CSS Grid

1. Qu'est-ce que CSS Grid ?

> `CSS Grid` est un système de mise en page CSS qui permet de structurer un contenu à la fois en lignes et en colonnes (de manière bidimensionnelle).
> Il est idéal pour créer des mises en page complexes et flexibles grâce à un contrôle total des alignements, des espacements et des proportions.

1.1. Les principaux concepts de CSS Grid

CSS Grid repose sur deux éléments principaux :

- Le conteneur grid : Défini par `display: grid;` , c'est l'élément parent.
- Les éléments de grille : Les enfants du conteneur qui s'organisent automatiquement selon les réglages du conteneur.

1.1.1. Conteneur grid

> Le conteneur grid est configuré par des propriétés CSS qui définissent l'agencement des lignes et colonnes.

1.1.1.1. Propriétés principales

1.2. Définir des colonnes avec `grid-template-columns`

HTML :
Ce code crée un conteneur avec 3 divs enfants.

```html
<div class="grid-container">
 <div>1</div>
 <div>2</div>
 <div>3</div>
</div>
```

CSS :
Ici, 3 colonnes de tailles différentes sont définies : 100px, 200px et la troisième occupe toute la largeur restante.

```
.grid-container {
 display: grid;
 grid-template-columns: 100px 200px 1fr;
 gap: 10px; /* Espace entre les colonnes */
}

.grid-container div {
 background-color: lightblue;
 text-align: center;
 padding: 10px;
}
```

1.3. Ajouter des colonnes dynamiques avec `repeat()`

> La fonction `repeat()` est utile pour générer des colonnes répétées et dynamiques.

HTML :

```html
<div class="grid-container">
 <div>1</div>
 <div>2</div>
 <div>3</div>
 <div>4</div>
 <div>5</div>
</div>
```

CSS :

```css
.grid-container {
 display: grid;
 grid-template-columns: repeat(3, 1fr); /* 3 colonnes de tailles égales */
 gap: 10px;
}

.grid-container div {
 background-color: lightgreen;
 text-align: center;
 padding: 10px;
}
```

- Explication :
  - `repeat(3, 1fr)` crée 3 colonnes de tailles égales.
  - On peut aussi combiner avec `auto-fill` et `minmax()` , par exemple :

```css
grid-template-columns: repeat(auto-fill, minmax(150px, 1fr));
```

Ce code crée autant de colonnes que possible avec une largeur minimum de `150px` et s'adaptant à l'espace disponible.

1.4. Gérer les lignes avec *grid-template-rows*

> La propriété `grid-template-rows` est un outil essentiel pour contrôler la hauteur des lignes dans une grille CSS.
>
> Elle vous permet de définir précisément la taille de chaque ligne, que ce soit en unités fixes (comme `px` ou `rem`), proportionnelles (comme `fr`) ou automatiques (`auto`).
>
> Grâce à cette flexibilité, vous pouvez créer des mises en page claires et structurées, adaptées à tout type de contenu.

HTML :

```html
<div class="grid-container">
 <div>A</div>
 <div>B</div>
 <div>C</div>
</div>
```

CSS :

```css
.grid-container {
 display: grid;
 grid-template-rows: 50px 100px 1fr; /* Hauteur des lignes */
 gap: 10px; /* Espace entre les lignes */
}

.grid-container div {
 background-color: lightcoral;
 text-align: center;
 padding: 10px;
}
```

2. Placement d'élément au sein de la grille

2.1. Étendre des éléments avec *grid-row-start*, *grid-column-start*, ...

> Les propriétés `grid-row-start`, `grid-row-end`, `grid-column-start` et `grid-column-end` permettent de positionner précisément un élément dans une grille en indiquant ses points de départ et de fin. Sans utiliser l'abréviation `span`, vous avez un contrôle très détaillé sur la portée des éléments dans la grille.

HTML : Voici une structure avec 4 éléments dans une grille.

```html
<div class="grid-container">
  <div>1</div>
  <div>2</div>
  <div>3</div>
  <div>4</div>
</div>
```

CSS :

```css
.grid-container {
  display: grid;
  grid-template-columns: repeat(3, 1fr); /* Trois colonnes égales */
```

```
  grid-template-rows: 50px 50px; /* Deux lignes égales */
  gap: 10px;
}

.grid-container div {
  background-color: lightgray;
  text-align: center;
  padding: 10px;
}

/* Etendre l'élément 1 sur deux colonnes */
.grid-container div:nth-child(1) {
  grid-column-start: 1; /* Commence sur la colonne 1 */
  grid-column-end: 3;   /* Se termine sur la colonne 3 */
}

/* Etendre l'élément 2 sur deux lignes */
.grid-container div:nth-child(2) {
  grid-row-start: 1; /* Commence sur la ligne 1 */
  grid-row-end: 3;   /* Se termine sur la ligne 3 */
}
```

Résultat :

- L'élément `1` s'étend sur les deux premières colonnes de la première ligne.
- L'élément `2` occupe les deux lignes de la première colonne.

Astuce : Bien qu'efficace, cette méthode peut devenir répétitive sur des grilles complexes.

2.2. Étendre des éléments avec *grid-row* et *grid-column*

> Les propriétés CSS `grid-column` et `grid-row` permettent permettent de positionner et d'étendre des éléments dans une grille en spécifiant directement les lignes et colonnes de départ et de fin. Cette méthode offre une grande précision sans utiliser l'abréviation `span`.

2.2.1. Exemple pratique :

Imaginons une grille composée de 4 colonnes et 3 lignes, contenant six éléments.

HTML :

```html
<div class="grid-container">
  <div>1</div>
  <div>2</div>
  <div>3</div>
  <div>4</div>
  <div>5</div>
  <div>6</div>
</div>
```

CSS :

```css
.grid-container {
  display: grid;
  grid-template-columns: repeat(4, 1fr); /* Quatre colonnes de tailles égales */
  grid-template-rows: 50px 50px 50px;   /* Trois lignes égales */
  gap: 10px;
}

.grid-container div {
  background-color: lightgray;
  text-align: center;
  padding: 10px;
}

/* Étendre l'élément 1 sur deux colonnes */
.grid-container div:nth-child(1) {
  grid-column: 1 / 3; /* De la première colonne à la troisième */
  grid-row: 1 / 2;    /* Sur la première ligne */
}

/* Étendre l'élément 2 sur deux lignes */
.grid-container div:nth-child(2) {
  grid-column: 3 / 4; /* Sur la troisième colonne */
  grid-row: 1 / 3;    /* De la première ligne à la troisième */
}
```

2.2.2. Explications des propriétés utilisées :

- `grid-column` : Spécifie les colonnes sur lesquelles un élément s'étend. Par exemple, `grid-column: 1 / 3` signifie que l'élément commence à la première colonne et se termine juste avant la troisième.
- `grid-row` : Indique les lignes occupées par un élément. Par exemple, `grid-row: 1 / 2` positionne l'élément sur la première ligne.

Dans cet exemple :

- L'élément `1` occupe deux colonnes (la première et la deuxième) et une ligne (la première).
- L'élément `2` s'étend verticalement sur deux lignes (de la première à la deuxième), tout en restant sur une seule colonne (la troisième).

2.2.3. Résultat visuel attendu :

Avec cette disposition :

- L'élément `1` apparaît sur les deux premières colonnes de la première ligne.
- L'élément `2` occupe la troisième colonne et s'étale sur deux lignes verticalement.

Utiliser `grid-row` et `grid-column` de cette manière fournit un contrôle précis tout en simplifiant la gestion des mises en page dans une grille CSS.

2.3. `grid-area` (format A/B/C/D)

> La propriété `grid-area` est un outil puissant pour positionner et dimensionner un élément dans une grille.
>
> Elle utilise un format basé sur les lignes de départ et de fin, tant pour les lignes que pour les colonnes.
> Cette flexibilité permet aux développeurs de contrôler précisément l'emplacement d'un élément, tout en optimisant l'espace et la structure de la mise en page.

HTML :

```html
<div class="grid-container">
 <div>1</div>
 <div>2</div>
 <div>3</div>
 <div>4</div>
</div>
```

CSS :

```css
.grid-container {
 display: grid;
 grid-template-columns: 1fr 1fr 1fr; /* 3 colonnes */
 grid-template-rows: 50px 50px; /* 2 lignes */
```

```
 gap: 10px;
}

.grid-container div {
 background-color: lightgray;
 text-align: center;
 padding: 10px;
}

/* Placer l'élément 1 sur les lignes 1 à 2 et colonnes 1 à 3 */
.grid-container div:nth-child(1) {
 grid-area: 1 / 1 / 3 / 3; /* Ligne de départ : 1, Colonne de départ : 1, Ligne de fin : 3, Colonne de fin : 3 */
}
```

- Explication :
  - grid-area au format A/B/C/D
    - `A` = Ligne de départ
    - `B` = Colonne de départ
    - `C` = Ligne de fin
    - `D` = Colonne de fin
  - Exemple : `grid-area: 1 / 1 / 3 / 3;` définit une zone englobant la 1ère à la 2ème ligne (exclut la 3ème) et la 1ère à la 2ème colonne (exclut la 3ème).

2.4. Etendre une zone avec *span*

> En utilisant `span`, vous pouvez déclarer directement le nombre de colonnes ou de lignes qu'un élément doit occuper. Cela rend le code plus concis.

CSS :

```css
.grid-container {
  display: grid;
  grid-template-columns: repeat(3, 1fr); /* Trois colonnes égales */
  grid-template-rows: 50px 50px; /* Deux lignes égales */
  gap: 10px;
}

.grid-container div {
```

```
  background-color: lightgray;
  text-align: center;
  padding: 10px;
}

/* Étendre l'élément 1 sur deux colonnes */
.grid-container div:nth-child(1) {
  grid-column: 1 / span 2; /* S'étend sur 2 colonnes à partir de la colonne 1 */
}

/* Étendre l'élément 2 sur deux lignes */
.grid-container div:nth-child(2) {
  grid-row: 1 / span 2; /* S'étend sur 2 lignes à partir de la ligne 1 */
}
```

Résultat :

- Les éléments occupent les mêmes positions que dans l'exemple précédent, mais avec un code simplifié grâce à `span`.

> Avantages du `span` : C'est une méthode plus lisible, surtout si le design de votre grille demande des ajustements fréquents.

Avec ces deux étapes, vous pouvez choisir entre précision ou simplicité pour étendre vos éléments dans une grille CSS, selon vos besoins de mise en page.

3. Zones nommées avec `grid-template-areas`

> La propriété `grid-template-areas` simplifie l'organisation des grilles en permettant de nommer les différentes zones dans le conteneur.
> Elle offre une méthode visuelle et intuitive pour définir la disposition des éléments, en utilisant des noms personnalisés pour chaque section.
> Cela rend la mise en page plus lisible et facilite le positionnement des éléments, même pour des grilles complexes.

HTML :

```html
<div class="grid-container">
 <div class="header">Header</div>
 <div class="sidebar">Sidebar</div>
 <div class="main">Main</div>
 <div class="footer">Footer</div>
</div>
```

CSS :

```css
.grid-container {
 display: grid;
 grid-template-areas:
   "header header"
   "sidebar main"
   "footer footer";
 grid-template-columns: 1fr 3fr; /* Largeur des colonnes */
 grid-template-rows: 50px 1fr 30px; /* Hauteur des lignes */
 gap: 10px;
}

.header {
 grid-area: header;
 background-color: lightgreen;
}

.sidebar {
 grid-area: sidebar;
 background-color: lightyellow;
}

.main {
 grid-area: main;
 background-color: lightpink;
}

.footer {
 grid-area: footer;
 background-color: lightskyblue;
```

```
}
```

4. À RETENIR

- CSS Grid c'est quoi ? Un système de mise en page bidimensionnel.
- Points principaux :
  - `grid-template-columns` : Définit les colonnes.
  - `grid-template-rows` : Définit les lignes.
  - `gap` : Ajoute des espaces entre les éléments.
  - `grid-template-areas` : Crée des zones nommées pour placer les éléments.
  - `grid-column` , `grid-row` et `grid-area` : Contrôlent la position et l'étendue des éléments.
- Astuce : Combinez CSS Grid avec Flexbox pour encore plus de contrôle dans vos mises en page complexes.