UAA3 – Site Web

Partie 2 : Coder en CSS

1. Introduction au CSS 2
2. Intégration du CSS 4
3. Sélecteurs CSS 8
4. Propriétés de texte 13
5. Propriétés de boîte 17
6. Positionnement 22
7. Affichage 27
8. Flexbox 30
9. Couleurs 35
10. Unités de mesure 38
11. Notions avancées 41
12. Liste des propriétés CSS 44

1. INTRODUCTION AU CSS

Le CSS (Cascading Style Sheets) est un langage qui permet de styliser les éléments HTML d'une page web. En d'autres termes, il permet de définir l'apparence visuelle du contenu : couleurs, polices, mise en page, etc.

A. Sélecteurs, propriétés et valeurs

Le CSS fonctionne sur un système de règles composées de trois éléments :

- **Sélecteur :** Indique à quel(s) élément(s) HTML la règle doit s'appliquer.
    - Exemple : p (sélectionne tous les paragraphes), h1 (sélectionne tous les titres de niveau 1), .ma-classe (sélectionne tous les éléments avec la classe "ma-classe").
- **Propriété :** Spécifie l'aspect visuel à modifier (couleur, taille, police, etc.).
    - Exemple : color, font-size, font-family, background-color.
- **Valeur :** Définit la valeur de la propriété.
    - Exemple : red, 16px, Arial, blue.

Exemple de règle CSS à tester:

CSS

```css
p {
  color: blue;
  font-size: 14px;
}
```

Cette règle définit la couleur du texte de tous les paragraphes en bleu et la taille de la police à 14 pixels.

Syntaxe CSS :

Une règle CSS se compose de :

- Un sélecteur suivi d'accolades {}.
- Une ou plusieurs déclarations à l'intérieur des accolades.
- Chaque déclaration est composée d'une propriété, suivie de deux points : et d'une valeur, et se termine par un point-virgule ;.

Exercice :

Objectif : Mettre en pratique les bases du CSS en modifiant la couleur et la taille du texte d'un paragraphe.

Consignes :

1. Créer un fichier HTML avec un paragraphe de texte.
2. Créer un fichier CSS (ex: style.css).
3. Dans le fichier CSS, écrire une règle pour :
    - Sélectionner le paragraphe (avec le sélecteur p).
    - Modifier la couleur du texte en rouge (color: red;).
    - Modifier la taille de la police à 20 pixels (font-size: 20px;).
4. Lier le fichier CSS au fichier HTML (voir chapitre suivant sur l'intégration du CSS).

Exemple de code HTML :

HTML

```html
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <title>Introduction au CSS</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>

  <p>Ce texte est un paragraphe.</p>

</body>
</html>
```

Exemple de code CSS (style.css) :

CSS

```css
p {
  color: red;
  font-size: 20px;
}
```

En ouvrant le fichier HTML dans votre navigateur, vous devriez voir le texte du paragraphe en rouge et avec une taille de police de 20 pixels.

2. Intégration du CSS

Il existe trois façons d'intégrer du code CSS à une page web :

1. Feuilles de style externes

- C'est la méthode la plus courante et la plus recommandée.
- Le code CSS est écrit dans un fichier séparé avec l'extension .css.
- Ce fichier est lié à la page HTML via la balise <link> dans la section <head>.

Exemple à tester:

HTML

```html
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <title>Feuille de style externe</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>

  <p>Ce texte est stylisé par une feuille de style externe.</p>

</body>
</html>
```

Avantages :

- **Séparation du contenu et de la présentation :** Facilite la maintenance et l'évolution du site.
- **Réutilisation du code :** Un même fichier CSS peut être utilisé pour styliser plusieurs pages HTML.
- **Meilleure organisation :** Le code est plus clair et plus facile à lire.
- **Cache du navigateur :** Le fichier CSS est mis en cache par le navigateur, ce qui accélère le chargement des pages.

2. Feuilles de style internes

- Le code CSS est écrit directement dans la page HTML, dans la section <head>, à l'intérieur d'une balise <style>.

Exemple à tester :

HTML

```html
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <title>Feuille de style interne</title>
  <style>
    p {
      color: green;
      font-size: 18px;
    }
  </style>
</head>
<body>

  <p>Ce texte est stylisé par une feuille de style interne.</p>

</body>
</html>
```

Avantages :

- **Simple à mettre en place :** Pas besoin de créer un fichier séparé.
- **Utile pour des styles spécifiques à une page :** Si vous avez des styles qui ne s'appliquent qu'à une seule page.

Inconvénients :

- **Moins flexible :** Difficile de réutiliser le code CSS pour d'autres pages.
- **Code moins organisé :** Le code HTML et CSS sont mélangés.

3. Styles en ligne

- Le code CSS est écrit directement dans l'attribut style d'une balise HTML.

Exemple à tester :

```
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <title>Style en ligne</title>
</head>
<body>

  <p style="color: blue; font-size: 16px;">Ce texte est
stylisé avec un style en ligne.</p>

</body>
</html
```

Avantages :

- **Utile pour des styles très spécifiques :** Si vous avez besoin d'appliquer un style unique à un seul élément.

Inconvénients :

- **Très peu flexible :** Difficile à maintenir et à modifier.
- **Code très peu organisé :** Le code CSS est mélangé au code HTML.
- **Non recommandé :** À éviter autant que possible, car il rend le code difficile à lire et à maintenir.

Exercice :

Objectif : Créer une feuille de style externe et l'appliquer à une page HTML.

Consignes :

1. Créer un nouveau fichier HTML (ex: index.html).
2. Créer un nouveau fichier CSS (ex: style.css).
3. Dans le fichier CSS, écrire une règle pour modifier la couleur de fond du body en gris clair (background-color: lightgray;) et la couleur du texte de tous les paragraphes en bleu (color: blue;).
4. Lier le fichier CSS au fichier HTML en utilisant la balise <link> dans la section <head>.
5. Ajouter du contenu (titres, paragraphes) au fichier HTML pour observer les effets du CSS.

Exemple de code HTML (index.html) :

```HTML
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <title>Feuille de style externe</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>

  <h1>Titre principal</h1>
  <p>Ceci est un paragraphe de texte.</p>

</body>
</html>
```

Exemple de code CSS (style.css) :

```CSS
body {
  background-color: lightgray;
}

p {
  color: blue;
}
```

En ouvrant le fichier HTML dans votre navigateur, vous devriez voir le fond de la page en gris clair et le texte des paragraphes en bleu.

3. SÉLECTEURS CSS

Les sélecteurs CSS permettent de cibler précisément les éléments HTML auxquels appliquer des styles. Il existe différents types de sélecteurs, offrant une grande flexibilité pour styliser votre page web.

1. Sélecteurs de type

- Ciblent les éléments HTML en fonction de leur nom de balise.
- Exemple : p (sélectionne tous les paragraphes), h1 (sélectionne tous les titres de niveau 1), div (sélectionne toutes les divisions).

2. Sélecteurs de classe

- Ciblent les éléments HTML qui possèdent un attribut class avec une valeur spécifique.
- Syntaxe : .nom_de_la_classe
- Permettent de styliser plusieurs éléments de manière identique, même s'ils sont de types différents.

Exemple à tester :

HTML

```html
<p class="important">Ce paragraphe est important.</p>
<div class="important">Cette division est importante.</div>
```

CSS

```css
.important {
  font-weight: bold;
  color: red;
}
```

3. Sélecteurs d'ID

- Ciblent un élément HTML unique qui possède un attribut id avec une valeur spécifique.
- Syntaxe : #nom_de_l_id
- Un ID doit être unique dans une page HTML.

Exemple à tester:

HTML

```html
<h1 id="titre-principal">Titre principal</h1>
```

CSS

```css
#titre-principal {
  font-size: 36px;
  text-align: center;
}
```

4. Sélecteurs descendants

- Ciblent les éléments qui sont descendants d'un autre élément.
- Syntaxe : element1 element2 (sélectionne tous les element2 qui sont descendants de element1).

Exemple à teser :

HTML

```html
<div class="conteneur">
  <p>Paragraphe dans un conteneur.</p>
</div>
```

CSS

```css
.conteneur p {
  color: green;
}
```

5. Sélecteurs enfants

- Ciblent les éléments qui sont enfants directs d'un autre élément.
- Syntaxe : element1 > element2 (sélectionne tous les element2 qui sont enfants directs de element1).

Exemple à tester :

HTML

```html
<div class="conteneur">
  <p>Paragraphe enfant direct.</p>
  <div>
    <p>Paragraphe enfant indirect.</p>
  </div>
</div>
```

CSS

```css
.conteneur > p {
  color: blue;
}
```

Exercice :

Objectif : Mettre en pratique les différents types de sélecteurs CSS.

Consignes :

1. Créer un nouveau fichier HTML avec différents éléments (titres, paragraphes, divisions, etc.).
2. Ajouter des classes et des ID à certains éléments.
3. Créer un nouveau fichier CSS.
4. Dans le fichier CSS, écrire des règles pour :
    - Modifier la couleur du texte de tous les paragraphes en gris.
    - Mettre en gras les éléments avec la classe "important".
    - Centrer le titre avec l'ID "titre-principal".
    - Modifier la couleur de fond des paragraphes qui sont enfants directs d'une division avec la classe "conteneur".

Exemple de code HTML :

HTML

```html
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <title>Sélecteurs CSS</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>

  <h1 id="titre-principal">Titre principal</h1>

  <p>Paragraphe 1.</p>

  <div class="conteneur">
    <p>Paragraphe enfant direct.</p>
    <div>
      <p>Paragraphe enfant indirect.</p>
    </div>
  </div>

  <p class="important">Paragraphe important.</p>

</body>
</html>
```

Exemple de code CSS (style.css) :

```css
p {
  color: gray;
}

.important {
  font-weight: bold;
}

#titre-principal {
  text-align: center;
}

.conteneur > p {
  background-color: lightblue;
}
```

En ouvrant le fichier HTML dans votre navigateur, vous devriez observer les différents styles appliqués aux éléments en fonction des sélecteurs utilisés.

4. PROPRIÉTÉS DE TEXTE

Les propriétés de texte en CSS permettent de contrôler l'apparence du texte affiché sur une page web. Voici les principales propriétés :

1. Couleur

- color : Définit la couleur du texte.
- Valeurs possibles :
    - Noms de couleurs prédéfinies (ex: red, blue, green).
    - Valeurs hexadécimales (ex: #FF0000, #0000FF).
    - Valeurs RGB (ex: rgb(255, 0, 0), rgb(0, 0, 255)).

2. Taille

- font-size : Définit la taille de la police.
- Valeurs possibles :
    - Pixels (ex: 16px).
    - Pourcentages (ex: 150%).
    - Ems (ex: 1.2em).
    - Mots-clés (ex: small, medium, large).

3. Police

- font-family : Définit la famille de polices à utiliser.
- Valeurs possibles :
    - Noms de polices (ex: Arial, Verdana, Times New Roman).
    - Familles génériques (ex: serif, sans-serif, monospace).

4. Style

- font-style : Définit le style de la police.
- Valeurs possibles :
    - normal (par défaut)
    - italic (italique)
    - oblique (oblique)

5. Alignement

- text-align : Définit l'alignement du texte.
- Valeurs possibles :
    - left (gauche, par défaut)
    - center (centre)
    - right (droite)
    - justify (justifié)

6. Décoration

- text-decoration : Définit la décoration du texte.
- Valeurs possibles :
    - none (aucune, par défaut)
    - underline (souligné)
    - overline (surligné)
    - line-through (barré)

Exercice :

Objectif : Mettre en forme un texte avec différentes propriétés CSS.

Consignes :

1. Créer un nouveau fichier HTML avec un paragraphe de texte.
2. Créer un nouveau fichier CSS.
3. Dans le fichier CSS, écrire une règle pour le paragraphe (p) et appliquer les styles suivants :
    - Couleur du texte : bleu foncé.
    - Taille de la police : 18 pixels.
    - Famille de polices : Verdana, sans-serif.
    - Style de la police : italique.
    - Alignement du texte : justifié.
    - Décoration du texte : souligné.

Exemple de code HTML :

HTML

```html
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <title>Propriétés de texte</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>

  <p>Ce texte est un paragraphe.</p>

</body>
</html>
```

Exemple de code CSS (style.css) :

```css
p {
  color: darkblue;
  font-size: 18px;
  font-family: Verdana, sans-serif;
  font-style: italic;
  text-align: justify;
  text-decoration: underline;
}
```

En ouvrant le fichier HTML dans votre navigateur, vous devriez observer le texte du paragraphe avec les styles appliqués.

Remarques :

- Il est possible de combiner plusieurs propriétés de texte pour obtenir l'effet désiré.
- N'hésitez pas à expérimenter avec différentes valeurs et à observer les résultats.
- Pour une mise en forme plus avancée, vous pouvez utiliser d'autres propriétés de texte, comme font-weight, line-height, letter-spacing, etc.

5. PROPRIÉTÉS DE BOÎTE

Chaque élément HTML est considéré comme une boîte rectangulaire. Les propriétés de boîte permettent de contrôler les dimensions et l'espacement de ces boîtes.

1. Marges (margin)

- Définissent l'espace **extérieur** à la boîte, entre l'élément et les éléments voisins.
- Propriétés :
    - margin-top : Marge supérieure.
    - margin-right : Marge droite.
    - margin-bottom : Marge inférieure.
    - margin-left : Marge gauche.
    - margin : Raccourci pour définir les quatre marges en une seule déclaration.

Exemple à tester :

```css
div {
  margin-top: 20px;
  margin-right: 10px;
  margin-bottom: 30px;
  margin-left: 10px;
}

/* Équivalent à : */
div {
  margin: 20px 10px 30px 10px;
}
```

2. Bordures (border)

- Définissent les bordures de la boîte.
- Propriétés :
    - border-width : Épaisseur de la bordure.
    - border-style : Style de la bordure (ex: solid, dashed, dotted).
    - border-color : Couleur de la bordure.
    - border : Raccourci pour définir les trois propriétés précédentes en une seule déclaration.
    - Il est également possible de définir chaque côté de la bordure séparément (ex: border-top, border-right, etc.).

Exemple à tester:

```css
div {
  border-width: 2px;
  border-style: solid;
  border-color: black;
}

/* Équivalent à : */
div {
  border: 2px solid black;
}
```

3. Padding (padding)

- Définit l'espace **intérieur** à la boîte, entre la bordure et le contenu.
- Propriétés :
    - padding-top : Padding supérieur.
    - padding-right : Padding droit.
    - padding-bottom : Padding inférieur.
    - padding-left : Padding gauche.
    - padding : Raccourci pour définir les quatre paddings en une seule déclaration.

Exemple à tester:

```css
div {
  padding: 10px; /* 10px de padding sur les 4 côtés */
}
```

4. Largeur (width) et hauteur (height)

- Définissent les dimensions de la boîte.
- Valeurs possibles :
    - Pixels (ex: 200px).
    - Pourcentages (ex: 50%).
    - auto (valeur par défaut, la largeur/hauteur s'adapte au contenu).

Exemple à tester:

```css
div {
  width: 300px;
  height: 200px;
}
```

Exercice :

Objectif : Créer des boîtes avec différentes marges, bordures et padding.

Consignes :

1. Créer un nouveau fichier HTML avec plusieurs divisions ( ).
2. Créer un nouveau fichier CSS.
3. Dans le fichier CSS, écrire des règles pour chaque division en appliquant des styles différents :
    - Marges : variez les marges sur les différents côtés.
    - Bordures : variez l'épaisseur, le style et la couleur des bordures.
    - Padding : ajoutez du padding intérieur.
    - Largeur et hauteur : définissez des largeurs et hauteurs fixes.

Exemple de code HTML :

HTML

```html
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <title>Propriétés de boîte</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>

  <div class="boite1">Boîte 1</div>
  <div class="boite2">Boîte 2</div>
  <div class="boite3">Boîte 3</div>

</body>
</html>
```

Exemple de code CSS (style.css) :

```css
.boite1 {
  margin: 20px;
  border: 3px solid red;
  padding: 15px;
  width: 200px;
  height: 100px;
}

.boite2 {
  margin-top: 50px;
  border-bottom: 1px dashed blue;
  padding: 10px;
}

.boite3 {
  margin-left: 100px;
  border: 2px dotted green;
  padding: 5px;
  width: 150px;
}
```

En ouvrant le fichier HTML dans votre navigateur, vous devriez observer les différentes boîtes avec les styles de marges, bordures, padding, largeur et hauteur que vous avez définis.

6. POSITIONNEMENT

La propriété position en CSS permet de contrôler la manière dont un élément est positionné dans la page web. Il existe différents types de positionnement :

1. Positionnement statique (static)

- C'est le positionnement par défaut de tous les éléments HTML.
- L'élément est placé dans le flux normal du document, selon l'ordre dans lequel il apparaît dans le code HTML.
- Les propriétés top, right, bottom et left n'ont aucun effet sur un élément positionné statiquement.

2. Positionnement relatif (relative)

- L'élément est décalé par rapport à sa position **normale** dans le flux du document.
- Les propriétés top, right, bottom et left permettent de définir le décalage par rapport à la position d'origine.

Exemple à tester :

```css
.boite {
  position: relative;
  top: 20px;
  left: 30px;
}
```

La boîte sera décalée de 20 pixels vers le bas et de 30 pixels vers la droite par rapport à sa position normale.

3. Positionnement absolu (absolute)

- L'élément est retiré du flux normal du document et positionné par rapport à son **ancêtre positionné** le plus proche.
- Si aucun ancêtre n'est positionné, l'élément est positionné par rapport à la fenêtre du navigateur.
- Les propriétés top, right, bottom et left permettent de définir la position par rapport à l'ancêtre positionné ou à la fenêtre.

Exemple à tester :

HTML

```html
<div class="conteneur">
  <div class="boite">Boîte absolue</div>
</div>
```

CSS

```css
.conteneur {
  position: relative; /* L'ancêtre doit être positionné */
  width: 300px;
  height: 200px;
  border: 1px solid black;
}

.boite {
  position: absolute;
  top: 50px;
  left: 80px;
}
```

La boîte sera positionnée à 50 pixels du haut et 80 pixels de la gauche du coin supérieur gauche du conteneur.

4. Positionnement fixe (fixed)

- L'élément est retiré du flux normal du document et positionné par rapport à la **fenêtre du navigateur**.
- L'élément reste à la même position même lorsque l'utilisateur fait défiler la page.
- Les propriétés top, right, bottom et left permettent de définir la position par rapport à la fenêtre.

Exemple à tester:

```css
.menu {
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  background-color: lightgray;
}
```

Le menu restera en haut de la fenêtre, même lorsque l'utilisateur fait défiler la page.

Exercice :

Objectif : Positionner des éléments de manière absolue et relative.

Consignes :

1. Créer un nouveau fichier HTML avec une division principale ( ) et plusieurs éléments à l'intérieur (ex: paragraphes, images).
2. Créer un nouveau fichier CSS.
3. Dans le fichier CSS, appliquer les styles suivants :
    - Positionner la division principale de manière relative.
    - Positionner un élément à l'intérieur de la division de manière absolue, en utilisant les propriétés top et left pour le placer à un endroit précis.
    - Positionner un autre élément de manière relative, en utilisant les propriétés top et left pour le décaler légèrement par rapport à sa position normale.

Exemple de code HTML :

HTML

```html
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <title>Positionnement</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>

  <div class="conteneur">
    <p>Paragraphe 1</p>
    <img src="image.jpg" alt="Image">
    <p class="relatif">Paragraphe 2</p>
  </div>

</body>
</html>
```

Exemple de code CSS (style.css) :

```CSS
.conteneur {
  position: relative;
  width: 400px;
  height: 300px;
  border: 1px solid black;
}

img {
  position: absolute;
  top: 50px;
  left: 100px;
}

.relatif {
  position: relative;
  top: 20px;
  left: 50px;
}
```

En ouvrant le fichier HTML dans votre navigateur, vous devriez observer l'image positionnée de manière absolue dans le conteneur, et le paragraphe "Paragraphe 2" décalé par rapport à sa position normale.

7. AFFICHAGE

La propriété display en CSS contrôle la manière dont un élément est affiché dans la page web. Elle détermine comment l'élément interagit avec les autres éléments et comment il occupe l'espace.

Valeurs principales de display :

- **block :**
  - L'élément est affiché comme un bloc qui occupe toute la largeur disponible.
  - Il commence sur une nouvelle ligne et force les éléments suivants à se placer en dessous.
  - Les propriétés width et height peuvent être appliquées.
  - Exemples d'éléments block par défaut : <h1>, <p>, .
- **inline :**
  - L'élément est affiché en ligne avec le texte qui le précède et le suit.
  - Il n'occupe que l'espace nécessaire pour son contenu.
  - Les propriétés width et height ne peuvent pas être appliquées.
  - Exemples d'éléments inline par défaut : , <a>, .
- **inline-block :**
  - L'élément est affiché en ligne avec le texte qui le précède et le suit, mais il se comporte comme un bloc.
  - Il peut avoir une largeur et une hauteur définies.
  - Permet de combiner les avantages des éléments block et inline.
- **none :**
  - L'élément n'est pas affiché du tout.
  - Il est complètement retiré du flux du document, comme s'il n'existait pas.
  - Utile pour masquer des éléments dynamiquement avec JavaScript.

Exercice :

Objectif : Modifier l'affichage des éléments d'une page web avec la propriété display.

Consignes :

1. Créer un nouveau fichier HTML avec différents éléments (titres, paragraphes, liens, etc.).
2. Créer un nouveau fichier CSS.
3. Dans le fichier CSS, écrire des règles pour modifier l'affichage de certains éléments :
    - Transformer un paragraphe en élément inline.
    - Transformer un lien en élément block.
    - Créer un élément inline-block avec une largeur et une hauteur définies.
    - Masquer un élément avec display: none;.

Exemple de code HTML :

HTML

```html
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <title>Affichage</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>

  <h1>Titre principal</h1>
  <p>Ceci est un paragraphe.</p>
  <a href="#">Un lien</a>
  <div class="boite">Une boîte</div>
  <p class="cache">Ce paragraphe est caché.</p>

</body>
</html>
```

Exemple de code CSS (style.css) :

```css
p {
  display: inline;
}

a {
  display: block;
  background-color: lightblue;
  padding: 10px;
}

.boite {
  display: inline-block;
  width: 100px;
  height: 50px;
  background-color: lightgreen;
}

.cache {
  display: none;
}
```

En ouvrant le fichier HTML dans votre navigateur, vous devriez observer les changements d'affichage :

- Le paragraphe est affiché en ligne.
- Le lien occupe toute la largeur et est affiché comme un bloc.
- La boîte est un élément inline-block avec une largeur et une hauteur définies.
- Le paragraphe avec la classe "cache" est masqué.

8. FLEXBOX

Flexbox (Flexible Box Layout) est un module CSS qui offre un système flexible et puissant pour organiser, aligner et distribuer l'espace entre les éléments d'un conteneur, même lorsque leurs dimensions sont inconnues ou dynamiques.

Concepts fondamentaux

- **Conteneur flexible (flex container) :** L'élément parent auquel on applique display: flex;. Il devient alors un conteneur flexible et ses enfants directs deviennent des éléments flexibles.
- **Éléments flexibles (flex items) :** Les enfants directs du conteneur flexible.
- **Axe principal (main axis) :** L'axe principal selon lequel les éléments flexibles sont disposés. Par défaut, il est horizontal (de gauche à droite).
- **Axe secondaire (cross axis) :** L'axe perpendiculaire à l'axe principal.

Propriétés clés pour le conteneur flexible

- **display: flex; :** Active Flexbox sur l'élément.
- **flex-direction :** Définit l'orientation de l'axe principal :
    - row (par défaut) : axe horizontal, de gauche à droite.
    - row-reverse : axe horizontal, de droite à gauche.
    - column : axe vertical, de haut en bas.
    - column-reverse : axe vertical, de bas en haut.
- **justify-content :** Aligne les éléments flexibles le long de l'axe principal :
    - flex-start (par défaut) : début de l'axe.
    - flex-end : fin de l'axe.
    - center : centre de l'axe.
    - space-between : espace uniformément réparti entre les éléments.
    - space-around : espace uniformément réparti autour des éléments.
- **align-items :** Aligne les éléments flexibles le long de l'axe secondaire :
    - flex-start : début de l'axe.
    - flex-end : fin de l'axe.
    - center : centre de l'axe.
    - stretch (par défaut) : étire les éléments pour remplir le conteneur.
- **flex-wrap :** Contrôle le retour à la ligne des éléments flexibles :
    - nowrap (par défaut) : pas de retour à la ligne.

    - wrap : retour à la ligne si nécessaire.
    - wrap-reverse : retour à la ligne en sens inverse.
- **align-content :** Aligne les lignes d'éléments flexibles le long de l'axe secondaire (utile avec flex-wrap: wrap).

Propriétés clés pour les éléments flexibles

- **order :** Définit l'ordre d'affichage des éléments flexibles.
- **flex-grow :** Définit la capacité d'un élément flexible à grandir pour remplir l'espace disponible.
- **flex-shrink :** Définit la capacité d'un élément flexible à rétrécir si nécessaire.
- **flex-basis :** Définit la taille initiale d'un élément flexible.
- **flex :** Raccourci pour flex-grow, flex-shrink et flex-basis.

Exercice :

Objectif : Créer une navigation horizontale responsive avec Flexbox.

Consignes :

1. Créer un nouveau fichier HTML avec une liste non ordonnée (<ul>) contenant des liens de navigation (<a>).
2. Créer un nouveau fichier CSS.
3. Dans le fichier CSS, appliquer les styles suivants :
    - Transformer la liste (<ul>) en conteneur flexible avec display: flex.
    - Aligner les éléments de la liste horizontalement avec flex-direction: row.
    - Distribuer l'espace entre les éléments avec justify-content: space-around.
    - Centrer verticalement les éléments de la liste avec align-items: center.
    - Supprimer les puces de la liste.
    - Styliser les liens de navigation (couleur, padding, etc.).

Exemple de code HTML :

HTML

```html
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <title>Navigation Flexbox</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>

  <nav>
    <ul>
      <li><a href="#">Accueil</a></li>
      <li><a href="#">À propos</a></li>
      <li><a href="#">Services</a></li>
      <li><a href="#">Contact</a></li>
    </ul>
  </nav>

</body>
</html>
```

Exemple de code CSS :

```css
nav ul {
  display: flex;
  flex-direction: row;
  justify-content: space-around;
  align-items: center;
  list-style: none; /* Supprimer les puces */
  padding: 0; /* Remettre le padding à zéro */
}

nav li a {
  display: block; /* Permettre de définir un padding */
  padding: 10px 20px;
  color: blue;
  text-decoration: none;
}
```

En ouvrant le fichier HTML dans votre navigateur, vous devriez observer une navigation horizontale avec les liens espacés uniformément.

Bonus : Ajoutez une media query pour que la navigation devienne verticale lorsque la largeur de l'écran est inférieure à une certaine valeur. Cela rendra votre navigation responsive !

CSS

```css
nav ul {
  display: flex;
  flex-direction: row;
  justify-content: space-around;
  align-items: center;
  list-style: none;
  padding: 0;
}

nav li a {
  display: block;
  padding: 10px 20px;
  color: blue;
  text-decoration: none;
}

/* Media query pour écrans plus petits */
@media (max-width: 768px) {
  nav ul {
    flex-direction: column; /* Navigation verticale */
    align-items: flex-start; /* Alignement à gauche */
  }

  nav li {
    width: 100%; /* Les éléments occupent toute la largeur */
  }
}
```

9. Couleurs

Le CSS offre plusieurs façons de définir les couleurs des éléments sur une page web. Voici les méthodes les plus courantes :

1. Couleurs nommées

- CSS supporte un certain nombre de noms de couleurs prédéfinies.
- Exemples : red, blue, green, black, white, orange, purple, yellow.
- Avantage : facile à retenir et à utiliser.
- Inconvénient : nombre limité de couleurs disponibles.

2. Couleurs hexadécimales

- Les couleurs sont représentées par un code hexadécimal à 6 chiffres précédé d'un dièse (#).
- Chaque paire de chiffres représente l'intensité d'une couleur primaire (rouge, vert, bleu) en hexadécimal (de 00 à FF).
- Exemples : #FF0000 (rouge), #0000FF (bleu), #00FF00 (vert), #FFFFFF (blanc), #000000 (noir).
- Avantage : permet de définir un large éventail de couleurs.
- Inconvénient : peut être moins intuitif que les noms de couleurs.

3. Couleurs RGB

- Les couleurs sont définies en utilisant la fonction rgb() avec trois valeurs entre parenthèses, représentant l'intensité de chaque couleur primaire (rouge, vert, bleu).
- Chaque valeur est un nombre entier compris entre 0 et 255.
- Exemples : rgb(255, 0, 0) (rouge), rgb(0, 0, 255) (bleu), rgb(0, 255, 0) (vert).
- Avantage : permet de définir un large éventail de couleurs et offre plus de contrôle sur l'intensité de chaque couleur.

4. Couleurs RGBA

- Similaire à RGB, mais avec un quatrième paramètre pour l'alpha (transparence).
- L'alpha est une valeur entre 0 (transparent) et 1 (opaque).
- Exemples : rgba(255, 0, 0, 0.5) (rouge semi-transparent), rgba(0, 0, 255, 1) (bleu opaque).

Exercice :

Objectif : Utiliser différentes méthodes pour définir les couleurs en CSS.

Consignes :

1. Créer un nouveau fichier HTML avec plusieurs éléments (titres, paragraphes, divisions, etc.).
2. Créer un nouveau fichier CSS.
3. Dans le fichier CSS, écrire des règles pour modifier la couleur de fond et/ou la couleur du texte de chaque élément en utilisant différentes méthodes :
    - Couleurs nommées.
    - Couleurs hexadécimales.
    - Couleurs RGB.
    - Couleurs RGBA (pour ajouter de la transparence).

Exemple de code HTML :

HTML

```html
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <title>Couleurs CSS</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>

  <h1>Titre principal</h1>
  <p>Ceci est un paragraphe.</p>
  <div class="boite">Une boîte</div>

</body>
</html>
```

Exemple de code CSS (style.css) :

```css
h1 {
  color: blue; /* Couleur nommée */
}

p {
  color: #FF0000; /* Couleur hexadécimale */
}

.boite {
  background-color: rgb(0, 255, 0); /* Couleur RGB */
  color: rgba(255, 255, 255, 0.8); /* Couleur RGBA avec transparence */
}
```

En ouvrant le fichier HTML dans votre navigateur, vous devriez observer les différents éléments avec les couleurs que vous avez définies en utilisant les différentes méthodes.

10. UNITÉS DE MESURE

Les unités de mesure en CSS permettent de définir les dimensions des éléments (largeur, hauteur, marges, padding, etc.) et la taille du texte. Il existe différentes unités, chacune ayant ses propres caractéristiques et utilisations.

1. Pixels (px)

- Unité absolue qui correspond à un point sur l'écran.
- La taille d'un pixel peut varier légèrement en fonction de la résolution de l'écran.
- Facile à utiliser et à comprendre.
- Exemple : width: 100px; (largeur de 100 pixels).

2. Pourcentages (%)

- Unité relative qui dépend de la taille de l'élément parent.
- Exemple : width: 50%; (largeur égale à 50% de la largeur de l'élément parent).
- Permet de créer des mises en page flexibles qui s'adaptent à différentes tailles d'écran.

3. Ems (em)

- Unité relative qui dépend de la taille de la police de l'élément courant.
- 1em est égal à la taille de la police courante.
- Exemple : font-size: 1.2em; (taille de la police 20% plus grande que la taille de la police courante).
- Permet de créer des mises en page qui s'adaptent à la taille de la police choisie par l'utilisateur.

4. Rems (rem)

- Unité relative qui dépend de la taille de la police de l'élément racine (l'élément <html>).
- Similaire à em, mais plus facile à utiliser pour des mises en page complexes car la taille de référence est toujours la même.

Autres unités:

- pt (points) : Unité typographique.
- cm (centimètres) : Unité de mesure absolue.
- in (pouces) : Unité de mesure absolue.
- vh (viewport height) : Pourcentage de la hauteur de la fenêtre du navigateur.
- vw (viewport width) : Pourcentage de la largeur de la fenêtre du navigateur.

Exercice :

Objectif : Utiliser différentes unités de mesure pour définir les dimensions des éléments.

Consignes :

1. Créer un nouveau fichier HTML avec plusieurs éléments (titres, paragraphes, divisions, etc.).
2. Créer un nouveau fichier CSS.
3. Dans le fichier CSS, écrire des règles pour modifier les dimensions des éléments en utilisant différentes unités de mesure :
    - Pixels (px) pour définir des dimensions fixes.
    - Pourcentages (%) pour des dimensions relatives à l'élément parent.
    - Ems (em) pour des dimensions relatives à la taille de la police.

Exemple de code HTML :

HTML

```html
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <title>Unités de mesure</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>

  <div class="conteneur">
    <p>Paragraphe</p>
  </div>

</body>
</html>
```

Exemple de code CSS (style.css) :

```css
.conteneur {
  width: 500px; /* Largeur en pixels */
  height: 300px; /* Hauteur en pixels */
  border: 1px solid black;
}

p {
  font-size: 1.2em; /* Taille de la police en ems */
  width: 80%; /* Largeur en pourcentage de l'élément parent */
  margin: 1em; /* Marge en ems */
  padding: 10px; /* Padding en pixels */
}
```

En ouvrant le fichier HTML dans votre navigateur, vous devriez observer les différents éléments avec les dimensions définies en utilisant les différentes unités de mesure.

Remarques :

- Le choix de l'unité de mesure dépend du contexte et de l'effet désiré.
- Les unités relatives (%, em, rem) permettent de créer des mises en page plus flexibles et adaptatives.
- Il est important de bien comprendre les différences entre les unités de mesure pour choisir celle qui convient le mieux à chaque situation.

11. NOTIONS AVANCÉES

Ce chapitre aborde des notions CSS plus avancées qui vous permettront de créer des styles plus complexes et dynamiques.

1. Pseudo-classes

- Les pseudo-classes sont des mots-clés ajoutés à un sélecteur pour cibler un état spécifique d'un élément.
- Exemples :
    - :hover : cible l'élément lorsqu'on le survole avec la souris.
    - :active : cible l'élément lorsqu'il est cliqué.
    - :focus : cible l'élément lorsqu'il a le focus (ex: un champ de formulaire).
    - :first-child : cible le premier enfant d'un élément.
    - :nth-child(n) : cible le nième enfant d'un élément.

Exemple à tester :

```css
a:hover {
  color: red;
  text-decoration: underline;
}
```

Ce code change la couleur du lien en rouge et ajoute un soulignement lorsque l'utilisateur le survole avec la souris.

2. Pseudo-éléments

- Les pseudo-éléments sont des mots-clés ajoutés à un sélecteur pour cibler une partie spécifique d'un élément.
- Exemples :
    - ::before : insère du contenu avant l'élément.
    - ::after : insère du contenu après l'élément.
    - ::first-letter : cible la première lettre d'un élément.
    - ::first-line : cible la première ligne d'un élément.

Exemple à tester:

```css
p::first-letter {
  font-size: 2em;
  font-weight: bold;
}
```

Ce code agrandit et met en gras la première lettre de chaque paragraphe.

3. Media Queries

- Les media queries permettent d'appliquer des styles différents en fonction des caractéristiques de l'appareil (taille d'écran, orientation, résolution, etc.).
- C'est la base du responsive design, qui permet de créer des sites web qui s'adaptent à tous les types d'écrans.

Exemple à tester :

```css
@media (max-width: 768px) {
  body {
    font-size: 14px;
  }
}
```

Ce code réduit la taille de la police du body lorsque la largeur de l'écran est inférieure ou égale à 768 pixels.

Exercice :

Objectif : Créer un effet de survol sur un lien en utilisant une pseudo-classe.

Consignes :

1. Créer un nouveau fichier HTML avec un lien hypertexte.
2. Créer un nouveau fichier CSS.
3. Dans le fichier CSS, écrire une règle pour le lien (a) et utiliser la pseudo-classe :hover pour :
    - Changer la couleur du lien en rouge au survol.
    - Ajouter un soulignement au lien au survol.

Exemple de code HTML :

```HTML
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <title>Effet de survol</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>

  <a href="#">Lien avec effet de survol</a>

</body>
</html>
```

Exemple de code CSS (style.css) :

```CSS
a:hover {
  color: red;
  text-decoration: underline;
}
```

En ouvrant le fichier HTML dans votre navigateur, vous devriez observer l'effet de survol lorsque vous passez la souris sur le lien.

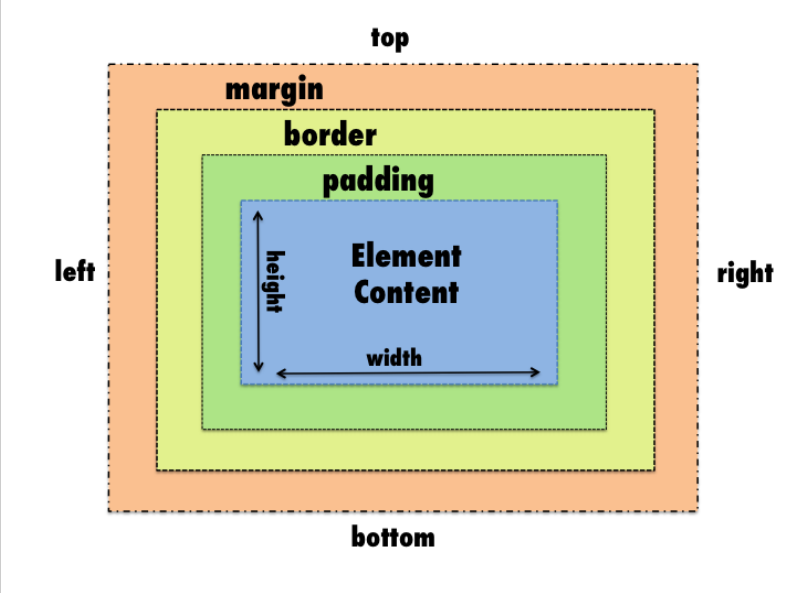
12. Liste des propriétés CSS

La liste complète des propriétés CSS (avec démos interactives) est disponible ici :
https://developer.mozilla.org/fr/docs/Web/CSS/Reference

Effet	Propriété	Valeur
Arrière-plan		
Définir la couleur de fond	background-color	<couleur>
		transparent
Définir l'image de fond	background-image	<lien>
Définir la taille de l'image de fond	background-size	<taille> cover contain auto
Bordure		
Définir la couleur de la bordure inférieure/supérieure gauche/droite	border-bottom-color border-top-color border-left-color border-right-color	<couleur>
Définir le style de la bordure inférieure/supérieure gauche/droite	border-bottom-style border-top-style border-left-style border-right-style	dotted dashed solid …
Définir l'épaisseur de la bordure inférieure/supérieure gauche/droite	border-bottom-width border-top-width border-left-width border-right-width	<taille> thin medium thick
Définir complètement la bordure	border	<border-width> <border-style> <border-color>
Définir l'épaisseur de la bordure (4 cotés)	border-width	<taille> thin medium

		thick
Définir la couleur de la bordure (4 cotés)	border-color	<couleur>
Définir les coins arrondis de la bordure (4 cotés)	border-radius	<taille> <taille> <taille> <taille>
Définir le style de la bordure (4 cotés)	border-style	dotted dashed solid …
Ajouter des ombres	box-shadow	<taille> <taille> <taille> <taille> <couleur>
Détermine si les bordures du tableau sont fusionnées ou séparées	border-collapse	collapse separate
Police		
Définir la famille de police	font-family	<police>
Définir la taille de la police	font-size	<taille>
Sélectionner une fonte italique ou oblique	font-style	italic oblique …
Définir des paramètres typographiques	font-variant	normal small-caps
Définir la graisse	font-weight	bold bolder lighter …
Texte		

Définir l'espace entre les lettres	letter-spacing	<taille>
Définir l'alignement	text-align	start end left right center justify
Définir des décorations	text-decoration	none underline overline line-through blink
Appliquer des transformations	text-transform	none capitalize uppercase lowercase
Définir l'espace entre les mots	word-spacing	<taille>
Couleur		
Définir la couleur	color	<couleur>
Définir l'opacité	opacity	inherit <nombre>
Listes et marqueurs		
Définir une image comme puce de liste	list-style-image	<lien>
Définir la position du marqueur	list-style-position	inside outside
Définir le style du marqueur	list-style-type	asterisks check circle

		diamond ...
Box model  		
Définir l'affichage utilisé pour le rendu d'un élément et la disposition utilisée pour ses éléments fils	display	none inline block inline-block ...
Indiquer qu'un élément doit être retiré du flux normal et doit être placé sur le coté droit/gauche de son conteneur. Cet élément devient alors flottant	float	left right none
Indique si un élément peut être situé à coté d'éléments flottants qui le précédent ou s'il doit être déplacé vers le bas pour être en dessous de ces éléments	clear	left right both none
Définir la hauteur du conteneur	height	auto <taille>
Définir la largeur du conteneur	width	auto <taille>
Définir la hauteur maximale/minimale d'un conteneur	max-height min-height	none <taille>

Définir la largeur maximale/minimale d'un conteneur	max-width min-width	none <taille>
Définir la taille des marges sur les 4 cotés du conteneur	margin	<margin-top> <margin-right> <margin-bottom> <margin-left>
Définir la taille des marges sur le coté inférieur/gauche/droit/supérieur	margin-bottom margin-left margin-right margin-top	auto length
Définir les écarts de remplissage sur les 4 cotés du conteneur	padding	<padding-top> <padding-right> <padding-bottom> <padding-left>
Définir l'écart des remplissages sur le coté inférieur/gauche/droit/supérieur	padding-bottom padding-left padding-right padding-top	length
Définir comment gérer le dépassement du contenu d'un élément dans son conteneur	overflow-x overflow-y	visible hidden scroll auto
Cacher un élément tout en conservant l'espace occupé dans lequel il aurait été visible	visibility	visible hidden collapse

Liste des sélecteurs CSS

La liste complète des sélecteurs CSS est disponible ici :
https://developer.mozilla.org/fr/docs/Learn/CSS/Building_blocks/Selectors

Nom du sélecteur	**Informations**	**Exemple(s)**

Universel	Tous les éléments	* {...}
Type	Tous les éléments de ce type	**h1** {...}
Groupement	Tous les éléments parmi ces balises	**h1**, **h2**, **h3** {...}
Classe	Plusieurs éléments de balises différentes sans forcément tous les sélectionner	.**maClasse** {...} **p**.**maClasse** {...}
Identifiant	Un seul élément en particulier	#**monId** {...} **div**#**monId** {...}
Descendant	Un élément qui se trouve dans un autre élément (peu importe le niveau)	.**maClasse h1** {...}
Enfant	Un élément qui se trouve dans un autre élément et qui est son enfant direct	.**maClasse** > **h1** {...}
Frère adjacent	Tous les éléments qui possèdent le même parent et qui sont adjacents	**h1** + **p** {...}
Frère général	Tous les éléments qui possèdent le même parent	**h1** ~ **p** {...}
Pseudo-classe[1]	Tous les éléments qui possède cette pseudo-classe	**p**:first-child {...} **p**:last-child {...} **p**:nth-child(**3**) {...} **p**:hover {...}
Attribut	Tous les éléments qui possède cet attribut	**img**[**height**="100"]{...} **img**[**src***=".png"]{...} **input**[**type**="text"]{...}

Quand un groupe de déclarations s'applique à plusieurs éléments distincts, on peut combiner les sélecteurs en une liste. Par exemple, si l'on souhaite cibler les paragraphes et les titre 1 pour qu'ils s'affichent en rouge, on pourrait écrire :

```
p {
color: red
```

```
}

h1 {
color: red
}
```

Mais dans ce cas, il est préférable de combiner les sélecteurs en une seule règle, en les séparant par une virgule :

```
p, h1 {
color: red
}
```

Liste des pseudo-classes CSS

La liste complète des pseudo-classes est disponible ici :
https://developer.mozilla.org/fr/docs/Web/CSS/Pseudo-classes

Pseudo-classe	**Information**	**Exemple**
:active	Cible un élément lorsque celui-ci est activé par l'utilisateur	*/* ne cible <a> que lorsqu'il est activé */* */* par exemple quand on clique dessus */* **a**:active { **color**: **red**; }
:checked	Cible un bouton radio, case à cocher ou option qui est coché ou activé par l'utilisateur	*/* cible n'importe quel bouton radio sélectionné, case à cocher cochée ou option sélectionnée */* **input**:checked { **margin-left**: 25px; }
:first-child	Cible un élément qui est le premier enfant de son parent	*/* Cible n'importe quel élément <p> qui est */* */* le premier fils de son élément parent */* **p**:first-child { **color**: **lime**; }

:first-of-type	Cible le premier élément d'un type donné parmi ceux d'un même élément parent	*/* Cible le premier élément <p> d'un type donné */* */* parmi ses éléments voisins */* **p**:first-of-type { **color**: **red**; }
:hover	Cible un élément qui est survolé par le pointeur de la souris	*/* Cible n'importe quel élément <a> lorsque */* */* celui-ci est survolé */* **a**:hover { **background-color**: **gold**; }
:last-child	Cible un élément qui est le dernier enfant de son parent	*/* Cible n'importe quel élément tant que */* */* celui-ci est le dernier enfant de son élément */* */* parent */* **li**:last-child { **background-color**: **lime**; }
:last-of-type	Cible un élément qui est le dernier enfant d'un type donné dans la liste des enfants de l'élément parent	*/* Cible n'importe quel paragraphe qui est */* */* le dernier paragraphe de son élément parent */* **p**:last-of-type { **color**: **lime**; }
:nth-child	Cible un élément qui est le nième enfant de son parent	*/* Cible n'importe quel paragraphe qui est */* */* le 2ième paragraphe de son élément parent */* **p**:nth-child(**2**) { **background-color**: **lightblue**; }
:link	Cible les liens qui n'ont pas encore été visités	*/* Cible les liens qui n'ont pas encore */* */* été visités */* **a**:link { **color**: **red**; }

:visited	Cible les liens qui ont été visités	*/* Cible les liens qui ont été visités */* **a**:visited { **color**: **red**; }