

TP CSS : des sélecteurs à la mise en page responsive

- Utilisez Visual Studio Code pour réaliser les exercices.
 - Consultez les raccourcis **Emmet** sur <https://docs.emmet.io/cheatsheet-a5.pdf> pour générer plus rapidement le code HTML.
 - Chaque exercice doit être réalisé dans **deux fichiers distincts** : un fichier HTML et un fichier CSS.
 - Les fichiers doivent être nommés **exercice1.html** et **exercice1.css**, puis **exercice2.html** et **exercice2.css**, etc.
 - Chaque fichier HTML doit être correctement relié à son fichier CSS correspondant à l'aide de la balise `<link>`.
-

Avant de commencer, vérifiez que chaque exercice est réalisé dans son propre couple de fichiers :

```
tp-css/  
|  
+-- exercice1.html  
+-- exercice1.css  
|  
+-- exercice2.html  
+-- exercice2.css  
|  
+-- exercice3.html  
+-- exercice3.css  
|  
+-- ...
```

À la fin de chaque exercice, vérifiez le résultat dans un navigateur et assurez-vous que le code HTML reste correctement structuré et valide.

Exercice 1 : sélectionner des éléments sans classe ni identifiant

En utilisant les classes, écrivez le code **CSS** permettant de colorer la liste suivante. La liste doit respecter les couleurs indiquées :

- Langages de programmation
 - **Java**
 - **C++**
 - **PHP**
- Éditeurs de texte
 - **Sublime text**
 - **Atom**
 - **Notepad++**

Dans une deuxième version, refaites ce travail sans utiliser ni `class`, ni `id`.

La construction de la liste doit être **W3C Valide**.

Exercice 2 : sélectionner des éléments avec deux classes

En utilisant **seulement deux classes** et **aucun identifiant**, écrivez le code **CSS** permettant de colorer la liste suivante.

- France
 - Bleu
 - Blanc
 - Rouge
- Tunisie
 - Blanc
 - Rouge
- Uruguay
 - Blanc
 - Bleu

Contraintes :

- utiliser exactement deux classes
- ne créer aucun identifiant
- utiliser les classes de manière à distinguer les couleurs

La construction de la liste doit être **W3C Valide**.

Exercice 3 (facultatif) : réutiliser une seule classe

Refaites l'exercice précédent en utilisant **une seule classe** et **aucun identifiant**.

Exercice 4 : pseudo-sélecteurs et relations entre éléments

Étant donnée une balise `<div class="container">` contenant plusieurs balises `<p>` (au moins trois), écrivez le code **CSS** permettant d'obtenir les comportements suivants lorsque la souris est déplacée sur un paragraphe :

- le paragraphe **survolé** est affiché en **gras** ;
- les paragraphes situés **avant** le paragraphe survolé sont affichés en *italique* ;
- les paragraphes situés **après** le paragraphe survolé sont soulignés.

Vous pouvez partir d'une structure HTML de ce type :

```
<div class="container">
  <p>Paragraphe 1</p>
  <p>Paragraphe 2</p>
  <p>Paragraphe 3</p>
  <p>Paragraphe 4</p>
</div>
```

Quelques indices

- utiliser `:hover` pour identifier le paragraphe survolé
- utiliser le combinateur de frères généraux `~` pour cibler les paragraphes suivants
- utiliser `:has()` avec `~` pour cibler les paragraphes précédents.

Contrainte : aucune nouvelle classe et aucun identifiant ne doivent être ajoutés aux balises `<p>`.

Exercice 5 : sélecteurs d'attributs

On considère le code HTML suivant :

```
<nav>
  <a href="index.html">Accueil</a>
  <a href="https://developer.mozilla.org/">MDN</a>
  <a href="https://www.w3.org/">W3C</a>
  <a href="documents/cours-css.pdf">Cours CSS</a>
  <a href="documents/exercices.html">Exercices</a>
</nav>

<form>
  <input type="text" placeholder="Nom">
  <input type="email" placeholder="Adresse e-mail" required>
  <input type="password" placeholder="Mot de passe">
  <input type="text" placeholder="Champ désactivé" disabled>
</form>
```

Écrivez le code **CSS** permettant de différencier les éléments selon leurs attributs.

- les liens dont l'attribut `href` commence par `https://` doivent avoir une apparence particulière
- les liens dont l'attribut `href` se termine par `.pdf` doivent avoir une autre apparence
- les liens dont l'attribut `href` contient `documents` doivent être facilement identifiables
- les champs possédant l'attribut `required` doivent être distingués des autres champs
- les champs possédant l'attribut `disabled` doivent avoir une apparence adaptée

Indices : vous pouvez utiliser plusieurs formes de sélecteurs d'attributs :

```
[attr]
[attr="valeur"]
[attr^="début"]
[attr$="fin"]
[attr*="contenu"]
```

Contrainte : ne modifiez pas le HTML et n'ajoutez ni classe ni identifiant.

Exercice 6 : créer un menu vertical

On considère le code HTML suivant :

```
<nav>
  <ul id="menu">
    <li><a href="page1.html">menu1</a>
      <ul class="sous-menu">
        <li><a href="page11.html">menu11</a></li>
        <li><a href="page12.html">menu12</a></li>
      </ul>
    </li>
    <li><a href="page2.html">menu2</a></li>
    <li><a href="page3.html">menu3</a></li>
    <li><a href="page4.html">menu4</a>
      <ul class="sous-menu">
        <li><a href="page41.html">menu41</a></li>
        <li><a href="page42.html">menu42</a></li>
      </ul>
    </li>
  </ul>
</nav>
```

En vous basant sur ce code HTML :

1. écrivez un premier code **CSS** permettant de créer un **menu vertical**
2. écrivez un deuxième code **CSS** permettant de créer un **menu horizontal** (facultatif)

Pour les deux versions, travaillez notamment sur :

- la suppression des puces
- les marges et espacements
- l'apparence des liens
- l'organisation des éléments principaux du menu
- l'organisation des sous-menus

La version horizontale, si elle est réalisée, peut être testée dans un second fichier CSS.

Exercice 7 : mise en page avec Flexbox ou Grid

L'objectif de cet exercice est de construire une page Web occupant l'intégralité de la fenêtre du navigateur en utilisant Flexbox ou Grid pour organiser les différentes zones de la page.

Commencez par choisir la technique qui vous convient le mieux : Flexbox ou Grid.

Si vous avez suffisamment de temps, réalisez ensuite une seconde version de la même page avec l'autre technique.

Commencez par créer une structure HTML5 utilisant des balises sémantiques :

```
<header>...</header>
<nav>...</nav>
<main>...</main>
<footer>...</footer>
```

La page devra être organisée de la manière suivante :

- un **header** occupant environ **20% de la hauteur** de la fenêtre et **100% de sa largeur**
- un **nav** placé à gauche, occupant environ **20% de la largeur**
- un **main** occupant la largeur restante
- un **footer** occupant environ **20% de la hauteur** et placé tout en bas de la page.

La structure visuelle attendue peut être représentée ainsi :

HEADER 20 %	
NAV 20 %	MAIN
FOOTER 20 %	

Contraintes :

- utiliser des balises sémantiques HTML5
- choisir Flexbox ou CSS Grid pour réaliser la première version de la mise en page
- si le temps le permet, réaliser une seconde version avec l'autre technique
- ne pas utiliser Flexbox et Grid simultanément dans une même version
- la page doit occuper toute la hauteur de la fenêtre
- le **header** et le **footer** doivent s'étendre sur toute la largeur
- le menu doit rester à gauche du contenu principal
- chaque zone (**header**, **nav**...) doit posséder une **couleur de fond différente**
- ajouter également une bordure et un espacement si nécessaire afin de rendre la structure facilement identifiable

Exercice 8 : adapter la page avec les Media Queries

Reprenez la page réalisée à l'exercice 7 et rendez-la adaptée aux petits écrans à l'aide des **media queries**.

Sur un écran suffisamment large, la page doit conserver la mise en page précédente :

- header en haut
- nav à gauche
- main à droite
- footer en bas

Pour les petits écrans, par exemple lorsque la largeur de la fenêtre est inférieure à **768 px** :

- supprimer l'affichage du **footer**
- supprimer l'affichage du **nav**
- augmenter la largeur du **main** afin qu'il occupe l'espace restant

Vous utiliserez une règle de type :

```
@media (max-width: 768px) {  
    /* règles appliquées sur les petits écrans */  
}
```

Contraintes :

- utiliser une media query
- ne pas créer une deuxième page HTML pour la version mobile
- modifier uniquement le CSS pour adapter l'affichage
- conserver des couleurs de fond distinctes pour les zones visibles afin de vérifier facilement la réorganisation responsive.