Le display: grid en CSS est une méthode puissante pour créer des mises en page complexes et structurées en utilisant une grille. Il permet de diviser l'espace disponible en lignes et colonnes et de placer les éléments dans cette grille de manière flexible. Voici un aperçu complet de display: grid, avec des exemples et des exercices corrigés pour vous aider à mieux comprendre son fonctionnement.

1. Introduction à display: grid

display: grid permet de créer une grille où les éléments enfants sont positionnés dans une structure de lignes et de colonnes. La grille est définie par deux axes :

- **L'axe des lignes** (horizontal).
- **L'axe des colonnes** (vertical).

Propriétés principales :

- **grid-template-columns** : définit le nombre et la largeur des colonnes.
- **grid-template-rows** : définit le nombre et la hauteur des lignes.
- **grid-gap** (ou gap) : espace entre les éléments de la grille (en lignes et en colonnes).
- **grid-column** et **grid-row** : pour positionner un élément spécifiquement dans la grille.
- **justify-items**, **align-items** : pour contrôler l'alignement des éléments enfants dans la cellule.

2. **Syntaxe de base**

```css
.container {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr; /* 3 colonnes égales */
  grid-template-rows: auto; /* lignes avec hauteur automatique */
  gap: 10px; /* espace entre les éléments */
}
```

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

3. **Propriétés principals**

3.1 grid-template-columns et grid-template-rows

La propriété grid-template-columns définit la largeur des colonnes. Vous pouvez utiliser des unités telles que px, %, fr (fraction), auto.

- **fr** (fraction) : divise l'espace restant disponible en fractions égales.
- **px** : unités fixes.
- **%** : pour des colonnes ou lignes avec un pourcentage de l'espace disponible.

```
.container {
    display: grid;
    grid-template-columns: 1fr 2fr 1fr; /* Première colonne 1 fraction, la seconde 2 fractions, la troisième 1 fraction */
    grid-template-rows: 100px auto 200px; /* Hauteurs fixes pour les lignes */
    gap: 10px;
        }
```

3.2 grid-gap

La propriété gap permet de définir l'espace entre les éléments de la grille (lignes et colonnes).

```
.container {
  display: grid;
  grid-template-columns: repeat(3, 1fr); /* 3 colonnes égales */
  grid-gap: 15px; /* Espacement entre les éléments */
}
```

3.3 grid-column et grid-row

Ces propriétés vous permettent de spécifier où un élément particulier doit commencer et se terminer dans la grille.

```
.item {
  grid-column: 2 / 4; /* L'élément occupe les colonnes 2 à 4 */
  grid-row: 1 / 3; /* L'élément occupe les lignes 1 à 3 */
}
```

4. Alignement dans la grille

- **justify-items** : aligne les éléments sur l'axe des colonnes (horizontal).
- **align-items** : aligne les éléments sur l'axe des lignes (vertical).
- **justify-self** et **align-self** : permettent de surcharger l'alignement pour un élément spécifique.

Exemple d'alignement de contenu :

```
.container {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  gap: 10px;
  justify-items: center; /* Centre les éléments horizontalement */
  align-items: center; /* Centre les éléments verticalement */
}
```

5. **Exemple complet : Mise en page de blog avec grid**

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

```css
.blog-container {
  display: grid;
  grid-template-columns: 1fr 3fr; /* Sidebar 1/4, Main Content 3/4 */
  grid-template-rows: auto 1fr auto; /* Header en haut, footer en bas */
  gap: 20px;
}

.header {
  grid-column: span 2; /* Le header occupe les 2 colonnes */
}

.footer {
  grid-column: span 2; /* Le footer occupe les 2 colonnes */
}
```

Exercices Corriges :

Exercice 1 : Créer une grille avec 4 colonnes égales

Objectif : Créer une grille avec 4 colonnes égales et 2 lignes. Chaque élément doit occuper une cellule.

Solution :

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

```css
.grid-container {
  display: grid;
  grid-template-columns: repeat(4, 1fr); /* 4 colonnes égales */
  gap: 10px;
}
```

Exercice 2 : Créer une grille responsive

Objectif : Créer une grille qui affiche 2 colonnes sur les petits écrans (moins de 600px) et 4 colonnes sur les grands écrans (plus de 600px).

Solution :

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

```css
.grid-container {
  display: grid;
  grid-template-columns: repeat(2, 1fr); /* 2 colonnes par défaut */
  gap: 10px;
}

@media (min-width: 600px) {
  .grid-container {
    grid-template-columns: repeat(4, 1fr); /* 4 colonnes pour les grands écrans */
  }
}
```

Exercice 3 : Placer un élément dans une zone spécifique de la grille

Objectif : Utiliser grid-column et grid-row pour placer un élément dans une cellule spécifique.

Solution :

```html
<div class="grid-container">
  <div class="item1">Item 1</div>
  <div class="item2">Item 2</div>
  <div class="item3">Item 3</div>
  <div class="item4">Item 4</div>
</div>
```

```css
.grid-container {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr; /* 3 colonnes égales */
  grid-template-rows: auto; /* lignes automatiques */
  gap: 10px;
}

.item2 {
  grid-column: 2; /* L'élément occupe la 2ème colonne */
  grid-row: 1; /* L'élément occupe la 1ère ligne */
}
```

Exercice 1 : Utilisation de span pour faire occuper plusieurs colonnes à un élément

Objectif : Créez une grille avec 3 colonnes et 2 lignes. L'élément de la première ligne doit occuper les 2 premières colonnes (en utilisant span), et l'élément de la deuxième ligne doit occuper les 3 colonnes.

Solution :

```html
<div class="grid-container">
  <div class="item1">Item 1</div>
  <div class="item2">Item 2</div>
  <div class="item3">Item 3</div>
</div>
```

Exercice 2 : Utilisation de grid-template-areas pour nommer les zones de la grille

Objectif : Créez une grille avec 3 colonnes et 3 lignes. Utilisez grid-template-areas pour nommer les zones de la grille et positionner les éléments dans ces zones.

Solution :

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

```css
.grid-container {
        display: grid;
        grid-template-columns: 200px 1fr; /* Sidebar de 200px, et une colonne de contenu fluide */
        grid-template-areas:
            "header header"
            "sidebar main"
            "footer footer"; /* Définit la disposition des zones */
        gap: 10px;
        }
.header {
grid-area: header;
}
.sidebar {
grid-area: sidebar;
}
.main {
grid-area: main;
}
.footer {
grid-area: footer;
}
```

Explication :

- Utilisation de grid-template-areas pour définir des zones claires : "header header", "sidebar main", et "footer footer".
- Les éléments .header, .sidebar, .main, et .footer sont ensuite assignés à leurs zones respectives avec grid-area.

Exercice 3 : Aligner les éléments avec justify-items et align-items

Objectif : Créez une grille de 2 colonnes et 2 lignes et alignez les éléments dans la grille de manière centrée à la fois horizontalement et verticalement.

Solution :

```html
<div class="grid-container">
  <div class="item1">Item 1</div>
  <div class="item2">Item 2</div>
  <div class="item3">Item 3</div>
  <div class="item4">Item 4</div>
</div>
```

```css
.grid-container {
  display: grid;
  grid-template-columns: 1fr 1fr; /* 2 colonnes égales */
  grid-template-rows: 100px 100px; /* 2 lignes de 100px de hauteur */
  gap: 10px;
  justify-items: center; /* Centre les éléments horizontalement */
  align-items: center; /* Centre les éléments verticalement */
}
```

Explication :

- justify-items: center; aligne les éléments horizontalement au centre de chaque cellule.
- align-items: center; aligne les éléments verticalement au centre de chaque cellule.

Exercice 4 : Placement spécifique avec grid-column et grid-row

Objectif : Créez une grille avec 3 colonnes et 2 lignes. Placez chaque élément dans une position spécifique en utilisant grid-column et grid-row.

Solution :

```html
<div class="grid-container">
  <div class="item1">Item 1</div>
  <div class="item2">Item 2</div>
  <div class="item3">Item 3</div>
  <div class="item4">Item 4</div>
</div>
```

```css
.grid-container {
    display: grid;
    grid-template-columns: 1fr 1fr 1fr; /* 3 colonnes égales */
    grid-template-rows: auto auto; /* 2 lignes */
    gap: 10px;
    }
.item1 {
grid-column: 1; /* Premier élément dans la première colonne */
grid-row: 1; /* Premier élément dans la première ligne */
}
.item2 {
grid-column: 2; /* Deuxième élément dans la deuxième colonne */
grid-row: 1; /* Deuxième élément dans la première ligne */
}
.item3 {
grid-column: 3; /* Troisième élément dans la troisième colonne */
grid-row: 2; /* Troisième élément dans la deuxième ligne */
}
.item4 {
grid-column: 1 / 4; /* L'élément occupe les 3 colonnes */
grid-row: 2; /* L'élément occupe la deuxième ligne */
}
```

Explication :

- grid-column: 1; place l'élément dans la première colonne, et ainsi de suite pour les autres éléments.
- .item4 utilise grid-column: 1 / 4; pour occuper toute la largeur de la grille (les 3 colonnes).

Exercice 5 : Créer une grille responsive avec `grid-template-columns`

Objectif : Créez une grille responsive qui affiche 3 colonnes sur les grands écrans et 1 seule colonne sur les petits écrans.

Solution :

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

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

@media (max-width: 600px) {
  .grid-container {
    grid-template-columns: 1fr; /* Une seule colonne sur les petits écrans */
  }
}
```

Explication :

- Sur les grands écrans, la grille utilise grid-template-columns: repeat(3, 1fr); pour afficher 3 colonnes égales.
- Avec la règle @media (max-width: 600px), la grille devient une seule colonne sur les petits écrans (moins de 600px).

Exercice 6 : Utilisation de grid-auto-flow **pour contrôler l'ordre des éléments**

Objectif : Créez une grille où les éléments sont automatiquement placés en ligne, et utilisez grid-auto-flow pour définir l'ordre (par défaut en ligne, mais vous pouvez changer cela en colonnes).

Solution :

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

```css
.grid-container {
  display: grid;
  grid-template-columns: 1fr 1fr; /* 2 colonnes */
  gap: 10px;
  grid-auto-flow: column; /* Les éléments seront placés en colonne plutôt qu'en ligne */
}
```