

# CONCEPTS DE BASE

# LES SÉLECTEURS CSS : SÉLECTEURS SIMPLES

# SÉLECTEUR D'ÉLÉMENT

Le **sélecteur d'élément** cible tous les éléments HTML correspondants.

Exemple :

```
p {  
    color: red;  
    font-size: 16px;  
}
```

Ce code applique une couleur rouge et une taille de police de 16px à tous les éléments de type <p> dans la page HTML.

## Sélecteur Cible

---

p                  Tous les éléments <p>

---

h1                Tous les éléments <h1>

---

nav              Tous les éléments <nav>

---

img              Tous les éléments <img>

---

## SYNTAXE

```
nom_element {  
    propriete: valeur;  
}
```

## EXEMPLES D'UTILISATION

```
p {  
  color: blue;  
}  
  
h1 {  
  font-size: 24px;  
}
```

# SÉLECTEUR DE CLASSE

Le **sélecteur de classe** cible tous les éléments HTML ayant la **classe spécifiée**.

Exemple :

```
/* CSS */  
.classe-exemple {  
    color: red;  
}  
  
<!-- HTML -->  
<p class="classe-exemple">Texte en rouge</p>
```

## SYNTAXE

```
.nom_classe {  
    propriete: valeur;  
}
```

### Tableau des propriétés CSS courantes :

Propriété	Description
color	Définit la couleur du texte
background	Définit la couleur de l'arrière-plan
font-size	Définit la taille du texte
border	Définit la bordure des éléments
margin	Définit l'espace extérieur des éléments
padding	Définit l'espace intérieur des éléments

## EXEMPLES D'UTILISATION

```
.alert {  
  color: red;  
}  
  
.primary {  
  background-color: blue;  
}
```

# SÉLECTEUR D'ID

Le **sélecteur d'ID** cible l'élément HTML ayant l'**ID spécifié**. Les ID sont **uniques** par page.

Exemple :

```
#monID {  
    font-size: 24px;  
    color: red;  
}
```

Dans cet exemple, le sélecteur d'ID `#monID` cible l'élément HTML ayant l'ID "monID" et lui applique les styles définis entre les accolades.

## SYNTAXE

```
#id_element {  
    propriete: valeur;  
}
```

## EXEMPLES D'UTILISATION

```
#header {  
    height: 70px;  
}  
  
#footer {  
    background-color: gray;  
}
```

# SÉLECTEURS D'ATTRIBUTS

# SÉLECTEUR D'ATTRIBUT SIMPLE

Le sélecteur d'attribut simple permet de sélectionner un **élément** en fonction de la présence d'un **attribut**.

Syntaxe	Description
[attribut]	Sélectionne tous les éléments ayant cet attribut
[attribut=valeur]	Sélectionne les éléments ayant cet attribut avec la valeur spécifiée

Exemple :

```
/* Sélectionne tous les éléments <a> possédant un attribut "href" */  
a[href] {  
    color: blue;  
}  
  
/* Sélectionne les éléments <input> ayant un attribut "type" égal à "button" */  
input[type="button"] {  
    background-color: green;  
}
```

## SYNTAXE

```
[element[attr]]
```

## EXEMPLES D'UTILISATION

```
/* Sélectionne tous les éléments <input> ayant un attribut "disabled" */
input[disabled] {
    background-color: grey;
}

/* Sélectionne toutes les balises <a> ayant un attribut "title" */
a[title] {
    color: blue;
}
```

# SÉLECTEUR D'ATTRIBUT AVEC VALEUR SPÉCIFIQUE

Le **sélecteur d'attribut** avec **valeur spécifique** permet de sélectionner un élément en fonction de la valeur d'un attribut.

## Exemple :

```
/* Pour sélectionner un lien externe */
a[href^="http"] {
  color: red;
}

/* Pour sélectionner un lien ayant une extension de fichier .pdf */
a[href$=".pdf"] {
  font-weight: bold;
}

/* Pour sélectionner un lien contenant le mot "documentation" */
a[href*="documentation"] {
  text-decoration: underline;
}
```

## SYNTAXE

```
[element[attr="value"]]
```

## EXEMPLES D'UTILISATION

```
/* Sélectionne tous les éléments <input> ayant un attribut "type" dont la valeur est "checkbox" */  
input[type="checkbox"] {  
    margin: 5px;  
}  
  
/* Sélectionne toutes les balises <a> ayant un attribut "target" dont la valeur est "_blank" */  
a[target="_blank"] {  
    text-decoration: underline;  
}
```

# SÉLECTEUR D'ATTRIBUT AVEC PRÉFIXE OU SUFFIXE

Le **sélecteur d'attribut** avec préfixe ou suffixe permet de sélectionner un élément en fonction d'une partie spécifique de la valeur d'un attribut.

Exemple de préfixe : `[attr^="pre"]`

Exemple de suffixe : `[attr$="suf"]`

Syntaxe	Description
<code>[attr^="value"]</code>	Sélectionne les éléments dont l'attribut commence par "value"
<code>[attr\$="value"]</code>	Sélectionne les éléments dont l'attribut se termine par "value"

Exemple en CSS :

```
/* Sélectionne tous les liens dont l'attribut href commence par "https" */
a[href^="https"] {
    color: green;
}

/* Sélectionne tous les liens dont l'attribut href se termine par ".pdf" */
a[href$=".pdf"] {
    color: red;
}
```

## SYNTAXE

```
[element[attr^="prefix"]] /* prefix */  
[element[attr$="suffix"]] /* suffix */  
[element[attr*= "part"]] /* part of string */
```

## EXEMPLES D'UTILISATION

```
/* Sélectionne tous les éléments <a> dont l'attribut "href" commence par "https://" */

```

# SÉLECTEURS DE RELATION

## SÉLECTEUR DESCENDANT (E F)

Ce sélecteur sélectionne tous les éléments F qui sont **descendants** de E.

Exemple :

```
/* Sélectionne tous les éléments <p> à l'intérieur d'une <div> */  
div p {  
    color: red;  
}
```

## SYNTAXE

```
E F {  
  /* règles CSS */  
}
```

## EXEMPLES D'UTILISATION

```
article p {  
  color: blue;  
}
```

Tous les **éléments** < p > descendants d'un **élément** < article > seront en **bleu**.

## SÉLECTEUR D'ENFANT (E > F)

Ce sélecteur sélectionne tous les éléments F qui sont des **enfants directs** de E.

Exemple :

```
.parent > .child {  
  color: blue;  
}
```

Ceci changera la couleur du texte des éléments de classe .child qui sont les enfants directs de l'élément de classe .parent.

## SYNTAXE

```
E > F {  
  /* règles CSS */  
}
```

## EXEMPLES D'UTILISATION

```
article > p {  
  color: green;  
}
```

Les éléments `<p>` **enfants directs** d'un élément `<article>` seront en **vert**.

## SÉLECTEUR D'ADJACENCE DIRECTE (E + F)

Ce sélecteur sélectionne l'élément F qui suit **immédiatement** l'élément E.

Exemple d'utilisation :

```
h1 + p {  
    font-size: 1.2em;  
}
```

Dans cet exemple, tous les éléments <p> qui suivent **directement** un élément <h1> auront une taille de police de 1,2em.

## SYNTAXE

```
E + F {  
  /* règles CSS */  
}
```

## EXEMPLES D'UTILISATION

```
h2 + p {  
  font-weight: bold;  
}
```

Les éléments `<p>` suivant immédiatement un élément `<h2>` auront un texte en **gras**.

## SÉLECTEUR D'ADJACENCE GÉNÉRALE (E ~ F)

Ce sélecteur sélectionne tous les éléments F qui suivent E au sein du **même parent**.

Exemple :

```
E ~ F {  
  /* les styles ici seront appliqués aux éléments F qui suivent E */  
}
```

- E : premier élément
- F : élément(s) suivant(s) dans la même hiérarchie

## SYNTAXE

```
E ~ F {  
  /* règles CSS */  
}
```

## EXEMPLES D'UTILISATION

```
h2 ~ p {  
  font-style: italic;  
}
```

Les éléments <p> suivant un élément <h2> dans le même parent auront un texte en **italique**.

# SÉLECTEURS DE PSEUDO-CLASSES

# :HOVER, :FOCUS, :ACTIVE

Les **pseudo-classes** `:hover`, `:focus` et `:active` permettent de définir des **styles** pour un élément lorsqu'il est **survolé**, possède le **focus** ou est **activé**.

```
/* Exemple de code CSS */
a:hover {
    background-color: yellow;
}

a:focus {
    color: red;
}

a:active {
    font-weight: bold;
}
```

## SYNTAXE

```
élément:hover { propriété: valeur; }  
élément:focus { propriété: valeur; }  
élément:active { propriété: valeur; }
```

## EXEMPLES D'UTILISATION

```
button:hover {  
    background-color: blue;  
}  
  
input:focus {  
    border-color: red;  
}  
  
button:active {  
    color: white;  
}
```

# :FIRST-CHILD, :LAST-CHILD, :NTH-CHILD

Ces **pseudo-classes** permettent de cibler un élément selon sa position parmi ses frères et sœurs.

Pseudo-classe	Description
:first-child	Cible le premier élément parmi les frères
:last-child	Cible le dernier élément parmi les frères
:nth-child(n)	Cible le n-ème élément parmi les frères

## Exemple d'utilisation :

```
p:first-child {  
  color: red;  
}  
  
p:last-child {  
  color: blue;  
}  
  
p:nth-child(2) {  
  color: green;  
}
```

## SYNTAXE

```
élément:first-child { propriété: valeur; }
élément:last-child { propriété: valeur; }
élément:nth-child(expression) { propriété: valeur; }
```

## EXEMPLES D'UTILISATION

```
li:first-child {  
    font-weight: bold;  
}  
  
li:last-child {  
    font-style: italic;  
}  
  
li:nth-child(2n) {  
    background-color: lightgray;  
}
```

# :CHECKED, :DISABLED, :ENABLED

Les pseudo-classes **:checked**, **:disabled** et **:enabled** permettent de cibler des éléments selon leur état dans les **formulaires**.

Pseudo-classe	Description
:checked	Cible les éléments cochés
:disabled	Cible les éléments désactivés
:enabled	Cible les éléments activés et prêts à être utilisés

Exemple d'utilisation :

```
input:checked {  
    background-color: green;  
}  
  
input:disabled {  
    background-color: lightgray;  
}  
  
input:enabled {  
    background-color: white;  
}
```

## SYNTAXE

```
élément:checked { propriété: valeur; }
élément:disabled { propriété: valeur; }
élément:enabled { propriété: valeur; }
```

Pseudo Classe	Description	Exemple
:checked	Cible les éléments qui sont coché ( <i>checkbox, radio</i> )	input[type="checkbox"]:checked
:disabled	Cible les éléments inactifs ( <i>input, button</i> )	input[type="text"]:disabled
:enabled	Cible les éléments actifs ( <i>input, button</i> )	input[type="text"]:enabled

## EXEMPLES D'UTILISATION

```
input[type="checkbox"]::checked {  
    background-color: green;  
}  
  
input:disabled {  
    opacity: 0.5;  
}  
  
input:enabled:hover {  
    border-color: red;  
}
```

# SÉLECTEURS DE PSEUDO-ÉLÉMENTS

# ::BEFORE, ::AFTER

Ces sélecteurs permettent d'**insérer du contenu** avant ou après l'élément sélectionné.

Exemple :

```
h1::before {  
    content: "Avant - ";  
}  
  
h1::after {  
    content: " - Après";  
}
```

## Sélecteur   Action

---

::before      Insère du contenu **avant** l'élément sélectionné

---

::after        Insère du contenu **après** l'élément sélectionné

## SYNTAXE

```
element::before {  
    content: "texte";  
}  
element::after {  
    content: "texte";  
}
```

## EXEMPLES D'UTILISATION

```
p::before {  
    content: "Important : ";  
    font-weight: bold;  
}  
p::after {  
    content: " (*)";  
}
```

## ::FIRST-LETTER, ::FIRST-LINE

Ces sélecteurs permettent de sélectionner la **première lettre** ou **ligne** de l'élément sélectionné.

```
p::first-letter {  
    font-size: 2em;  
    font-weight: bold;  
}  
  
p::first-line {  
    text-decoration: underline;  
}
```

## SYNTAXE

```
element::first-letter {  
    font-size: 2em;  
}  
element::first-line {  
    text-transform: uppercase;  
}
```

## EXEMPLES D'UTILISATION

```
p::first-letter {  
    font-size: 2em;  
}  
p::first-line {  
    text-transform: uppercase;  
}
```

# ::SELECTION

Ce sélecteur permet d'appliquer un **style spécifique** sur la **partie sélectionnée** du texte.

Exemple :

```
::selection {  
background-color: yellow;  
color: black;  
}
```

## SYNTAXE

```
element::selection {  
    background-color: #ccc;  
    color: #000;  
}
```

## EXEMPLES D'UTILISATION

```
p::selection {  
    background-color: #ccc;  
    color: #000;  
}
```

# COMBINER PLUSIEURS SÉLECTEURS

# GROUPEMENT DE SÉLECTEURS

Le **groupeement de sélecteurs** permet de cibler plusieurs éléments avec une seule règle CSS.

```
h1, h2, h3 {  
    font-family: Arial, sans-serif;  
}
```

## SYNTAXE

```
sélecteur1, sélecteur2, sélecteur3 {  
    propriété: valeur;  
}
```

## EXEMPLES D'UTILISATION

```
h1, h2, h3 {  
    color: blue;  
}
```

# COMBINAISONS DE SÉLECTEURS

Combinaisons de sélecteurs permet de cibler des **éléments spécifiques** en fonction de leur **relation**.

Syntaxe	Description
A, B	Cible les éléments A et B
A B	Cible les éléments B descendant d'A
A > B	Cible les éléments B enfants d'A
A + B	Cible les éléments B immédiatement après A
A ~ B	Cible les éléments B précédés par A

## SYNTAXE

```
sélecteur1 sélecteur2 {  
    propriété: valeur;  
}
```

## EXEMPLES D'UTILISATION

```
ul.nav li {  
    margin-right: 10px;  
}
```

```
h1 + p {  
    font-style: italic;  
}
```

# PROPRIÉTÉS DE TEXTE

# COLOR

La propriété `color` est utilisée pour changer la couleur du texte.

```
p {  
  color: red;  
}
```

# SYNTAXE

```
p {  
  color: blue;  
}  
h1 {  
  color: #00ff00;  
}
```

## EXEMPLES D'UTILISATION

Dans l'exemple de syntaxe, le premier bloc de code change la couleur de tout texte à l'intérieur d'un élément `<p>` en **bleu**. Le deuxième bloc de code change la couleur de tout texte à l'intérieur d'un élément `<h1>` en **vert clair**.

# FONT-SIZE

La propriété `font-size` est utilisée pour changer la taille du texte.

- Exemple : `p { font-size : 15px; }`
- Exemple : `h1 { font-size : 200%; }`

# SYNTAXE

```
p {  
  font-size: 20px;  
}  
h1 {  
  font-size: 2em;  
}
```

## EXEMPLES D'UTILISATION

Dans l'exemple de syntaxe, le premier bloc de code change la taille du texte à l'intérieur d'un élément `<p>` en **20 pixels**. Le deuxième bloc de code change la taille du texte à l'intérieur d'un élément `<h1>` en **2 fois** la taille de la police actuelle.

# TEXT-ALIGN

La propriété `text-align` est utilisée pour aligner le texte.

## Valeurs Description

---

`left` Aline le texte à gauche.

`right` Aline le texte à droite

`center` Centre le texte.

`justify` Les lignes de texte sont étirées de façon à aligner les deux bords du texte avec les bords gauche et droit de la ligne.

# SYNTAXE

```
p {  
    text-align: center;  
}  
h1 {  
    text-align: left;  
}
```

## EXEMPLES D'UTILISATION

Dans l'exemple de syntaxe, le premier bloc de code aligne le texte à l'intérieur d'un élément `<p>` au **centre**.  
Le deuxième bloc de code aligne le texte à l'intérieur d'un élément `<h1>` à **gauche**.

```
p {  
    text-align: center;  
}  
  
h1 {  
    text-align: left;  
}
```

# INTRODUCTION AUX PROPRIÉTÉS DE FOND

# background-color

La propriété **background-color** permet de définir la **couleur de fond** d'un élément HTML.

**Exemple :**

```
p {  
    background-color: red;  
}
```

# COULEURS NOMMÉES

Couleurs nommées	Codes hexadécimaux	Codes RGB
Red	#FF0000	rgb(255, 0, 0)
Blue	#0000FF	rgb(0, 0, 255)
Green	#008000	rgb(0, 128, 0)

## EXEMPLES D'UTILISATION

```
/* Couleur de fond en nom de couleur */
body {
    background-color: red;
}

/* Couleur de fond en code HEX */
h1 {
    background-color: #FF5733;
}

/* Couleur de fond en RGB */
p {
    background-color: rgb(255, 255, 0);
}
```

## OPACITÉ ET TRANSPARENCE

Pour définir la **transparence** d'une couleur, utilisez la fonction **\*\*rgba\*\*** avec un **quatrième paramètre alpha**.

```
/* Couleur semi-transparente */  
div {  
    background-color: rgba(255, 255, 255, 0.5);  
}
```

Couleur	Code rgba
Transparent	rgba(255, 255, 255, 0)
Semi-transparent	rgba(255, 255, 255, 0.5)
Opaque	rgba(255, 255, 255, 1)

# BACKGROUND-IMAGE

# BACKGROUND-IMAGE

La propriété **background-image** permet d'ajouter une image en arrière-plan d'un élément **HTML**.

```
.selector {  
    background-image: url ('chemin/vers/image.jpg');  
}
```

Propriété	Valeurs possibles
-----------	-------------------

background-image	url, none, gradient
------------------	---------------------

# SYNTAXE

Pour définir une **image de fond**, utilisez la syntaxe suivante :

```
selector {  
  background-image: url("image-url.jpg");  
}
```

# EXEMPLES D'UTILISATION

```
/* Image de fond pour le **corps de la page** */
body {
    background-image: url("background.jpg");
}

/* Image de fond pour un **div spécifique** */
#my-div {
    background-image: url("my-div-background.png");
}
```

# TYPES D'IMAGES SUPPORTÉES

Les formats d'images supportés par la propriété background-image sont :

- **JPEG** (.jpg ou .jpeg)
- **PNG** (.png)
- **GIF** (.gif)
- **SVG** (.svg)
- **WebP** (.webp)
- **BMP** (.bmp)

# PROPRIÉTÉS DE DIMENSION

# PROPRIÉTÉS DE DIMENSION

En CSS, les **propriétés de dimension** nous permettent de contrôler la **hauteur** et la **largeur** des éléments.

Propriété	Description
width	définit la largeur de l'élément
height	définit la hauteur de l'élément

# WIDTH

La propriété `width` définit la largeur d'un élément.

- Si la valeur est spécifiée en pixels, la largeur totale est déterminée par `width` plus les marges, les bordures et le padding.
- Si la valeur est spécifiée en pourcentage, la largeur totale est calculée en fonction de la largeur de l'élément parent.

# SYNTAXE

La syntaxe pour utiliser la propriété de **largeur** est `width: valeur;`

```
div {  
    width: 100px;  
}
```

# EXEMPLES D'UTILISATION

```
div.exemple {  
    width: 50%;  
}  
p.exemple {  
    width: auto;  
}
```

# HEIGHT

La propriété `height` définit la **hauteur** d'un élément.

# SYNTAXE

La syntaxe pour utiliser la propriété de hauteur est `height: valeur;`

```
div {  
    height: 100px;  
}
```

# EXEMPLES D'UTILISATION

```
div.exemple {  
    height: 50%;  
}  
p.exemple {  
    height: auto;  
}
```

# MODÈLE DE BOÎTE CSS (BOX MODEL)

# MODÈLE DE BOÎTE CSS

Le modèle de boîte CSS est un concept fondamental dans le styling CSS. Il décrit comment le **contenu**, la **padding**, la **bordure** et la **marge** d'un élément se combinent pour former le box.

# PADDINGS

Le **padding** est l'espace entre le contenu de l'élément et sa bordure. Il est transparent.

Propriété CSS	Description
padding-top	Espacement du haut
padding-bottom	Espacement du bas
padding-right	Espacement à droite
padding-left	Espacement à gauche

## PROPRIÉTÉ 'PADDING'

La propriété padding est utilisée pour contrôler l'espace entre le contenu de l'élément et sa bordure.

Syntaxe :

```
element {  
  padding: 10px;  
}
```

## VALEURS UNIQUES ET MULTIPLES

La propriété padding peut accepter une, deux, trois, ou quatre valeurs :

- padding: 10px; tous les côtés ont le même padding.
- padding: 10px 20px; top/bottom ont 10px et left/right ont 20px.
- padding: 10px 15px 20px; top a 10px, left/right ont 15px, et bottom a 20px.
- padding: 10px 20px 30px 40px; spécifie le padding pour top, right, bottom, et left.

# MARGES

La marge est l'espace **autour** de l'élément. Elle est également **transparente**.

## PROPRIÉTÉ 'MARGIN'

La propriété margin est utilisée pour contrôler l'espace **autour** de l'élément.

Syntaxe :

```
element {  
  margin: 10px;  
}
```

## VALEURS UNIQUES ET MULTIPLES

La propriété margin peut accepter une, deux, trois, ou quatre valeurs tout comme padding.

- Si une **valeur unique** est fournie, elle sera utilisée pour **tous les côtés**.
- Avec **deux valeurs**, la première représente le haut et le bas, la deuxième la gauche et la droite.
- Avec **trois valeurs**, la première représente le haut, la deuxième la gauche et la droite, et la dernière le bas.
- Avec **quatre valeurs**, elles représentent respectivement le haut, la droite, le bas, et la gauche.

# BORDURES

La bordure entoure le **padding** (si il est présent) et le **contenu**.

## PROPRIÉTÉ 'BORDER'

La propriété `border` est utilisée pour contrôler le **style**, la **largeur** et la **couleur** de la bordure de l'élément.

Syntaxe :

```
element {  
  border: 1px solid black;  
}
```

## STYLES, LARGEURS ET COULEURS DE BORDURE

La propriété `border` peut être divisée en `border-width`, `border-style`, et `border-color`. Chaque propriété peut avoir une à quatre valeurs, spécifiant les côtés.

- `border-width` : détermine la **largeur** de la bordure. Les valeurs peuvent être en pixels ou en pourcentages. Par exemple : `border-width: 2px;`
- `border-style` : définit le **style** de la bordure. Certaines valeurs possibles sont `solid`, `dotted`, `dashed` etc. Par exemple : `border-style: solid;`
- `border-color` : spécifie la **couleur** de la bordure. Les valeurs peuvent être n'importe quel terme de couleur CSS standard ou valeurs hexadécimales. Par exemple : `border-color: #ff0000;`

Propriété	Description	Exemple de valeur
<code>border-width</code>	Largeur de la bordure	<code>2px</code>
<code>border-style</code>	Style de la bordure	<code>solid</code>
<code>border-color</code>	Couleur de la bordure	<code>#ff0000</code>

# INTRODUCTION AUX BASES DE GRID

# DÉFINITION DE CSS GRID

**CSS Grid** est un système de mise en page **bidimensionnel** qui permet de positionner les éléments HTML dans une grille.

# CONCEPT DE BASE DE CSS GRID

## GRILLE

Une grille est composée de **lignes** et de **colonnes** qui créent des espaces pour placer les éléments HTML.

## LIGNES ET COLONNES

Les **lignes** sont les rangées horizontales et les **colonnes** sont les rangées verticales de la grille.

## ESPACEMENT ET ALIGNEMENT

Avec **CSS Grid**, il est facile d'espacer et d'aligner les éléments de la grille.

Propriété	Description
grid-gap	Permet de définir l'espacement entre les lignes et les colonnes.
align-items	Permet d'aligner les éléments de la grille sur l'axe des blocs.
justify-items	Permet d'aligner les éléments de la grille sur l'axe en ligne.

Exemple:

```
.container {  
    display: grid;  
    grid-template-columns: repeat(3, 1fr);  
    grid-gap: 10px;  
    align-items: center;  
    justify-items: center;  
}
```

# SYNTAXE DE CSS GRID

## display: grid

Pour commencer à utiliser **CSS Grid**, définissez la propriété `display` d'un élément HTML à `grid`:

```
.container {  
  display: grid;  
}
```

## grid-template-rows / grid-template-columns

Définissez la **structure** de lignes et de colonnes de la grille en utilisant les propriétés `grid-template-rows` et `grid-template-columns` :

```
.container {  
    grid-template-rows: 100px 200px;  
    grid-template-columns: 1fr 2fr 1fr;  
}
```

## grid-gap / grid-row-gap / grid-column-gap

Créez des espacements entre les **lignes** et les **colonnes** en utilisant les propriétés grid-gap, grid-row-gap et grid-column-gap :

```
.container {  
  grid-gap: 10px;  
  grid-row-gap: 20px;  
  grid-column-gap: 30px;  
}
```

## grid-template-areas

Utilisez grid-template-areas pour définir des **zones nommées** dans la grille :

```
.container {  
  grid-template-areas:  
    "header header header"  
    "sidebar main main"  
    "footer footer footer";  
}
```

## grid-auto-rows / grid-auto-columns

Définissez une **taille automatique** pour les lignes et les colonnes qui ne sont pas explicitement spécifiées avec `grid-auto-rows` et `grid-auto-columns`:

```
.container {  
    grid-auto-rows: 100px;  
    grid-auto-columns: 1fr;  
}
```

## grid-auto-flow

Contrôlez l'ordre de placement des éléments avec `grid-auto-flow`:

```
.container {  
    grid-auto-flow: column;  
}
```

Valeur	Description
row	Place les éléments par rangées
column	Place les éléments par colonnes
dense	Remplit les espaces vides

# EXEMPLES D'UTILISATION DE CSS GRID

## MISE EN PAGE BASIQUE

```
<div class="container">
  <div class="item">1</div>
  <div class="item">2</div>
  <div class="item">3</div>
</div>
```

```
.container {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-gap: 10px;
}

.item {
  background-color: coral;
  padding: 20px;
}
```

## MISE EN PAGE RESPONSIVE

```
.container {  
    display: grid;  
    grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));  
    grid-gap: 10px;  
}
```

# INTRODUCTION AUX BASES DE FLEXBOX

# CSS FLEXBOX

**Flexbox** est un modèle de mise en page **unidimensionnel** permettant de bien organiser et aligner les éléments HTML de manière simple et flexible.

## CONCEPTS DE BASE DE FLEXBOX

- **Flex container:** élément parent qui applique la propriété display: flex
- **Flex items:** éléments enfants du container
- **Ordre et alignement:** contrôle de la position des éléments

# SYNTAXE DE FLEXBOX

Pour créer un **flex container**, utilisez la propriété `display: flex;`

```
.container {  
  display: flex;  
}
```

## PROPRIÉTÉS POUR LE CONTAINER

Propriété	Description
<b>flex-direction</b>	Définit la direction des éléments
<b>flex-wrap</b>	Contrôle si les éléments peuvent se casser ou pas
<b>justify-content</b>	Alignement des éléments sur l'axe principal
<b>align-items</b>	Alignement des éléments sur l'axe transversal
<b>align-content</b>	Alignement des éléments lorsqu'il y a plusieurs lignes

## PROPRIÉTÉS POUR LES ITEMS

Propriété	Description
<b>order</b>	Contrôle l'ordre des éléments
<b>flex-grow</b>	Contrôle la proportion de la croissance des éléments
<b>flex-shrink</b>	Contrôle la proportion de la réduction des éléments
<b>flex-basis</b>	Définit la taille initiale des éléments
<b>flex</b>	Raccourci pour définir flex-grow, flex-shrink, et flex-basis

# EXEMPLES D'UTILISATION DE FLEXBOX

## ALIGNEMENT D'ÉLÉMENTS

```
.container {  
  display: flex;  
  justify-content: center;  
  align-items: center;  
}
```

## MISE EN PAGE BASIQUE

```
.container {  
  display: flex;  
  flex-wrap: wrap;  
}  
  
.item {  
  flex-basis: 30%;  
}
```

## MISE EN PAGE RESPONSIVE

```
.container {  
  display: flex;  
  flex-wrap: wrap;  
}  
  
.item {  
  flex-basis: calc(100%/3);  
}  
  
@media (max-width: 768px) {  
  .item {  
    flex-basis: 50%;  
  }  
}
```

