CSS et Flexbox

# CSS (Cascading Style Sheets)

Le CSS est un langage de feuille de style utilisé pour décrire la présentation et l'apparence d'un document HTML. Il permet de contrôler les couleurs, les polices, les marges, les espacements, les mises en page et d'autres aspects visuels d'une page web. Le CSS fonctionne en associant des règles de style à des éléments HTML, ce qui permet de séparer la structure du contenu (HTML) de sa présentation (CSS).

Voici un exemple de fichier CSS externe :

/\* styles.css \*/

body {

font-family: Arial, sans-serif;

background-color: #f2f2f2;

}

h1 {

color: #333;

font-size: 24px;

margin-bottom: 20px;

}

p {

color: #666;

line-height: 1.5;

}



### 

### Intégration du CSS dans le HTML :

Pour intégrer le fichier CSS externe dans votre document HTML, vous pouvez utiliser la balise **<link>** dans la section **<head>** :

<!DOCTYPE html>

<html>

**<head>**

**<link rel="stylesheet" href="styles.css">**

**</head>**

<body>

<!-- Contenu de votre page -->

</body>

</html>



### Quelques propriétés de base du CSS

Voici quelques exemples de propriétés CSS couramment utilisées :

**color** : définit la couleur du texte.

**font-size** : définit la taille de la police.

**background-color** : définit la couleur de fond.

**margin** : définit les marges autour d'un élément.

**padding** : définit l'espacement à l'intérieur d'un élément.

**border** : définit les bordures d'un élément.

**width** : définit la largeur d'un élément.

**height** : définit la hauteur d'un élément.

**display** : définit comment un élément est affiché (par exemple, en bloc ou en ligne).

Ces propriétés ne sont qu'un aperçu et il en existe de nombreuses autres pour personnaliser et styliser vos pages web.

Le CSS introduit plusieurs manières de penser l’affichage des différents éléments sur une page. L'attribut display en CSS permet de spécifier le comportement de positionnement d'un élément HTML. Il offre différentes valeurs qui définissent les types de mise en page disponibles. Voici quelques-uns des types de mise en page couramment utilisés :

**block** : Cela définit l'élément comme un bloc qui occupe toute la largeur disponible et commence sur une nouvelle ligne. Les éléments de type div sont généralement affichés en tant que blocs.

**inline** : Cela définit l'élément comme un élément en ligne qui ne crée pas de rupture de ligne. Il occupe seulement l'espace nécessaire pour son contenu. Les éléments de type span sont souvent affichés en tant qu'éléments en ligne.

**inline-block** : Cela combine les caractéristiques des éléments en ligne et des blocs. Il permet à l'élément de s'afficher en tant qu'élément en ligne tout en conservant la possibilité de définir une largeur et une hauteur.

**flex** : Cela permet de créer des mises en page flexibles en utilisant le modèle de flexbox. Il offre un contrôle puissant sur l'alignement, l'espacement et l'ordre des éléments à l'intérieur d'un conteneur.

**grid** : Cela permet de créer des mises en page en grille en utilisant le modèle de grille CSS. Il permet de définir des colonnes et des lignes pour positionner et organiser les éléments dans un conteneur.

Voici un exemple minimal de ce qui est possible avec display: grid pour transformer le conteneur .container en une grille avec deux colonnes égales. Les éléments enfants du conteneur, tels que **<h1>**, **<p>** et **<ul>**, sont affichés en tant qu'éléments en ligne à l'intérieur de chaque cellule de la grille.

<div class="container">

<h1>Titre principal</h1>

<p>Paragraphe de texte.</p>

<ul>

<li>Élément 1</li>

<li>Élément 2</li>

<li>Élément 3</li>

</ul>

</div>



.container {

display: grid;

grid-template-columns: 1fr 1fr;

grid-gap: 20px;

}

h1, p, ul {

/\* Les éléments enfants du conteneur s'afficheront en tant qu'éléments en ligne \*/

display: inline;

}



Grâce à l'utilisation de **display: grid**, nous pouvons organiser et positionner les éléments à l'intérieur du conteneur de manière flexible et réactive. Cela nous offre un contrôle précis sur la mise en page et la structure de notre contenu.

# 

# Flexbox

Le module des boîtes flexibles, aussi appelé « *flexbox* » est un module CSS qui facilite la création de mises en page flexibles et réactives. Il offre un contrôle puissant sur la disposition des éléments d'un conteneur, en leur permettant de se redimensionner automatiquement et de s'ajuster à l'espace disponible.

Lorsqu'on décrit les boîtes flexibles comme une méthode de disposition unidimensionnelle, on indique en fait que les *flexbox* gèrent une seule dimension à la fois : une ligne ou une colonne.

Lorsqu'on travaille avec les boîtes flexibles, deux axes interviennent : l'axe principal (main axis en anglais) et l'axe secondaire (cross axis en anglais). L'axe principal est défini par la propriété flex-direction et l'axe secondaire est l'axe qui lui est perpendiculaire. Tout ce que nous manipulons avec les boîtes flexibles fera référence à ces axes.

### Propriétés des Flexbox

**display** : définit un conteneur comme un conteneur flex. Tous les enfants d’une Flexbox disposent automatiquement du contexte de la flexbox qui les contient

.container {

display: flex; /\* or inline-flex \*/

}

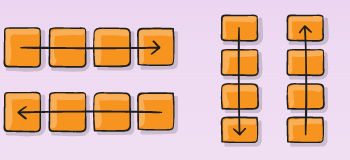


**flex-direction** : définit la direction principale (main axis) des éléments dans le conteneur flex. La valeur de cette propriété peut être **row** (en ligne), **row-reverse** (en ligne inversée), **column** ou **column-reverse**

.container {

flex-direction: row | row-reverse | column | column-reverse;

}



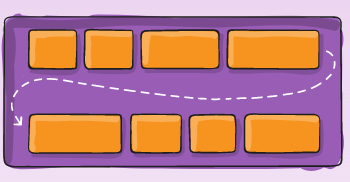
**flex-wrap** : définit si les éléments flex doivent être ajustés sur une seule ligne ou plusieurs lignes. Les valeurs de la propriété peuvent être **nowrap**, **wrap** ou **wrap-reverse**. nowrap (par défaut): tous les objets sont sur la même ligne, wrap: les objets flex s’adaptent et vont à la ligne automatiquement, de haut en bas et wrap-reverse: les objets flex s’adaptent et vont à la ligne automatiquement, de bas en haut

.container {

flex-wrap: nowrap | wrap | wrap-reverse;

}





**justify-content** : définit l'alignement des éléments le long de l'axe principal du conteneur flex, ce qui permet d’aligner le long de l’axe horizontal les objets contenus dans la flexbox.

.container {

justify-content: flex-start | flex-end | center | space-between | space-around | space-evenly | start | end | left | right ... + safe | unsafe;

}



Les valeurs de l’attribut sont divisées en deux avec **flex-start** (par défaut), **flex-end**, **start**, **end**, **left**, **right**, **center**, **space-around**, **space-evenly** d’une part et **safe** and **unsafe** de l’autre.

**flex-start** : les éléments sont alignés vers le début de la direction de flexion.

**flex-end** : les éléments sont alignés vers la fin de la direction de flexion.

**start** : les éléments sont alignés vers le début de la direction d'écriture.

**end** : les éléments sont alignés vers la fin de la direction d'écriture.

**left** : les éléments sont alignés vers le bord gauche du conteneur, sauf si cela n'a pas de

sens avec la direction de flexion, auquel cas il se comporte comme start.

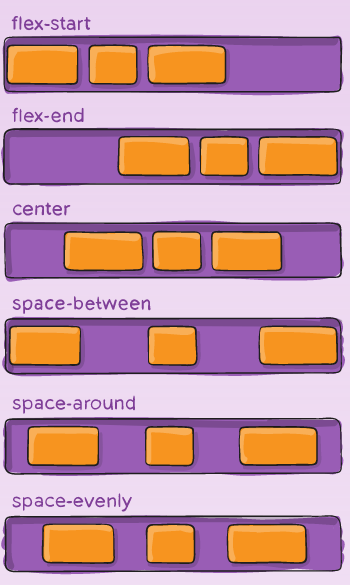
**right** : les éléments sont alignés vers le bord droit du conteneur, sauf si cela n'a pas de sens avec la direction de flexion, auquel cas il se comporte comme end.

**center** : les éléments sont centrés le long de la ligne.

**space-between** : les éléments sont répartis uniformément dans la ligne ; le premier élément est à la ligne de départ, le dernier élément est à la ligne de fin.

**space-around** : les éléments sont répartis uniformément dans la ligne avec un espace égal autour d'eux. Notez que visuellement, les espaces ne sont pas égaux, car tous les éléments ont un espace égal des deux côtés. Le premier élément aura un espace d'une unité contre le bord du conteneur, mais deux unités d'espace entre le prochain élément, car ce dernier a son propre espacement qui s'applique.

**space-evenly** : les éléments sont répartis de manière à ce que l'espacement entre deux éléments (et l'espace aux extrémités) soit égal.



Notez que la prise en charge de ces valeurs par les navigateurs est nuancée. Par exemple, space-between n'a jamais été pris en charge par certaines versions de Edge, et start/end/left/right ne sont pas encore disponibles dans Chrome. MDN dispose de graphiques détaillés à ce sujet. Les valeurs les plus sûres sont flex-start, flex-end et center.

Il existe également deux mots-clés supplémentaires que vous pouvez associer à ces valeurs : safe et unsafe. En utilisant safe, vous vous assurez que quelle que soit la manière dont vous effectuez ce type de positionnement, vous ne pouvez pas pousser un élément de manière à ce qu'il se rende hors de l'écran (par exemple, en dehors du haut) de telle manière que le contenu ne puisse pas être défilé également.

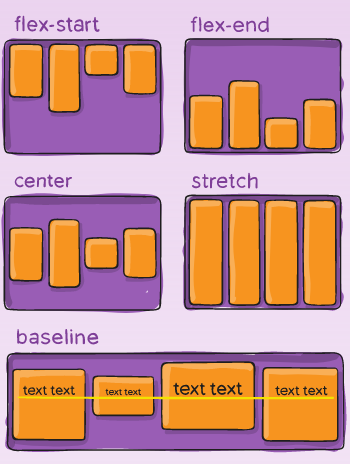
**align-items** : définit l'alignement des éléments le long de l'axe transversal du conteneur flex. Les propriétés de cet attribut peuvent être : **stretch**, **flex-start**, **flex-end**, **center**, **baseline**, **first baseline**, **last baseline**, **start**, **end**, **self-start**, **self-end** accompagné de **safe** ou **unsafe**

.container {

align-items: stretch | flex-start | flex-end | center | baseline | first baseline | last baseline | start | end | self-start | self-end + ... safe | unsafe;

}





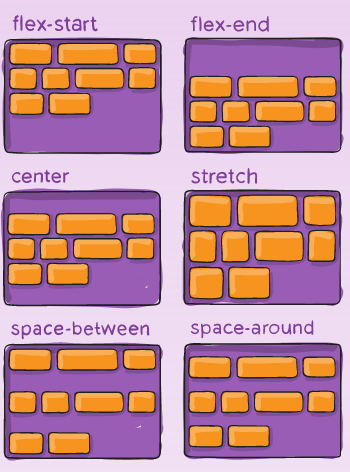
**align-content** : définit l'alignement des lignes de flex lorsque celles-ci s'étendent sur plusieurs lignes.

.container {

align-content: flex-start | flex-end | center | space-between | space-around | space-evenly | stretch | start | end | baseline | first baseline | last baseline + ... safe | unsafe;

}





De plus, chaque enfant d'un conteneur flex peut avoir ses propres propriétés flex :

**order** : définit l'ordre dans lequel les éléments apparaissent dans le conteneur flex.

**flex-grow** : définit la capacité d'un élément à se développer pour remplir l'espace disponible.

**flex-shrink** : définit la capacité d'un élément à se rétrécir lorsqu'il y a moins d'espace disponible.

**flex-basis** : définit la taille de base d'un élément avant que l'espace supplémentaire ou manquant ne soit distribué.

Ces propriétés Flexbox vous permettent de créer des mises en page réactives et flexibles en utilisant des conteneurs et des éléments flex.

### 

### Exemple d'utilisation de Flexbox :

Voici un exemple d'utilisation de Flexbox pour aligner horizontalement des éléments. Ce code crée un conteneur flex avec trois éléments à l'intérieur. Les éléments sont centrés horizontalement et verticalement dans le conteneur.

<!DOCTYPE html>

<html>

<head>

**<style>**

**.container {**

**display: flex;**

**justify-content: center;**

**align-items: center;**

**height: 300px;**

**}**

**.item {**

**margin: 10px;**

**padding: 20px;**

**background-color: #f2f2f2;**

**}**

**</style>**

</head>

<body>

**<div class="container">**

**<div class="item">Élément 1</div>**

**<div class="item">Élément 2</div>**

**<div class="item">Élément 3</div>**

**</div>**

</body>

</html>



Sources

[A Complete Guide to Flexbox | CSS-Tricks](https://css-tricks.com/snippets/css/a-guide-to-flexbox/)

[Les concepts de base pour flexbox - CSS : Feuilles de style en cascade | MDN](https://developer.mozilla.org/fr/docs/Web/CSS/CSS_flexible_box_layout/Basic_concepts_of_flexbox)