

Two vertical bars on the left side of the slide: a light blue bar and a darker blue bar, both of equal height and width, positioned side-by-side.

CSS modernes

GRID et FLEXBOX

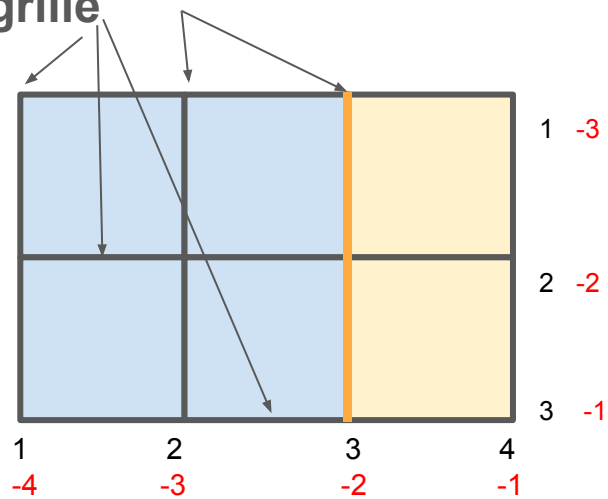
PARTIE 6.1

Un peu de vocabulaire

Un peu de vocabulaire

- Conteneur (**grid container**)
- Cellule (**grid cell**)
- un élément de la grille (**grid item**) (plusieurs cellule) - grid item en anglais
- **Zone** (**grid area**) : pour créer une zone dans une grille , exemple une zone Header, une zone contenu, une zone sidebar, une zone
- Ligne dans une grille (**grid line**)
- Les pistes (**grid track**): bande verticale ou horizontale

Ligne dans une grille



Les pistes : bande verticale ou horizontale

Display: **grid | inline-grid;**

```
*{  
    margin:0;  
    padding:0;  
}  
  
.container{  
    display:grid; | inline-grid;  
}
```

PARTIE 6.1

grid-template-columns

grid-template-columns et la propriété repeat (x , x)

```
*{  
  margin:0;  
  padding:0;  
}  
  
.container{  
  display:grid;  
  grid-template-columns : xx xx xx;  
}
```

le principe de la
propriété
grid-template-colum
ns et la notation
fonctionnelle
repeat

```
grid-template-columns : xx xx xx;  
grid-template-columns : repeat(3, 100px);
```

Une nouvelle unité : le fr

```
grid-template-column : 25% 50% 25%;  
  
// j'aurai 3 colonnes de 100px;  
grid-template-columns : repeat(3, 100px);  
  
// j'aurai 10 colonnes de 100px;  
grid-template-columns : repeat(10, 100px);
```

```
// j'aurai 10 colonnes de 100px;  
grid-template-columns : repeat(10, 100px);  
// je divise en 10 l'espace disponible  
grid-template-columns : repeat(10, 1fr);  
// si j'ai une premiere colonne qui fait 100px  
puis 10 en 1fr, je divise en 10 l'espace disponible  
grid-template-columns : 100px repeat(10, 1fr);
```

le fr est une nouvelle unité de mesure : c'est une fraction, une taille variable en fonction de la place qu'il reste ou de la taille de son container.

il est possible de mettre des valeurs fixes AVANT ou APRÈS le repeat;

min-content, max-content, minmax

```
// j'aurai 3 colonnes de 100px;  
grid-template-columns : repeat(3, minmax(100px, 300px));
```

les colonnes peuvent
faire entre 100px
minimum et maximum
300px !

auto

```
// la taille des colonnes sera AUTOMATIQUE en fonction du contenu  
grid-template-columns : repeat(15, auto);
```

PARTIE 6.1

grid-template-rows

min-content, max-content, minmax

```
// j'aurai 3 colonnes de 100px;  
grid-template-columns : repeat(3, minmax(100px, 300px));
```

les colonnes peuvent
faire entre 100px
minimum et maximum
300px !

auto

```
// la taille des colonnes sera AUTOMATIQUE en fonction du contenu  
grid-template-columns : repeat(15, auto);
```

PARTIE 6.1

**raccourcis
grid-template et grid**

min-content, max-content, minmax

```
.container{  
display: grid;  
height: 100vh;  
grid-template-columns : repeat(3, auto);  
grid-template-rows: 50% 50%;  
}
```

les colonnes peuvent
faire entre 100px
minimum et maximum
300px !

raccourcis avec `grid-template`

```
// grid-template réunit grid-template-columns et grid-template-rows  
grid-template : 50% 50% / repeat(3, auto);
```

raccourcis avec `grid`

```
// grid-template réunit grid-template-columns et grid-template-rows et  
autres déclarations implicites  
grid-template : 50% 50% / repeat(3, auto);  
grid :
```

grid-column

```
.item:first-child {  
  grid-column-start: 1;  
  grid-column-end: 5;  
}
```

// Pour indiquer la taille de la colonne
grid-column-start : 1;
grid-column-end: 5;

// Pour indiquer la taille de la colonne
grid-column : 1 / 5;

```
10 <body>  
11   <div class="container">  
12     <div class="item">item1</div>  
13     <div class="item">item2</div>  
14     <div class="item">item3</div>  
15     <div class="item">item4</div>  
16   </div>  
17 </body>
```

```
12 .item:first-child {  
13   grid-column: 1 / 5;  
14 }
```

On peut aussi avoir le même résultat avec
grid-column-start : span 4 ;

notre 1ere colonne occupera 4 colonnes

```
.item:first-child {  
  grid-column-start: span 4;  
}
```

c'est exactement
la même chose

```
.item:first-child{  
  grid-column-start : span 4;  
}
```

```
.item:first-child{  
  grid-column-end : span 4;  
}
```

```
.item:first-child{  
  grid-column: span 4;  
}
```

```
11  
12 .item:first-child {  
13     grid-column: span 4;  
14     grid-row: span 2;  
15  
16     grid-area: 1 / 1 / -1 / 5;  
17 }  
18
```

```
.item:first-child{  
grid-column : span 4;  
grid-row : span 4;  
  
grid-area: 1 / 1 / -1 / 5;  
/* grid-area : row-start / column-start / row-end / column-end; */  
}
```



```
.item:first-child{  
  /* grid-column : span 4;  
  grid-row : span 2; */  
  grid-area: span 2 / span 4;  
  /* grid-area : grid-row / grid-span ; */  
}
```

PARTIE 2

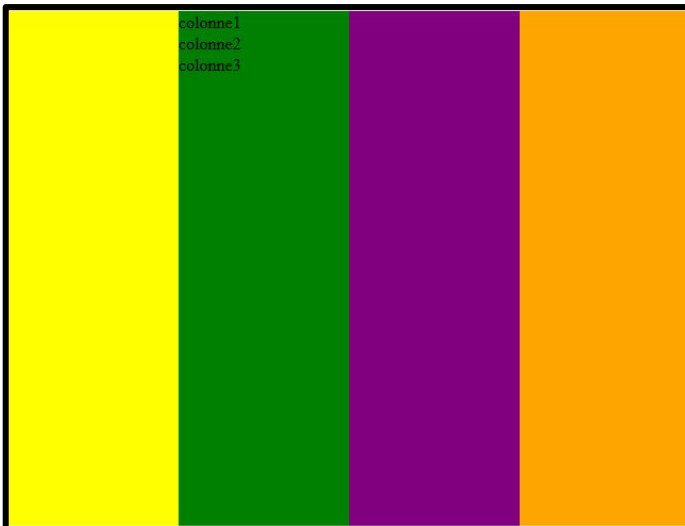
Une structure de site

Le html

```
<div class="container">
  <header></header>
  <main>
    <div>colonne1</div>
    <div>colonne2</div>
    <div>colonne3</div>
  </main>
  <aside></aside>
  <footer></footer>
</div>
```

Le CSS

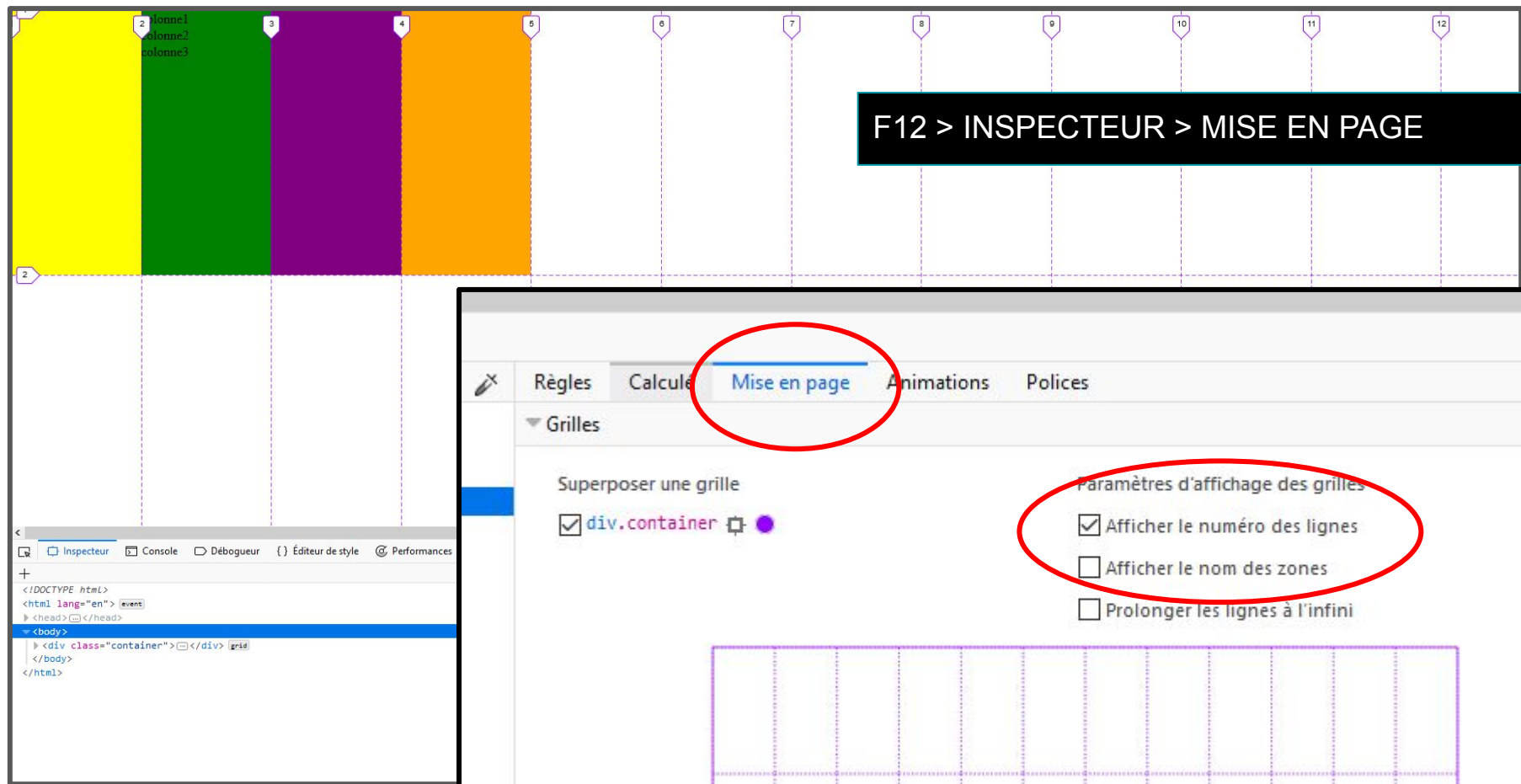
```
*{ margin:0; padding: 0;}
.container{
  display: grid;
  height: 100vh;
  grid: 50% 50% / repeat(12, 1fr);
}
header { background-color: yellow;}
main { background-color: green;}
aside { background-color: purple;}
footer { background-color: orange;}
```



Visualisation dans le navigateur

j'ai 4 éléments enfants de .container, ils occupent bien par défaut 1 colonne chacun sur 1 ligne

```
<div class="container">  
  <header></header>  
  <main>  
    <div>colonne1</div>  
    <div>colonne2</div>  
    <div>colonne3</div>  
  </main>  
  <aside></aside>  
  <footer></footer>  
</div>
```



On va maintenant créer des zones dans nos éléments HTML

CRÉATION DE ZONE DANS NOS ÉLÉMENTS HTML

01- Je commence par indiquer une hauteur avec grid-template-row

```
.container{  
  display: grid;  
  grid-template-rows: repeat(4, 100px);  
}
```



Pour créer une zone, on va utiliser grid-template-areas

CRÉATION DE ZONE DANS NOS ÉLÉMENTS HTML

02- création d'une zone avec **grid-template-areas**

```
.container{  
  display: grid;  
  grid-template-rows: repeat(4, 100px);  
  grid-template-areas :  
    "header"  
    "aside"  
    "main"  
    "footer"  
  ;  
}
```

On met les noms de zone que l'on souhaite, ici j'ai repris les noms des blocs html, mais on n'est pas obligé !

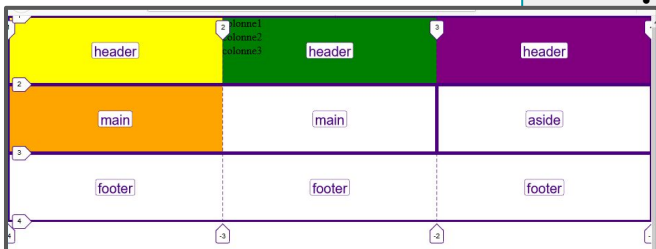


Pour créer une zone, on va utiliser grid-template-areas

CRÉATION DE ZONE DANS NOS ÉLÉMENTS HTML

03- Pour mieux comprendre on va insérer un media query...

```
@media (min-width:768px){  
  .container{  
    grid-template-columns: repeat(3, 1fr);  
    grid-template-rows: repeat(3, 100px);  
    grid-template-areas :  
      "header header header"  
      "main main aside"  
      "footer footer footer"
```




PARTIE 6.1

**Pour créer une
gouttière**

Pour créer une zone, on va utiliser grid-template-areas

CRÉATION DE ZONE DANS NOS ÉLÉMENTS HTML

Pour créer une gouttière deux propriétés :
grid-column-gap et
grid-row-gap



```
@media (min-width:768px){  
  .container{  
    grid-template-columns: repeat(3, 1fr);  
    grid-template-rows: repeat(3, 100px);  
    grid-template-areas :  
      "header header header"  
      "main main aside"  
      "footer footer footer"  
    ;  
    grid-column-gap: 2rem;  
    grid-row-gap: 2rem  
  }  
}
```

Pour créer une GOUTTIERE
grid-gap;

CRÉATION DE GOUTTIÈRES

Il existe une propriété
raccourcis pour gérer les
gouttières :
grid-gap: valeur du row
valeur de la colonne

```
@media (min-width:768px){  
  .container{  
    grid-template-columns: repeat(3, 1fr);  
    grid-template-rows: repeat(3, 100px);  
    grid-template-areas :  
      "header header header"  
      "main main aside"  
      "footer footer footer"  
  ;  
  grid-column-gap: 2rem;  
  grid-row-gap: 2rem  
  grid-gap: 2rem 2rem;  
  /* grid-gap: la 1ere valeur pour le row puis la colonne */  
}
```

PARTIE 6.1

**Pour créer une
gouttière**