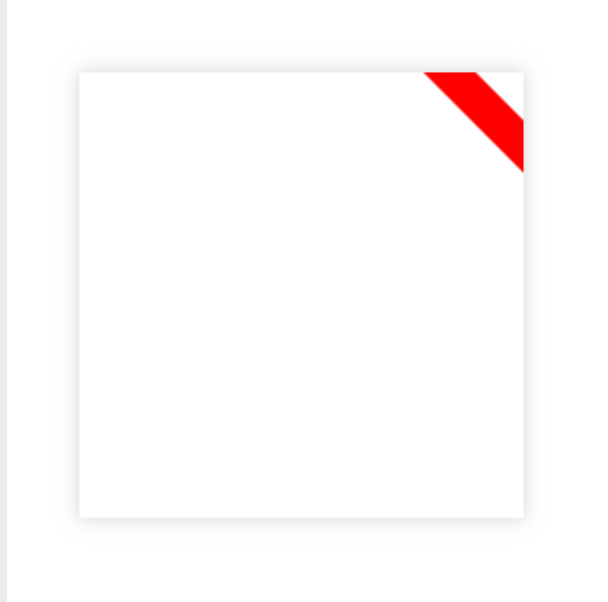
De la même manière, l'utilisation du mot-clé `!important` est à proscrire dans le code CSS. Il force la priorité d'une règle en ignorant tout le contexte de la cascade, c'est un *anti-pattern* en CSS.

**Complément** Ne pas être trop spécifique

Un autre piège à éviter est un sélecteur CSS qui aurait des styles trop précis, qui ferait trop de choses. Pour respecter la structure de CSS et pour améliorer la maintenabilité du code, il est préférable d'utiliser des styles plus atomiques et réutilisables. C'est d'ailleurs la philosophie du *utility-first*.

**Exemple** Afficher un ruban

Dans cet exemple, on affiche une boîte avec un effet de ruban dans le coin supérieur droit. On pourrait dire que ce sélecteur est trop précis : on voudrait pouvoir par exemple réutiliser l'effet de ruban sur un autre élément que notre boîte. Dans l'état de ce sélecteur, c'est impossible, il faudrait dupliquer le code du ruban.



```
1 .card-with-ribbon {  
2   position: relative;  
3   background: white;  
4   height: 200px;  
5   width: 200px;  
6   padding: 20px;  
7   overflow: hidden;  
8   box-shadow: 0 0 10px lightgray;  
9 }  
10  
11 .card-with-ribbon::after {  
12   content: '';  
13   position: absolute;  
14   top: -10px;  
15   right: -30px;  
16   background: red;  
17   height: 20px;  
18   width: 140px;  
19   transform: rotate(45deg);  
20 }
```

En revanche, si on modifie le code pour séparer les deux effets, d'un côté l'aspect `card`, et de l'autre le ruban, on peut toujours appliquer le ruban à notre `card`, mais il est maintenant possible d'appliquer le ruban à d'autres éléments également.

```
1 .card {  
2   position: relative;  
3   background: white;  
4   height: 200px;  
5   width: 200px;  
6   padding: 20px;  
7   overflow: hidden;  
8   box-shadow: 0 0 10px lightgray;
```

```
9 }
10
11 .with-ribbon::after {
12   content: '';
13   position: absolute;
14   top: -10px;
15   right: -30px;
16   background: red;
17   height: 20px;
18   width: 140px;
19   transform: rotate(45deg);
20 }
```

Ce code permet d'obtenir le même résultat, mais les deux styles sont maintenant découplés et peuvent fonctionner indépendamment l'un de l'autre.

#### **Syntaxe**    **À retenir**

- Il faut veiller à utiliser les noms corrects des propriétés et des valeurs.
- Il faut essayer de ne pas dupliquer des styles en les remontant sur un parent commun ou en créant un sélecteur CSS multiple.
- Il faut veiller à ne pas être trop spécifique dans les sélecteurs CSS.
- Il faut penser ses styles pour qu'ils soient atomiques et réutilisables.

## V. Exercice : Appliquez la notion

Vous disposez du fichier CSS suivant :

```
1 body {
2   font-family: 'Lato', sans-serif;
3   margin: 0;
4 }
5
6 header {
7   text-align: center;
8   font-style: italique;
9   font-size: 14px;
10 }
11
12 h1 {
13   font-weight: bold;
14 }
15
16 h2 {
17   font-style: italique;
18   font-weight: bold;
19 }
20
21 nav {
22   padding: 10px;
23   background-color: lightgrey;
24   margin-bottom: 10px;
25   font-size: 14px;
26 }
```

```

27
28 nav li {
29 }
30
31 main {
32 display: block-inline;
33 width: 74%;
34 font-size: 14px;
35 }
36
37 aside {
38 display: block-inline;
39 width: 25%;
40 vertical-align: top;
41 font-size: 14px;
42 }
43
44 section {
45 padding: 20px;
46 font-size: 14px;
47 }
48
49 article {
50 padding: 20px;
51 font-size: 14px;
52 }
53
54 footer {
55 text-align: center;
56 font-style: italic;
57 background-color: #005cbf;
58 color: white;
59 margin-top: 20px;
60 padding: 10px;
61 font-size: 14px;
62 }

```

## Question

[solution n°2 p.29]

- Vérifiez que l'ensemble des propriétés utilisées sont bien des propriétés existantes.
- Modifiez ce fichier afin d'éviter les répétitions en factorisant les propriétés qui peuvent l'être.

Vous pouvez utiliser le fichier HTML suivant pour vous aider :

```

1 <!DOCTYPE html>
2 <html lang="fr">
3   <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <title>Document</title>
7     <link href="style.css" rel="stylesheet">
8   </head>
9   <body>
10    <header>
11      <h1>Mon site personnel</h1>
12    </header>
13    <div class="content">
14      <aside>
15        <nav>

```

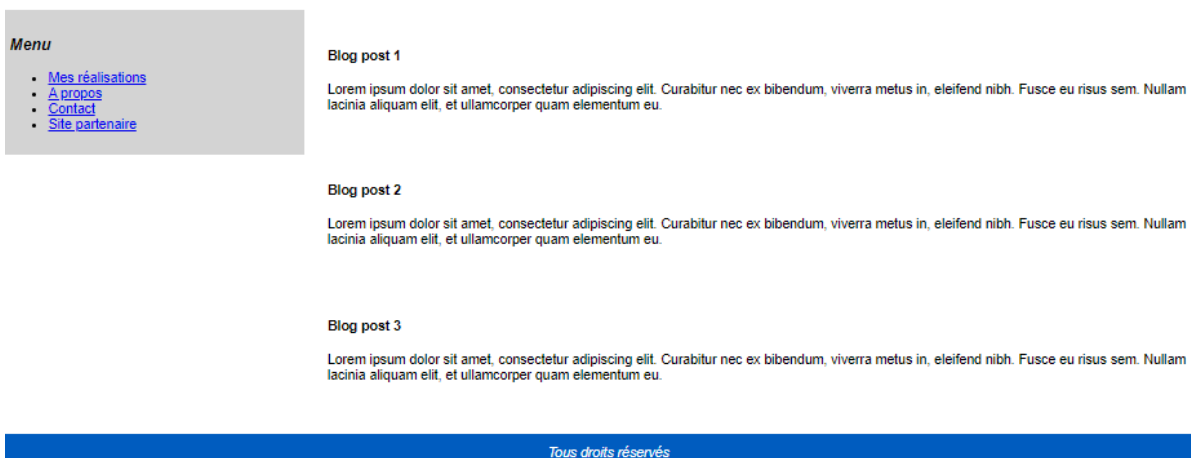
```

16         <h2>Menu</h2>
17         <ul>
18             <li><a href="#">Mes réalisations</a></li>
19             <li><a href="#">A propos</a></li>
20             <li><a href="#">Contact</a></li>
21             <li><a href="#">Site partenaire</a></li>
22         </ul>
23     </nav>
24 </aside>
25 <main>
26     <article>
27         <h3>Blog post 1</h3>
28         <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur nec
29         ex bibendum, viverra metus in,
30         ullamcorper quam elementum eu.</p>
31     </article>
32     <article>
33         <h3>Blog post 2</h3>
34         <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur nec
35         ex bibendum, viverra metus in,
36         ullamcorper quam elementum eu.</p>
37     </article>
38     <article>
39         <h3>Blog post 3</h3>
40         <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur nec
41         ex bibendum, viverra metus in,
42         ullamcorper quam elementum eu.</p>
43     </article>
44 </main>
45 </div>
46 <footer>Tous droits réservés</footer>
47 </body>
48 </html>

```

Voici l'affichage que vous devriez obtenir si le style est correctement appliqué :

## Mon site personnel



## VI. Les normes de développement

### Objectifs

- Observer quelques éléments de syntaxe à améliorer
- Comprendre les différences de point de vue sur certaines pratiques de développeur

### Mise en situation

Pour faire face aux problèmes engendrés par la permissivité des langages web, il peut être bon de s'imposer des règles supplémentaires par rapport à ce que le langage exige au minimum pour son fonctionnement. Ces règles sont souvent regroupées dans ce qu'on appelle des **normes de développement**, ou des *code styles* en anglais.

Ces code styles sont souvent très suivis par les développeurs, ce qui présente un autre avantage : si tout le monde développe avec les mêmes règles arbitraires, le code est beaucoup plus homogène. Il est donc plus facile à comprendre par les nouveaux développeurs dans un projet, puisqu'ils retrouvent la syntaxe qu'ils ont déjà l'habitude d'utiliser.

#### Méthode Se fixer une limite de caractères

Cela aide également à respecter une autre règle, qu'essaient d'appliquer les développeurs : ne pas écrire de lignes de code qui dépassent un certain nombre de caractères. Il convient généralement d'éviter de dépasser 100 caractères, mais certains développeurs préfèrent limiter à 80 caractères.

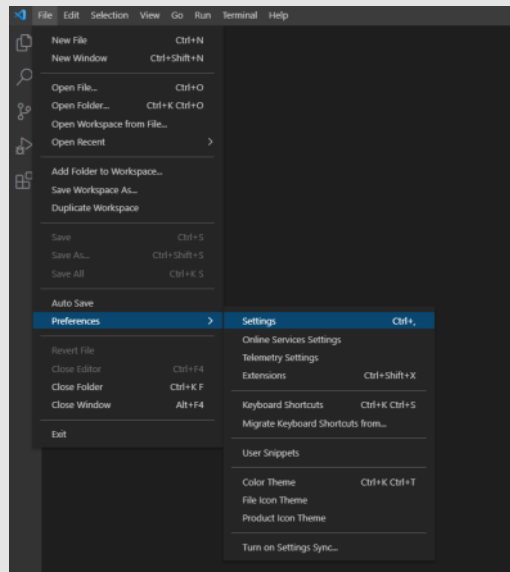
VSCode propose un paramètre à rajouter dans la configuration pour ajouter une ligne verticale à un certain caractère, comme un indicateur pour ne pas dépasser.

```
1 "editor.rulers": [
2   {
3     "column": 100,
4     "color": "#0984e3"
5   }
6 ]
```

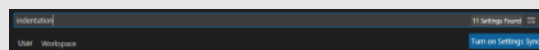
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link href="https://unpkg.com/tailwindcss@1.0/dist/tailwind.min.css" rel="stylesheet">
  <link rel="stylesheet" href="style.css">
  <title>Voyage</title>
</head>
<body class="bg-yellow-200 w-screen h-screen flex items-center justify-center">
  <div class="relative bg-white w-2/3 md:w-3/6 lg:w-2/6 xl:w-2/6 p-10">
    <h3 class="text-2xl font-bold font-sans text-gray-900">Idée voyage</h3>
    <div class="w-full flex flex-col items-center">
      
      <h2 class="text-2xl font-bold font-sans uppercase">Découvrez</h2>
      <h1 class="text-4xl font-bold font-sans text-gray-900">Oahu - Hawaï</h1>
    </div>
    <button
      type="button"
      class="bg-indigo-600 text-white text-2xl font-bold font-sans py-4 rounded-lg">
      2590 €
    </button>
  </div>
</body>
</html>
```

L'indentation est également un sujet très important à respecter : certains développeurs préfèrent indenter avec quatre espaces.

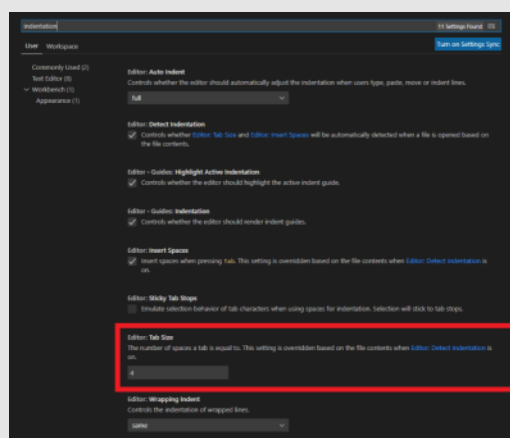
Il vous faut aller dans “File” -> “Preferences” -> “Settings”. Vous pouvez aussi utiliser le raccourci Ctrl+,.



Ensuite, écrire “indentation” dans la barre de recherche.



Puis changer l’indentation comme souhaité, ici l’indentation est à 4 :



### Exemple

```

1 <!-- 2 espaces d'indentation -->
2
3 <html lang="en">
4   <head>
5     <meta charset="UTF-8">
6     <meta name="viewport" content="width=device-width, initial-scale=1.0">
7     <title>Document</title>
8   </head>
9   <body>
10    <h1>Un titre</h1>
11  </body>
12 </html>

```

```

1 <!-- 4 espaces d'indentation -->
2
3 <html lang="en">
4   <head>
5     <meta charset="UTF-8">
6     <meta name="viewport" content="width=device-width, initial-scale=1.0">
7     <title>Document</title>
8   </head>
9   <body>
10    <h1>Un titre</h1>
11  </body>
12 </html>

```

### Méthode Améliorer la lisibilité du code

Lorsqu'il y a de nombreux attributs sur un élément HTML, passer chaque attribut à la ligne est considéré comme une bonne pratique. Par exemple :

```

1 <form class="container form card lef-side" id="contact-form" action="/contact.php"
  method="POST"></form>
2
3 <form
4   class="container form card lef-side"
5   id="contact-form"
6   action="/contact.php"
7   method="POST"
8 >
9 </form>

```

### Méthode Les commentaires

Enfin, parmi les types de commentaires qu'il est possible d'écrire, certains développeurs favorisent les commentaires concis, en une ligne :

```

1 <!-- Mon commentaire HTML -->
2
1 /* Mon commentaire CSS */

```

D'autres préfèrent systématiquement ajouter des *block-comments* pour plus de visibilité :

```

1 <!--
2   Mon commentaire
3 -->
4
1 /*
2   Mon commentaire CSS
3 */

```

Le respect de ces règles n'est pas obligatoire pour que le code fonctionne, mais c'est une bonne habitude à prendre que de suivre un code style avec une grosse communauté et de s'y tenir.

### Syntaxe À retenir

- Il y a des règles de syntaxe qu'il convient de s'imposer : passer les attributs sur une ligne séparée lorsqu'il y en a beaucoup, ne pas écrire une ligne de code trop longue, ou encore respecter l'indentation établie, par exemple.
- Il faut choisir un code style et essayer de l'appliquer du mieux possible.



## VII. Exercice : Appliquez la notion

On dispose du code HTML suivant :

```

1 <!DOCTYPE html>
2 <html lang="fr">
3 <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <title>Document</title>
7     <link rel="stylesheet" href="style_correct.css">
8 </head>
9 <body>
10 <header>
11     <h1>Contactez-moi</h1>
12 </header>
13
14 <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur nec ex bibendum, viverra
    metus in, eleifend nibh. Fusce eu risus sem. Nullam lacinia aliquam elit, et ullamcorper quam
    elementum eu.</p>
15
16 <form>
17 <div class="form-group row">
18     <label for="email" class="col-sm-2 col-form-label">Email</label>
19     <div class="col-sm-10">
20         <input type="email" readonly class="form-control-plaintext" id="email"
21             placeholder="email@example.com" data-type="email" data-invalid-message="Merci de saisir une
22             adresse e-mail correct">
23     </div>
24 </div>
25 <div class="form-group row">
26     <label for="name" class="col-sm-2 col-form-label">Name</label>
27     <div class="col-sm-10">
28         <input type="text" readonly class="form-control-plaintext" id="name" placeholder="John Doe"
29             data-type="text">
30     </div>
31 </div>
32 <div class="form-group row">
33     <label for="subject" class="col-sm-2 col-form-label">Subject</label>
34     <div class="col-sm-10">
35         <input type="text" readonly class="form-control-plaintext" id="subject" placeholder="Sujet de
36             votre message"
37             data-type="text">
38     </div>
39 </div>
40
41 <div class="form-group row">
42     <label for="message">Textarea</label>
43     <textarea class="form-control" id="message" placeholder="Votre message" data-type="text"
44         required></textarea>
45 </div>
46
47 <button class="btn btn-primary" type="submit">Envoyer le message</button>
48 </form>
49
50 </body>
51 </html>
52

```

Dans notre équipe, on considère notamment que :

- L'indentation du code doit systématiquement être respectée avec une tabulation
- Lorsqu'un élément dispose de 3 attributs ou plus, chaque attribut sera affiché sur une seule ligne
- Une ligne ne peut dépasser 120 caractères

### Question

[solution n°3 p.31]

Adaptez le code ci-dessus afin de respecter les lignes de conduite évoquées.

## VIII. Les outils pour améliorer la qualité

### Objectifs

- Analyser la qualité de son code avec le validateur W3C
- Utiliser un `linter` directement depuis la ligne de commande pour le HTML et le CSS

### Mise en situation

Certains outils permettent de valider la qualité du code JavaScript, CSS ou HTML.

#### Méthode Le W3C

Le premier qui vient à l'esprit lorsque l'on parle de validation est l'outil mis à disposition par le W3C. Il est accessible sous la forme d'un site Internet disponible à cette adresse<sup>1</sup>.

L'outil propose de tester la qualité du code d'un site, soit via son URL, soit par l'envoi d'un fichier, soit par la saisie directe de code dans un champ de texte.

<sup>1</sup> <http://validator.w3.org/>

Le code suivant passe au validateur sans générer aucune erreur :

```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <title>Document</title>
7   </head>
8   <body>
9     <h1>Un titre</h1>
10  </body>
11 </html>
```

## Nu Html Checker

This tool is an ongoing experiment in better HTML checking, and its behavior remains subject to change

### Showing results for contents of text-input area

Checker Input

Show ☒ source ☐ outline ☐ image report [Options...](#)

Check by  ☐ css

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
  </head>
  <body>
    <h1>Un titre</h1>
  </body>
</html>
```

[Check](#)

Use the Message Filtering button below to hide/show particular messages, and to see total counts of errors and warnings.

[Message Filtering](#)

**Document checking completed. No errors or warnings to show.**

### Source

```
1. <!DOCTYPE html>↵
2. <html lang="en">↵
3.   <head>↵
4.     <meta charset="UTF-8">↵
5.     <meta name="viewport" content="width=device-width, initial-scale=1.0">↵
6.     <title>Document</title>↵
7.   </head>↵
8.   <body>↵
9.     <h1>Un titre</h1>↵
10.  </body>↵
11. </html>
```

Used the HTML parser.

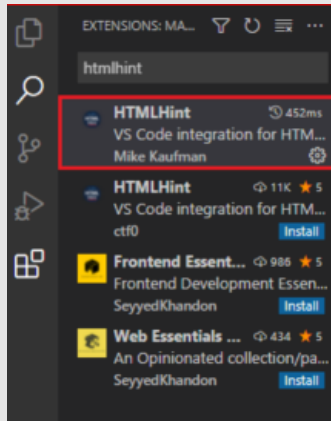
Total execution time 1 milliseconds.

## Complément

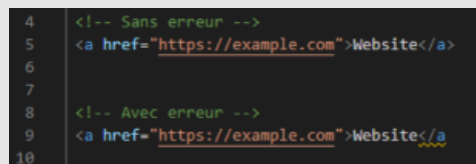
Ce validateur permet de vérifier la validité du code HTML, mais a été remplacé avec les années par des outils mieux intégrés aux outils de développement, ce qui permet de pouvoir bénéficier d'une validation sans sortir de son éditeur de code.

## Méthode HTMLHint

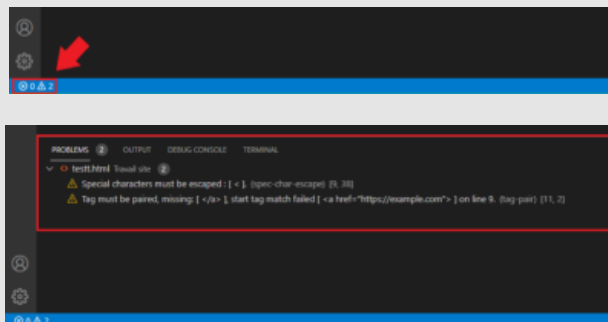
Un outil de *lint* comme HTMLHint<sup>1</sup> permet une analyse du même type que ce que propose le validateur W3C, il suffit d'installer l'extension HTMLHint sur Visual Studio Code, comme ci-dessous :



L'extension HTMLHint affiche l'erreur grâce à une petite vague de couleur jaune, comme ci-dessous :



Le détail de l'erreur s'affiche dans la console, dans la rubrique "problems".

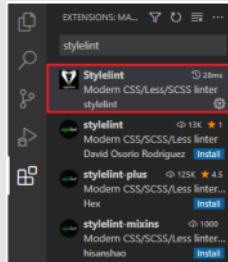


<sup>1</sup> <https://github.com/htmlhint/HTMLHint>

**Méthode** Analyser du CSS avec StyleLint

Un outil similaire existe, pour analyser cette fois-ci le code CSS : Stylelint<sup>1</sup>.

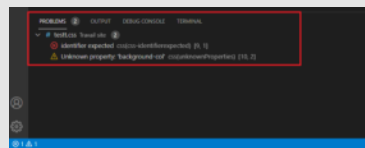
Stylelint s'installe de la même manière, via l'extension sur Visual Studio Code :



L'extension Stylelint affiche l'erreur grâce à une petite vague de couleur rouge et une decouleur jaune en fonction de l'erreur, comme ci-dessous :

```
1 /* Sans erreur */
2 .class
3 {
4   background-color: red;
5 }
6
7 /* Avec erreur */
8 .class
9 {
10  background-col: red;
11 }
```

Le détail de l'erreur s'affiche dans la console, dans la rubrique "problems".

**Syntaxe** À retenir

- Le validateur W3C permet de vérifier que le code HTML est valide.
- HTMLHint permet de faire ces vérifications également, mais il est disponible depuis la ligne de commande.
- StyleHint est un outil équivalent à HTMLHint pour analyser le code CSS.

**IX. Exercice : Appliquez la notion****Question 1**

[solution n°4 p.32]

Identifiez comment valider un code HTML directement depuis le site de W3C.

<sup>1</sup> <https://github.com/stylelint/stylelint/blob/master/docs/user-guide/get-started.md>

## Question 2

[solution n°5 p.32]

Identifiez les problèmes du code HTML suivant au moyen du validateur W3C, corrigez-les et validez le code HTML produit.

```

1 <html>
2 <head>
3   <meta charset="utf-8">
4   <meta name="viewport" content="width=device-width, initial-scale=1.0">
5   <link rel="stylesheet" href="style.css">
6 </head>
7 <body>
8 <header>
9   <h1>Recette - La pâte à crêpes</h1>
10 </header>
11 <div class="content">
12   <aside>
13     <nav>
14       <h3>Menu</h3>
15       <ul>
16         <li><a href="#">Mes dernières recettes</a></li>
17         <li><a href="#">A propos</a></li>
18         <li><a href="#">Contact</a></li>
19       </ul>
20     </nav>
21   </aside>
22   <main>
23     <section>
24       <ul>
25         <li>300g de farine</li>
26         <li>3 oeufs entiers</li>
27         <li>3 cuillères à soupe de sucre</li>
28         <li>2 cuillères à soupe d'huile</li>
29         <li>50g de beurre fondu</li>
30         <li>60cl de lait</li>
31         <li>5cl de rhum</li>
32       </ul>
33     </section>
34     <section>
35       
36       <ol>
37         <li>
38           Mettre la farine dans une terrine et former un puits.
39         </li>
40         <li>
41           Y déposer les oeufs entiers, le sucre, l'huile et le beurre.
42         </li>
43         <li>
44           Mélanger délicatement avec un fouet en ajoutant au fur et à mesure le
45           lait. La pâte ainsi obtenue
46           doit
47           avoir une consistance d'un liquide légèrement épais.
48         </li>
49         <li>
50           Parfumer de rhum.
51         </li>
52         <li>
53           Faire chauffer une poêle antiadhésive et la huiler très légèrement. Y
54           verser une louche de pâte, la

```

```

53         répartir dans la poêle puis attendre qu'elle soit cuite d'un côté avant de
54 la retourner. Cuire ainsi toutes les crêpes à feu doux.
55     </li>
56 </ol>
57 </section>
58
59 </main>
60 </div>
61
62 <footer>Tous droits réservés</footer>
63 </body>
64 </html>
65

```

## X. Auto-évaluation

### A. Exercice final

#### Exercice 1

[solution n°6 p.34]

#### Exercice

Lorsque l'on définit le `doctype` d'un site récent, il convient d'utiliser la déclaration...

- ☐ `<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">`
- ☐ `<!DOCTYPE html>`
- ☐ `<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">`

#### Exercice

Qu'importe l'ordre de fermeture des balises, la page sera toujours considérée comme sémantiquement valide.

- ☐ Vrai
- ☐ Faux

#### Exercice

On souhaite englober deux éléments `li` dans un élément parent `div`, car c'est considéré comme une syntaxe sémantiquement autorisée.

```

1 <ul>
2   <li><a href="#">Accueil</a></li>
3   <div>
4     <li><a href="#">Favoris</a></li>
5     <li><a href="#">Profil</a></li>
6   </div>
7   <li><a href="#">Déconnexion</a></li>
8 </ul>

```

- ☐ Vrai
- ☐ Faux

#### Exercice

En HTML ou CSS, si l'on souhaite agir sur la mise en page, il convient d'utiliser...

- ☐ Les propriétés CSS `display`
- ☐ Les tableaux HTML
- ☐ Les propriétés CSS `flexbox`

#### Exercice

Pour faciliter la compréhension de la sémantique d'une page, mieux vaut utiliser...

- ☐ Des identifiants
- ☐ Des classes CSS
- ☐ Des balises HTML `div`
- ☐ Des balises HTML spécifiques (`nav`, `header`...)

#### Exercice

Dans la console développeur d'un navigateur, si l'on constate l'affichage suivant, cela signifie...

```
element.style {
  position: block;
  position: relative;
  min-height: 100%;
  top: 0px;
}
```

- ☐ Que les propriétés ne sont pas utilisées
- ☐ Que les propriétés sont inexistantes

#### Exercice

On dispose du code HTML suivant. Quel sélecteur CSS faudrait-il privilégier pour cibler le nom de la personne ?

```
1 <body>
2   <main class="content">
3     <span class="contact-name">
4       John Doe
5     </span>
6   </main>
7 </body>
```

- ☐ `span`
- ☐ `body > main.content > span.contact-email`
- ☐ `span.contact-name`

#### Exercice

Dans la déclaration CSS suivante, quelles propriétés aurait-il fallu mutualiser ?

```
1 header {
2   font-size: 18px;
3   color: lightyellow;
4   margin: 10px;
5 }
```



```

6
7 main {
8   font-size: 18px;
9   margin: 10px;
10 }
11
12 footer {
13   font-size: 18px;
14   padding: 10px;
15   background: black;
16 }

```

- ☐ font-size
- ☐ color
- ☐ margin
- ☐ padding
- ☐ background

#### Exercice

Lorsque l'on parle de philosophie *utility-first*, on parle de styles que l'on considère comme...

- ☐ Atomiques et réutilisables
- ☐ Spécifiques et définis pour être appliqués uniquement à certains éléments

#### Exercice

Quel consortium célèbre met à disposition un outil permettant de valider la qualité d'un code HTML ?

### B. Exercice : Défi

Suite à la lecture de ce cours, vous avez décidé de valider la qualité de votre site personnel, laissé à l'abandon depuis maintenant quelques années. Voici le code source que vous avez retrouvé :

```

1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
2 <html lang="fr">
3 <head>
4   <meta charset="utf-8">
5   <title>Mon site</title>
6   <link href="https://fonts.googleapis.com/css?family=Lato:400,100,300" rel="stylesheet">
7   <link rel="stylesheet" href="style.css">
8 </head>
9 <body>
10 <table>
11   <tr class="header">
12     <td colspan="2"><h1>Mon site</td></h1>
13   </tr>
14   <tr class="nav">
15     <td colspan="2">
16       <ul>
17         <li><a href="#" class="nav-link">Ma présentation</li></a>
18         <li><a href="#" class="nav-link">Mes projets</a></li>
19         <li><a href="#" class="nav-link">Me contacter</li></a>
20       </ul>
21     </td>

```

```

22     </tr>
23     <tr>
24         <td colspan="2">
25             <h2>Tout savoir sur moi</h2>
26         </td>
27     </tr>
28     <tr>
29         <td class="section col-3-4">
30             <div class="article">
31                 <h3>Mon site personnel</h3>
32                 <p>
33                     Lorem ipsum dolor sit amet, consectetur adipiscing elit.
34                     Curabitur fringilla nisi neque, non commodo diam tincidunt quis.
35                     Donec a nibh sed ipsum fermentum vehicula vel ac lectus.
36                     Nulla congue feugiat tellus, ut imperdiet enim elementum a.
37                     Sed sed odio eu nibh gravida ultrices.
38                 </p>
39             </div>
40             <p>
41                 <div class="article ">
42                     <h3>Mon parcours</h3>
43                     <p>
44                         Lorem ipsum dolor sit amet, consectetur adipiscing elit.
45                         Curabitur fringilla nisi neque, non commodo diam tincidunt quis.
46                         Donec a nibh sed ipsum fermentum vehicula vel ac lectus.
47                         Nulla congue feugiat tellus, ut imperdiet enim elementum a.
48                         Sed sed odio eu nibh gravida ultrices.
49                     </p>
50                 </div>
51             </p>
52         </td>
53         <td>
54             <div class="aside">
55                 <div class="article">
56                     <h2>Qui suis-je ?</h2>
57                     <p>
58                         Lorem ipsum dolor sit amet, consectetur adipiscing elit.
59                     </p>
60                     <h3>Mes passions</h3>
61                     <ul>
62                         <div><li>Les chats</li></div>
63                         <div><li>Le sport</li></div>
64                         <div><li>La musique</li></div>
65                     </ul>
66                 </div>
67             </div>
68         </td>
69     </tr>
70     <tr class="footer">
71         <td colspan="2">
72             <p>Suivez moi sur les réseaux sociaux !</p>
73             <div>
74                 
75                 
76                 
77             </div>
78             <p>Tous droits réservés</p>
79         </td>

```

```
80     </tr>
81 </table>
82 </body>
83 </html>
84
1  body {
2     font-family: 'Lato', sans-serif;
3     margin: 0;
4 }
5
6  table {
7     width: 100%;
8 }
9
10 td {
11     padding: 20px;
12 }
13
14 h1 {
15     font-weight: bold;
16 }
17
18 h2 {
19     font-weight: bold;
20 }
21
22 h3 {
23     font-style: italic;
24 }
25
26 .header {
27     text-align: center;
28 }
29
30 .nav {
31     padding: 10px;
32     background-color: lightgrey;
33     margin-bottom: 10px;
34 }
35
36 .nav li {
37     display: inline;
38     margin: 10px;
39 }
40
41 .col-3-4 {
42     width: 75%;
43 }
44
45 .footer {
46     text-align: center;
47     font-style: italic;
48     background-color: #607D8B;
49     color: white;
50     margin-top: 20px;
51     padding: 10px;
52 }
53
```

**Question 1**

[solution n°7 p.36]

- Le design de cette page a été implémenté il y a maintenant quelque temps, l'ensemble des éléments ont été alignés grâce à un tableau. C'est fonctionnel, mais ce n'est pas le rôle de ces balises. Remplacez cela par les balises sémantiquement adaptées, en vous aidant des classes définies. Vous adapterez le CSS pour conserver un affichage identique.
- Modifiez le `doctype` pour qu'il corresponde à la déclaration d'un site HTML5.
- Vérifiez l'imbrication des différentes balises et corrigez-la au besoin.

Une fois vos correctifs terminés, vous validerez la structure de la page sur le site du W3C pour vous assurer que celle-ci est valide.

**Indice :**

Pensez à la propriété `display:inline-block` pour aligner des éléments sur une même ligne.

**Question 2**

[solution n°8 p.39]

Il ne vous aura certainement pas échappé que le CSS utilisé aurait pu être simplifié : de nombreuses propriétés sont redéfinies systématiquement, avec la même valeur. Le CSS est fonctionnel, mais difficilement maintenable. La philosophie du *utility-first* se prêterait à ce cas, l'idée étant de créer des classes "utilitaires" plutôt que de spécifier et répéter systématiquement certaines propriétés.

Par exemple, on pourrait créer une classe `.text-bold` pour éviter de répéter la définition de `font-weight:bold`, ou `.p-20` pour indiquer que l'on souhaite appliquer un padding de 20 px à un élément.

Analysez le code CSS proposé dans la solution précédente et essayez d'adapter la philosophie de celui-ci afin de décomplexifier le CSS et de faciliter la compréhension du rendu HTML.

**Solutions des exercices**

## p. 6 Solution n°1

Voici ce que vous avez dû obtenir :

```

1 <!DOCTYPE html>
2 <html lang="fr">
3 <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <title>Document</title>
7 </head>
8 <body>
9 <header>Mon site personnel</header>
10 <div class="content">
11     <aside>
12         <nav>
13             <h2>Menu</h2>
14             <ul>
15                 <li><a href="#">Mes réalisations</a></li>
16                 <li><a href="#">A propos</a></li>
17                 <li><a href="#">Contact</a></li>
18                 <li><a href="#">Site partenaire</a></li>
19             </ul>
20         </nav>
21     </aside>
22     <main>
23         <article>
24             <h3>Blog post 1</h3>
25             <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur nec ex
26 bibendum, viverra metus in,
27             eleifend nibh. Fusce eu risus sem. Nullam lacinia aliquam elit, et
28 ullamcorper quam elementum eu.</p>
29         </article>
30         <article>
31             <h3>Blog post 2</h3>
32             <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur nec ex
33 bibendum, viverra metus in,
34             eleifend nibh. Fusce eu risus sem. Nullam lacinia aliquam elit, et
35 ullamcorper quam elementum eu.</p>
36         </article>
37         <article>
38             <h3>Blog post 3</h3>
39             <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur nec ex
40 bibendum, viverra metus in,
41             eleifend nibh. Fusce eu risus sem. Nullam lacinia aliquam elit, et
42 ullamcorper quam elementum eu.</p>
43         </article>
44     </main>
45 </div>
46 <footer>Tous droits réservés</footer>
47 </body>
48 </html>

```

## p. 12 Solution n°2

```

1 body {
2   font-family: 'Lato', sans-serif;
3   margin: 0;
4   font-size: 14px;
5 }
6
7 h1, h2 {
8   font-weight: bold;
9 }
10
11 h2, header, footer {
12   font-style: italic;
13 }
14
15 header, footer {
16   text-align: center;
17 }
18
19 footer {
20   background-color: #005cbf;
21   color: white;
22   margin-top: 20px;
23   padding: 10px;
24 }
25
26 nav {
27   padding: 10px;
28   background-color: lightgrey;
29   margin-bottom: 10px;
30 }
31
32 main {
33   display: inline-block;
34   width: 74%;
35 }
36
37 aside {
38   display: inline-block;
39   width: 25%;
40   vertical-align: top;
41 }
42
43 section, article {
44   padding: 20px;
45 }
46

```

Nous pouvons également optimiser le code en utilisant :

```

1 nav, footer {
2   padding: 10px;
3 }
4 main, aside {
5   display: inline-block;
6 }
7

```

## p. 18 Solution n°3

```

1 <!DOCTYPE html>
2 <html lang="fr">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Document</title>
7   <link rel="stylesheet" href="style_correct.css">
8 </head>
9 <body>
10 <header>
11   <h1>Contactez-moi</h1>
12 </header>
13
14 <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur nec ex bibendum,
    viverra metus in, eleifend nibh.
15     Fusce eu risus sem. Nullam lacinia aliquam elit, et ullamcorper quam elementum eu.</p>
16
17 <form>
18   <div class="form-group row">
19     <label for="email" class="col-sm-2 col-form-label">Email</label>
20     <div class="col-sm-10">
21       <input
22         type="email"
23         readonly
24         class="form-control-plaintext"
25         id="email"
26         placeholder="email@example.com"
27         data-type="email"
28         data-invalid-message="Merci de saisir une adresse e-mail correct"
29       >
30     </div>
31   </div>
32   <div class="form-group row">
33     <label for="name" class="col-sm-2 col-form-label">Name</label>
34     <div class="col-sm-10">
35       <input
36         type="text"
37         readonly
38         class="form-control-plaintext"
39         id="name" placeholder="John Doe"
40         data-type="text"
41       >
42     </div>
43   </div>
44   <div class="form-group row">
45     <label for="subject" class="col-sm-2 col-form-label">Subject</label>
46     <div class="col-sm-10">
47       <input
48         type="text"
49         readonly
50         class="form-control-plaintext"
51         id="subject"
52         placeholder="Sujet de votre message"
53         data-type="text"
54       >
55     </div>

```

```

56     </div>
57     <div class="form-group row">
58         <label for="message">Textarea</label>
59         <textarea
60             class="form-control"
61             id="message"
62             placeholder="Votre message"
63             data-type="text"
64             required
65         ></textarea>
66     </div>
67
68     <button class="btn btn-primary" type="submit">Envoyer le message</button>
69 </form>
70
71 </body>
72 </html>
73

```

#### p. 21 Solution n°4

Pour valider un code HTML depuis le site du W3C, on peut le copier-coller directement depuis cette URL : [https://validator.w3.org/#validate\\_by\\_input](https://validator.w3.org/#validate_by_input).

#### p. 22 Solution n°5

Voici les retours de la validation W3C :

- Warning** Consider adding a `lang` attribute to the `html` start tag to declare the language of this document.  
From line 1, column 1 to line 1, column 6  
`<html>--<head`  
For further guidance, consult [Declaring the overall language of a page](#) and [Choosing language tags](#).  
If the HTML checker has misidentified the language of this document, please [file an issue report](#) or [send e-mail to report the problem](#).
- Error** Start tag seen without seeing a doctype first. Expected `<!DOCTYPE html>`.  
From line 1, column 1 to line 1, column 6  
`<html>--<head`
- Error** Element `head` is missing a required instance of child element `title`.  
From line 5, column 1 to line 5, column 7  
`<!--</head>--<body`  
Content model for element `head`:  
If the document is an `iframe` source document or if title information is available from a higher-level protocol: Zero or more elements of `metadata content`, of which no more than one is a `title` element and no more than one is a `base` element.  
Otherwise: One or more elements of `metadata content`, of which exactly one is a `title` element and no more than one is a `base` element.
- Warning** Section lacks heading. Consider using `h2-h6` elements to add identifying headings to all sections.  
From line 22, column 9 to line 22, column 17  
`<section>--`
- Error** An `img` element must have an `alt` attribute, except under certain conditions. For details, consult [guidance on providing text alternatives for images](#).  
From line 34, column 13 to line 34, column 99  
`--`
- Warning** Section lacks heading. Consider using `h2-h6` elements to add identifying headings to all sections.  
From line 33, column 9 to line 33, column 17  
`<section>--`

Et le code une fois modifié pour respecter les normes W3C :

```

1 <!DOCTYPE html>
2 <html lang="fr">
3 <head>
4     <title>Recette - La pâte à crêpes</title>
5     <meta charset="utf-8">
6     <meta name="viewport" content="width=device-width, initial-scale=1.0">
7     <link rel="stylesheet" href="style.css">

```



```

8 </head>
9 <body>
10 <header>
11   <h1>Recette - La pâte à crêpes</h1>
12 </header>
13 <div class="content">
14   <aside>
15     <nav>
16       <h3>Menu</h3>
17       <ul>
18         <li><a href="#">Mes dernières recettes</a></li>
19         <li><a href="#">A propos</a></li>
20         <li><a href="#">Contact</a></li>
21       </ul>
22     </nav>
23   </aside>
24   <main>
25     <section>
26       <h2>Ingrédients</h2>
27       <ul>
28         <li>300g de farine</li>
29         <li>3 oeufs entiers</li>
30         <li>3 cuillères à soupe de sucre</li>
31         <li>2 cuillères à soupe d'huile</li>
32         <li>50g de beurre fondu</li>
33         <li>60cl de lait</li>
34         <li>5cl de rhum</li>
35       </ul>
36     </section>
37     <section>
38       
39       <h2>Déroulé des opérations</h2>
40       <ol>
41         <li>
42           Mettre la farine dans une terrine et former un puits.
43         </li>
44         <li>
45           Y déposer les oeufs entiers, le sucre, l'huile et le beurre.
46         </li>
47         <li>
48           Mélanger délicatement avec un fouet en ajoutant au fur et à mesure le
49           lait. La pâte ainsi obtenue
50           doit
51           avoir une consistance d'un liquide légèrement épais.
52         </li>
53         <li>
54           Parfumer de rhum.
55         </li>
56         <li>
57           Faire chauffer une poêle antiadhésive et la huiler très légèrement. Y
58           verser une louche de pâte, la
59           répartir dans la poêle puis attendre qu'elle soit cuite d'un côté avant
60           de la retourner. Cuire ainsi
61           toutes les crêpes à feu doux.
62         </li>
63       </ol>
64     </section>

```

```

63     </main>
64 </div>
65
66 <footer>Tous droits réservés</footer>
67 </body>
68 </html>
69

```

## Exercice p. 23 Solution n°6


### Exercice

Lorsque l'on définit le `doctype` d'un site récent, il convient d'utiliser la déclaration...

- ☐ `<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">`
- ☒ `<!DOCTYPE html>`
- ☐ `<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">`

### Exercice

Qu'importe l'ordre de fermeture des balises, la page sera toujours considérée comme sémantiquement valide.

- ☐ Vrai
  - ☒ Faux
-  Même si la page est fonctionnelle à l'écran dans certains cas, cela ne signifie pas pour autant que sa sémantique sera considérée comme valide.


### Exercice

On souhaite englober deux éléments `li` dans un élément parent `div`, car c'est considéré comme une syntaxe sémantiquement autorisée.

```

1 <ul>
2   <li><a href="#">Accueil</a></li>
3   <div>
4     <li><a href="#">Favoris</a></li>
5     <li><a href="#">Profil</a></li>
6   </div>
7   <li><a href="#">Déconnexion</a></li>
8 </ul>

```

- ☐ Vrai
  - ☒ Faux
-  Non, la balise `<ul>` ne peut contenir que des balises `<li></li>`.

### Exercice

En HTML ou CSS, si l'on souhaite agir sur la mise en page, il convient d'utiliser...

- ☒ Les propriétés CSS `display`
- ☐ Les tableaux HTML
- ☒ Les propriétés CSS `flexbox`

### Exercice

Pour faciliter la compréhension de la sémantique d'une page, mieux vaut utiliser...

- ☐ Des identifiants
- ☐ Des classes CSS
- ☐ Des balises HTML `div`
- ☒ Des balises HTML spécifiques (`nav`, `header`...)

### Exercice

Dans la console développeur d'un navigateur, si l'on constate l'affichage suivant, cela signifie...

```
element.style {  
  ⚠ position: block;  
  ⚠ position: relative: true;  
  min-height: 100%;  
  top: 0px;  
}
```

- ☒ Que les propriétés ne sont pas utilisées
- ☐ Que les propriétés sont inexistantes

### Exercice

On dispose du code HTML suivant. Quel sélecteur CSS faudrait-il privilégier pour cibler le nom de la personne ?

```
1 <body>  
2   <main class="content">  
3     <span class="contact-name">  
4       John Doe  
5     </span>  
6   </main>  
7 </body>
```

- ☐ `span`  
Ce sélecteur est beaucoup trop générique, on risquerait de cibler trop d'éléments.
- ☐ `body > main.content > span.contact-email`  
Ce sélecteur est beaucoup trop spécifique, si la structure de notre page change, l'élément ne sera plus ciblé.
- ☒ `span.contact-name`

### Exercice


Dans la déclaration CSS suivante, quelles propriétés aurait-il fallu mutualiser ?

```

1 header {
2   font-size: 18px;
3   color: lightyellow;
4   margin: 10px;
5 }
6
7 main {
8   font-size: 18px;
9   margin: 10px;
10 }
11
12 footer {
13   font-size: 18px;
14   padding: 10px;
15   background: black;
16 }

```

- ☒ font-size
- ☐ color
- ☒ margin
- ☐ padding
- ☐ background

 font-size et margin sont répétées plusieurs fois avec la même valeur, nous aurions pu améliorer la déclaration du style afin de ne pas nous répéter.

### Exercice

Lorsque l'on parle de philosophie *utility-first*, on parle de styles que l'on considère comme...

- ☒ Atomiques et réutilisables
- ☐ Spécifiques et définis pour être appliqués uniquement à certains éléments

### Exercice

Quel consortium célèbre met à disposition un outil permettant de valider la qualité d'un code HTML ?

W3C

### p. 28 Solution n°7

```

1 <!DOCTYPE html>
2 <html lang="fr">
3 <head>
4   <meta charset="utf-8">
5   <title>Mon site</title>
6   <link href="https://fonts.googleapis.com/css?family=Lato:400,100,300" rel="stylesheet">
7   <link rel="stylesheet" href="style.css">
8 </head>
9 <body>
10 <header>
11   <h1>Mon site</h1>
12 </header>
13 <nav>
14   <ul>

```

```

15     <li><a href="#" class="nav-link">Ma présentation</a></li>
16     <li><a href="#" class="nav-link">Mes projets</a></li>
17     <li><a href="#" class="nav-link">Me contacter</a></li>
18 </ul>
19 </nav>
20 <main>
21     <section>
22         <h2>Tout savoir sur moi</h2>
23         <article>
24             <h3>Mon site personnel</h3>
25             <p>
26                 Lorem ipsum dolor sit amet, consectetur adipiscing elit.
27                 Curabitur fringilla nisi neque, non commodo diam tincidunt quis.
28                 Donec a nibh sed ipsum fermentum vehicula vel ac lectus.
29                 Nulla congue feugiat tellus, ut imperdiet enim elementum a.
30                 Sed sed odio eu nibh gravida ultrices.
31             </p>
32         </article>
33         <article>
34             <h3>Mon parcours</h3>
35             <p>
36                 Lorem ipsum dolor sit amet, consectetur adipiscing elit.
37                 Curabitur fringilla nisi neque, non commodo diam tincidunt quis.
38                 Donec a nibh sed ipsum fermentum vehicula vel ac lectus.
39                 Nulla congue feugiat tellus, ut imperdiet enim elementum a.
40                 Sed sed odio eu nibh gravida ultrices.
41             </p>
42         </article>
43     </section>
44 </main>
45 <aside>
46     <article>
47         <h2>Qui suis-je ?</h2>
48         <p>
49             Lorem ipsum dolor sit amet, consectetur adipiscing elit.
50         </p>
51         <h3>Mes passions</h3>
52         <ul>
53             <li>Les chats</li>
54             <li>Le sport</li>
55             <li>La musique</li>
56         </ul>
57     </article>
58 </aside>
59 <footer>
60     <div>
61         <p>Suivez moi sur les réseaux sociaux !</p>
62         <div>
63             
64             
65             
66         </div>
67     </div>
68     <p>Tous droits réservés</p>
69 </footer>
70 </body>
71 </html>

```

```

1 body {
2     font-family: 'Lato', sans-serif;
3     margin: 0;
4 }
5
6 header {
7     text-align: center;
8     font-style: italic;
9 }
10
11 h1 {
12     font-weight: bold;
13 }
14
15 h2 {
16     font-weight: bold;
17 }
18
19 h3 {
20     font-style: italic;
21 }
22
23 nav {
24     padding: 10px;
25     background-color: lightgrey;
26     margin-bottom: 10px;
27 }
28
29 nav li {
30     display: inline;
31     margin: 10px;
32 }
33
34 main {
35     display: inline-block;
36     width: 74%;
37 }
38
39 aside {
40     display: inline-block;
41     width: 25%;
42 }
43
44 section {
45     padding: 20px;
46 }
47
48 article {
49     padding: 20px;
50 }
51
52 footer {
53     text-align: center;
54     font-style: italic;
55     background-color: #607D8B;
56     color: white;
57     margin-top: 20px;
58     padding: 10px;

```

```
59 }  
60
```

### p. 28 Solution n°8

Voici comment l'on pourrait définir des classes utilitaires dans le CSS proposé ci-dessus. Le fait de mettre en place une telle structure permet également de se poser des questions quant à la rationalisation du design d'une application. Ces classes prédéfinies permettront par exemple de systématiser les valeurs utilisées. Certains frameworks CSS, tels que TailwindCSS<sup>1</sup>, orientent leur philosophie dans ce sens.

Le style spécifique de la page en devient également plus lisible.

```
1 /** Text style */  
2 .text-center {  
3     text-align: center;  
4 }  
5  
6 .text-italic {  
7     font-style: italic;  
8 }  
9  
10 .text-bold {  
11     font-weight: bold;  
12 }  
13  
14 /** Padding */  
15 .p-10 {  
16     padding: 10px;  
17 }  
18  
19 .p-20 {  
20     padding: 20px;  
21 }  
22  
23 /** Margin */  
24 .m-10 {  
25     margin: 10px;  
26 }  
27  
28 .mb-10 {  
29     margin-bottom: 10px;  
30 }  
31  
32 .mt-20 {  
33     margin-top: 20px;  
34 }  
35  
36 /** Page specific style */  
37 body {  
38     font-family: 'Lato', sans-serif;  
39     margin: 0;  
40 }  
41  
42 nav {  
43     background-color: lightgrey;  
44 }
```

---

1 <https://tailwindcss.com/>

```

45
46 nav li {
47     display: inline;
48 }
49
50 main {
51     display: inline-block;
52     width: 74%;
53 }
54
55 aside {
56     display: inline-block;
57     width: 25%;
58 }
59
60 footer {
61     background-color: #607D8B;
62     color: white;
63 }
64

```

Et l'adaptation du fichier HTML :

```

1 <!DOCTYPE html>
2 <html lang="fr">
3 <head>
4     <meta charset="utf-8">
5     <title>Mon site</title>
6     <link href="https://fonts.googleapis.com/css?family=Lato:400,100,300" rel="stylesheet">
7     <link rel="stylesheet" href="style.css">
8 </head>
9 <body>
10 <header class="text-center text-italic">
11     <h1 class="text-bold">Mon site</h1>
12 </header>
13 <nav class="p-10 mb-10">
14     <ul>
15         <li class="m-10"><a href="#" class="nav-link">Ma présentation</a></li>
16         <li class="m-10"><a href="#" class="nav-link">Mes projets</a></li>
17         <li class="m-10"><a href="#" class="nav-link">Me contacter</a></li>
18     </ul>
19 </nav>
20 <main>
21     <section class="p-20">
22         <h2 class="text-bold">Tout savoir sur moi</h2>
23         <article class="p-20">
24             <h3 class="text-italic">Mon site personnel</h3>
25             <p>
26                 Lorem ipsum dolor sit amet, consectetur adipiscing elit.
27                 Curabitur fringilla nisi neque, non commodo diam tincidunt quis.
28                 Donec a nibh sed ipsum fermentum vehicula vel ac lectus.
29                 Nulla congue feugiat tellus, ut imperdiet enim elementum a.
30                 Sed sed odio eu nibh gravida ultrices.
31             </p>
32         </article>
33         <article class="p-20">
34             <h3 class="text-italic">Mon parcours</h3>
35             <p>
36                 Lorem ipsum dolor sit amet, consectetur adipiscing elit.

```



```
37         Curabitur fringilla nisi neque, non commodo diam tincidunt quis.
38         Donec a nibh sed ipsum fermentum vehicula vel ac lectus.
39         Nulla congue feugiat tellus, ut imperdiet enim elementum a.
40         Sed sed odio eu nibh gravida ultrices.
41     </p>
42 </article>
43 </section>
44 </main>
45 <aside>
46     <article class="p-20">
47         <h2 class="text-bold">Qui suis-je ?</h2>
48         <p>
49             Lorem ipsum dolor sit amet, consectetur adipiscing elit.
50         <h3 class="text-italic">Mes passions</h3>
51         <ul>
52             <li>Les chats</li>
53             <li>Le sport</li>
54             <li>La musique</li>
55         </ul>
56     </article>
57 </aside>
58 <footer class="text-italic text-center p-10 mt-20">
59     <div>
60         <p>Suivez moi sur les réseaux sociaux !</p>
61         <div>
62             
63             
64             
65         </div>
66     </div>
67     <p>Tous droits réservés</p>
68 </footer>
69 </body>
70 </html>
71
```