# 🎨 Cours CSS – Bases et notions modernes

## 1. Introduction au CSS

Le rôle du CSS est de **séparer le contenu** (ce que dit la page, écrit en HTML) de sa **présentation** (comment elle s'affiche). Sans CSS, une page web est fonctionnelle mais trés épurée. Avec le CSS, on peut construire une véritable identité visuelle.

☞ Trois façons d'appliquer du CSS :

1. **Inline** : directement dans la balise HTML via l'attribut `style`. → rapide mais difficile à maintenir.
2. **Interne** : dans un bloc `<style>` placé dans le `<head>`. → pratique pour des tests ou petites pages.
3. **Externe** : dans un fichier `.css` lié avec `<link>`. → la méthode recommandée en projet réel.

---

## 2. Les sélecteurs

Un sélecteur dit au navigateur *"à quels éléments appliquer les styles"*.

- **Balise** : cible tous les éléments d'un type (`p {}` → tous les paragraphes).
- **Classe** : commence par un `.` et peut être réutilisée (`.card {}`).
- **ID** : commence par `#`, unique dans la page (`#menu {}`).

On peut combiner ces sélecteurs :

- `div p` → tous les `<p>` à l'intérieur d'un `<div>`.
- `div > p` → uniquement les `<p>` enfants directs du `<div>`.

☞ Il existe aussi des sélecteurs plus précis :

- **Attribut** : `input[type="text"]` cible seulement certains champs.
- **Pseudo-classes** : `a:hover` change un lien quand la souris passe dessus.
- **Pseudo-éléments** : `p::first-line` agit sur la première ligne du texte.

---

## 3. Les propriétés de base

Les propriétés définissent **ce qu'on change** : couleur, police, taille, alignement... Exemples :

- `color` : couleur du texte.
- `background-color` : couleur de fond.
- `font-family` : police d'écriture.
- `line-height` : hauteur de ligne pour l'aération.

Un exemple simple :

```
body {
    color: #333; /* Texte en gris foncé */
    background-color: #f5f5f5; /* Fond gris clair */
```

```
    font-family: Arial, sans-serif;
}
```

---

## 4. Le Box Model

Tout élément HTML est vu comme une **boîte** par le navigateur. Cette boîte est composée de :

- **content** : le texte ou l'image.
- **padding** : l'espace intérieur (entre le contenu et la bordure).
- **border** : la bordure visible autour.
- **margin** : l'espace extérieur (séparation avec les autres éléments).

☞ Comprendre ce modèle est fondamental pour bien positionner et espacer les éléments.

---

## 5. Dimensions et unités

Il existe plusieurs façons de définir tailles et espacements :

- **px (pixels)** : fixe, indépendant de l'écran.
- **%** : proportion par rapport au parent.
- **em/rem** : relatif à la taille de police (du parent ou de la racine).
- **vh/vw** : proportion par rapport à la hauteur/largeur de la fenêtre.

☞ Exemple :

```
h1 {
    font-size: 2rem;
} /* 2x la taille de base */
.container {
    width: 80%;
} /* 80% de la largeur du parent */
```

---

## 6. Couleurs et arrière-plans

On peut exprimer les couleurs de plusieurs manières :

- **noms** (`red`, `blue`).
- **hexadécimal** (`#ff0000`).
- **rgb/rgba** (`rgb(255,0,0)` ou `rgba(255,0,0,0.5)`).
- **hsl** (`hsl(0, 100%, 50%)`).

On peut aussi utiliser des **dégradés** et des **images de fond** :

```
div {
    background: linear-gradient(to right, red, yellow);
```

```
}
```

---

## 7. Positionnement

Le CSS définit comment placer les éléments sur la page :

- `static` : par défaut, l'élément suit le flux naturel.
- `relative` : permet de déplacer l'élément **par rapport à sa position normale**.
- `absolute` : position par rapport au premier parent **positionné en relative (body par défaut)**.
- `fixed` : reste fixé même si on scrolle (ex. barre de navigation).
- `sticky` : se comporte comme `relative` mais devient fixe au scroll.

☞ Exemple : une barre fixée en haut :

```
.menu {
    position: fixed;
    top: 0;
    width: 100%;
}
```

---

## 8. Display et mise en page

Chaque élément a une valeur `display` qui définit sa nature :

- `block` : occupe toute la largeur (ex : `<div>`).
- `inline` : reste dans le flux du texte (ex : `<span>`).
- `inline-block` : comme inline mais avec dimensions que l'on peut modifier au CSS.
- `none` : l'élément disparaît.

Pour la mise en page moderne :

- **Flexbox** : organisation en ligne ou colonne, alignements.

Voir :

- l'article de CSS Tricks
- L'article de [Josh COMEAU] (https://www.joshwcomeau.com/css/interactive-guide-to-flexbox/)
- Les jeux pour apprendre comme flexbox froggy
- **Grid** : organisation en tableau, plus structurée.

Voir :

- l'article de CSS Tricks
- L'article de [Josh COMEAU] (https://www.joshwcomeau.com/css/interactive-guide-to-grid/)
- Les jeux pour apprendre comme grid garden ou grid attack.

## 9. Responsive Design

Le web doit s'adapter à tous les écrans.

### Media queries

```
@media (max-width: 600px) {
    body {
        font-size: 14px;
    }
}
```

### Container queries (nouveauté CSS moderne)

Elles adaptent un composant à la taille de son **contenant**, pas seulement de l'écran.

```
.card {
    container-type: inline-size;
}

@container (max-width: 400px) {
    .card {
        flex-direction: column;
    }
}
```

## 10. Transitions et animations

Pour ajouter du mouvement :

- **transition** : effet doux entre deux états.
- **@keyframes** : animations personnalisées.

Exemple :

```
button {
    transition: background 0.3s;
}
button:hover {
    background: orange;
}
```

## 11. Outils modernes CSS

### Variables CSS

Permettent de stocker des valeurs réutilisables.

```
:root {
    --couleur-primaire: #3498db;
}
h1 {
    color: var(--couleur-primaire);
}
```

### Fonctions modernes

- `clamp()` : définit une valeur adaptable entre min et max.
- `:has()` : permet de sélectionner un parent si un enfant existe ou valide une condition, comme la présence d'une classe.
- `aspect-ratio` : fixe les proportions (utile pour vidéos/images).

---

## 🌙 Bonus : Dark mode

CSS peut s'adapter automatiquement au thème du système.

```
@media (prefers-color-scheme: dark) {
    body {
        background: #111;
        color: #eee;
    }
}
```