CSS3 – Grid Layout

- La taille dans une grille
- Flux des éléments
- Distribution des éléments
- Dense ou sparse
- Placement des éléments
- Fonction minmax()
- Auto-fill et auto-fit
- Ressources

La taille dans une grille

- Quelle taille est prise en compte ? Celle de la cellule logique ou de l'élément HTML physique ?
- **Par défaut**, un élément occupe toute la largeur et la hauteur de sa cellule
- Si la largeur d'un élément est **exprimée en %**, cette valeur se rapporte à la **cellule occupée**
- Si l'élément est **plus grand que sa cellule**, il dépasse, au risque de se superposer à ses voisins (pas de modification de taille comme dans un tableau HTML)

Flux des éléments

- Dans le modèle Grid Layout, lorsque les éléments ont la liberté de se placer automatiquement, ils adoptent un flot pouvant être horizontal ou vertical.
- C’est que l’on appelle **le « flux »**
- Que la grille soit explicite ou implicite, il n’est pas forcément nécessaire de préciser quelle place est occupée par chacun des enfants.
- Ils bénéficient **par défaut** d’une distribution et d’un placement automatiques.

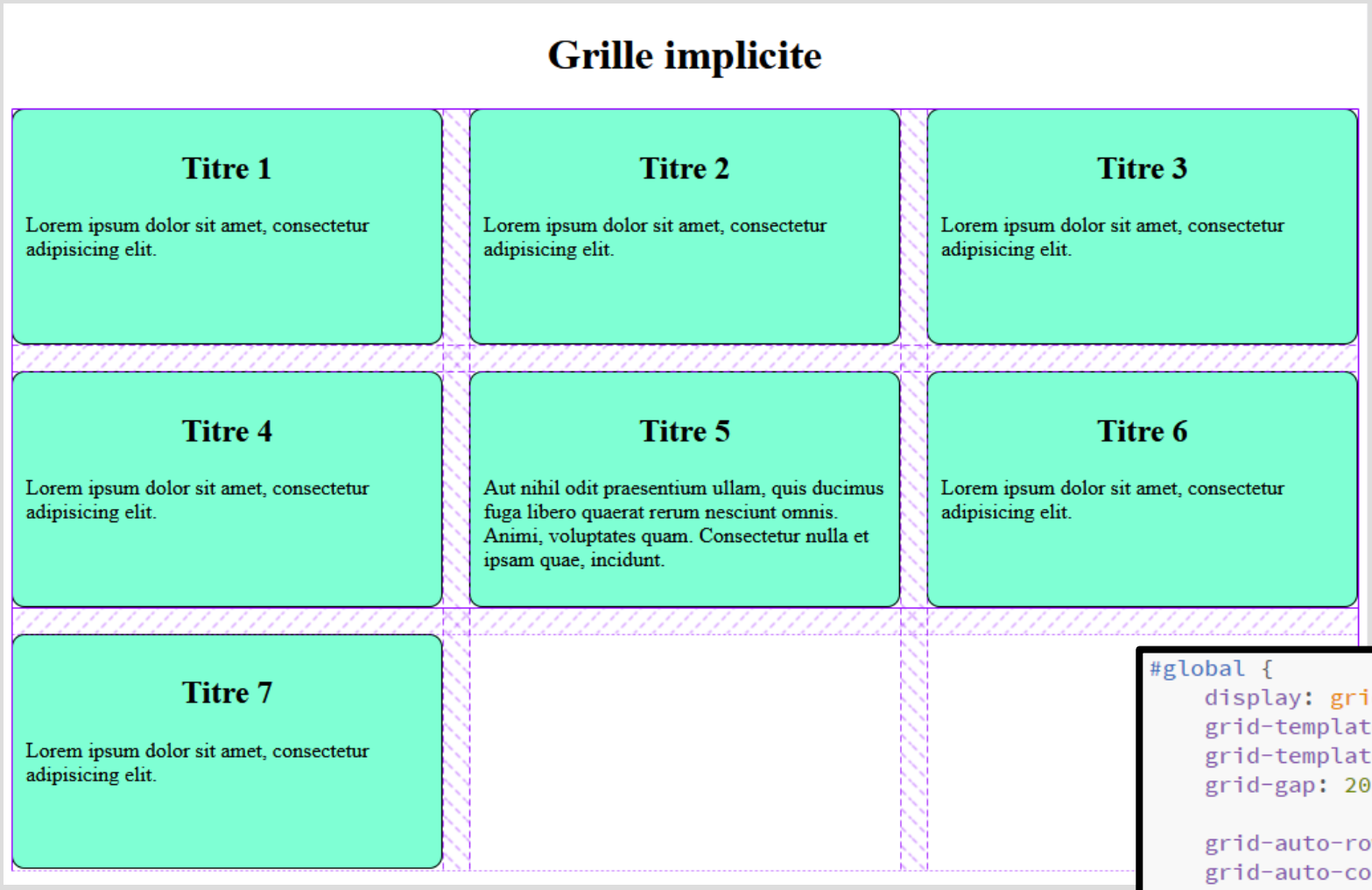
Distribution par défaut

- **Par défaut**, le **remplissage d'une grille** se fait **sous forme horizontale** : chaque grid item occupe une grid cell, et se place sur la même ligne et à la droite de l'élément précédent.
- Le principe de la **distribution** au sein de Grid Layout repose sur **deux règles** simples :
  - On remplit une rangée après l'autre. Quand une rangée est remplie, on passe à la suivante.
  - Quand il n'y a plus de rangée prévue pour accueillir les éléments, une nouvelle rangée implicite est générée.

Changer l’axe de distribution

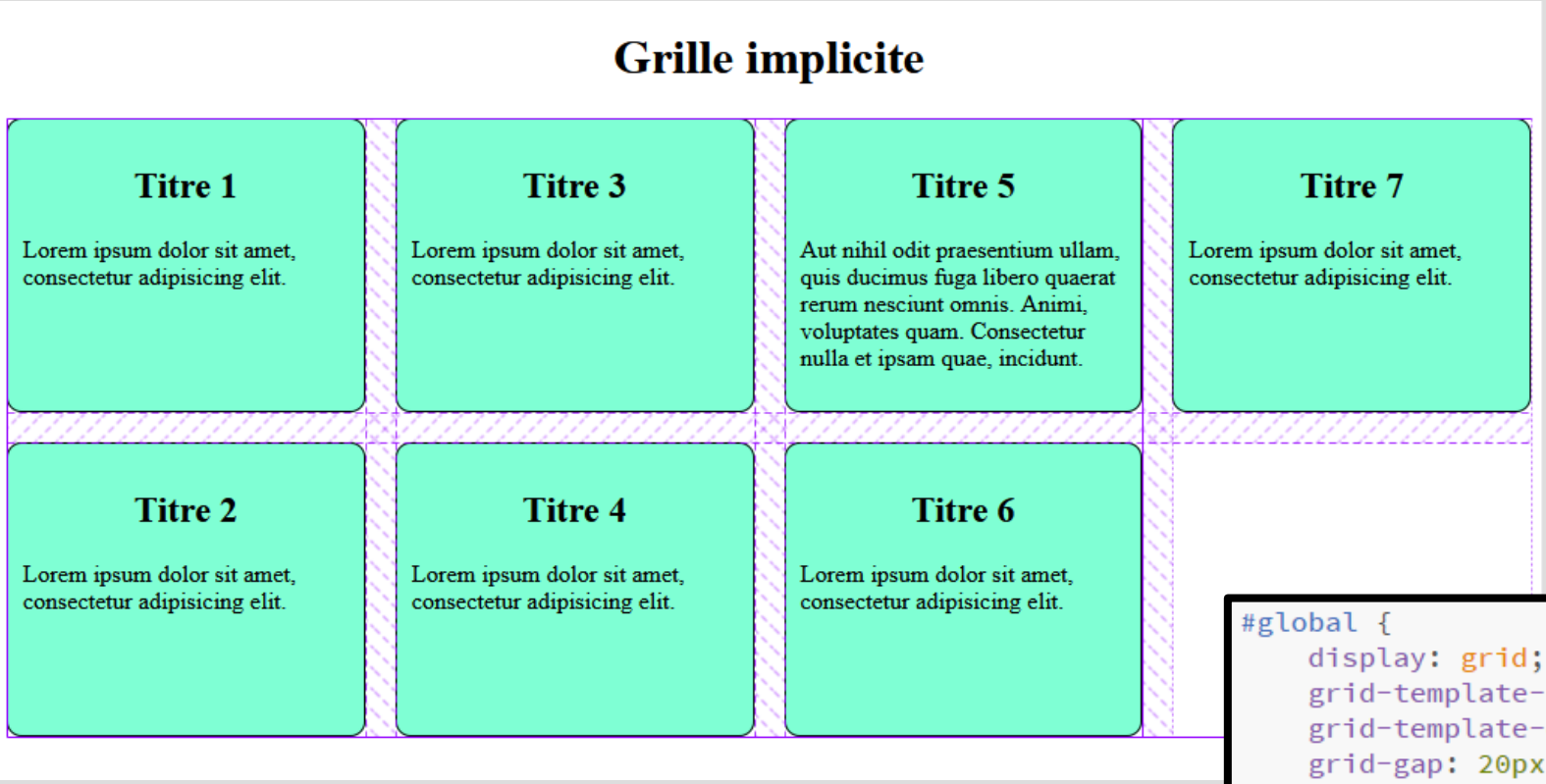
- L’axe de distribution automatique des boîtes est régulé par la propriété **grid-auto-flow**.
  - **grid-auto-flow: row;** : la distribution se fait par ligne, les éléments s’écoulent horizontalement
  - **grid-auto-flow: column;** : l’empilement est vertical, une colonne après l’autre.
- Grâce à la valeur **dense**, la propriété grid-auto-flow va permettre à la grille de combler après coup les cellules laissées vides.
- La valeur opposée à dense est **sparse**, c’est la **valeur par défaut**

Exemple : grid-auto-flow: row;




```css
#global {
    display: grid;
    grid-template-rows: 1fr 1fr;
    grid-template-columns: 1fr 1fr 1fr;
    grid-gap: 20px;

    grid-auto-rows: 1fr;
    grid-auto-columns: 1fr;

    grid-auto-flow: row;
}
```

Exemple : grid-auto-flow: column;




```css
#global {
    display: grid;
    grid-template-rows: 1fr 1fr;
    grid-template-columns: 1fr 1fr 1fr;
    grid-gap: 20px;

    grid-auto-rows: 1fr;
    grid-auto-columns: 1fr;

    grid-auto-flow: column;
}
```

Distribution des éléments

- Donc, pour gérer les 2 axes : **grid-auto-flow** peut accepter 2 valeurs à la fois si elles sont séparées d'un espace. Finalement, il y a 4 cas :
  - **grid-auto-flow: row sparse;** (valeur par défaut) distribution horizontale non dense
  - **grid-auto-flow: row dense;** distribution horizontale dense
  - **grid-auto-flow: column sparse;** distribution verticale non dense
  - **grid-auto-flow: column dense;** distribution verticale dense.

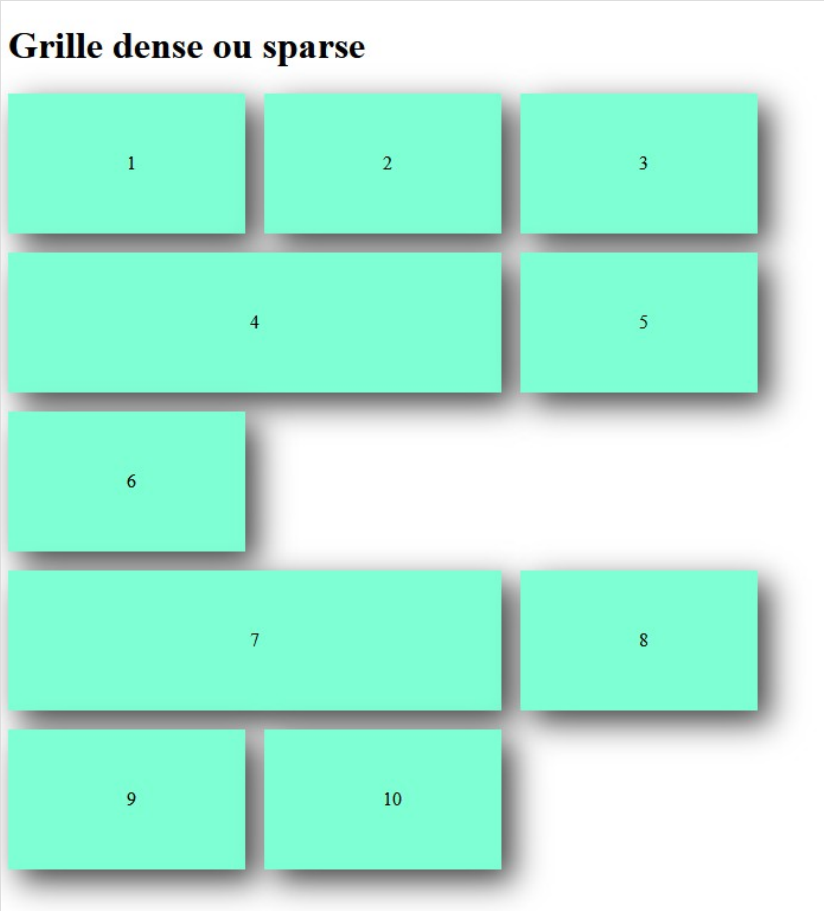
Exemple – HTML

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Grille dense ou sparse</title>
    <link rel="stylesheet" href="dense.css">
</head>
<body>
    <h1>Grille dense ou sparse</h1>
    <section>
        <article>1</article>
        <article>2</article>
        <article>3</article>
        <article class="en-avant">4</article>
        <article>5</article>
        <article>6</article>
        <article class="en-avant">7</article>
        <article>8</article>
        <article>9</article>
        <article>10</article>
    </section>
</body>
</html>
```

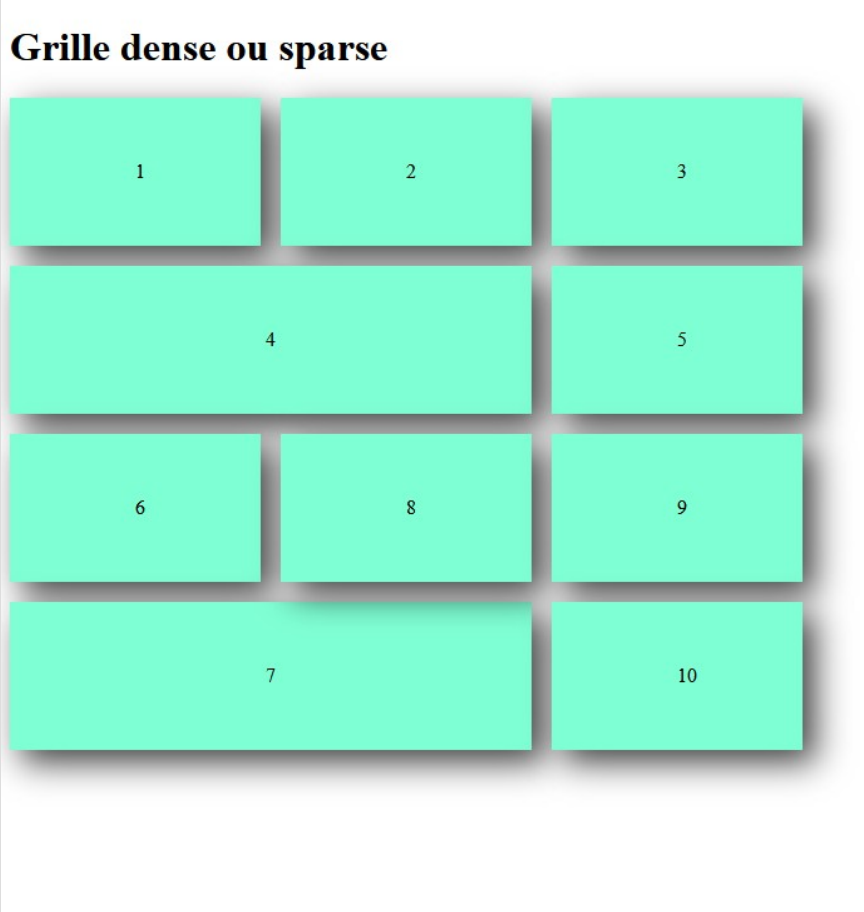
Exemple – CSS

```css
section {
    display: grid;

    grid-template-columns: repeat(3, 200px);
    grid-template-rows: repeat(3, 1fr);
    grid-gap: 1em;

    grid-auto-flow: row dense;
}

article {
    background-color: aquamarine;
    box-shadow: 10px 10px 25px 0px rgba(0,0,0,0.75);
    text-align: center;
    padding: 50px 100px;
}

.en-avant {
    grid-column: 1 / span 2;
}
```

Exemple de distribution dense



grid-auto-flow: row sparse;



grid-auto-flow: row dense;

Placement des éléments

- Trois types de placements d’éléments peuvent être rencontrés au sein d’une grille CSS :
  - le placement automatique (auto-placement)
  - le placement défini (definite position)
  - le placement semi-défini

Placement automatique

- Pour un grid item, la valeur **par défaut** des propriétés grid-row et grid-column est **auto**.
- Un élément dit « en **placement automatique** » ou « auto-placé », est tout simplement un **enfant** de grille **ayant conservé toutes ses valeurs par défaut** et qui observe les règles de distribution automatique.
- En fixant l'une ou l'autre valeur, l'élément sortira **partiellement ou totalement** du modèle automatique et correspondra soit à un placement **défini** soit à un placement **semi-défini**.

Placement défini ou semi-défini

- **Placement défini** (definite position)
  - Un grid item **défini** est un élément disposant d'un ***grid-row*** **ET** d'un ***grid-column*** **explicites**.
  - Ce type d'élément est toujours **prioritaire** lors du remplissage de la grille.
- **Placement semi-défini**
  - Un grid item **semi-défini** bénéficie d'un ***grid-row*** **OU** d'un ***grid-column*** **explicite**, l'autre propriété étant laissée à sa valeur auto par défaut.

Algorithme de placement

- Pour tenir compte de tous les cas possibles, l'algorithme de placement des objets sur la grille est plutôt complexe :
  - D'abord, les éléments en placement défini sont dessinés en priorité dans la grille.
  - Puis les éléments automatiques se placent un par un dans les espaces libres, selon la distribution en vigueur.
  - Ensuite pour les éléments semi-définis,
    - si grid-row est uniquement précisé, l'élément est placé en priorité dans la grille, puis les autres éléments auto-placés sont disposés, bouchant les espaces libres.
    - si grid-column est uniquement renseigné, l'élément est placé en générant une nouvelle rangée si nécessaire, puis les autres s'écoulent à sa suite. Des espaces libres peuvent demeurer dans la rangée précédente.

Fonction minmax( )

- Que l'on crée une grille explicite ou implicite, il peut être utile d'**assigner une taille minimum**, **qui s'agrandit pour s'adapter** au contenu.
- Par exemple, on veut que les rangées ne soient jamais moins hautes que 100px, mais qu'elles aillent jusqu'à 300 px si le contenu le nécessite.
- La fonction **minmax( )** permet ce comportement.

```
minmax(100px, 300px);
```

- Avec **auto** comme maximum, la taille s'adaptera à la hauteur du contenu.

Auto-fill et auto-fit

- La fonction **repeat()** permet de répéter des motifs correspondant aux tailles des pistes.
- Les mots-clés **auto-fit** et **auto-fill** augmentent les possibilités de repeat() pour la conception de grilles dynamiques
  - Ces valeurs magiques permettent d'**adapter** automatiquement le **nombre de pistes à l'espace disponible** au sein de la grille.
  - La **différence** entre auto-fit et auto-fill est que la première conserve les pistes vides et que la seconde les supprime.

Exemple – HTML

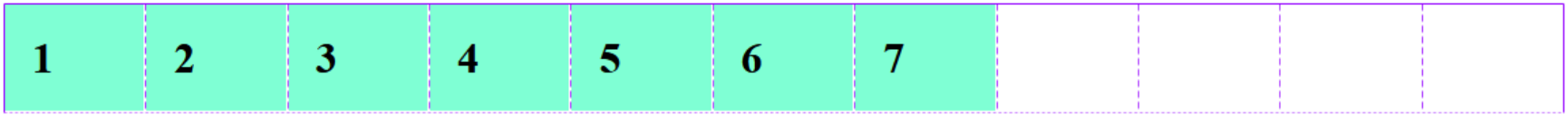
```html
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <link rel="stylesheet" href="fill-fit.css">
</head>
<body>
    <h2>Avec auto-fill</h2>
    <section class="fill">
      <article>1</article>
      <article>2</article>
      <article>3</article>
      <article>4</article>
      <article>5</article>
      <article>6</article>
      <article>7</article>
    </section>
    <hr>
    <h2>Avec auto-fit</h2>
    <section class="fit">
        <article>1</article>
        <article>2</article>
        <article>3</article>
        <article>4</article>
        <article>5</article>
        <article>6</article>
        <article>7</article>
    </section>
</body>
</html>
```

Exemple – CSS

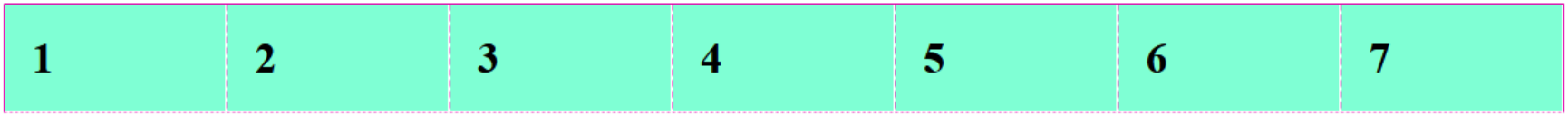
```css
body {
    padding: 2em;
}

hr {
    margin: 80px;
}

section {
    display: grid;
}

article {
  background-color: aquamarine;
  padding: 20px;
  color: #000;
  font-weight: bold;
  font-size: 2em;
  border: 1px solid #fff;
}

.fill {
    grid-template-columns: repeat(auto-fill, minmax(100px, 1fr));
}

.fit {
    grid-template-columns: repeat(auto-fit, minmax(100px, 1fr));
}
```

Exemple – Auto-fill et auto-fit

Avec auto-fill



Avec auto-fit



Ressources

- La Cascade
- MDN – CSS Grid
- Learn CSS Grid
- Grid By Example
- Grid Reference
- Débuter avec Grid
- 50 nuances de Grid
- Grid Garden
- Grid Cheat Sheet
- Pense-bête Grid
- Copier/Coller Grid
- Grid Builder
- Grid Generator
- Grid vs. Bootstrap