Comprendre et utiliser le CSS : donner du style à ses pages ✨

Premiers pas avec CSS

1. **Faire le lien HTML/CSS**
2. **Sélecteurs CSS : comment cibler efficacement les bons éléments**
    a. Le sélecteur universel
    b. Le sélecteur d'identifiant
    c. Le sélecteur de classe
    d. Le sélecteur universel
    e. Les combinateurs
3. **L'héritage en cascade**
4. **La propriété Display**
5. **Les modèles de boîtes**
6. **Les unités**
7. **Les typographies**
8. **Les couleurs & backgrounds**
9. **Variables en CSS**
10. **Reset / Normalize en CSS**


@ithumor
hotel-room.css
.air-conditioner {
margin-left: -25px;
}

Faire le lien HTML/CSS

Le code CSS s'écrit dans un fichier avec l'extension .css. Le lien entre le fichier HTML et le fichier CSS s'effectue dans le premier des deux, au niveau de l'en-tête.

Ajoutons ce lien dans le code minimal d'un fichier HTML.

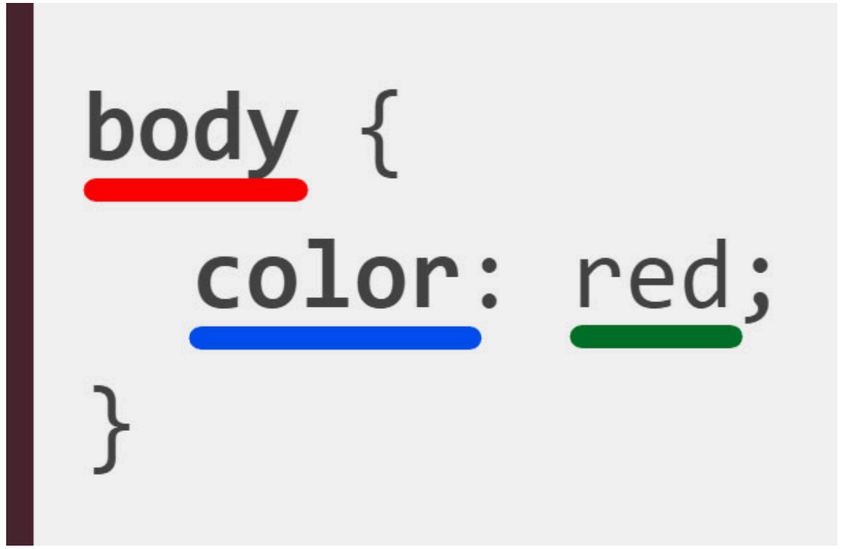
```
<head>
...
 <link rel="stylesheet" type="text/css" href="style.css" />
...
</head>
```

Premiers pas avec CSS

👉 Une feuille de style de CSS est composée d'un ensemble de règles. Une règle est composée de **3 notions** différentes :

1. **Sélecteur** : Indique le ou les éléments sur lesquels la mise en forme va s'appliquer.
2. **Propriété** : Précise la propriété CSS, nous allons les découvrir au fur et à mesure.
3. **Valeur** : Détermine la valeur de la propriété à appliquer sur le sélecteur.

Une règle débute par le sélecteur, puis le reste du contenu est englobé entre deux accolades : **{** et **}**.



Le sélecteur universel

Le **sélecteur universel *** permet de cibler toutes les balises sans exception.

Le caractère qui identifie le sélecteur universel est *. Il s'utilise de manière très simple.

```
* { color: blue; }
```

Le sélecteur d'identifiant (#id)

Le sélecteur d'identifiant utilise l'attribut **id=""** d'une balise HTML pour sélectionner un élément spécifique.

L'identifiant d'une balise devant **<u>être unique</u>** dans une page web, le sélecteur d'identifiant est donc utilisé pour sélectionner un élément unique.

```
<div id=”super-p”>Text</div>
```

```
#super-p { color: pink; }
```

Text

Le sélecteur de classe (.class)

A l'inverse d'un identifiant, un ou plusieurs éléments peuvent avoir la **même classe**. Le sélecteur de classe sélectionne les éléments avec **un attribut de classe spécifique**.

Le sélecteur de classe utilise l'attribut class="" d'une balise HTML pour sélectionner une ou plusieurs balises.

```
<div class=”mega-p”>Text</div>
<div class=”mega-p”>Texte 2</div>
```

```
.mega-p { color: yellow; }
```

```
Text
Text 2
```

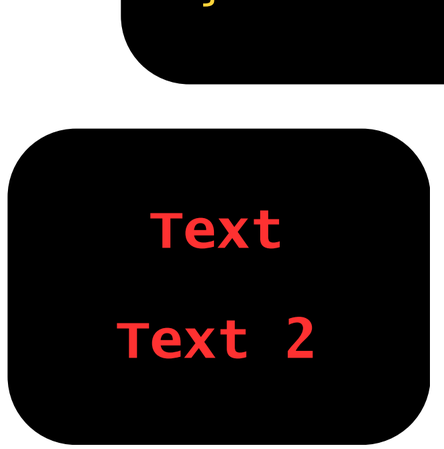
Les groupes de sélecteurs

Pour **grouper les sélecteurs,** séparez chaque sélecteur par une virgule.

Ainsi, vous pourrez définir **les mêmes styles** pour **plusieurs éléments** en une seule déclaration.

```
<div class=”mega-p”>Text</div>
<div id=”super-p”>Texte 2</div>
```

```
#super-p, .mega-p {
 color: red;
}
```



Les combinateurs

Les combinateurs permettent de combiner différents types de sélecteurs pour en former un nouveau plus précis.

On peut donc inclure entre chaque sélecteur un combinateur. **Il existe 4 grands types de combinateurs :**

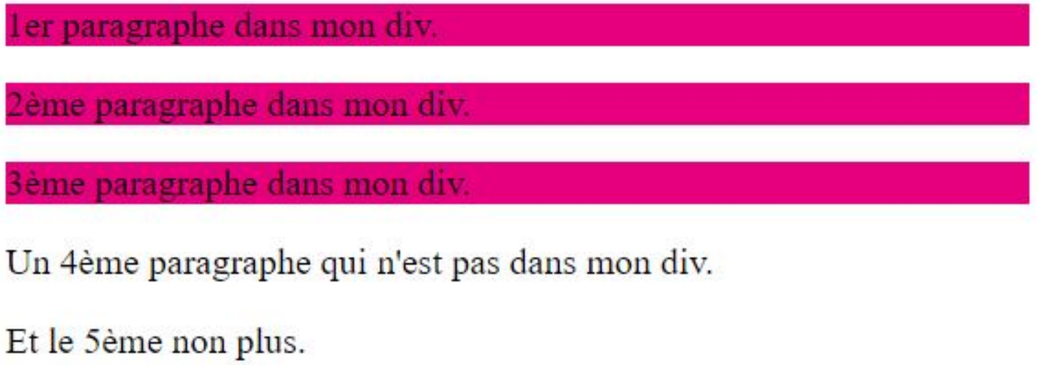
Combinateur descendant

Seul l'enfant est sélectionné et non les enfants d'enfants.

```
div p {
   background-color: #e6007e;
}
```

```
<div>
  <p>1er paragraphe dans mon div.</p>
  <p>2ème paragraphe dans mon div.</p>
  <span><p>3ème paragraphe dans mon div.</p></span>
</div>

<p>Un 4ème paragraphe qui n'est pas dans mon div.</p>
<p>Et le 5ème non plus.</p>
```



Les combinateurs

Les combinateurs permettent de combiner différents types de sélecteurs pour en former un nouveau plus précis.

On peut donc inclure entre chaque sélecteur un combinateur. **Il existe 4 grands types de combinateurs :**

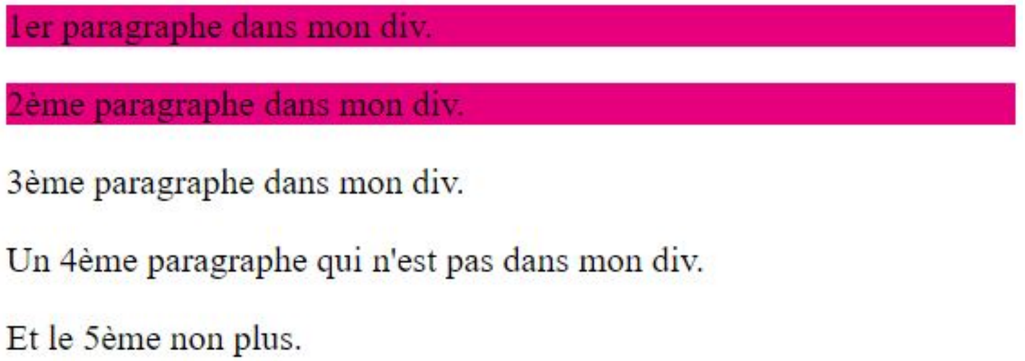
Combinateur enfant “>”

Seul l’enfant est sélectionné et non les enfants d’enfants.

```
div > p {
   background-color: #e6007e;
}
```

```
<div>
  <p>1er paragraphe dans mon div.</p>
  <p>2ème paragraphe dans mon div.</p>
  <span><p>3ème paragraphe dans mon div.</p></span>
</div>

<p>Un 4ème paragraphe qui n'est pas dans mon div.</p>
<p>Et le 5ème non plus.</p
```



Les combinateurs

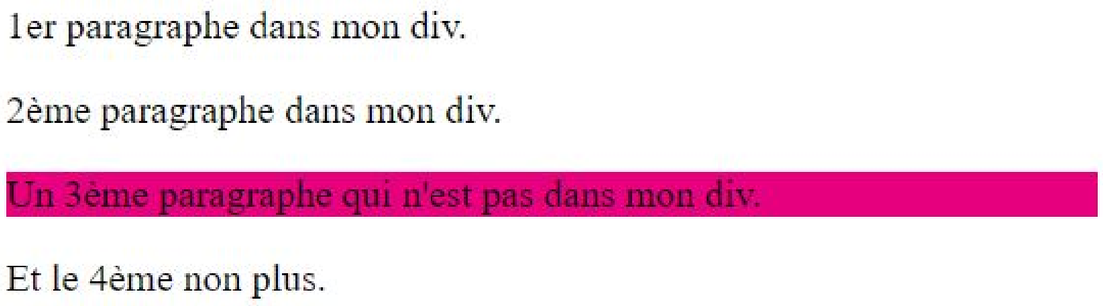
Combinateur adjacent “+”

Seul l’élément qui se trouve juste après le div est ciblé.

```
div + p {
   background-color: #e6007e;
}
```

```
<div>
 <p>1er paragraphe dans mon div.</p>
 <p>2ème paragraphe dans mon div.</p>
</div>

<p>Un 3ème paragraphe qui n'est pas dans mon div.</p>
<p>Et le 4ème non plus.</p>
```



Les combinateurs

Combinateur général sibling “~”

tous les frères qui suivent les paragraphes contenus dans le div sont ciblés.

```
div ~ p {
   background-color: #e6007e;
}
```

```
<p>Mon 1er paragraphe tout seul</p>

<div>
  <code>Un peu de code dans un div.</code>
  <p>Un 2è paragraphe dans un div.</p>
</div>

<p>Mon 3ème paragraphe tout seul.</p>
<code>Un peu de code tout seul.</code>
<p>Et un 4è paragraphe tout seul.</p>
```

Mon 1er paragraphe tout seul

`Un peu de code dans un div.`

Un 2è paragraphe dans un div.

Mon 3ème paragraphe tout seul.

`Un peu de code tout seul.`

Et un 4è paragraphe tout seul.

Les combinateurs : résumé

NOM	SYNTAXE	SÉLECTION
Combinateur descendant	A B	Ce combinateur, matérialisé par un simple espace, cible un élément B contenu dans un élément A, quelque soit son degré de parenté : un enfant ou un enfant d'enfant.
Combinateur enfant	A > B	Contrairement au précédent, ce combinateur, matérialisé par un supérier, cible un élément B qui est l'enfant direct d'un élément A.
Combinateur adjacent	A + B	Ce combinateur, matérialisé par un plus, cible un élément B immédiatement précédé par un élément A. Le mot adjacent est important : seul le premier élément après l'élément A sera ciblé.
Combinateur général sibling	A ~ B	Ce combinateur cible un élément B précédé par un élément A. Contrairement au combinateur adjacent où seul le premier frère est ciblé, ici, tous les frères sont concernés.

L'héritage en cascade

👉 **Chaque élément HTML a un parent**, sauf la balise <html>, qui est le parent ultime. L'imbrication des balises met en évidence cette hiérarchie.

```
<body>
  <h1 id="super">Roger, maître du HTML</h1>
  <div>
    <p>
      <span>Enfant</span>
    </p>
  </div>
</body>
```

- **<body>** est le parent de toutes les balises.
- **<h1>** a un parent (<body>) mais pas d'enfant.
- **** a plusieurs parents (<p>, , et <body>).

L'héritage et le poids des sélecteurs

👉 Les sélecteurs ont un poids, qui détermine leur priorité :

Sélecteur d'identifiant (#id) > Sélecteur de classe (.class) > Sélecteur d'élément (h1, p) > Sélecteur universel (*)

```
#top {
  color: green;
}
.mega {
  color: yellow;
}
h1 {
  color: orange;
}
body {
  color: red;
}
* {
  color: yellow;
}
```

```
<body>
  <h1 id="top" class=”mega”>
    Roger, maître du HTML
  </h1>
</body>
```

👉 Le **<h1>** sera vert car l'identifiant #top a le poids le plus élevé.

L’héritage CSS : ce qu’il faut retenir

L'héritage et la priorité des styles sont des notions fondamentales du CSS. Retenez que :

- Les styles **héritables** sont transmis aux enfants.
- **Les styles les plus précis** ou définis en dernier l'emportent.
- **Le poids des sélecteurs** influence leur priorité.

La propriété Display

👉 Chaque élément HTML possède une valeur par défaut pour la propriété **display**. Cette valeur varie selon le type d'élément et définit comment celui-ci sera affiché dans la page.

👉 Les 4 valeurs les plus répandues sont

display: block, **display: inline**, **display: flex** et **display: grid**

30 minutes of trying to fix something in CSS

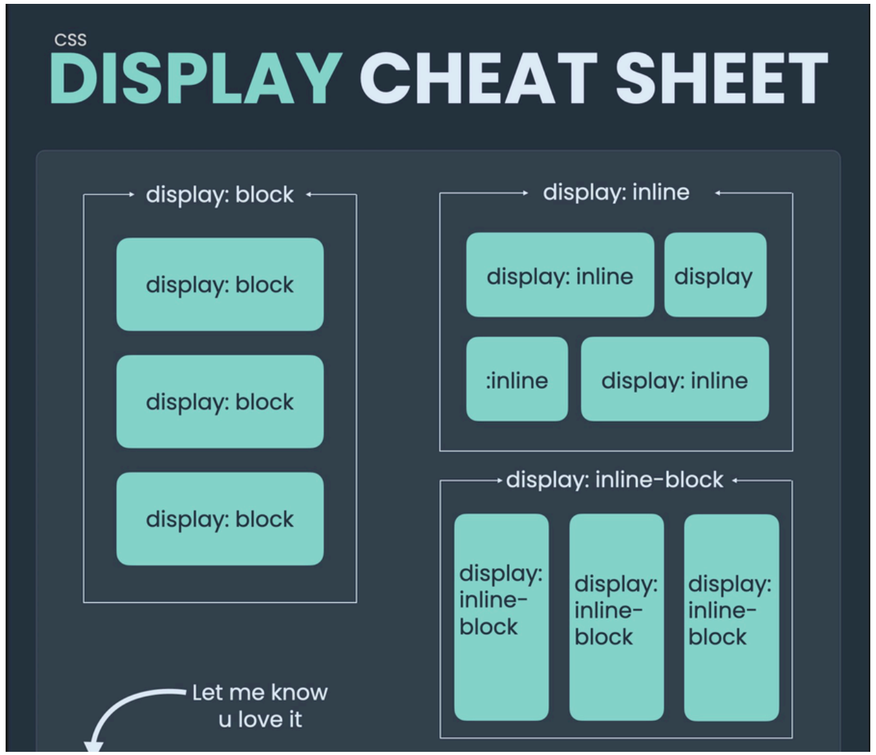
La propriété display:block

👉 Un élément de type **display:block** :

- Commence toujours sur une nouvelle ligne
- Occupe toute la largeur disponible, de la gauche à la droite

Quelques exemples d'éléments ayant un **display: block;** par défaut :

```
<p>, <h1>, <h2>, <h3>, <h4>, <h5>, <h6>, <div>, <form>
```



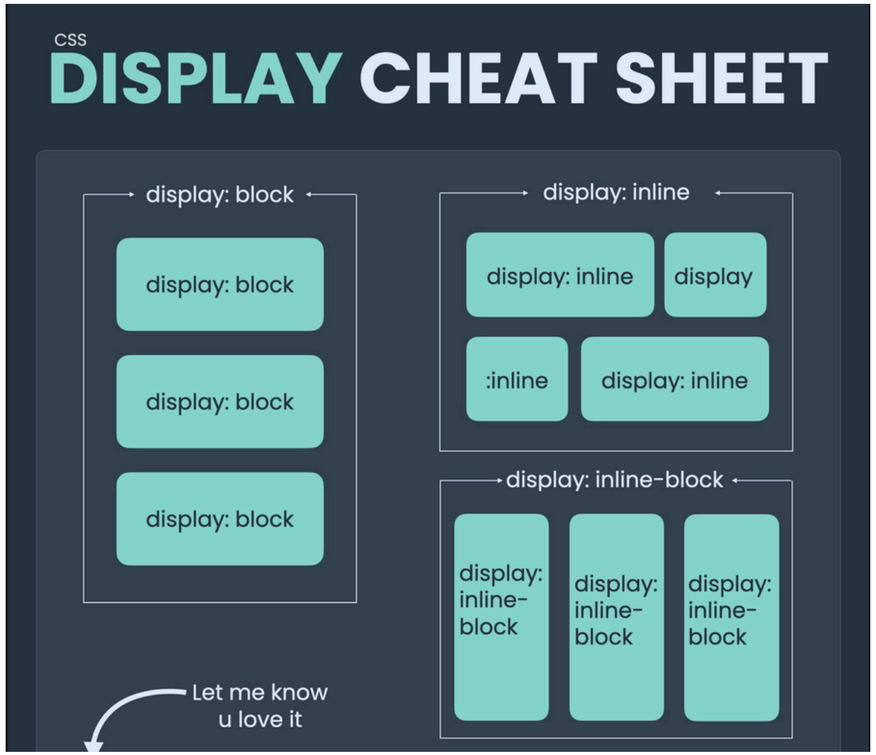
La propriété display:inline

👉 Un élément de type **display:inline** :

- Ne commence pas sur une nouvelle ligne
- Ne prend que la place dont il a besoin

Quelques exemples d'éléments ayant un **display: inline;** par défaut :

```
<span>, <a>, <img>
```



La propriété display:none

👉 **display:none** masque un élément et ses enfants.

Le navigateur ignore complètement cet élément.

👉 Cette propriété est surtout utilisée pour masquer des éléménts et les faires apparaître en

JavaScript : dropdown, menu, etc...

La propriété display:flex : organisation en ligne ou colonne

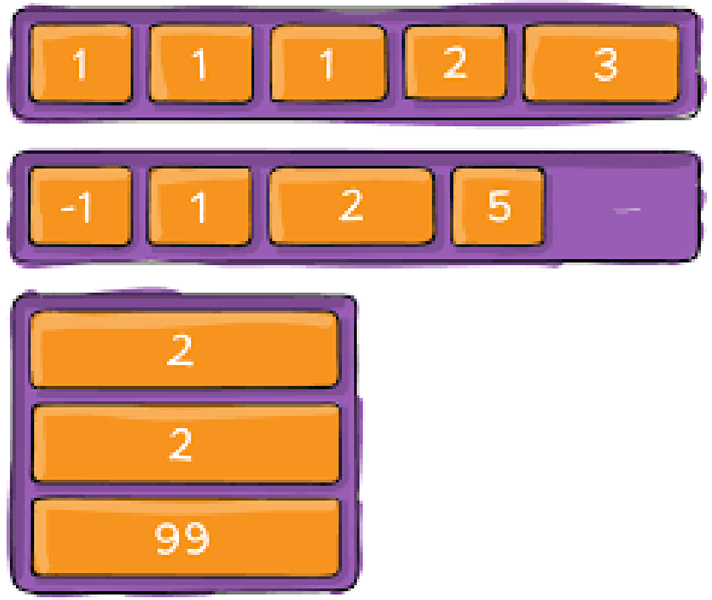
👉 **display: flex;** permet d'organiser les éléments le long d'un axe principal (horizontal ou vertical).

Activer flexbox :

```
.container {
  display: flex; /* Active le mode Flexbox */
  gap: 10px; /* Ajoute une marge de 10px */
}
```

Activer flexbox :

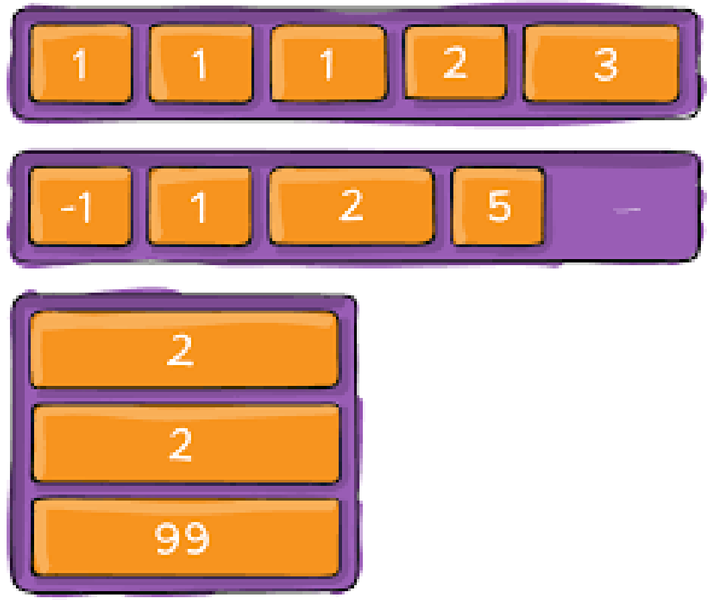
```
.container {
  flex-direction: row; /* en ligne horizontalement */
  flex-direction: column; /* empilés verticalement */
}
```



La propriété display:flex : organisation en ligne ou colonne

👉 Aligner les éléments avec **justify-content** et **align-items**

Propriété	Effet en row	Effet en column
justify-content: flex-start;	Aligné à gauche	Aligné en haut
justify-content: center;	Centré horizontalement	Centré verticalement
justify-content: space-between;	Espaces entre les éléments	Idem en colonne
align-items: center;	Aligné au centre verticalement	Aligné au centre horizontalement




```
.container {
  display: flex;
  flex-direction: row;
  justify-content: center;
  align-items: center;
  gap: 10px;
}
```

Ajuster la taille des éléments avec flex-grow, flex-shrink, flex-basis

👉 **flex-grow**: détermine la capacité d’un élément à s'agrandir pour remplir l’espace disponible

```
.item {
  flex-grow: 1; /* Les éléments prennent tout l’espace disponible */
  flex-grow: 2; /* Prend 2 parts, donc plus de place */
}
```

👉 **flex-shrink**: détermine si un élément peut rétrécir en cas de manque d’espace.

```
.item {
  flex-shrink: 1; /* L’élément peut rétrécir si nécessaire */
}
```

Si **flex-shrink: 0;**, l'élément ne se rétrécira pas, même s’il manque d’espace.

Ajuster la taille des éléments avec flex-grow, flex-shrink, flex-basis

👉 **flex-basis**: définit la taille initiale de l'élément avant application des autres propriétés flex.

```
.item {
  flex-basis: 200px; /* L'élément commencera avec 200px de largeur */
}
```

👉 On peut aussi utiliser :

```
.item {
  flex-basis: auto; /* L'élément prendra la taille de son contenu */
}
```

👉 Abréviation complète :

```
.item {
  flex: 1 1 100px; /* Équivaut à flex-grow: 1; flex-shrink: 1; flex-basis: 100px; */
}
```

Jeux CSS : Découverte 1h

https://codingfantasy.com/games/flexboxadventure/play

Plonge dans un univers médiéval où tu aides un chevalier à avancer… grâce aux propriétés Flexbox ! Ludique et scénarisé, parfait pour comprendre justify-content, align-items, etc.

http://www.flexboxdefense.com

Un tower defense original où tu repousses les ennemis… en plaçant tes tours à l'aide de Flexbox. Apprendre à positionner les éléments devient un vrai jeu de stratégie.

https://flukeout.github.io

Tu dois sélectionner les bons éléments sur une table… avec les sélecteurs CSS ! En 26 niveaux, ce jeu te fait progresser de façon visuelle et intuitive sur un concept fondamental.

https://flexboxfroggy.com/#fr

Aide des petites grenouilles à rejoindre leur nénuphar en manipulant Flexbox. Clair, progressif, et dispo en français !

https://cssgridgarden.com/#fr

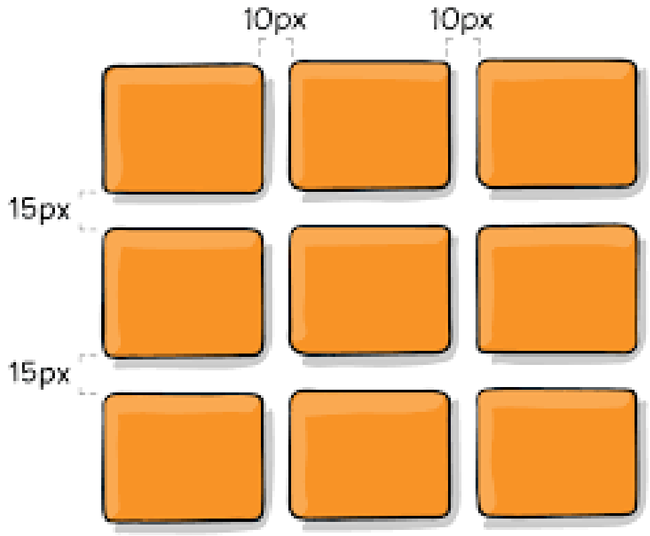
Fais pousser des carottes en arrosant la bonne case grâce à CSS Grid. Un excellent moyen d'appréhender le système de grilles CSS de façon concrète et fun.

La propriété display:grid

👉 **display: grid;** permet d'organiser des éléments en lignes et en colonnes simultanément.

Activer grid :

```
.container {
  display: grid; /* Active le mode Grid */
  gap: 20px; /* Ajoute 20px de marge */
}
```



Définir les colonnes & grilles :

```
.container {
  /* 3 colonnes de largeur variable */
  grid-template-columns: 1fr 2fr 1fr;
  /* 3 lignes */
  grid-template-rows: 100px auto 200px;
}
```

Unités utiles :

- **px** : Taille fixe.
- **%** : Proportion de la taille du parent.
- **fr** : Fraction de l'espace disponible.
- **auto** : S'adapte au contenu.

La propriété display:grid

👉 Positionner les éléments avec grid-column et grid-row

```css
.item {
  grid-column: 1 / 3; /* Occupe les colonnes 1 et 2 */
  grid-row: 1 / 2;    /* Occupe la première ligne */
}
```

👉 Aligner les Éléments (justify-items et align-items)

```css
.item {
  justify-items: center; /* centre horizontalement */
  align-items: center;   /* centre verticalement */
}
```

Grid vs Flexbox : que choisir ?

Critère	display: flex;	display: grid;
Organisation en ligne ou colonne	✅ Oui	❌ Non
Organisation en grille complète	❌ Non	✅ Oui
Alignement simple d'éléments	✅ Oui	✅ Oui
Contrôle précis de la disposition	❌ Non	✅ Oui
Adapté aux mises en page complexes	❌ Non	✅ Oui
Responsive Design	✅ Oui	✅ Oui

Bonnes pratiques :

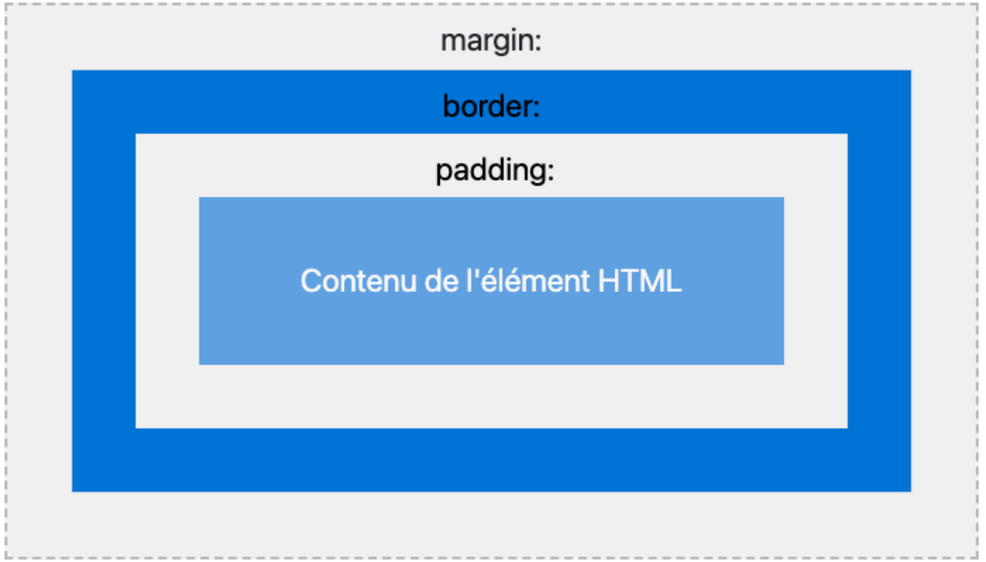
- Utilisez **Grid** pour les mises en page globales (ex: structure principale d'un site).
- Utilisez **Flexbox** pour aligner des éléments spécifiques (ex: boutons, menus).
- Utilisez **gap** au lieu de margin pour espacer les éléments en Grid/Flexbox.

Les modèles de boîtes : Introduction

👉 En CSS, chaque élément HTML est une boîte rectangulaire. Comprendre la composition et la structure de ces boîtes est essentiel pour bien maîtriser la mise en page.

👉 Le modèle de boîte CSS est constitué de plusieurs zones :

1. Le **contenu** (content)
2. La **marge intérieure** (padding)
3. La **bordure** (border)
4. La **marge extérieure** (margin)



Les modèles de boîtes : la marge intérieure (padding)

👉 Le **padding** correspond à l'espace entre le contenu d'un élément et sa bordure.

```
div {
    padding: 20px;
    background-color: lightblue;
}
```

👉 Cela crée **un espace de 20 pixels** autour du contenu, tout en gardant la bordure et la marge extérieure intactes.

👉 **Une notation abrégée est possible :**

```
div {
    /* Haut, Droite, Bas, Gauche */
    padding: 20px 20px 20px 20px;
}
```

On peut aussi définir des valeurs **différentes pour chaque côté** :

```
div {
    padding-top: 20px;
    padding-right: 20px;
    padding-bottom: 20px;
    padding-left: 20px;
}
```

Les modèles de boîtes : la marge extérieure (margin)

👉 Le **margin** représente la distance entre une boîte et les autres éléments.

```css
div {
    margin: 20px;
    background-color: lightgreen;
}
```

👉 La marge **n'affecte pas la taille de la boîte**, elle agit seulement sur l'espacement entre éléments.

```css
div {
    margin: 0 auto;
}
```

👉 Le cas particulier **margin: 0 auto;** centre horizontalement un élément par rapport à son parent.

👉 **Comme pour padding, on peut ajuster les valeurs individuellement :**

```css
div {
    /* Haut, Droite, Bas, Gauche */
    margin: 20px 20px 20px 20px;
}
```

Les modèles de boîtes : la bordure (border)

👉 La **border** est une ligne qui s'insère entre le padding et le margin. Elle peut être stylisée avec plusieurs propriétés :

```
div {
   border: 1px solid lightgreen;
}
```

👉 **On peut personnaliser chaque côté :**

```
div {
   border-top: 1px solid red;
   border-right: 1px dashed pink;
   border-bottom: 1px dotted green;
   border-left: 1px double blue;
   /* etc... */
}
```

👉 **On peut aussi accéder directement à certaines caractéristiques comme de nombreuses propriétés CSS:**

```
div {
   border-color: red;
   border-width: 1px;
   border-style: dotted;
}
```

Les modèles de boîtes : largeur (width) et hauteur (height)

👉 **La taille d'une boîte** dépend de plusieurs facteurs :

Taille	Calcul
width	width + padding-left + padding-right + border-left + border-right
height:	height + padding-top + padding-bottom + border-top + border-bottom

```
div {
  width: 200px;
  height: 100px;
  padding: 10px;
  border: 5px solid black;
  background-color: lightgray;
}
```

Dans ce cas, la largeur totale sera : **200px** (contenu) + **10px2** (padding) + **5px2** (border) = **230px**

Remarque : Par défaut, width et height ne prennent pas en compte padding et border.

Pour inclure ces valeurs dans le calcul, utilisez toujours :

```
*{ box-sizing: border-box; }
```

Les modèles de boîtes : min-width, min-height, max-width, max-height

👉 **La taille d'une boîte** peut également avoir une taille minimale ou maximale.

L'unité peut s'exprimer en PX,%, REM, etc...

Taille	
min-width	Largeur minimale : la boîte ne fera jamais moins de (min-width: 100px)
max-width	Largeur maximale, la boîte ne dépassera jamais (max-widh: 500px)
min-height	Hauteur minimale : la boîte ne fera jamais moins de (min-width: 100px)
max-height	Hauteur maximale, la boîte ne dépassera jamais (max-height: 500px)

```
div {
  max-width: 500px;
  max-height: 500px;
  min-width: 100px;
  min-height: 100px;
  width: 100%;
}
```

Les unités de mesure

👉 **Les principales catégories d’unités CSS sont fixes et flexibles.**

- **Unités fixes** : comme les pixels (px), ne changent pas de taille. Un pixel correspond toujours à un point sur votre écran. Cela les rend faciles à utiliser, mais elles peuvent nécessiter des ajustements pour la conception responsive.

- **Unités flexibles** : elles changent de taille en fonction d’autres facteurs, comme la taille de l’écran. Elles s’étirent et se rétrécissent pour s’adapter à différents appareils, ce qui les rend idéales pour la conception responsive.

Les unités de mesure : les 4 principales unités

👉 Il existe 4 principaux types d’unités CSS parmi lesquels choisir:

Unité	Basé sur	Exemple
em	Taille de la police du parent Sa valeur est calculée en fonction de la taille de la police de l'élément parent. Par exemple, si un élément a une taille de police de 16 pixels, 1 em équivaut à 16 pixels.	font-size: 2em; (2x plus grand)
rem	Taille de la police racine (html) Toujours calculée par rapport à la taille de la police de l’élément racine (<html>), et non à celle de l’élément parent.	font-size: 1.5rem;
px	Il représente un point sur l’écran, et sa valeur ne change pas, quelle que soit la taille de l’écran ou les paramètres de l’utilisateur.	font-size: 12px
vw	Largeur de la fenêtre (1vw = 1%)	width: 50vw; (50% de la largeur)
vh	Hauteur de la fenêtre (1vh = 1%)	height: 80vh; (80% de la hauteur)
%	Taille du parent	width: 50%; (50% du parent)

Les typographies : font-family

👉 La propriété **font-family** permet de définir la police de caractères d'un élément HTML.

```
div {
    font-family: "Times New Roman", Arial, serif;
}
```

Bonnes pratiques :

- Indiquez plusieurs polices (fallbacks) en cas d'indisponibilité de la première.
- Si le nom contient des espaces, il doit être placé entre guillemets.
- On termine par une famille générique (serif, sans-serif, monospace, etc.).

Les typographies : font-style

👉 La propriété **font-style** permet de définir le **style** de la police.

```css
div {
    font-style: normal;
    font-style: italic;
    font-style: oblique;
}
```

Indications :

- **normal** : texte standard.
- **italic** : texte en italique.
- **oblique** : texte incliné, proche de italic.

Les typographies : font-size

👉 La propriété **font-size** permet de définir la **taille** de la police.

```
div {
    font-size: 16px;
    font-size: 1.2em;
    font-size: 120%;
}
```

Rappel sur les unités :

- **px** : taille fixe en pixels.
- **em et rem** : taille relative (1em = 100% de la taille parent).
- **%** : pourcentage de la taille parent.

Les typographies : text-align

👉 La propriété **text-align** permet de définir **l'alignemement** d'un texte.

```
div {
    text-align: left;
    text-align: center;
    text-align: right;
    text-align: justify;
}
```

Bonnes pratiques :

- **left, right, center** : alignement classique.
- **justify** : ajuste les espacements pour aligner le texte sur les deux bords.

Nouveauté CSS :

text-align-last ajuste l'alignement de la dernière ligne.

Les typographies : text-decoration

👉 La propriété **font-weight** permet de définir **l'épaisseur** de la police.

```
div {
    text-decoration: none;
    text-decoration: underline;
    text-decoration: line-through;
    text-decoration: overline;
}
```

Bonnes pratiques :

- **none** : aucune décoration.
- **underline** : souligne le texte.
- **line-through** : barre le texte.
- **overline** : ligne au-dessus du texte.

Nouveauté CSS :

text-decoration-thickness pour contrôler l'épaisseur du soulignement.

Les typographies : text-transform

👉 La propriété **text-transform** permet de transformer le texte en majuscules/minuscules

```
div {
    text-transform: uppercase;
    text-transform: lowercase;
    text-transform: capitalize;
}
```

Bonnes pratiques :

- **uppercase** : met tout en majuscules.
- **lowercase** : met tout en minuscules.
- **capitalize** : met la première lettre de chaque mot en majuscule.

Les typographies : color

👉 La propriété **color** permet de définir la couleur d'un texte

```css
div {
    color: red;
    color: #FF0000;
    color: rgb(255,255,255);
    color: hsl(0,100%,50%);
}
```

Indication :

- **Nom de couleur** : red, blue, etc...
- **Code hexadécimal :** #FF0000.
- **RVB :** rgb(255, 0, 0).
- **HSL :** hsl(0, 100%, 50%).

Nouveauté CSS :

color-mix(in srgb, red, blue 50%) pour mélanger les couleurs.

Les typographies : line-height

👉 La propriété **line-height** définit l'espace entre les lignes d'un texte

```
div {
    line-height: normal;
    line-height: 1.5;
    line-height: 150%;
}
```

Indication :

- **normal** : espace par défaut.
- **Valeurs numériques** (ex: 1.5) : facteur multiplicatif.
- **Valeurs en %** : pourcentage de la taille de police.

Recommandation :

Une line-height de 1.5 améliore la lisibilité.

Les typographies : letter-spacing & word-spacing

👉 La propriété **letter-spacing** définit l'espace entre les lettres d'un texte

```
div {
   letter-spacing: 2px;
   letter-spacing: normal;
}
```

Indication :

- **normal** : espace standard.
- **2px** : ajoute un espace supplémentaire entre chaque lettre.

👉 La propriété **word-spacing** définit l'espace entre les mots d'un texte

```
div {
   word-spacing: 5px;
   word-spacing: normal;
}
```

Indication :

- **normal** : espace standard.
- **5px** : ajoute un espace supplémentaire entre chaque lettre.

Les couleurs en CSS

👉 La gestion des couleurs et des fonds en CSS est essentielle pour styliser une page web. CSS propose différentes façons de définir les couleurs et d'appliquer des arrière-plans aux éléments.

- **color** : Définit la couleur du texte.
- **background-color** : Définit la couleur de fond d’un élément.
- **background** : Propriété raccourcie pour gérer plusieurs aspects d’un fond.
- **opacity** : Ajuste la transparence d’un élément.
- **background-image** : Ajoute une image en arrière-plan.
- **background-size, background-position, background-repeat** : Contrôlent l'affichage des images de fond.

Les fonds en CSS

👉 Utiliser **background-color** pour une couleur de fond

```
p {
   background-color: #000000;
}
```

👉 Utiliser **background-image** pour une image de fond

```
p {
   background-image: url(‘mon_image.jpg’);
   background-image: url('image.jpg');
   background-size: cover;
   background-position: center;
   background-repeat: no-repeat;
}
```

Options utiles :

- **background-repeat: no-repeat;** : Évite la répétition de l’image.
- **background-size: cover;** : Ajuste l’image pour remplir l’élément.
- **background-position: center;** : Centre l’image.

Les variables CSS

👉 Les **variables CSS** permettent de stocker des valeurs réutilisables dans une feuille de style, facilitant ainsi la maintenance et l'organisation du code.

👉 Les variables CSS se déclarent dans un sélecteur (généralement :root) et commencent par --.

```
:root {
   --primary-color: #000000;
   --font-size-base: 16px;
}
```

- Utilisez **:root** pour les variables globales.
- Nommez vos variables de manière explicite (**--main-bg-color**, **--border-radius-small**).

Utiliser une variable CSS

👉 Pour utiliser une variable, on utilise la fonction **var()** :

```css
body {
    color: var(--primary-color);
    font-size: var(--font-size-base);
}
```

```css
:root {
    --primary-color: #000000;
    --font-size-base: 16px;
}
```

- Les variables peuvent être utilisées partout où une valeur est acceptée : couleurs, tailles, marges, espacements, etc.

👉 Si une **variable est indisponible** ou mal définie, on peut fournir une valeur de secours par défaut :

```css
p {
    color: var(--primary-color, #FF0000);
}
```

Modifier une Variable en Fonction du Contexte

👉 Si une variable est indisponible ou mal définie, on peut fournir une valeur par défaut :

```
.button {
   background-color: var(--primary-color);
}
```

```
.button:hover {
   --primary-color: #FFF;
}
```

- Les variables peuvent ainsi être modifiées dynamiquement via JavaScript.

```
document.documentElement.style.setProperty('--primary-color', '#ff5733');
```

https://www.afecdax.ovh/ressources/dev/css/css_demo_theme.html

Le reset CSS/normalize

👉 Lorsque vous créez une page web, **chaque navigateur applique ses propres styles par défaut aux éléments HTML**.
Ces styles peuvent varier, ce qui peut provoquer des incohérences d'affichage d'un navigateur à l'autre.

Pour résoudre ce problème, deux méthodes principales existent :

- **Reset CSS**
- **Normalize CSS.**

👉 **Pourquoi utiliser un reset ou un normalize CSS ?**

- Éviter les différences de styles par défaut entre navigateurs.
- Assurer une base cohérente pour le design.
- Mieux contrôler l'apparence de chaque élément HTML.
- Améliorer l'accessibilité et l'expérience utilisateur.

Le reset CSS/normalize

👉 **Qu’est-ce qu’un reset.css ?**

Un Reset CSS supprime complètement les styles par défaut des navigateurs.

Cela permet de repartir d'une base neutre pour appliquer vos propres styles.

👉 **Qu’est-ce qu’un normalize.css ?**

Contrairement au reset, Normalize CSS ne supprime pas tous les styles par défaut. Il harmonise les styles entre les navigateurs en conservant ceux qui sont utiles.

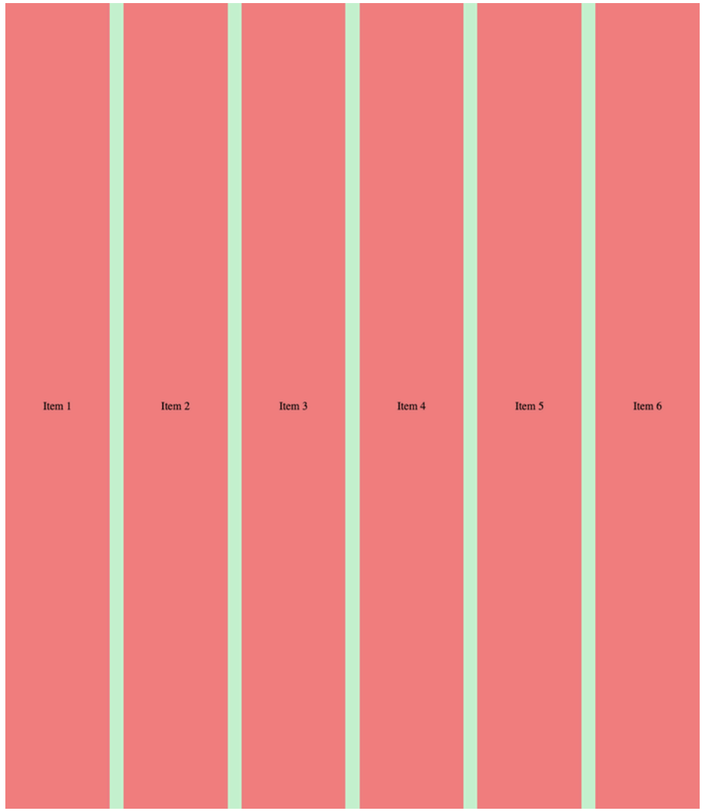
👉 **Exemple de normalize / reset prêt à l’emploi :**

- https://gist.github.com/rubensanroman/1670441
- https://necolas.github.io/normalize.css/

Exercice 1

Reproduire cette grille avec **grid**

- **6 colonnes**
- Espacement de **2%** entre colonnes



Exercice à reproduire :

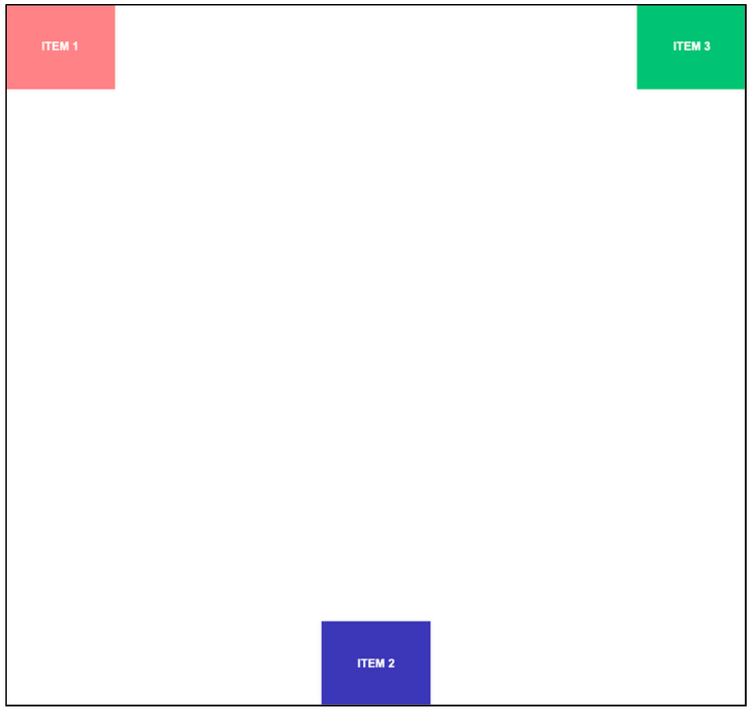
www.afecdax.ovh/ressources/dev/css/exercices/exercice_1.png

Base de départ :

https://www.afecdax.ovh/ressources/dev/css/exercices/css_exercice_1.html

Exercice 2

Reproduire cette grille avec **flex**



Exercice à reproduire :

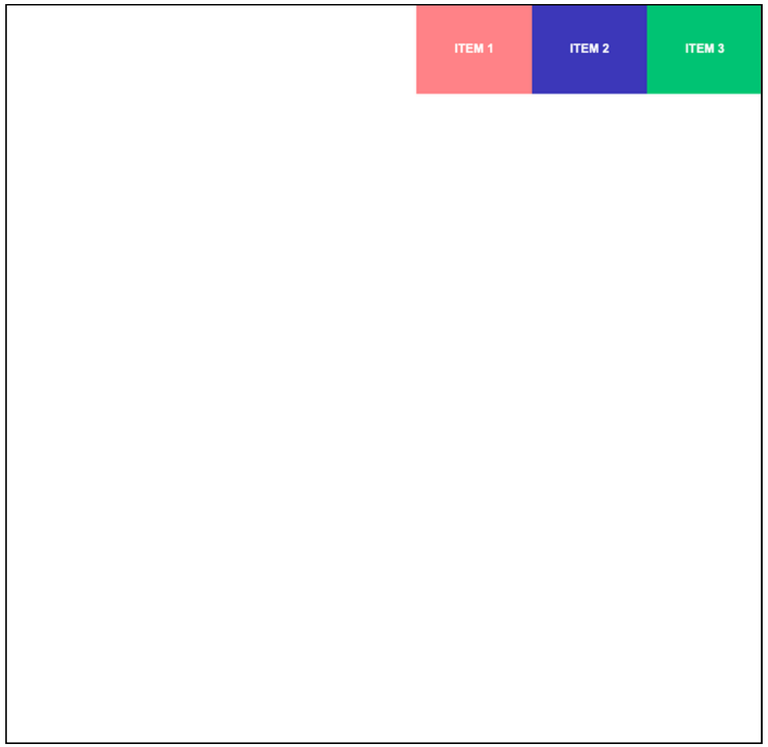
www.afecdax.ovh/ressources/dev/css/exercices/exercice_2.png

Base de départ :

https://www.afecdax.ovh/ressources/dev/css/exercices/css_exercice_2.html

Exercice 3

Reproduire cette grille avec **display:flex;**



Exercice à reproduire :

www.afecdax.ovh/ressources/dev/css/exercices/exercice_3.png

Base de départ :

https://www.afecdax.ovh/ressources/dev/css/exercices/css_exercice_3.html

Exercice 1

Reproduire cette capture d’écran

- Utiliser tous les moyens à votre disposition
- Consulter les documentations CSS
- Consulter la base du fichier HTML pour vous aider
- Utiliser la propriété box-shadow :

  https://developer.mozilla.org/en-US/docs/Web/CSS/box-shadow

www.afecdax.ovh/ressources/dev/css/exercices/exercice_4.png

www.afecdax.ovh/ressources/dev/css/exercices/exercice_4.html

À approfondir

- Présentation et fonctionnement des pseudo-elements en css
- Présentation et fonctionnement des pseudo-classes en css
- Présentation et fonctionnement de la syntaxe d'écriture BEM
- Présentation et fonctionnement des animations CSS
- Les dégradés en CSS
- fonction calc()
- Box-shadow
- Les position: relative; position: fixed; position:sticky et position: absolute

À approfondir

- Présentation et fonctionnement des pseudo-elements en css
- Présentation et fonctionnement des pseudo-classes en css
- Présentation et fonctionnement de la syntaxe d'écriture BEM
- Présentation et fonctionnement des animations CSS
- Les dégradés en CSS
- fonction calc()
- Box-shadow
- Les position: relative; position: fixed; position:sticky et position: absolute

QCM

Ouvrir le QCM

Ressources

CSS : Feuilles de style en cascade
https://developer.mozilla.org/fr/docs/Web/CSS

CSS Combinators
https://www.w3schools.com/css/css_combinators.asp

CSS Display
https://developer.mozilla.org/fr/docs/Web/CSS/display

CSS Les styles de bordures
https://developer.mozilla.org/fr/docs/Web/CSS/border-style

Les unités de mesure en CSS
https://www.codeur.com/tuto/css/unite-de-mesure-taille-px-em-rem/

Font-size-adjust
https://web.dev/articles/css-size-adjust

Text-align-last
https://developer.mozilla.org/en-US/docs/Web/CSS/text-align-last

Reset & normalize css
https://gist.github.com/rubensanroman/1670441
https://necolas.github.io/normalize.css/

Grid
https://css-tricks.com/snippets/css/complete-guide-grid/

Text Decoration Thickness
https://developer.mozilla.org/en-US/docs/Web/CSS/text-decoration-thickness

Color-mix()
https://developer.mozilla.org/en-US/docs/Web/CSS/color_value/color-mix

Flexbox
https://css-tricks.com/snippets/css/a-guide-to-flexbox/

Box-shadow:
https://developer.mozilla.org/en-US/docs/Web/CSS/box-shadow

Jeu pour apprendre CSS Grid
https://cssgridgarden.com/

Exercices
https://www.afecdax.ovh/ressources/dev/css/exercices/exercice_1.png
https://www.afecdax.ovh/ressources/dev/css/exercices/exercice_2.png
https://www.afecdax.ovh/ressources/dev/css/exercices/exercice_3.png
https://www.afecdax.ovh/ressources/dev/css/exercices/exercice_4.png
https://www.afecdax.ovh/ressources/dev/css/exercices/exercice_4.html

Démos
https://www.afecdax.ovh/ressources/dev/css/css_demo_theme.html