420-KB2-LG, Conception d’interfaces Web

Préparé par Saliha Yacoub

Les grilles CSS

Plan de la séance:

- Définition
- grid-template-columns
- grid-template-rows
- grid-gap
- Les lignes et les colonnes
- Les grilles et les éléments structurants
  - `grid-template-areas`

Préparé par Saliha Yacoub

Retour sur la dernière séance

Point de vue des étudiants

Point de vue de l’enseignante

Les grilles : Définition

- Le module CSS Grid Layout offre un système de mise en page basé sur une grille, avec des lignes et des colonnes, ce qui facilite la conception de pages Web sans avoir à utiliser de flotteurs (float) ni de positionnement.
- La stratégie de mise en page présentée ici, les grilles CSS (CSS grids), est la plus récente de toutes. Elle est aussi très intuitive au niveau conception.
- On commence par créer une grille virtuelle, puis on y place les éléments, ceux-ci pouvant recouvrir plusieurs cellules.
- La propriété display aura comme valeur grid ce qui permettra d'avoir une grille.
- Une grille est un ensemble de lignes et de colonnes.
- Pour montrer la puissance des grilles css, nous procéderons par des exemples simples.

Exemple1: Affichage sans la grille

- Voici le code html et le css sans la grille.

```
<div id="conteneur">
      <div>1</div>
      <div>2</div>
      <div>3</div>
      <div>4</div>
      <div>5</div>
      <div>6</div>
      <div>7</div>
      <div>8</div>
      <div>9</div>
 </div>
```

```
#conteneur {
   text-align: center;
  }
  #conteneur div:nth-child(even) {
    background:aquamarine;
    padding:10px;
    border: 1px solid black;
  }
  #conteneur div:nth-child(odd) {
    background:pink;
    padding: 10px;
    border: 1px solid black;
  }
```

Exemple1: Affichage sans la grille

Voici le code html et le css sans la grille. L'affichage sera des div les uns à la suite des autres (display block)

1
2
3
4
5
6
7
8
9

Exemple1: Affichage avec la grille

- Pour le même code html, nous allons utiliser la propriété display avec la valeur grid: `display: grid;`
- Nous avons 9 div dans le code html il faudra:
  1. définir le nombre de colonne que l'on souhaite avoir avec ces div.
  2. Et il faudra définir la taille de chacune des colonnes.

- La propriété `grid-template-columns` permet de faire les points 1 et 2

Affichage avec une grille. La taille en px.

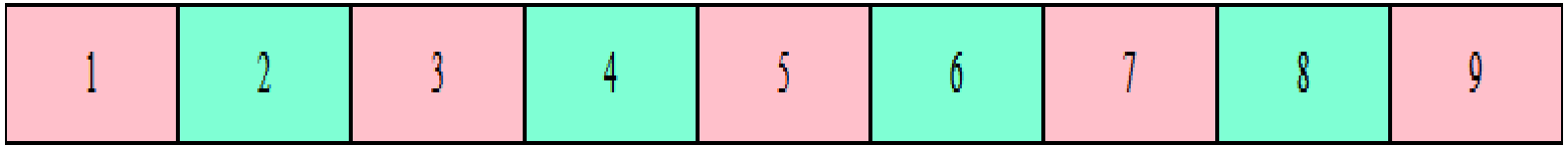
Il existe plusieurs façons de donner la taille des cellules. En px telle que le montre le code suivant: (les padding ont été ajoutés pour plus de clarté).

```
#conteneur {
   text-align: center;
   display:grid;
   grid-template-columns: 100px 100px 100px 100px 100px 100px 100px 100px 100px;
  }
  #conteneur div:nth-child(even) {
    background:aquamarine;
    padding:10px;
    border: 1px solid black;
  }
  #conteneur div:nth-child(odd) {
    background:pink;
    padding: 10px;
    border: 1px solid black;
  }
```

Affichage avec une grille. La taille en px.

- Le code HTML n’a pas changé. Il ne contient que les div.
- Le css de chaque div n’a pas changé non plus.
- Dans le div principal, nous avons ajouté les deux lignes suivantes qui permettent l’affichage sous forme de grille (image plus bas):

```
display:grid;
grid-template-columns: 100px 100px 100px 100px 100px 100px 100px 100px 100px;
```



Affichage avec une grille. La taille en pourcentage.

- Le problème avec l'affichage en px est que la taille des div est absolue et que l'affichage ne s'adaptera pas à la taille de l'écran. Lorsque vous diminuez la taille de l'écran l'affichage ne s'adapte pas.
- L'affichage en pourcentage permet de faire un affichage moins contraignant. Nous avons 9 div à afficher, chaque div a environ 11%.

```
#conteneur {
   text-align: center;
   display:grid;
   grid-template-columns: 11% 11% 11% 11% 11% 11% 11% 11% 11%;
  }
```

Affichage avec une grille. La taille fraction.

- L’affichage en fraction permet d’avoir une fraction de la même fenêtre.
- Ici nous avons 9 div chaque div aura une taille de 1/9 de la fenêtre.
- L’affichage va s’adapter à la taille de la fenêtre.

```
#conteneur {
   text-align: center;
   display:grid;
   grid-template-columns: 1fr 1fr 1fr 1fr 1fr 1fr 1fr 1fr 1fr ;

  }
```

Affichage avec une grille. Taille variable des cellules.

- Peu importe la façon dont on donne la largeur des colonnes (en px, en % ou en fr), on pourrait décider de leur attribuer des largeurs différentes.
- Dans le code suivant, on dit que la première colonne va avoir une fraction, la 2eme deux fractions et la troisième colonne une fraction.

```
#conteneur {
   text-align: center;
   display:grid;
   grid-template-columns: 1fr 2fr 1fr;
  }
```

Affichage avec une grille. Taille variable des cellules.

- On remarque que les 9 div ne s'affichent pas tous sur la même ligne.
- La tailles des cellules au milieu est deux fois plus grande que celles au début et à la fin.
- L'affichage s'est effectué ainsi:
  - Nous avons défini: `1fr 2fr 1fr,` ce qui veut dire déjà que nous avons rempli la fenêtre avec 3 colonnes de tailles différentes.
  - Les autres div se sont positionnés sur les lignes suivantes.
  - Si j'ajoute un 10eme div, il va se placer en bas du 7.
  - Si nous avions `0.5fr` on aura la moitié de la cellule

1	2	3
4	5	6
7	8	9

Affichage avec une grille. Taille variable des cellules.

- On aura la même grille si la taille des cellules est exprimée en pourcentage comme suit:

```
#conteneur {
   text-align: center;
   display:grid;
   grid-template-columns: 25% 50% 25%
  }
```

1	2	3
4	5	6
7	8	9

Utilisation de la fonction repeat.

- Cette fonction est utilisée lorsque toutes les cellules de la grille ont la même largeur.
- La fonction prend deux paramètres:
  - Le nombre de répétions et
  - la taille (en fraction ou en pourcentage)

```
#conteneur {
   text-align: center;
   display:grid;
   grid-template-columns: repeat(3,1fr);
  }
```

1	2	3
4	5	6
7	8	9

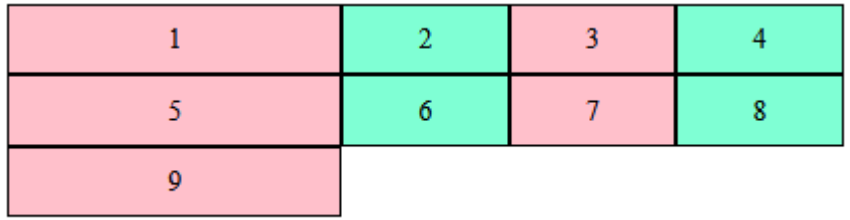
Utilisation de la fonction repeat.

- Cette fonction est utilisée lorsque toutes les cellules de la grille ont la même largeur.
- La fonction prend deux paramètres:
  - Le nombre de répétions et
  - la taille (en fraction ou en pourcentage).

```
#conteneur {
   text-align: center;
   display:grid;
   grid-template-columns: repeat(3,25%);
  }
```

Exemple on mélange la fonction repeat et les fractions

```
#conteneur {
   text-align: center;
   display:grid;
    grid-template-columns: 2fr  repeat(3,1fr);
   max-width:480px;
   margin:auto;
  }
  #conteneur div:nth-child(even) {
    background:aquamarine;
    border: 1px solid black;
    padding:10px;
  }
  #conteneur div:nth-child(odd) {
    background:pink;
    border: 1px solid black;
    padding:10px;
  }
```

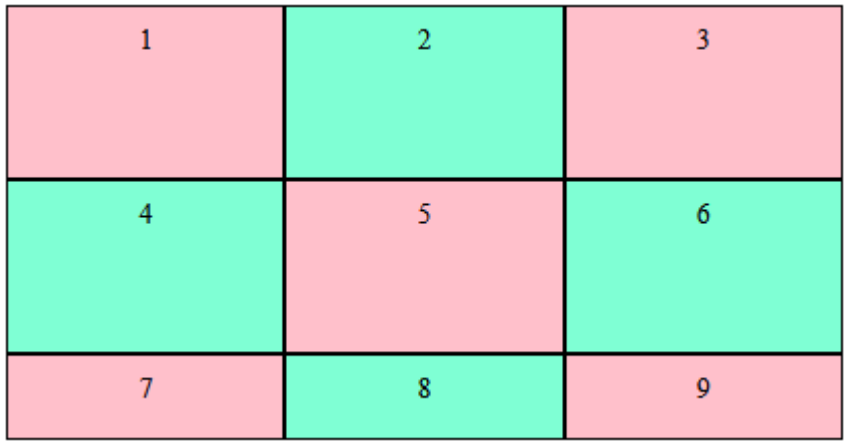


Et les lignes ??

Par défaut, la hauteur des rangées s'ajuste au contenu (en plus de la valeur de la propriété padding). Mais on peut aussi fixer la hauteur des rangées: `grid-template-rows`

- Voici l'exemple d'une grille de 6 colonnes et de 3 rangées (les deux premières d'une hauteur de 100px, la troisième de 50px) :

```
#conteneur {
   text-align: center;
   display:grid;
   grid-template-columns: repeat(3,1fr);
   max-width:480px;
   margin:auto;
   grid-template-rows: 100px 100px 50px;}
```



Les lignes

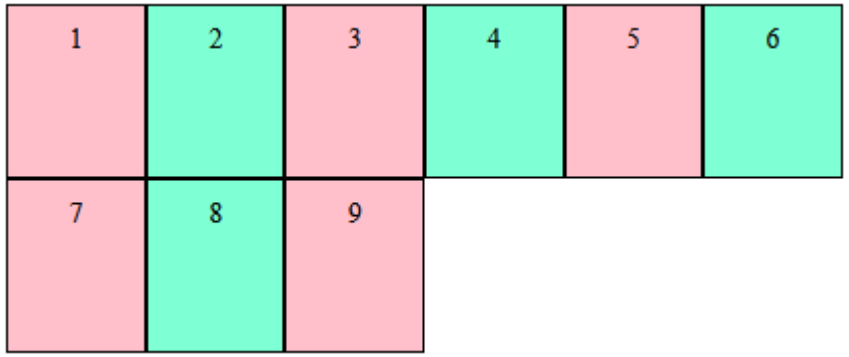
- Tout comme pour les cellules, on pourrait donner la taille en fraction.

```
grid-template-rows: 2fr 2fr 1fr;
```

- Peu importe le nombre de lignes que nous avons, on pourrait décider qu'elles ont toutes la même hauteur avec

```
grid-auto-rows: 100px
```

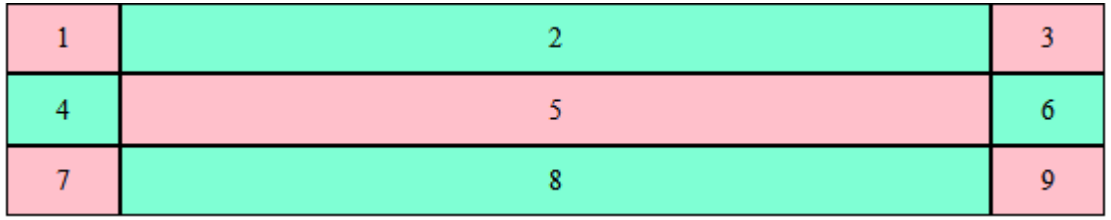
```
#conteneur {
   text-align: center;
   display:grid;
   grid-template-columns: repeat(6,1fr);
   max-width:480px;
   margin:auto;
   grid-auto-rows: 100px;}
```



minmax pour les colonnes et les rangées

- On décide de donner à une cellule en particulier la taille minimale que celle-ci devrait avoir. De cette façon lorsque la largeur de l'écran est diminuée la cellule gardera quand même une longueur minimale. Ce qui peut être utile si on veut qu'un texte soit affichée au complet. L'exemple de la figure suivante est obtenu en réduisant l'affichage. On remarque la colonne du milieu garde sa largeur de 500px.

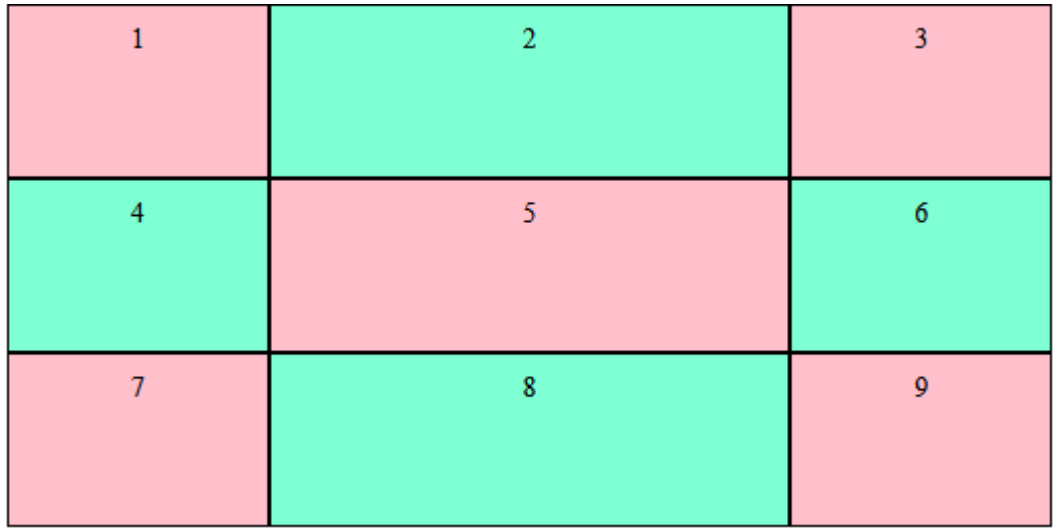
```
# #conteneur {
   text-align: center;
   display:grid;
   grid-template-columns: 1fr minmax(500px, 1fr) 1fr;

   margin:auto;
   grid-template-rows: repeat(3,2fr);}
```



minmax pour les colonnes et les rangées

- Exemple pour les lignes. Ici la ligne a minimum 100px.

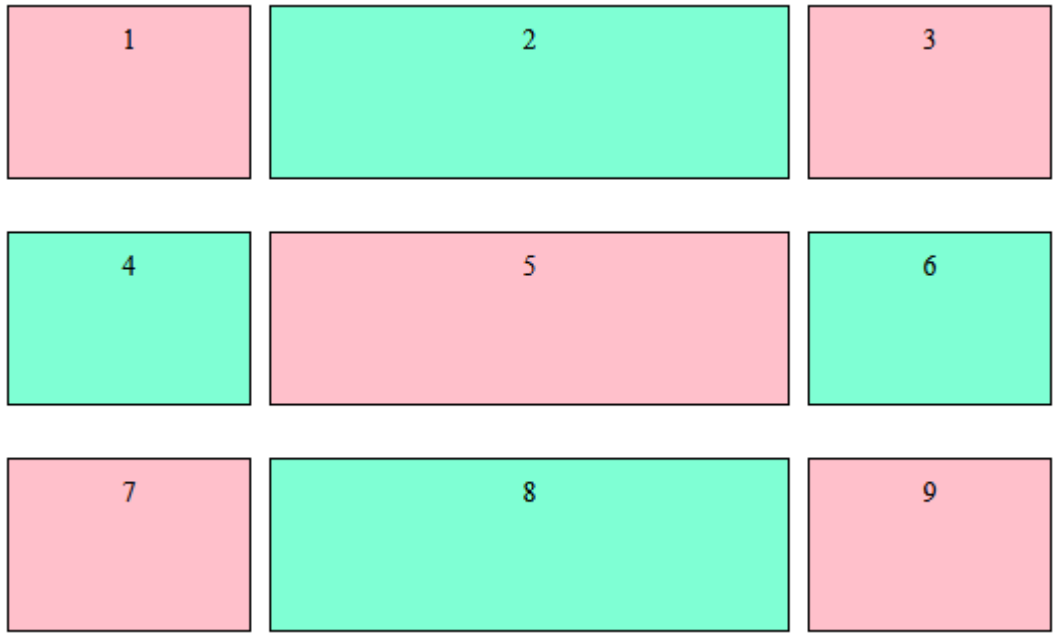
```
#conteneur {
   text-align: center;
   display:grid;
   grid-template-columns: 1fr minmax(300px, 1fr) 1fr;
   max-width: 600px;
   margin:auto;
   grid-template-rows: repeat(3,minmax(100px, 1fr));

}
```



Espacement entre les cellules

- On peut ajouter un espacement entre les colonnes avec la propriété `column-gap`
- On peut ajouter un espacement entre les lignes avec la propriété `row-gap;`
- Si l'espacement entre les lignes et les colonnes est identique on avoir : `grid-gap`

```
#conteneur {
    text-align: center;
    display:grid;
    grid-template-columns: 1fr minmax(300px, 1fr) 1fr;
    max-width: 600px;
    margin:auto;
    grid-template-rows: repeat(3,minmax(100px, 1fr));
    column-gap: 10px;
    row-gap: 30px;
}
```



Positionner les éléments: Les lignes et les colonnes

Nous avons vu qu'à l'intérieur d'un conteneur ayant la propriété display:grid, les éléments sont placés dans la grille virtuelle selon l'ordre de leur déclaration dans le fichier HTML. Mais il s'agit là du comportement par défaut.

Il existe une façon beaucoup plus souple de positionner les éléments, soit en utilisant les lignes de colonnes et les lignes de rangées.

Ces lignes, que l'on peut voir dans la figure suivante, débute à 1 et vont jusqu'à n+1 où n est le nombre de colonnes ou le nombre de rangées.

Les propriétés suivantes seront utilisées pour placer chaque élément dans la grille:

grid-column-start

grid-column-end

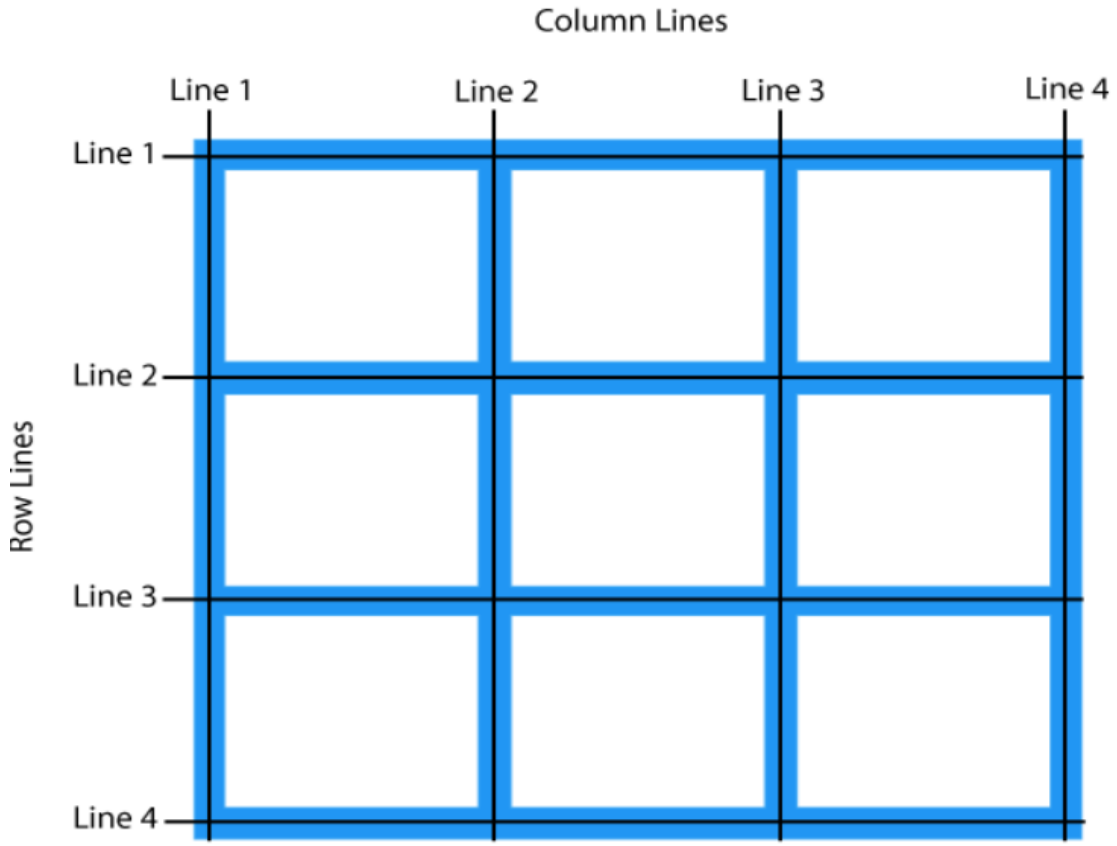
grid-row-start

grid-row-end

Positionner les éléments: Les lignes et les colonnes

Attention. Ici il ne s'agit pas du nombre de cellule mais bien du nombre de lignes « traits »

Dans les exemples suivants, la bordure a été ajoutée pour des raisons de clarté.

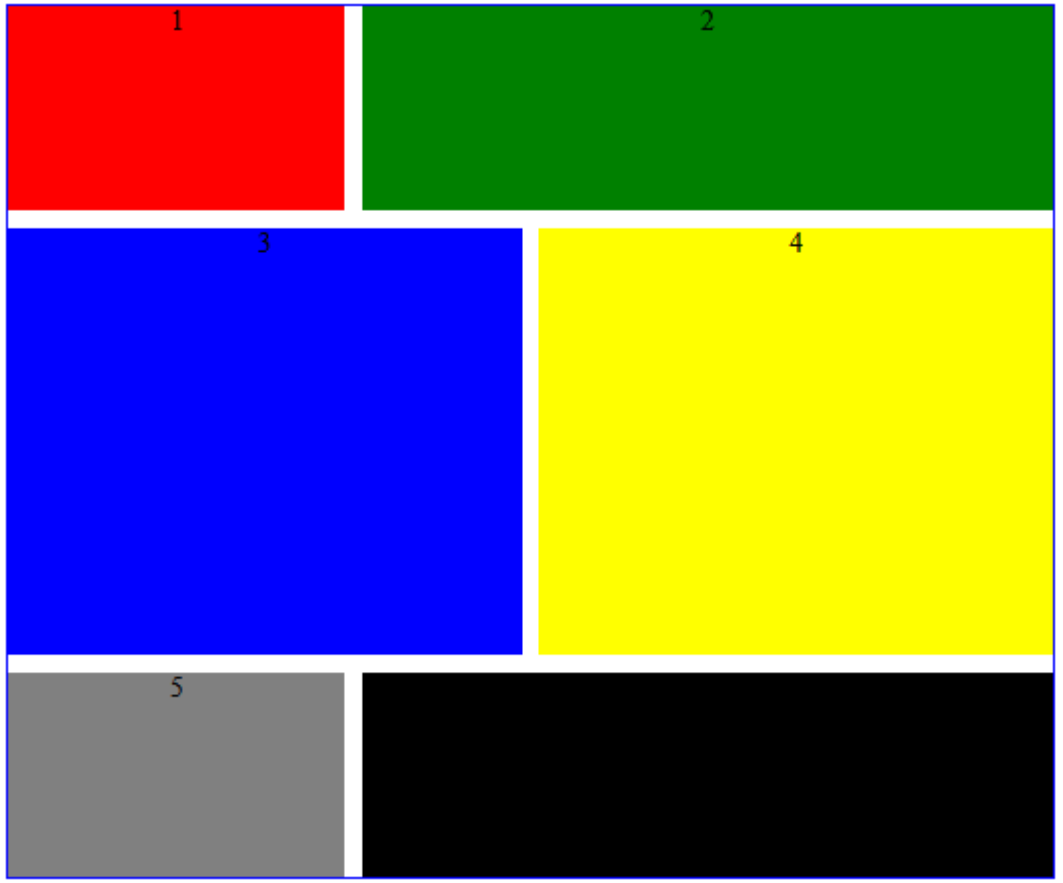


Source: https://www.w3schools.com/css/css_grid.asp

Positionner les éléments: Les lignes et les colonnes

Voici la grille de départ

Ce que nous voulons obtenir.



Positionner les éléments: Les lignes et les colonnes

Code html

```
<div id="conteneur">
    <div class="un">1</div>
    <div  class ="deux">2</div>
    <div class="trois">3</div>
    <div class="quatre">4</div>
    <div class="cinq">5</div>
    <div class="six">6</div>
</div>
```

Positionner les éléments: css de la grille obtenue

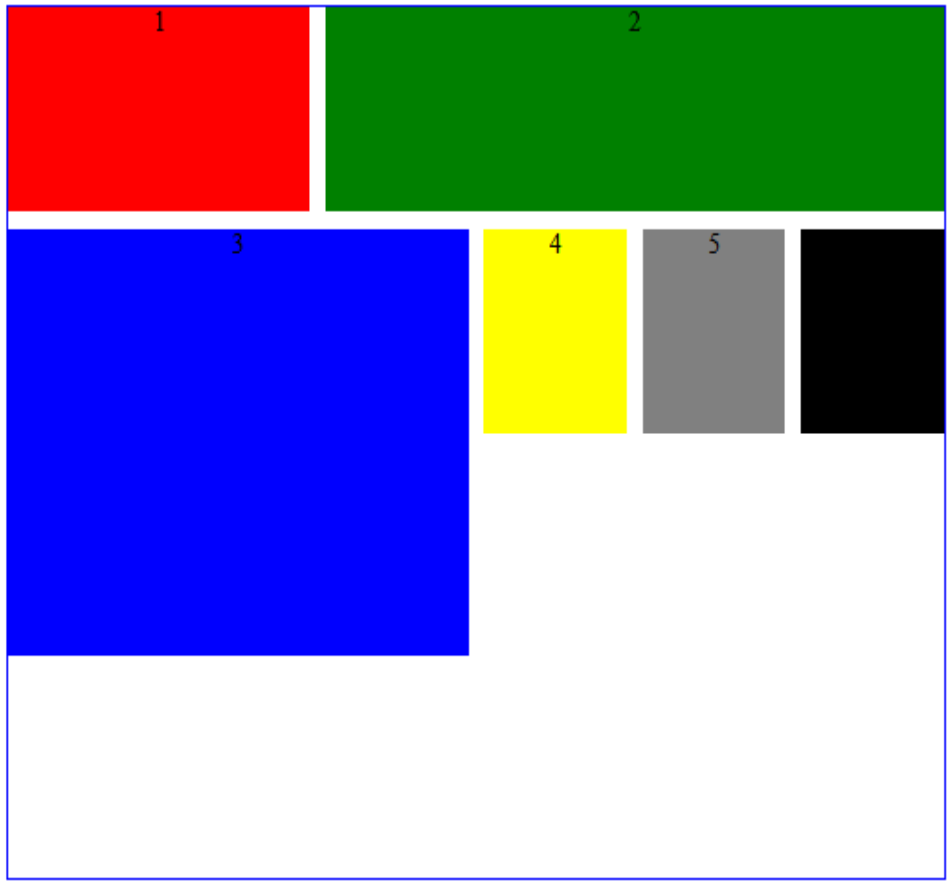
```css
#conteneur {
    text-align: center;
    display:grid;
    grid-template-columns: repeat(6,1fr);
    grid-template-rows: repeat(4, minmax(75px, auto));
    width: 600px;
    column-gap: 1px;
    height: 500px;
    border: 2px solid blue;
    margin:auto;
    grid-gap:10px;
   }

  .un{
    background-color: red;
    grid-column-start: 1;
    grid-column-end: 3;
  }
  .deux{
    background-color: green;
    grid-column-start: 3;
    grid-column-end: 7;
  }
```

```css
.trois{
    background-color: blue;
    grid-column-start: 1;
    grid-column-end: 4;
    grid-row-start: 2;
    grid-row-end:4
  }
  .quatre{
    background-color: yellow;
    grid-column-start: 4;
    grid-column-end: 7;
    grid-row-start: 2;
    grid-row-end:4
  }
  .cinq{
    background-color: gray;
    grid-column-start: 1;
    grid-column-end: 3;
    grid-row-start: 4;
    grid-row-end:5
  }
  .six{
    background-color: black;
    grid-column-start: 3;
    grid-column-end: 7;
    grid-row-start: 4;
    grid-row-end:5
```

Positionner les éléments: Les lignes et les colonnes

Voici la grille de départ

Ce que nous voulons obtenir.



Positionner les éléments: css de la grille obtenue

```css
#conteneur {
    text-align: center;
    display:grid;
    grid-template-columns: repeat(6,1fr);
    grid-template-rows: repeat(4, minmax(75px, auto));
    width: 600px;
    column-gap: 1px;
    height: 500px;
    border: 2px solid blue;
    margin:auto;
    grid-gap:10px;
   }

  .un{
    background-color: red;
    grid-column-start: 1;
    grid-column-end: 3;
  }
  .deux{
    background-color: green;
    grid-column-start: 3;
    grid-column-end: 7;
  }
```

```css
.trois{
    background-color: blue;
    grid-column-start: 1;
    grid-column-end: 4;
    grid-row-start: 2;
    grid-row-end:4 }
  .quatre{
    background-color: yellow;
    grid-column-start: 4;
    grid-column-end: 5;
    grid-row-start: 2;
    grid-row-end:3 }
  .cinq{
    background-color: gray;
    grid-column-start: 5;
    grid-column-end: 6;
    grid-row-start: 2;
    grid-row-end:3 }
  .six{
    background-color: black;
    grid-column-start: 6;
    grid-column-end: 7;
    grid-row-start: 2;
    grid-row-end:3; }
```

Positionner les éléments: écriture simplifiée

On peut remplacer les deux propriétés

`grid-column-start: 1;`

`grid-column-end: 3;`

par la seule propriété : `grid-column: 1/3;`

Il en est de même pour les propriétés:

grid-row-start: 2;

grid-row-end: 4 par `grid-row: 2/4;`

```
.un{
  background-color: red;
  grid-column: 1/3;
}
.deux{
  background-color: green;
  grid-column: 3/7;
}
.trois{
  background-color: blue;
  grid-column:1/4;
  grid-row: 2/4;
   }
.quatre{
  background-color: yellow;
  grid-column:4/7;
  grid-row: 2/4;
}
```

grid-area et grid-template-areas

- La propriété grid-area peut également être utilisée pour attribuer un nom à un élément de grille. Les éléments de grille nommés peuvent ensuite être référencés par la propriété `grid-template-areas` du conteneur de grille.
- peuvent ensuite être référencés et placés directement dans la grille.
- Nous allons toujours utiliser des div pour montrer un exemple

grid-area et grid-template-areas

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>grille</title>
    <link href="css/area1.css"rel="stylesheet">
</head>
<body>

        <div id="conteneur">
            <div class= "un"      id ="div1">Je suis en haut, je prends toute la place</div>
            <div class="deux"     id="div2"> Je suis au milieu et je prends de la place</div>
            <div class= "trois"   id="div3"> Je prends une petite place à gauche</div>
            <div class = "quatre" id="div4">Oups, je suis en bas, mais je prends toute la place</div>

        </div>
</body>
</html>
```

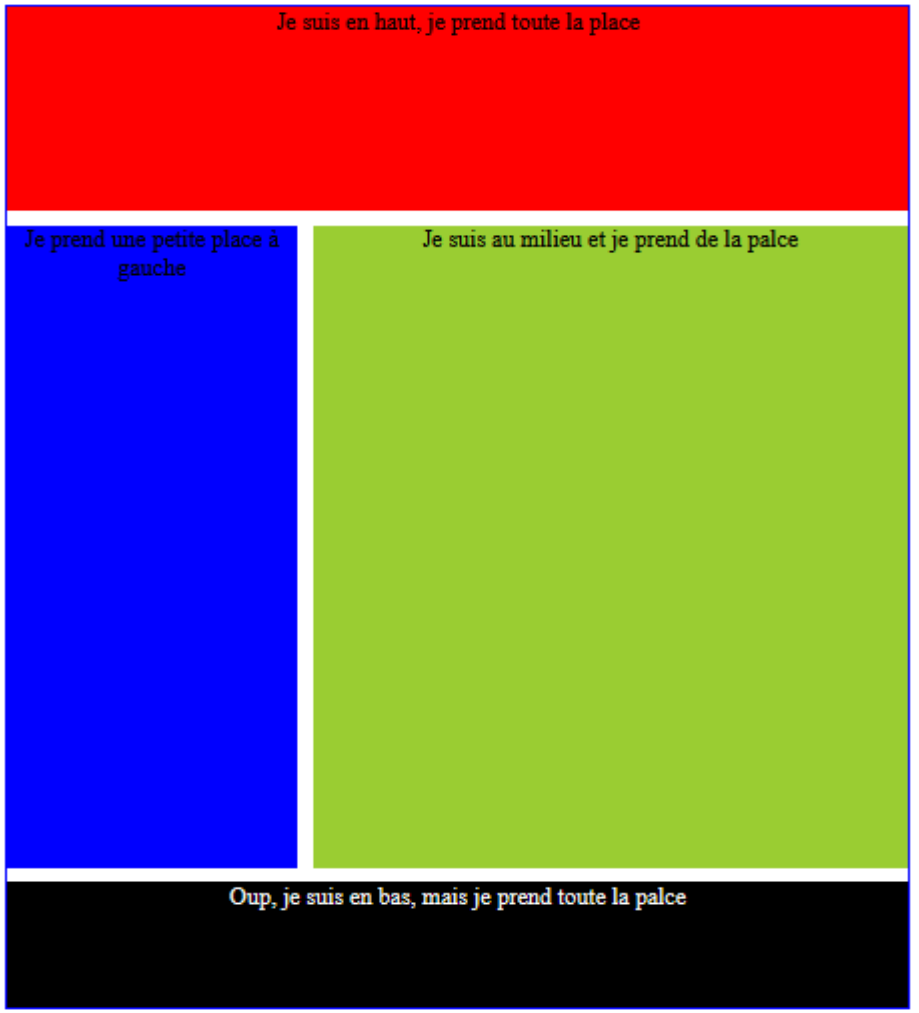
grid-area et grid-template-areas

```
#conteneur {
    text-align: center;
    display:grid;
    grid-template-columns: repeat(6,1fr);
    grid-template-rows: repeat(5, minmax(75px, auto));
    width: 600px;
    column-gap: 1px;
    height: 700px;
    border: 2px solid blue;
    margin:auto;
    grid-gap:10px;
    grid-template-areas:
    "div1 div1 div1 div1 div1 div1"
    "div3 div3 div2 div2 div2 div2"
    "div3 div3 div2 div2 div2 div2"
    "div3 div3 div2 div2 div2 div2"
    "div4 div4 div4 div4 div4 div4"
   }
```

```
#div1{
    grid-area: div1;
  }
  #div2{
    grid-area: div2;
  }
  #div3{
   grid-area: div3;
  }
  #div4{
    grid-area:div4;
  }
```

grid-area et grid-template-areas

```
.un{
   background-color: red;
   }
 .deux{
   background-color: yellowgreen;
    }
 .trois{
   background-color: blue;
    }
 .quatre{
   background-color: black;
   color: white;
 }
```



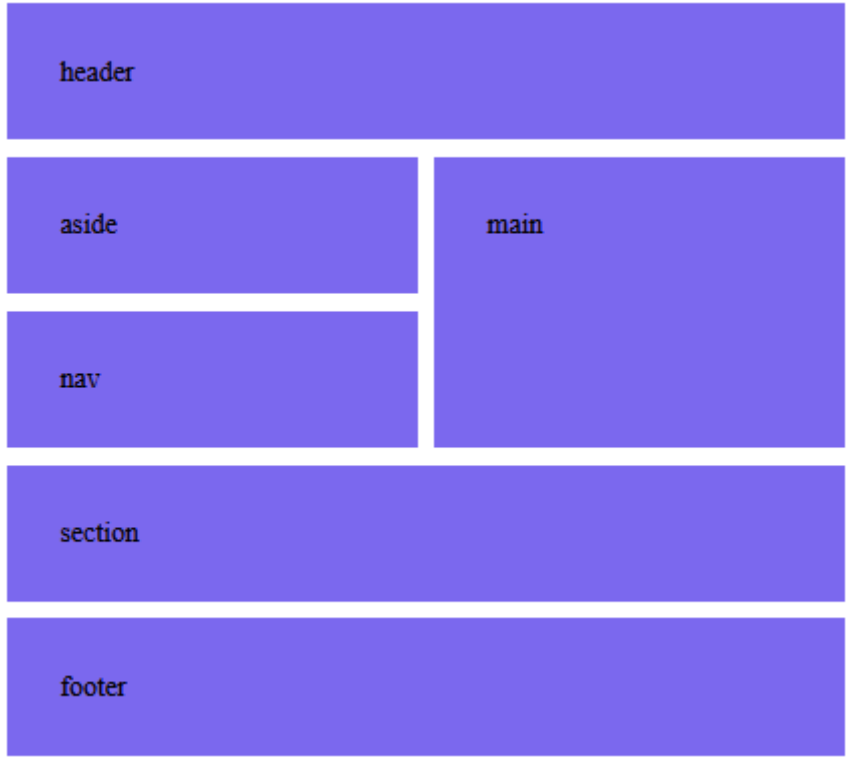
Les grilles et les éléments structurants

- La propriété grid-area peut également être utilisée pour attribuer un nom à un élément de grille. Les éléments de grille nommés peuvent ensuite être référencés par la propriété `grid-template-areas` du conteneur de grille.
- peuvent ensuite être référencés et placés directement dans la grille.
- Cette fois nous allons utiliser les éléments sémantiques appropriés et non plus des conteneurs généraux comme l'élément div
- Sans spécifier le nombre de colonnes et de rangées, ni la position exacte des éléments, ces derniers s'empilent simplement à la verticale.

Les grilles et les éléments structurants

- Imaginons que nous voulons obtenir la grille suivante:
- Nous avons un div principal, mais le reste des éléments sont structurants.
- Nous devons associer aux différents éléments structurants un grid-area de la façon suivante:

```
header {
    grid-area: header;
  }
  main {
    grid-area: main;
  }
  section {
    grid-area: section;
  }
Etc...
```



Les grilles et les éléments structurants

- Ensuite pour changer la position ou la superficie occupée par un élément se résume maintenant à "jouer" avec les zones du `grid-template-areas` dans le conteneur principal

```
grid-template-areas:
    "header  header  header  header"
    "aside   aside   main    main"
    "nav     nav     main    main"
    "section section section section"
    "footer  footer  footer  footer";
}
```

Les grilles et les éléments structurants

```
/*conteneur principal: la grille*/
#conteneur {
    display: grid;
    grid-template-columns: repeat(4, 1fr);  /* 4
colonnes */
    grid-auto-rows: minmax(75px, auto);      /*
autant de rangées qu'il faut */
    grid-gap: 10px;
    max-width: 480px;
    margin:  auto;
    grid-template-areas:
    "header  header  header  header"
    "aside   aside   main    main"
    "nav     nav     main    main"
    "section section section section"
    "footer  footer  footer  footer";
}
  #conteneur > * {
    background-color:mediumslateblue;
    padding: 30px;
  }
```

```
/*les grid area*/
header {
    grid-area:
header;
  }
  main {
    grid-area: main;
  }
  section {
    grid-area:
section;
  }
  aside {
    grid-area: aside;
  }
  nav {
    grid-area: nav;
  }
  footer {
    grid-area:
footer;
  }
```

```
    <!--le code HTML-->
 <div id="conteneur">
    <header>header</header>
    <main>main</main>
    <section>section</section>
    <aside>aside</aside>
     <nav>nav</nav>
      <footer>footer</footer>
   </div>
```

[Cliquez ici pour le code CSS]

Questions

CONCLUSION

QUESTIONS ??

Sources

- https://www.w3schools.com/html/
- https://prog101.com/cours/kb2/