Architecture et langages Web
(NFA040)

CSS : grid (tutoriel sur les concepts de base)

Remarque. Ce tutoriel est extrait de la section de MDN intitulée : Concepts de base de la disposition en grille. Les sources des exemples accompagnent ce sujet.

Le module CSS Grid Layout ajoute à CSS une grille à deux dimensions. Les grilles peuvent être utilisées pour agencer des pages entières ou de petits éléments d'interface. Cet article introduit CSS Grid Layout, et la terminologie de la spécification CSS Grid Layout Level 1. Les fonctionnalités évoquées seront expliquées plus en détail dans le reste du guide.

Qu'est-ce qu'une grille ?

Une grille est un ensemble de lignes horizontales et verticales qui se croisent – les premières définissant les rangées, et les secondes les colonnes. Les éléments sont placés sur la grille en fonction de ces rangées et colonnes. Les fonctionnalités sont les suivantes :

Pistes à taille fixe ou variable

On peut créer une grille avec des pistes à taille fixes en utilisant une unité comme le pixel. Pour les pistes à taille variable on peut utiliser le pourcentage ou l'unité `fr` créée à cet effet.

Placement des éléments

Pour placer les éléments sur la grille, on peut utiliser le numéro ou le nom d'une ligne, ou cibler une zone particulière. La grille contient aussi un algorithme pour placer les éléments qui n'ont pas été placés explicitement.

Création de pistes supplémentaires pour du contenu

Lorsqu'une grille explicite n'est pas définie, la spécification prend en charge le contenu défini en dehors d'une grille en ajoutant des colonnes et des rangées. Cela comprend des fonctionnalités telles qu'« ajouter autant de colonnes que possible dans le conteneur ».

Contrôle de l'alignement

On peut contrôler l'alignement des éléments dans une zone de la grille, ainsi que celui de l'ensemble de la grille.

Contrôle des contenus qui se chevauchent

Il peut arriver que l'on place plusieurs éléments dans une même cellule, ou que des zones se chevauchent. La superposition peut être contrôlée à l'aide de la propriété `z-index`.

La grille est une spécification puissante qui peut être combinée avec d'autres modules CSS tels que flexbox. Le point de départ est le **conteneur**.

1 Conteneur

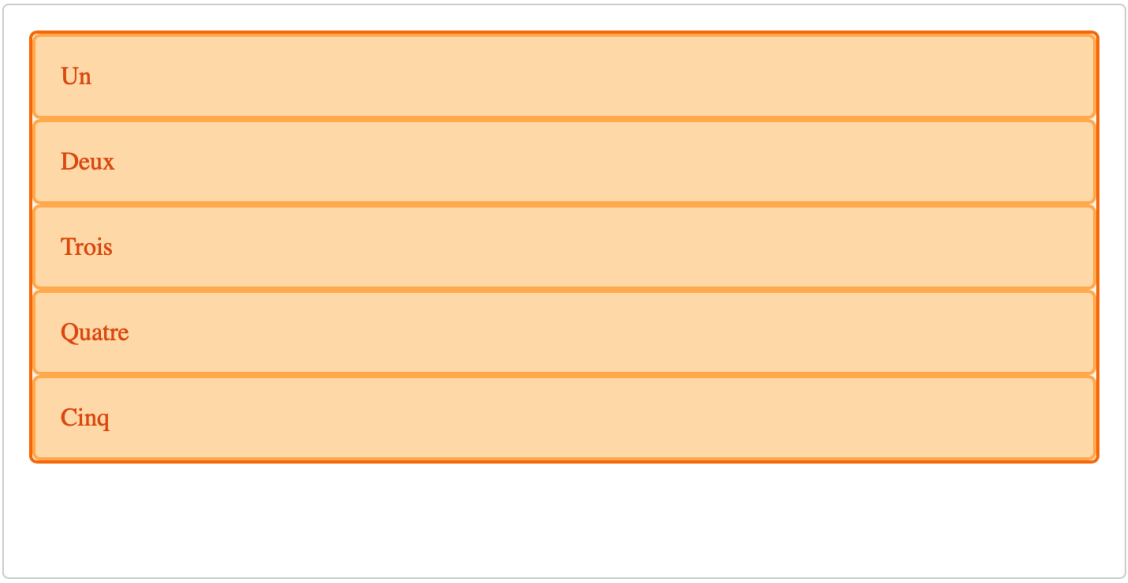
À partir du moment où on crée un *conteneur* en déclarant la propriété `display: grid` ou `display: inline-grid` sur un élément, tous les *enfants directs* de cet élément deviennet des *éléments de grille*.

Cet exemple montre une div avec une classe `.wrapper`, avec cinq éléments enfants.

```
<div class="wrapper">
  <div>Un</div>
  <div>Deux</div>
  <div>Trois</div>
  <div>Quatre</div>
  <div>Cinq</div>
</div>
```

On transforme `.wrapper` en conteneur.

```
.wrapper {
  display: grid;
}
```

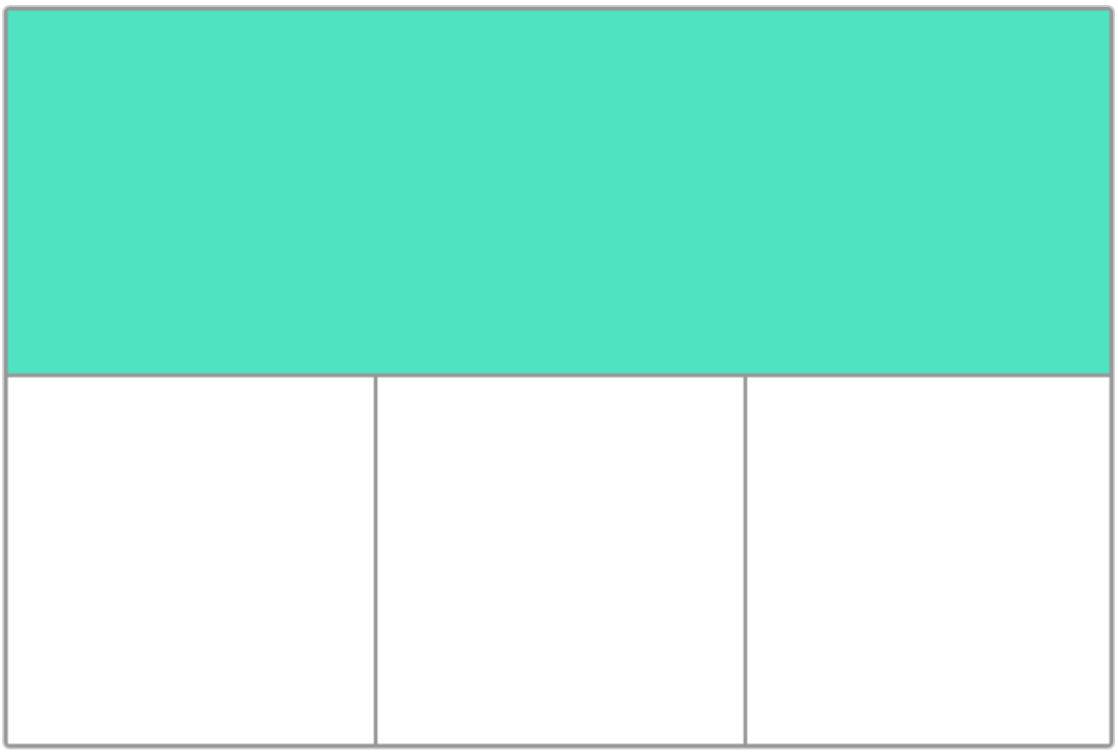


Tous les enfants directs sont maintenant des éléments de grille. On ne voit pas la différence dans un navigateur, car la grille n'a qu'une seule colonne. Vous trouverez sans doute utile de travailler avec Firefox ou Google Chrome, qui proposent un inspecteur de grille dans les outils de développement. Cet outil vous permettra de mieux comprendre le fonctionnement de CSS Grid Layout.

2 Pistes

Les propriétés `grid-template-columns` et `grid-template-rows` permettent de définir des colonnes et des rangées. Celles-ci définissent les pistes. Une *piste* est l'espace entre deux lignes

adjacentes d'une grille. L'image ci-dessous colore une piste de la grille, correspondant à la première rangée de la grille.



Les pistes sont définies dans la grille explicite à l'aide des propriétés `grid-template-columns` et `grid-template-rows`, ou des propriétés raccourcies `grid` ou `grid-template`. Les pistes sont aussi créées dans la grille implicite en positionnant un élément de grille en dehors des pistes créées dans la grille explicite.

Exemple simple

On peut ajouter la propriété `grid-template-columns` à notre exemple précédent, pour définir la taille des colonnes.

Nous avons créé une grille avec trois pistes de 200 pixels de large. Chaque élément sera disposé dans l'une des cellules de la grille.

```
<div class="wrapper">
  <div>Un</div>
  <div>Deux</div>
  <div>Trois</div>
  <div>Quatre</div>
  <div>Cinq</div>
</div>
```

```
.wrapper {
  display: grid;
  grid-template-columns: 200px 200px 200px;
}
```

3 L'unité `fr`

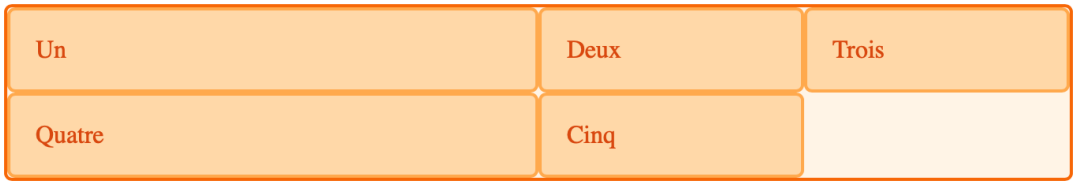
Les pistes peuvent être définies à l'aide de n'importe quelle unité de mesure. Les grilles proposent aussi une nouvelle unité de mesure pour aider à la création de pistes flexibles. Cette unité, `fr`, représente une fraction de l'espace disponible dans le conteneur de la grille. Le code suivant crée trois colonnes égales qui se redimensionnent en fonction de l'espace disponible.

```
.wrapper {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
}
```

4 Tailles différentes

L'exemple suivant crée une grille avec une colonne de `2fr`, et deux colonnes de `1fr`. L'espace disponible est divisé en quatre. Les deux premières fractions sont allouées à la première colonne, et chacune des colonnes suivante dispose d'une fraction.

```
.wrapper {
  display: grid;
  grid-template-columns: 2fr 1fr 1fr;
}
```



5 Mélanger des tailles flexibles et absolues

Dans ce dernier exemple nous utilisons à la fois des dimensions absolues et des relatives pour les pistes. La première piste faisant 500px, cette valeur est soustraite de l'espace disponible. L'espace restant est divisé en trois et alloué proportionnellement aux deux colonnes spécifiées avec l'unité relative `fr`.

```
.wrapper {
  display: grid;
  grid-template-columns: 500px 1fr 2fr;
}
```

Utiliser la notation `repeat()` pour définir les pistes

Pour les grilles comprenant de nombreuses pistes on peut utiliser la notation `repeat()` pour répéter toute ou une partie des pistes définies. Par exemple la définition de grille :

```
.wrapper {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
}
```

peut également s'écrire :

```
.wrapper {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
}
```

Dans l'exemple suivant on crée une grille avec une première colonne de 20px de large, puis une section répétant 6 fois une piste de 1fr, et enfin on termine par une colonne de 20px de large.

```
.wrapper {
  display: grid;
  grid-template-columns: 20px repeat(6, 1fr) 20px;
}
```

Cette notation accepte une liste de pistes, on peut donc l'utiliser pour répéter un motif. Dans l'exemple qui suit la grille aura 10 colonnes : une colonne de 1fr suivie d'une colonne de 2fr, ceci répété 5 fois.

```
.wrapper {
  display: grid;
  grid-template-columns: repeat(5, 1fr 2fr);
}
```

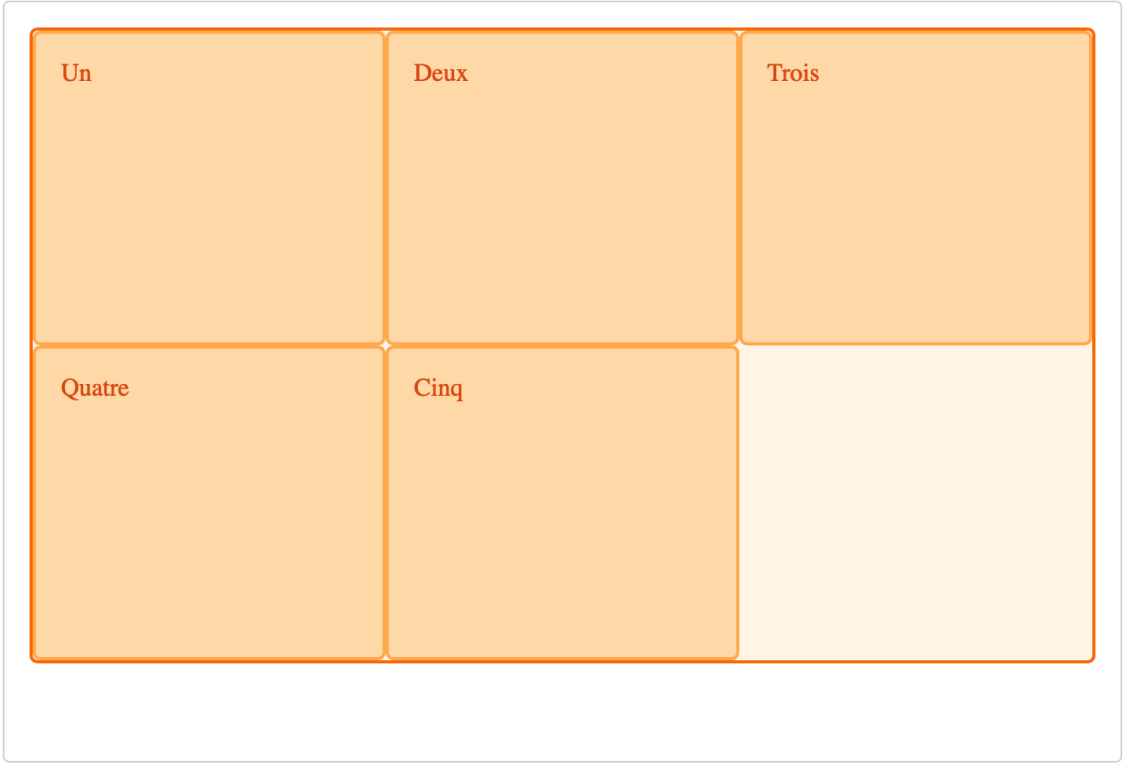
6 Grille implicite et grille explicite

Dans ces exemples nous avons défini nos colonnes à l'aide de la propriété `grid-template-columns`, et nous avons laissé la grille créer les rangées. Ces rangées font partie de la grille implicite. La grille explicite est constituée des pistes définies par les propriétés `grid-template-columns` et `grid-template-rows`. Si un élément est placé en dehors de la grille ainsi définie, ou que la quantité de contenu nécessite d'étendre la grille, alors la grille ajoute implicitement des colonnes et rangées. Les dimensions de ces pistes auront par défaut la valeur auto, c'est-à dire qu'elles s'ajusteront à leur contenu.

On peut définir une taille pour les pistes de la grille implicite grâce aux propriéts `grid-auto-rows` et `grid-auto-columns`.

Dans l'exemple ci-après nous utilisons `grid-auto-rows` pour que les rangées de la grille implicite fassent 200 pixels de haut.

```
.wrapper {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-auto-rows: 200px;
}
```



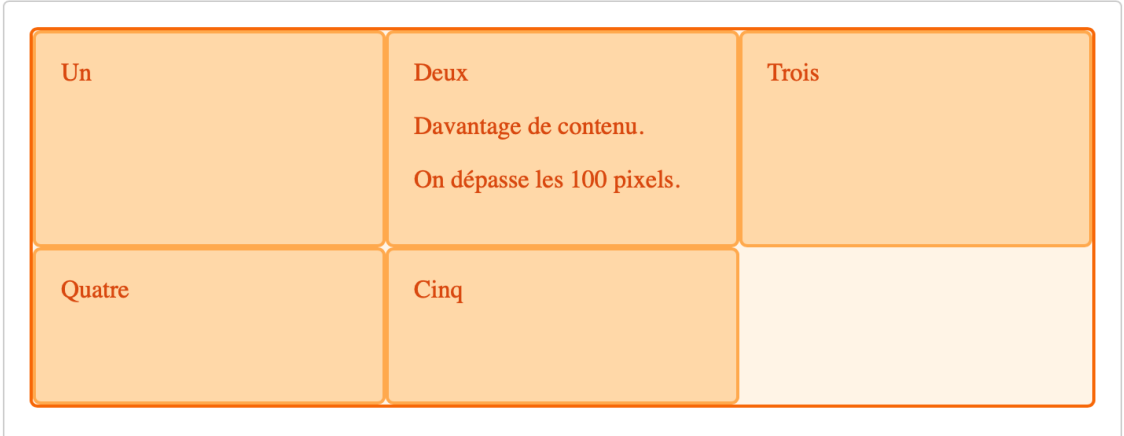
7 Dimensionner une piste avec `minmax`

Que l'on crée une grille explicite, ou que l'on définisse la taille des pistes créées implicitement, il peut être utile d'assigner une taille minimum, qui s'agrandit pour s'adapter au contenu. Par exemple on peut souhaiter que les rangées ne soient jamais moins hautes que 100 pixels, mais

qu'elles aillent jusqu'à 300 pixels de haut si le contenu le nécessite.

La fonction `minmax()` permet ce comportement. Dans l'exemple suivant nous utilisons `minmax()` comme valeur de la propriété `grid-auto-rows`. Les rangées créées automatiquement feront un minimum de 100 pixels, et un maximum de `auto`, ce qui signifie que la taille s'adaptera à la hauteur du contenu.

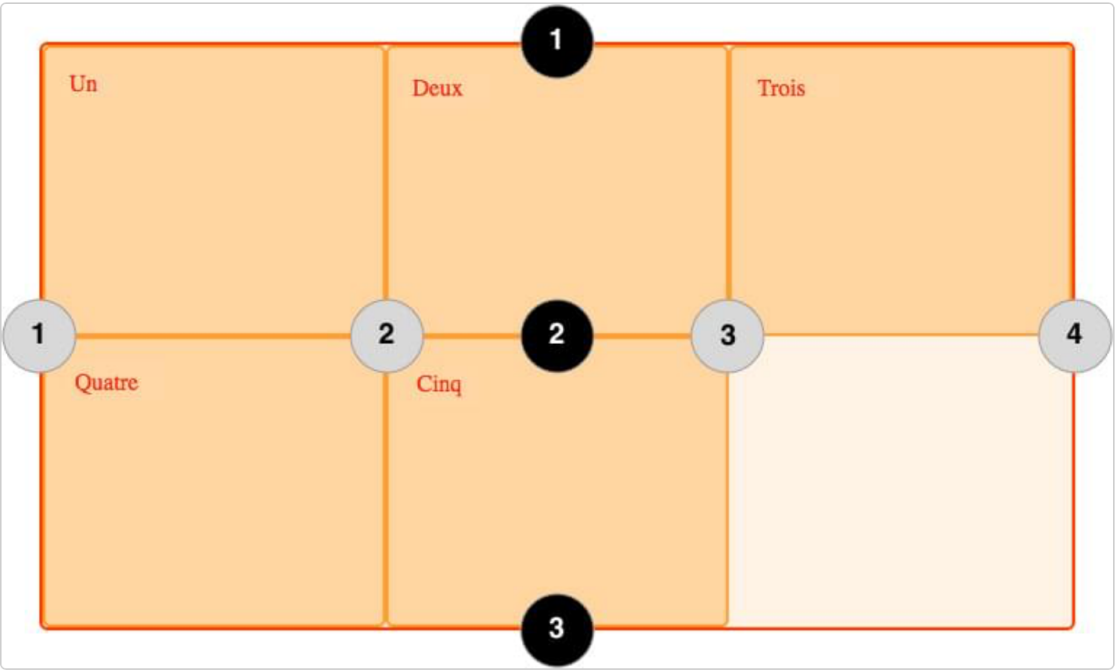
```
.wrapper {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-auto-rows: minmax(100px, auto);

<div class="wrapper">
   <div>Un</div>
   <div>Deux
     <p>Davantage de contenu.</p>
     <p>On dépasse les 100 pixels.</p>
   </div>
   <div>Trois</div>
   <div>Quatre</div>
   <div>Cinq</div>
</div>
```



Lignes de grille

Il faut noter que l'on définit les pistes d'une grille, et pas les lignes qui en résultent. La grille génère des lignes numérotées que l'on utilise pour positionner les éléments. Dans notre grille de

trois colonnes et deux rangées, nous avons quatre lignes de colonnes.



Les lignes sont numérotées selon le sens de lecture du document. Dans un langage qui se lit de gauche à droite, la ligne 1 est située à gauche, dans un langage qui se lit de droite à gauche elle est située à droite. Les lignes peuvent aussi être nommées, comme nous le verrons plus loin dans ces pages.

Positionnement des éléments sur les lignes

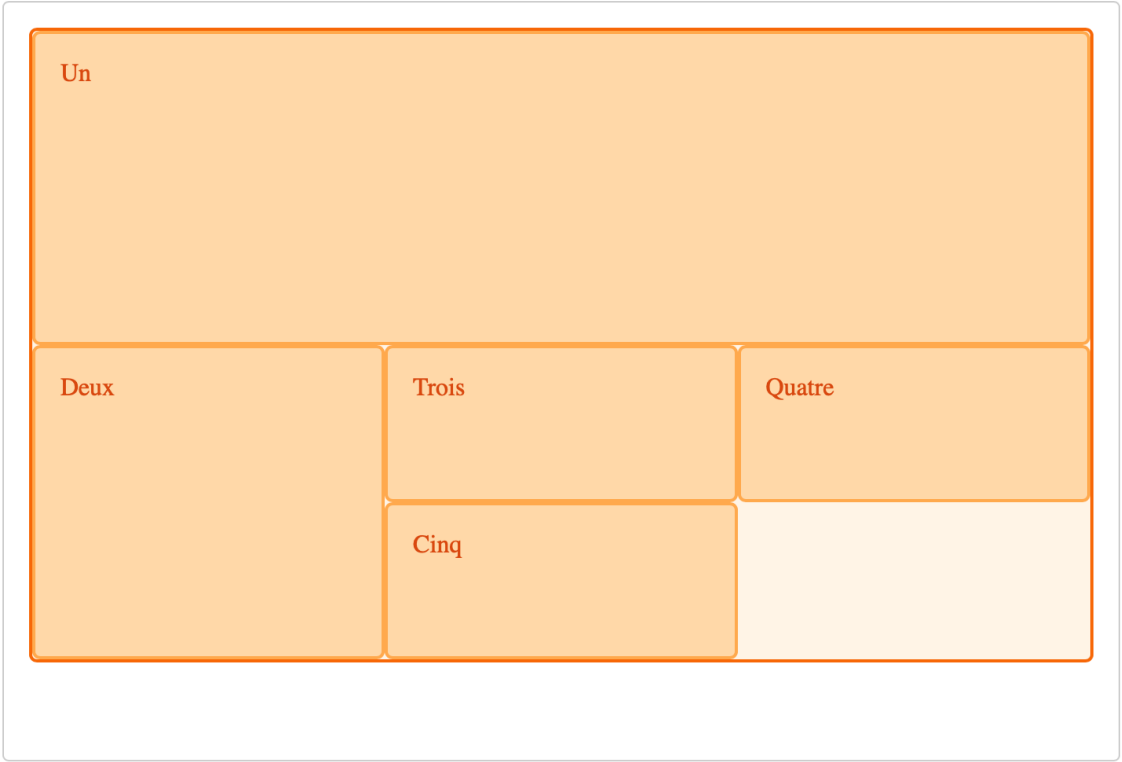
Nous explorerons le placement sur les lignes de manière détaillée dans un prochain article, l'exemple qui suit montre comment l'utiliser de façon simple. Lorsque nous plaçons un élément nous ciblons une ligne plutôt qu'une piste.

Nous plaçons ici les deux premiers éléments en utilisant les propriétés `grid-column-start`, `grid-column-end`, `grid-row-start` et `grid-row-end`. En allant de gauche à droite, le premier élément est placé sur la ligne de colonne 1, et va jusqu'à la ligne de colonne 4, qui dans ce cas est la dernière. Il est placé sur la ligne de rangée 1, et va jusqu'à la ligne 3, s'étendant ainsi sur deux rangées.

Le second élément commence sur la ligne de colonne 1 et s'étend sur une seule piste. C'est la largeur par défaut, donc il n'est pas nécessaire de spécifier la ligne de fin. Il s'étend aussi sur deux rangées de la ligne 3 à la ligne 5. Les autres éléments se placeront dans les espaces vides de la grille.

```
<div class="wrapper">
  <div class="box1">Un</div>
  <div class="box2">Deux</div>
  <div class="box3">Trois</div>
  <div class="box4">Quatre</div>
  <div class="box5">Cinq</div>
</div>
```

```css
.wrapper {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-auto-rows: 100px;
}
.box1 {
  grid-column-start: 1;
  grid-column-end: 4;
  grid-row-start: 1;
  grid-row-end: 3;
}
.box2 {
  grid-column-start: 1;
  grid-row-start: 3;
  grid-row-end: 5;
}
```



Note : Pensez à utiliser l'Inspecteur de grille dans les outils de développement du navigateur pour voir comment les éléments se placent sur les lignes d'une grille.

Raccourcis de positionnement sur ligne

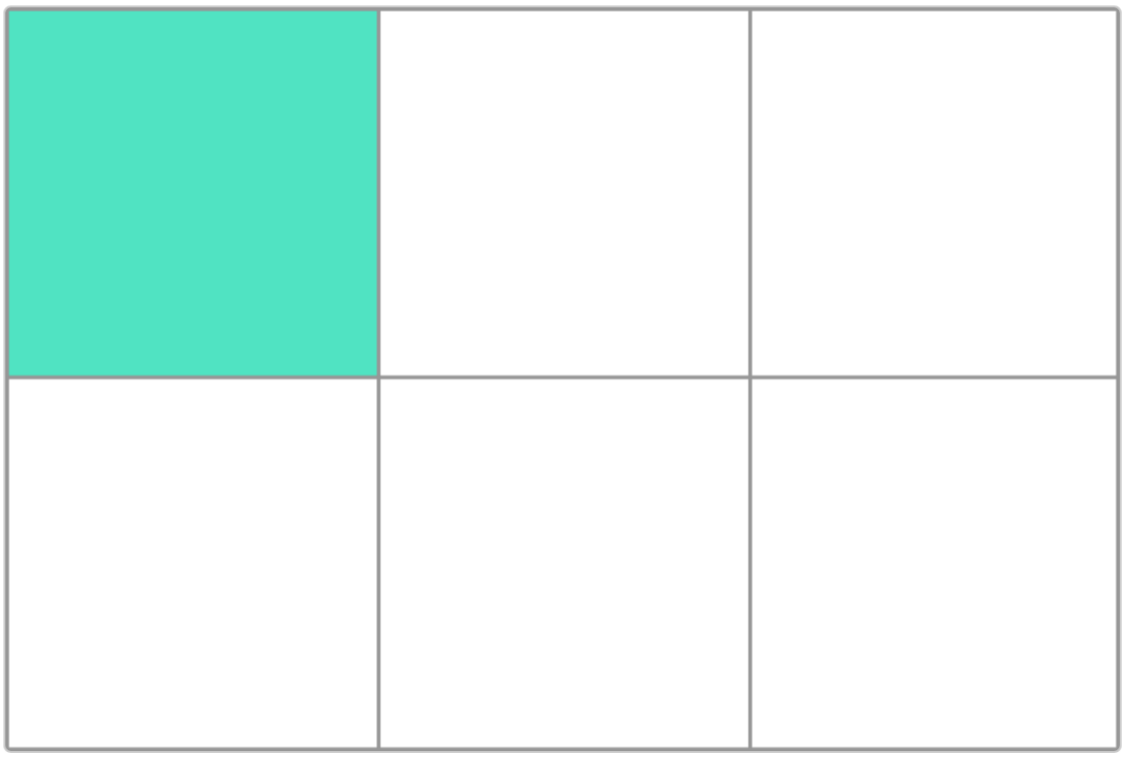
Les valeurs écrites en entier utilisées plus haut peuvent être abrégées en une ligne pour les colonnes avec `grid-column` et une ligne pour les rangées avec `grid-row`. L'exemple suivant donne le même positionnement que le code précédent, mais avec beaucoup moins de CSS. La valeur avant la barre oblique (/) représente la ligne de début, celle d'après représente la ligne de fin.

Vous pouvez omettre la valeur de la ligne de fin si la zone ne recouvre qu'une piste.

```css
.wrapper {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-auto-rows: 100px;
}

.box1 {
  grid-column: 1 / 4;
  grid-row: 1 / 3;
}

.box2 {
  grid-column: 1;
  grid-row: 3 / 5;
}
```

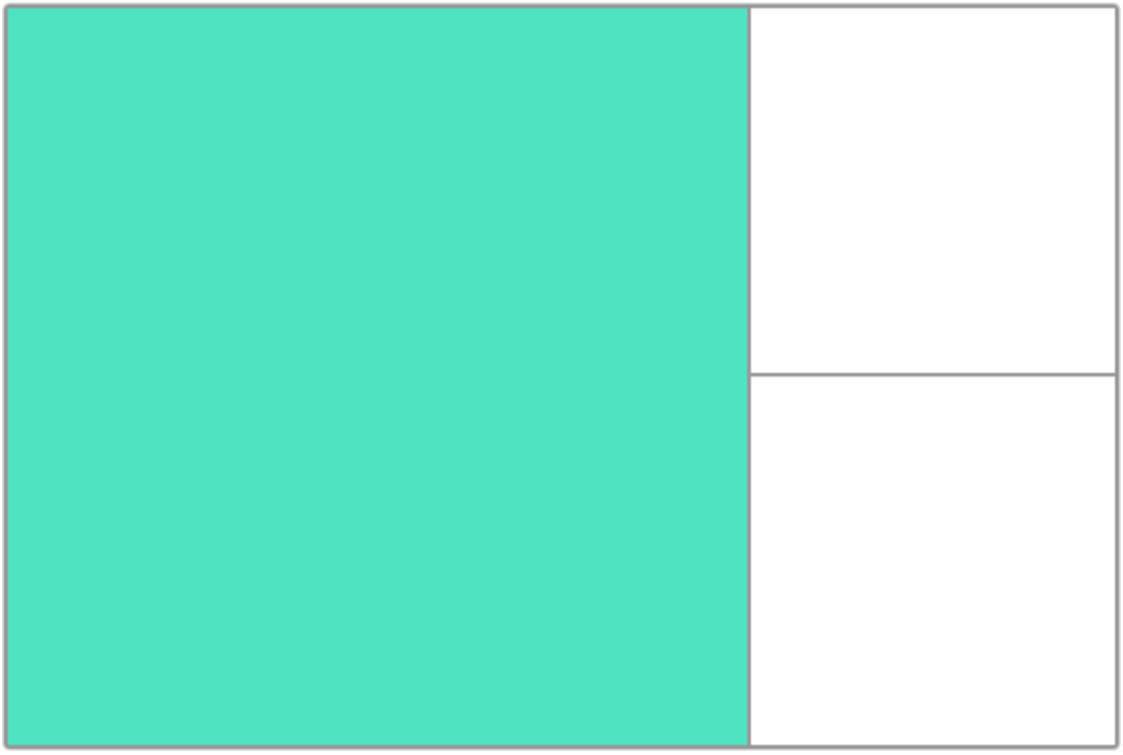
Les cellules

Une *cellule* est la plus petite unité sur une grille, conceptuellement similaire à une cellule de tableau. Comme nous l'avons vu lorsqu'une grille est définie sur un élément ses enfants viennent se positionner chacun dans l'une des cellules de la grille. Dans l'image ci-dessous la première cellule est colorée.



Les zones

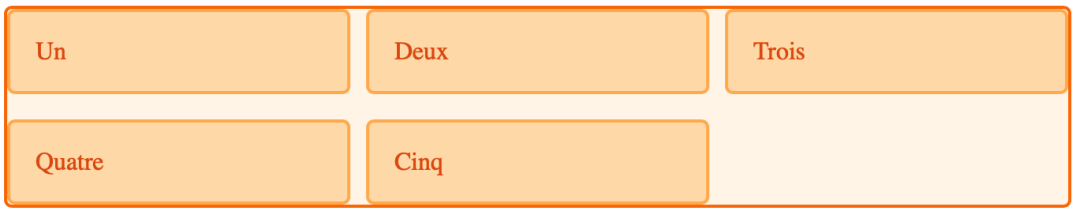
Un élément peut s'étendre sur plusieurs cellules d'une rangée ou d'une colonne, et cela crée une *zone*. Les zones doivent être rectangulaires - on ne peut pas créer de forme en L par exemple. La zone colorée ci-dessous s'étend sur deux rangées et deux colonnes.



Les gouttières

Les *gouttières* entre les cellules sont définies à l'aide des propriétés `grid-column-gap` et `row-gap`, ou de la propriété raccourcie `gap`. Dans l'exemple ci-dessous nous créons une gouttière de dix pixels de large entre les colonnes, et une gouttière de 1em de hauteur entre les rangées.

```
.wrapper {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-column-gap: 10px;
  grid-row-gap: 1em;
}

<div class="wrapper">
  <div>Un</div>
  <div>Deux</div>
  <div>Trois</div>
  <div>Quatre</div>
  <div>Cinq</div>
</div>
```



L'espace utilisé par les gouttières sera pris en compte avant l'assignation de la place restante aux pistes définies avec l'unité `fr`. La taille des gouttières est calculée comme celle des pistes, mais on ne peut pas placer d'élément dans une gouttière. Au niveau du positionnement des éléments sur les lignes, la gouttière se comporte comme une ligne épaisse.