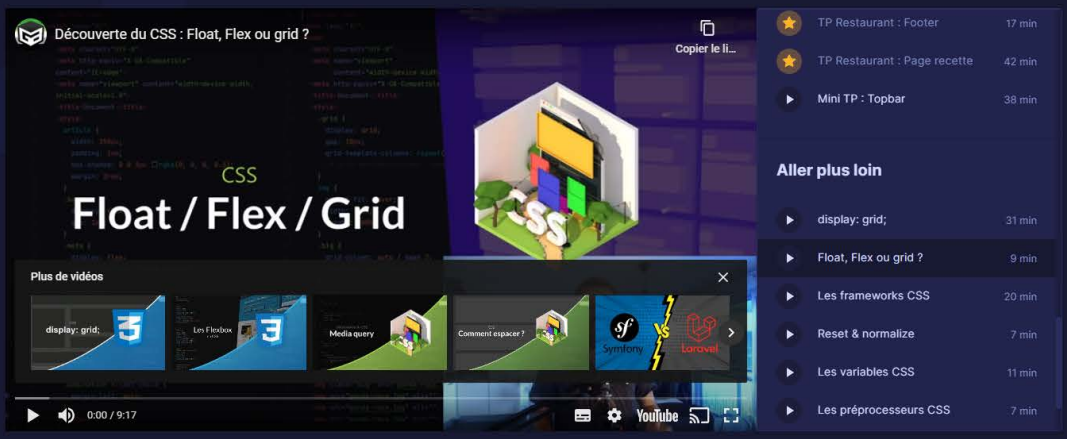
# Float, Flex ou grid ?




Débutant

Télécharger la vidéo

## À propos de ce tutoriel

Publié Il y a environ un an

Dans ce tutoriel nous allons faire le point sur les différentes méthodes de placement des éléments et leur cas d'utilisation.

### Technologies utilisées

CSS

**Auteur :**
Grafikart

### Partager

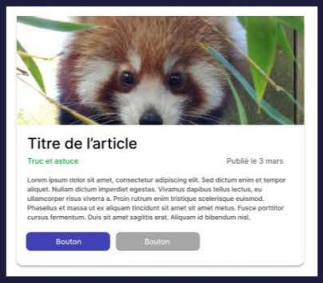
Proposer une correction

## Float

L'utitilisation de `float:left` permet de retirer un élément du flux normal et ne sera aujourd'hui utilisé que pour placer un élément à gauche ou à droite avec les autres éléments "inline" qui se placent autour (image d'illustration dans un article par exemple).

## Flex

Le `display:flex` permet de changer la disposition des éléments avec un positionnement en ligne ou en colonne. Les éléments enfants d'une **Flexbox** ont alors la responsabilité de définir leur largeur / hauteur via les propriétés `width/height` et via le `flex-basis`.



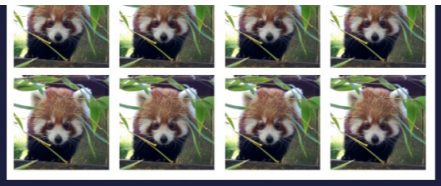

L'approche `flexbox` est donc utile pour changer la direction de placement des éléments (pour par exemple les placer les uns à côté des autres) sans forcément avoir de dimensions précises pour les enfants. On aura donc tendance à les utiliser dans les cas suivants :

- On veut aligner les élément verticalement ou horizontalement en insérant un espace avec la propriété `gap`.
- On a une grille qui n'est pas uniforme (structure des colonnes qui varient entre chaque ligne).

## Grid

La grosse particularité du `display: grid` est le fait que l'élément parent a la responsabilité de définir la structure via un `grid-template-columns` ou `grid-template-rows`. Cela permet de ne pas avoir à ajouter de règles sur les éléments enfants.

Typiquement, comme son nom l'indique, ce type de positionnement est très adapté à une disposition en grille

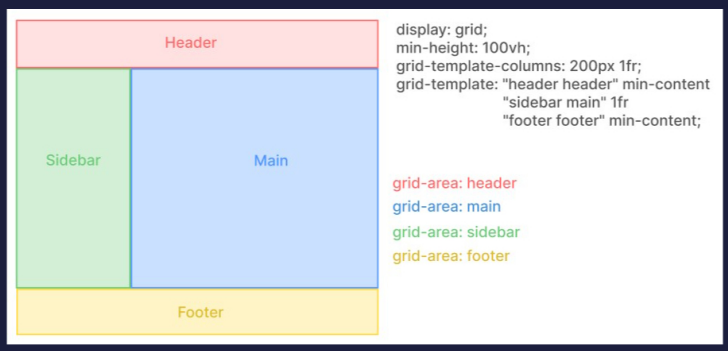


```
.grid {
    display: grid;
    gap: 10px;
    grid-template-columns: repeat(5, 1fr);
}
```

On peut aussi facilement créer une structure responsive qui adapte le nombre d'élément en fonction de la largeur souhaité pour les éléments

```
.grid {
    display: grid;
    gap: 10px;
    grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));
}
```

On peut aussi utiliser ce type de disposition pour créer des structures de page rapidement grâce notamment au `grid-area`.




```
<style>
.layout {
    display: grid;
    min-height: 100vh;
    grid-template: "header header" min-content
                   "sidebar main" 1fr
                   "footer footer" min-content;
    grid-template-columns: 250px 1fr;
}
header {
    grid-area: header;
}
aside {
    grid-area: sidebar;
}
main {
    grid-area: main;
}
footer {
    grid-area: footer;
}
</style>
<div class="layout">
    <header>Header</header>
    <aside>Sidebar</aside>
    <main>Contenu</main>
    <footer>Footer</footer>
</div>
```

Cette approche permet de piloter facilement la structure depuis l'élément parent `.layout` en changeant le template via la propriété `grid-template`. Ce type de positionnement est donc intéressant pour les cas suivante :

- On souhaite utiliser une grille simple
- On veut créer une structure qui peut être décomposer sous forme de colonnes (constantes entre chaque ligne).

## 4 commentaires

NOM D'UTILISATEUR

VOTRE MESSAGE