Display : grid en CSS

Scenario 1 : Créer une grille simple avec 3 colonnes égales.

CSS

```
.container {
  display: grid;
  grid-template-columns: repeat(3, 1fr); /* 3 colonnes égales */
  gap: 10px;
}
.item {
  background-color: lightblue;
  text-align: center;
  padding: 20px;
  font-size: 20px;
}
```

Html

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

Scenario 2 : Disposition asymétrique avec grid-template-areas

Css

```
.container {
  display: grid;
  grid-template-areas:
    "header header header"
    "sidebar main main"
    "footer footer footer";
  grid-template-rows: 50px 1fr 30px;
  grid-template-columns: 200px 1fr 1fr;
  gap: 10px;
}
.header {
  grid-area: header;
  background-color: lightcoral;
}
.sidebar {
  grid-area: sidebar;
```

```
  background-color: lightgreen;
}
.main {
  grid-area: main;
  background-color: lightblue;
}
.footer {
  grid-area: footer;
  background-color: lightgray;
}
```

HTML

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

Scenario 3 : Grille réactive avec minmax et auto-fit

css

```
.container {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(150px, 1fr));
  gap: 20px;
}
.item {
  background-color: lightpink;
  height: 150px;
}
```

Html

```
<div class="container">
  <div class="item"></div>
  <div class="item"></div>
  <div class="item"></div>
  <div class="item"></div>
  <div class="item"></div>
  <div class="item"></div>
</div>
```

Scenario 3 : Grille imbriquée (Nested Grid)

css

```
.container {
  display: grid;
  grid-template-columns: repeat(2, 1fr);
  gap: 20px;
}
.subgrid {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  gap: 10px;
}
.item {
  background-color: lightblue;
  padding: 10px;
}
```

HTML

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

Scenario 5 : Grille avec des colonnes de tailles variables

Css

```
.container {
  display: grid;
  grid-template-columns: 100px 1fr 2fr; /* Première colonne fixe, autres flexibles */
  gap: 10px;
}
.item {
  background-color: lightgray;
  padding: 20px;
}
```

HTML

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

Scenario 6 : Grille avec des items superposés (Overlapping Items)

CSS

```
.container {
  display: grid;
  grid-template-columns: 1fr 1fr;
  grid-template-rows: 100px 100px;
}
.item1 {
  grid-column: 1 / 3; /* S'étend sur 2 colonnes */
  grid-row: 1 / 2;
  background-color: lightblue;
}
.item2 {
  grid-column: 2 / 3;
  grid-row: 1 / 3; /* S'étend sur 2 rangées */
  background-color: lightcoral;
}
```

Html

```
<div class="container">
  <div class="item1">1</div>
  <div class="item2">2</div>
</div>
```

Senario 7 : Grille pour un dashboard complexe

CSS

```
.container {
  display: grid;
  grid-template-areas:
    "nav nav nav"
    "sidebar main main"
    "footer footer footer";
  grid-template-columns: 200px 1fr 1fr;
  grid-template-rows: 50px 1fr 50px;
  gap: 10px;
}
.nav {
  grid-area: nav;
  background-color: lightblue;
}
.sidebar {
  grid-area: sidebar;
  background-color: lightgreen;
}
.main {
  grid-area: main;
  background-color: lightpink;
}
.footer {
  grid-area: footer;
  background-color: lightgray;
}
```

HTML

```
<div class="container">
  <div class="nav">Navigation</div>
  <div class="sidebar">Sidebar</div>
  <div class="main">Main Content</div>
  <div class="footer">Footer</div>
</div>
```