CSS GRID

Introduction

- CSS a été créée 2017
- Permet de faire des **grilles** en **2 dimensions**.
- Facilite le design et l'agencement des sites
- Complément de **FlexBox** (1 dimension)

Première grille

Nous allons créer une **grille** de **3 colonnes** et de **2 lignes**.

Chaque **colonne** aura une largeur de **150px**.

Chaque **ligne** aura une **hauteur** de **150px**.

Le **gap** entre chaque **colonne** sera de **20px** et le **gap** entre chaque ligne sera de **30px**.

Code CSS du container :

```css
.container{
    background-color: #eee;
    width: 90%;
    margin: 20px auto;
    border: 2px solid blue;
    display : grid;
    grid-template-rows: 150px 150px;
    grid-template-columns: 150px 150px 150px;
    /*grid-row-gap: 30px;
    grid-column-gap: 20px;*/
    grid-gap: 30px 20px;
}
```

Repeat et Fractionnal Unit

Possibilité d'utiliser la fonction **repeat()** afin de définir les **dimensions** des **colonnes** et des **lignes**.

```
grid-template-rows: repeat(2,150px);
grid-template-columns: repeat(3,150px);
```

Autre possibilité

```
grid-template-rows: repeat(2,150px);
grid-template-columns: repeat(2,150px) 400px;
```

Fractionnal Unit : unité de mesure permettant de définir les **dimensions** des **colonnes** et des **lignes** en **fonction de l'espace disponible restant dans le container**.

Colonnes :

```
grid-template-rows: repeat(2,150px);
grid-template-columns: repeat(2,150px) 1fr;

grid-template-rows: repeat(2,150px);
grid-template-columns: repeat(3,1fr);

grid-template-rows: repeat(2,150px);
grid-template-columns: 1fr 2fr 1fr;
```

Lignes :

```
grid-template-rows: repeat(2,1fr);
grid-template-columns: 1fr 2fr 1fr;
```

Ici les lignes ont une hauteur en fonction du contenu.

Donnons maintenant une **hauteur à notre container :**

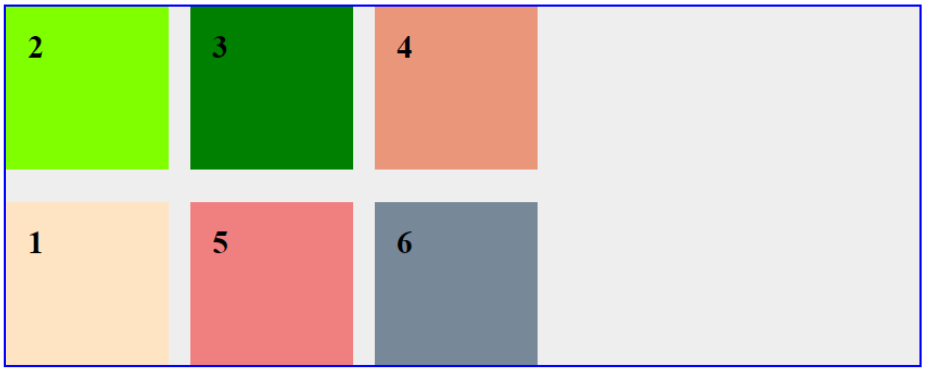
```
height: 800px;
```

Et regardons ce que cela produit !

Placement avec grid-row et grid-column

Possibilité de placer les items dans la grille à la position que l'on souhaite.

```
.item1 {
    background-color: bisque;
    grid-row-start: 2;
    grid-row-end: 3;
}
```

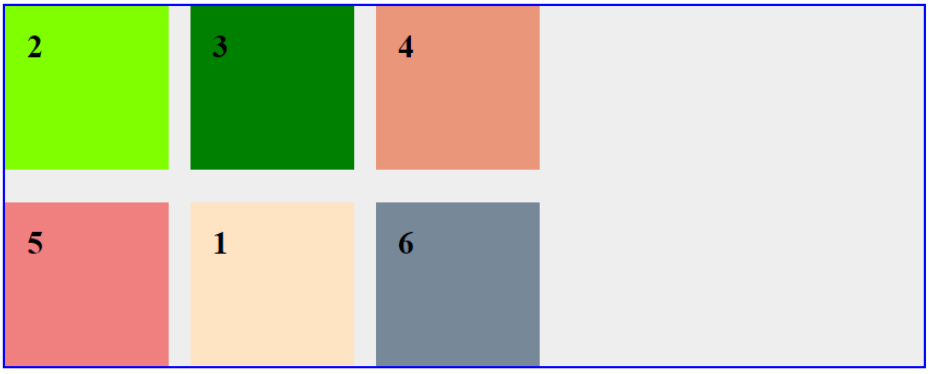


Simplification :

```
.item1 {
    background-color: bisque;
    grid-row: 2/3;
}
```

Autre exemple :

```
.item1 {
    background-color: bisque;
    grid-row: 2/3;
    grid-column-start: 2;
    grid-column-end: 3;
}
```



Simplification :

```
.item1 {
    background-color: bisque;
    grid-row: 2/3;
    grid-column: 2/3;
}
```

Simplification avec **grid-area** :

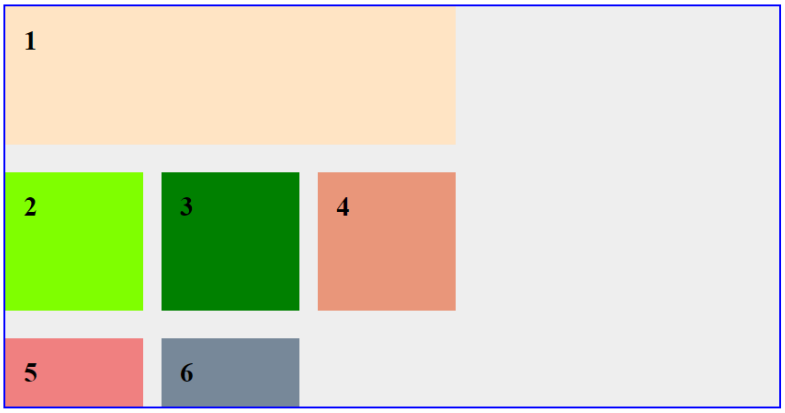
```
.item1 {
    background-color: bisque;
    grid-area: 2 / 2 / 3/ 3;
}
```

Placement sur plusieurs cellules

Possibilité de placer les items dans la grille sur plusieurs cellules.

On va définir des **aires** (**area**) dans laquelle placer les items.

```
.item1 {
    background-color: bisque;
    grid-row: 1 / 2;
    grid-column: 1 / 4;
}
```



On s'aperçoit qu'une **nouvelle ligne a été créée** ! On appelle cela une **grille implicite**.

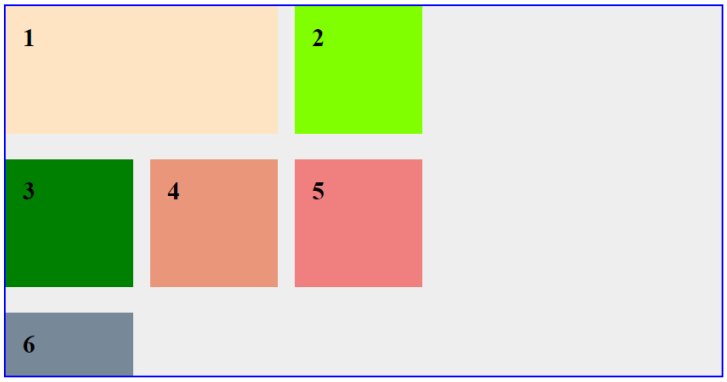
Elle a été créée car il n'y avait plus de place pour placer dans la **grille explicite** les items 5 et 6.

Pour aller à la **dernière colonne** (fin de la ligne) :

```
.item1 {
    background-color: bisque;
    grid-row: 1 / 2;
    grid-column: 1 / -1;
}
```

On peut utiliser également un **span** :

```
.item1 {
    background-color: bisque;
    grid-row: 1 / 2;
    grid-column: 1 / span 2;
}
```

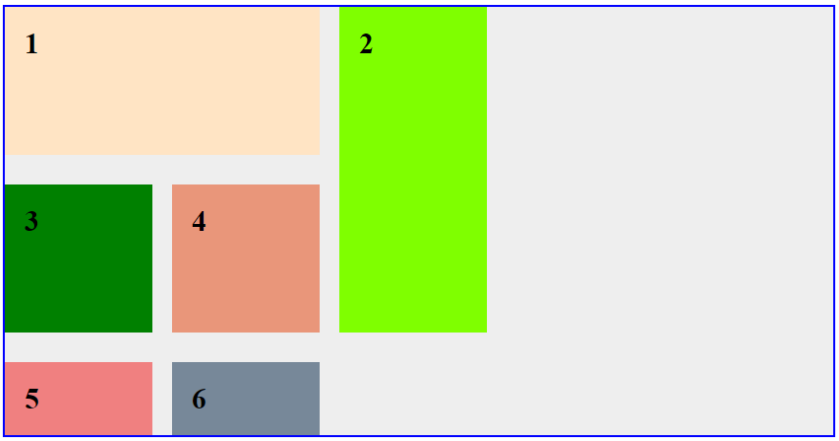


Simplification avec **grid-area** :

```
.item1 {
    background-color: bisque;
    grid-area: 1 / 1 / 2 / 3;
}
```

Placement d'un autre item sur plusieurs cellules :

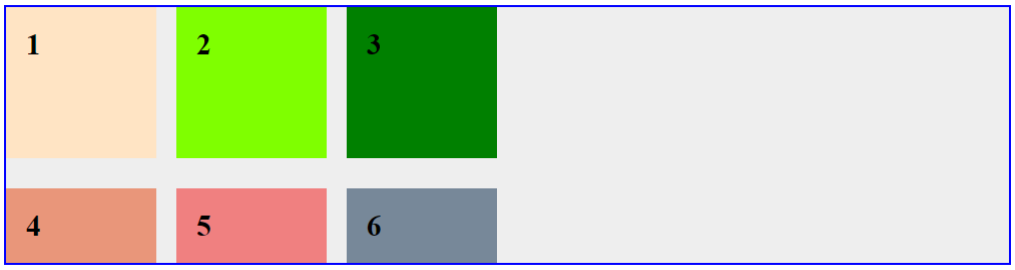
```
.item2 {
    background-color: chartreuse;
    grid-row: 1 / 3;
    grid-column: 3 / 4;
}

.item2 {
    background-color: chartreuse;
    grid-area: 1 / 3 / 3 / 4;
}
```



Grille explicite et grille implicite

Soit le container suivant définissant la **grille explicite** suivante :

```
display: grid;
grid-template-rows: repeat(1, 150px);
grid-template-columns: repeat(3, 150px);
grid-gap: 30px 20px;
```

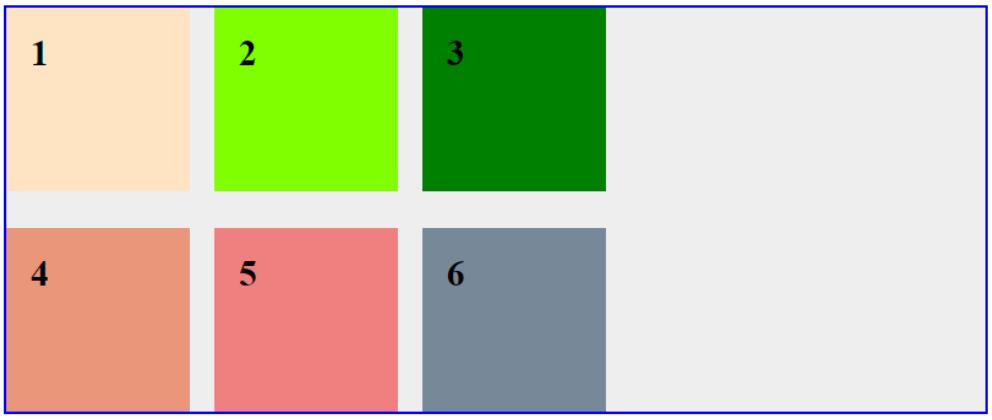


On s'aperçoit qu'une **nouvelle ligne a été créée** ! On appelle cela une **grille implicite**.

Par défaut, la **hauteur de la ligne** correspond à la **hauteur de son contenu**.

On peut spécifier une hauteur de ligne :

```
grid-auto-rows: 150px;
```



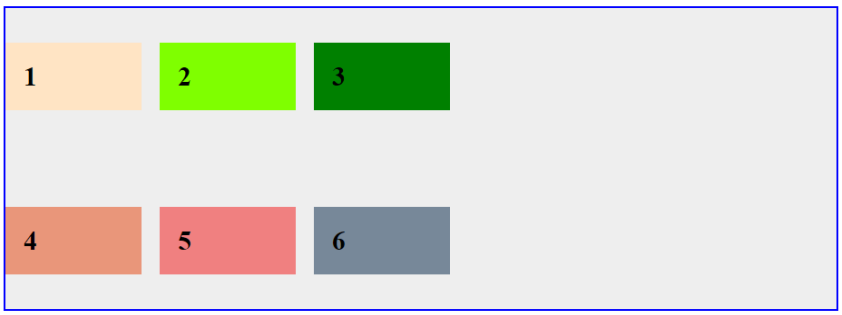
Aligner et center les items dans les cellules

Soit le container suivant définissant la **grille explicite** suivante :

```
display: grid;
grid-template-rows: repeat(2, 150px);
grid-template-columns: repeat(3, 150px);
grid-gap: 30px 20px;
```

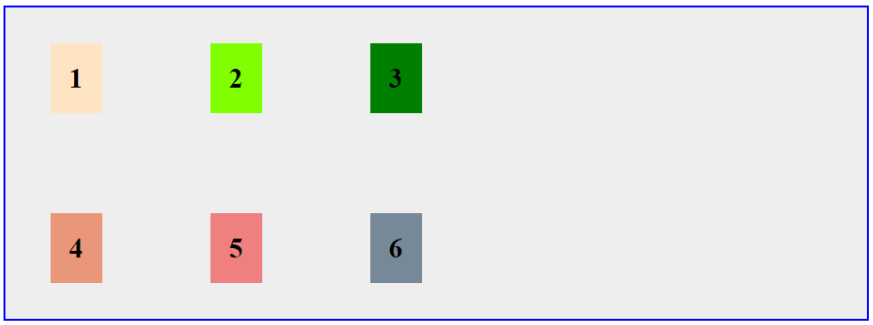
Pour **centrer** sur l'**axe y** :

```
align-items: center;
```



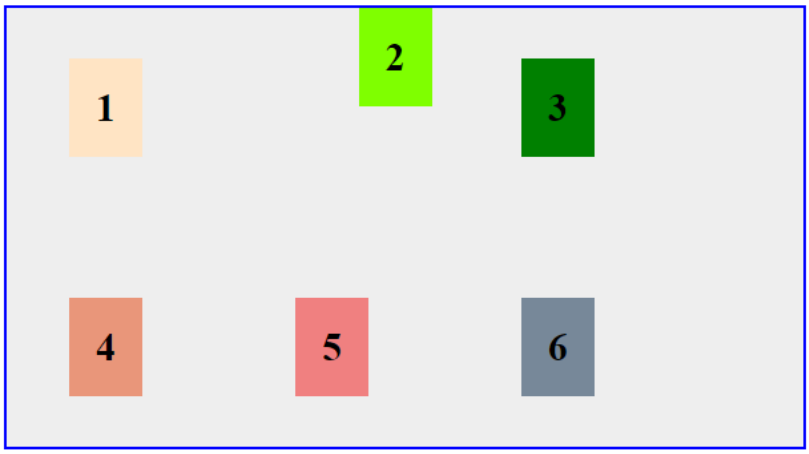
Pour **centrer** sur l'**axe x** :

```
justify-items: center;
```



Pour modifier l'alignement d'un item particulier :

```
.item2 {
    background-color: chartreuse;
    align-self: start;
    justify-self: end;
}
```



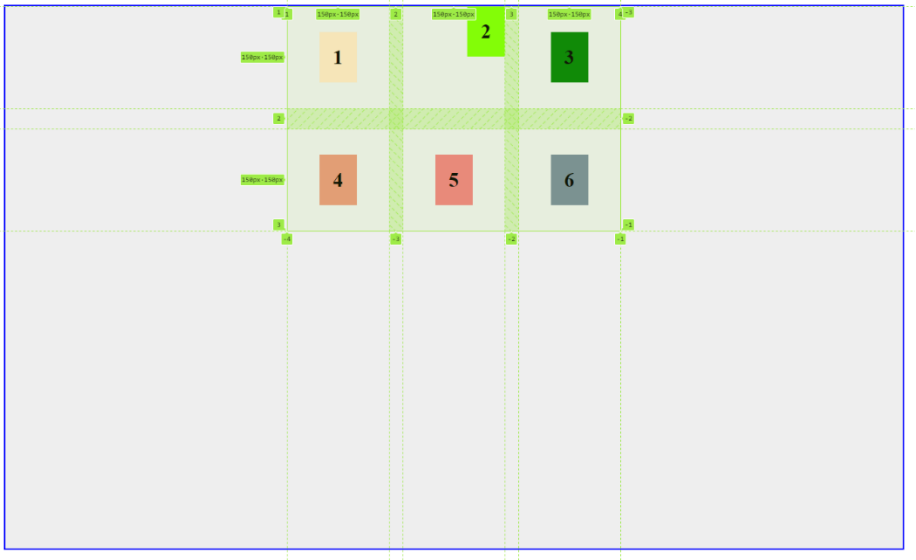
Aligner et center la grille

Donnons tout d'abord une hauteur à notre conteneur :

```
height: 800px;
```

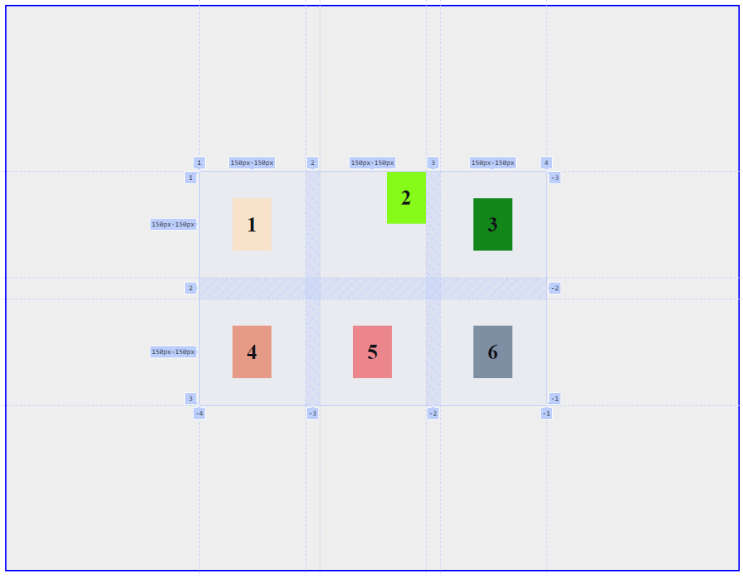
Pour **centrer** la **grille** dans le **conteneur** sur l'**axe x** :

```
justify-content: center;
```



Pour **centrer** la **grille** dans le **conteneur** sur l'**axe y** :

```
align-content: center;
```



La fonction minmax()

La fonction **minmax()** permet de définir un **intervalle de taille** pour une **colonne** ou une **ligne**.

Soit la grille explicite suivante :

```
.container{
    background-color: #eee;
    width: 90%;
    margin: 20px auto;
    border: 2px solid blue;
    display: grid;
    grid-template-rows: repeat(2, 150px);
    grid-template-columns: repeat(3, 150px);
}
```

Modifions la taille des colonnes :

1. `grid-template-columns: minmax(250px,500px) repeat(2,150px);`
2. `grid-template-columns: minmax(250px,500px) repeat(2,1fr);`

Grille responsive

Possibilité de créer simplement une **grille responsive** qui s'adapte aux différentes résolutions.

Tout d'abord, on va utiliser les propriétés **auto-fill** et **auto-fit**.

La propriété **auto-fill** :

```
grid-template-columns: repeat(auto-fill, 100px);
```

La propriété **auto-fit** :

```
grid-template-columns: repeat(auto-fit, 100px);
```

Pour faire du **responsive**, on va combiner les propriétés ci-dessous avec la fonction **minmax()**.

Avec **auto-fill** :

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

Avec **auto-fit** :

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