

# CSS GRID la suite

## Différence entre les grilles explicites et implicites

CSS grid utilise le concept de grille *explicite* et *implicite* . Ceci est un concept clé dont vous devez tenir compte lors de la création d'une grille, sinon vous pourriez vous retrouver avec un tas de lignes ou de colonnes inattendues!

Explicit Grid



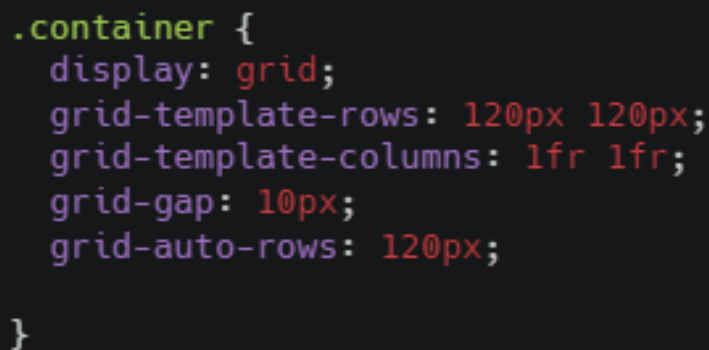
Implicit Grid

*Les grilles explicites et implicites.*

La grille explicite est la grille que vous définissez en utilisant les propriétés `grid-template-rows`, `grid-template-columns` et `grid-template-areas`.

Cependant, vous pouvez toujours avoir des éléments qui ne rentrent pas dans votre grille définie explicitement. Par exemple, supposons que vous définissiez une grille ne pouvant contenir que 6 éléments, alors que le conteneur de la grille en contient en réalité 9 éléments. Seuls 6 éléments entreront dans la grille explicite et 3 seront hors grille. C'est là qu'intervient la grille implicite.

La grille implicite est automatiquement générée par le conteneur de grille chaque fois que des éléments de la grille sont positionnés en dehors de la grille explicite. Le conteneur de grille génère des pistes de grille implicites en ajoutant des lignes de grille implicites à la grille. Ces lignes avec la grille explicite forment la grille implicite.

A dark-themed code editor window with three colored window control buttons (red, yellow, green) in the top-left corner. It contains CSS code for a grid container.

```
.container {  
  display: grid;  
  grid-template-rows: 120px 120px;  
  grid-template-columns: 1fr 1fr;  
  grid-gap: 10px;  
  grid-auto-rows: 120px;  
}
```

---

1	2
3	4
5	6

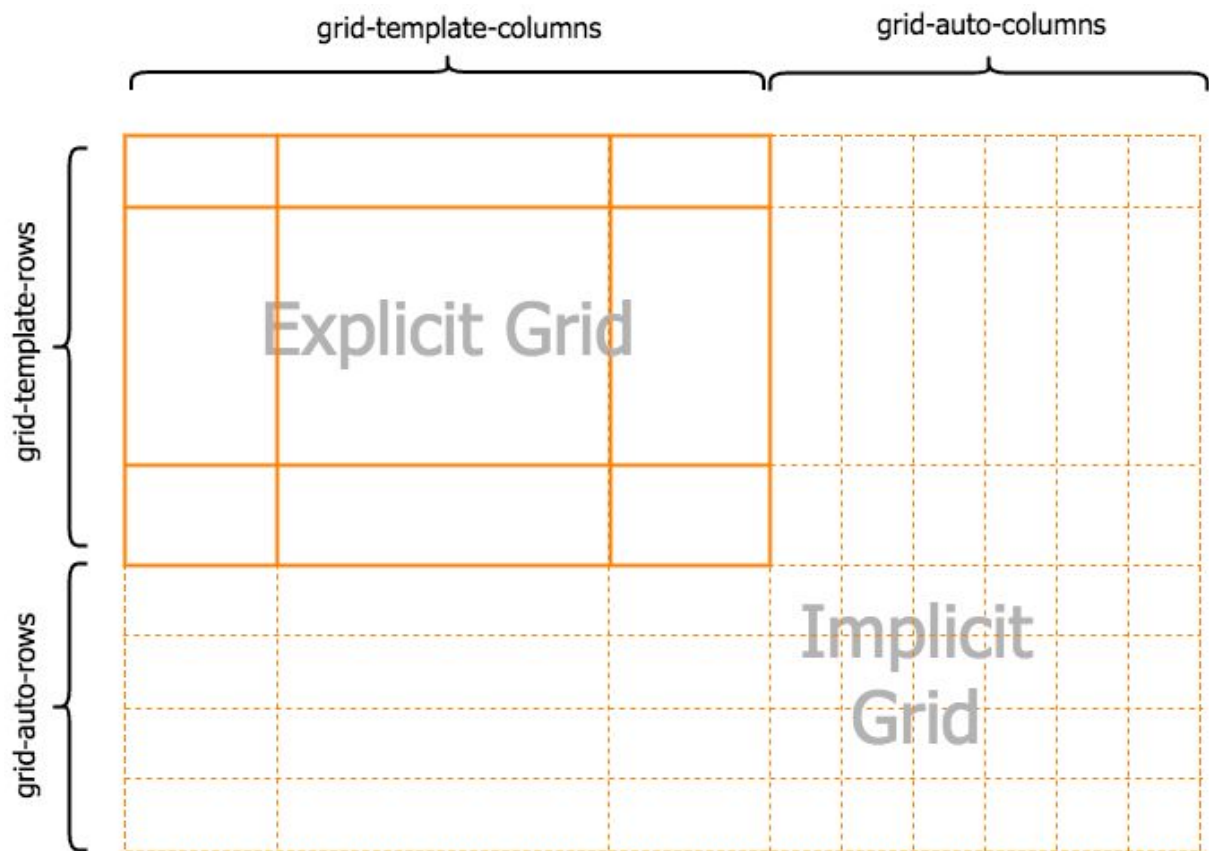
Dans cet exemple, nous définissons explicitement deux lignes et deux colonnes. Cela correspond à quatre éléments de la grille. Toutefois, comme il existe six éléments de grille, une grille implicite a été créée pour accueillir les deux éléments supplémentaires.

C'est une bonne chose. Si la grille implicite n'avait pas été créée, les deux éléments supplémentaires occasionnerait un désordre complet à la grille.

## Définition de la taille de la piste sur des grilles implicites

Vous remarquerez peut-être que la rangée supplémentaire n'est pas aussi haute que ses deux rangées précédentes. En effet, nous définissons la hauteur des lignes à l'aide de la propriété `grid-template-rows`, mais cela ne s'applique qu'aux lignes de la grille explicite. La hauteur de ligne sur la grille implicite doit être définie avec la propriété `grid-auto-rows`. Mais comme nous ne l'avons pas fait, la ligne implicite utilise une taille de piste de `auto` (basée sur le contenu).

Voici comment les propriétés de dimensionnement de piste s'intègrent:



*Propriétés de dimensionnement explicites des pistes de la grille par rapport aux propriétés implicites de dimensionnement des pistes de la grille.*

Alors c'est comme ça:

- Utilisations explicites de la grille `grid-template-rows` et `grid-template-columns`.
- Utilisations implicites de la grille `grid-auto-rows` et `grid-auto-columns`.

Dans l'exemple suivant, nous rendons les lignes explicites et implicites de la même hauteur (120px). Nous ajoutons donc la propriété `grid-auto-rows` pour définir la hauteur de la ligne générée implicitement:



```
.container {  
  display: grid;  
  grid-template-rows: 120px 120px;  
  grid-template-columns: 1fr 1fr;  
  grid-gap: 10px;  
  grid-auto-rows: 120px;  
}
```

---

Flux automatique entre les lignes et les colonnes

Jusqu'à présent, des lignes supplémentaires ont été créées pour accueillir les éléments de grille supplémentaires. Mais que faire si nous voulons des colonnes supplémentaires à la place?

Cela peut être fait en utilisant la propriété `grid-auto-flow`.

Cette propriété vous permet de spécifier s'il faut utiliser des lignes ou des colonnes pour les éléments placés automatiquement. En d'autres termes, vous pouvez spécifier si la grille implicite augmente les lignes ou les colonnes. La valeur par défaut est `row` donc ceci explique pourquoi l'exemple ci-dessus ajoute des lignes au lieu de colonnes. Si vous préférez ajouter des colonnes, vous pouvez le faire:

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

Cela garantira que tous les éléments supplémentaires sont placés dans des colonnes implicites au lieu de lignes.

Voici ce qui se passe lorsque nous appliquons cela au premier exemple:

```
.container {  
  display: grid;  
  grid-template-rows: 80px 80px;  
  grid-template-columns: 1fr 1fr;  
  grid-gap: 10px;  
  grid-auto-flow: column;  
}
```



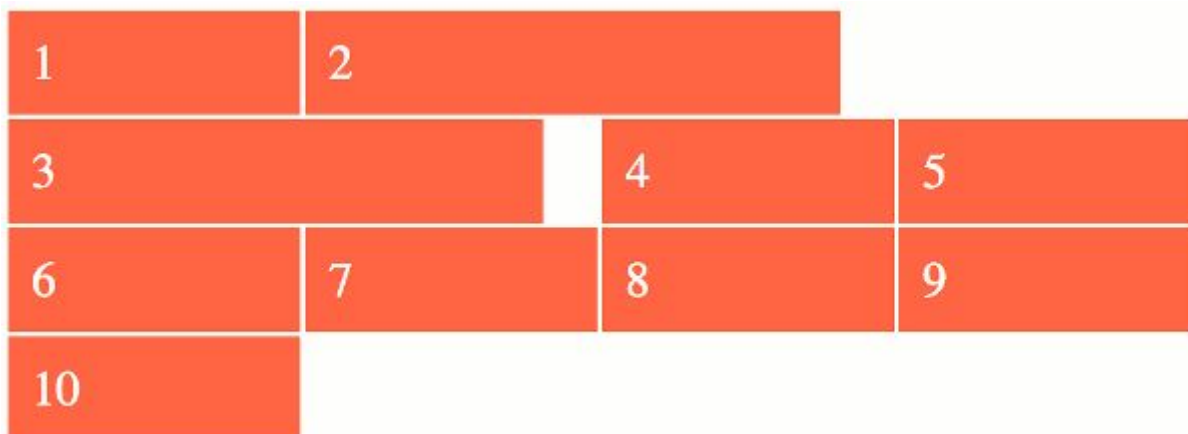
Bien sûr, vous pouvez continuer et utiliser la propriété `grid-auto-columns` pour modifier la largeur de la colonne générée automatiquement. Donc, si vous voulez que toutes les colonnes ci-dessus aient la même largeur, vous les utiliseriez `grid-auto-columns: 1fr`.

*Notez que si vous mettez `grid-auto-flow` à `column` vous changer le positionnement réel des éléments de la grille, en colonne et non plus en ligne.*

## Le mot clé *dense*

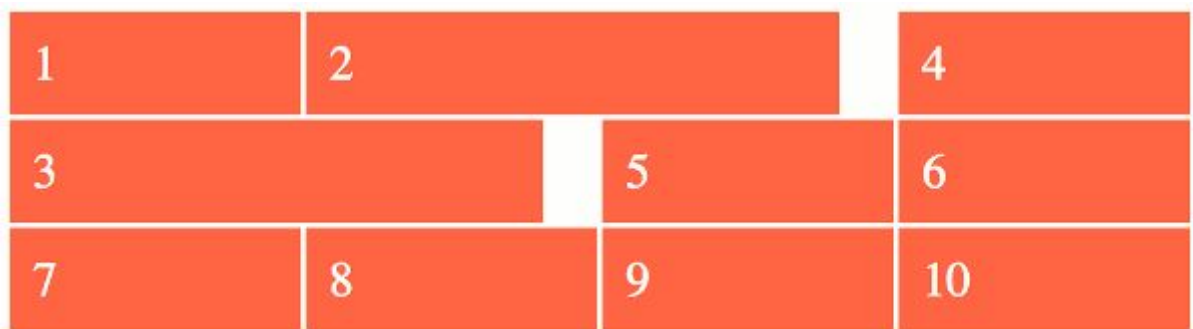
La propriété `grid-auto-flow` comporte également un mot-clé facultatif `dense` qui peut aider à maintenir votre grille compacte et à éviter de nombreux écarts dus à des éléments de grille de taille incohérente.

Par exemple, vous pouvez transformer une grille comme ceci:



*Exemple de grille sans le mot clé dense.*

Dans une grille comme celle-ci:



*Exemple de grille avec le mot clé dense.*

L'utilisation de ce mot clé peut modifier l'ordre d'apparition des éléments non conformes à l'ordre du code (car des éléments placés après peuvent revenir en arrière et combler les espaces apparus au dessus), ce qui ne convient pas à toutes les situations.

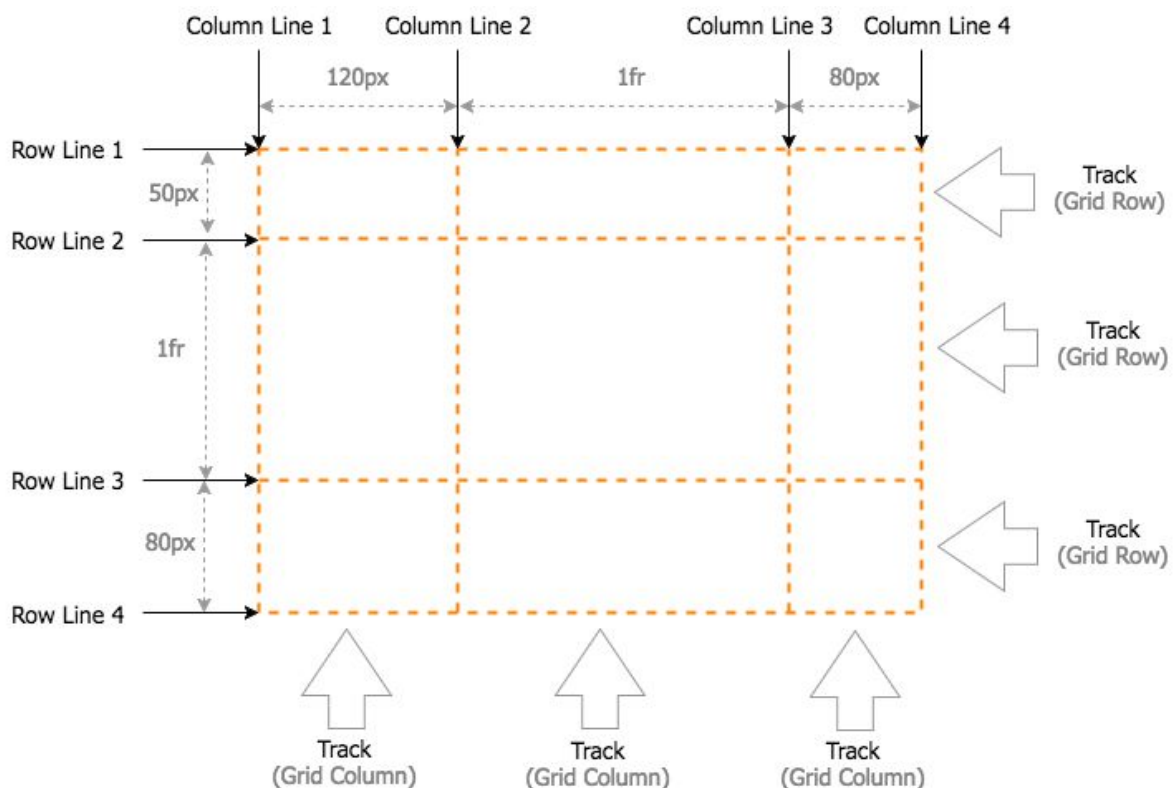
## Placement des éléments grid

Comprendre la manière dont les éléments de la grille sont placés dans la grille est crucial lorsque vous utilisez css grid.

Si vous ne comprenez pas bien comment les éléments de la grille sont placés dans la grille, vous risquez de perdre beaucoup de temps à essayer de faire quelque chose qui ne devrait prendre que quelques secondes.

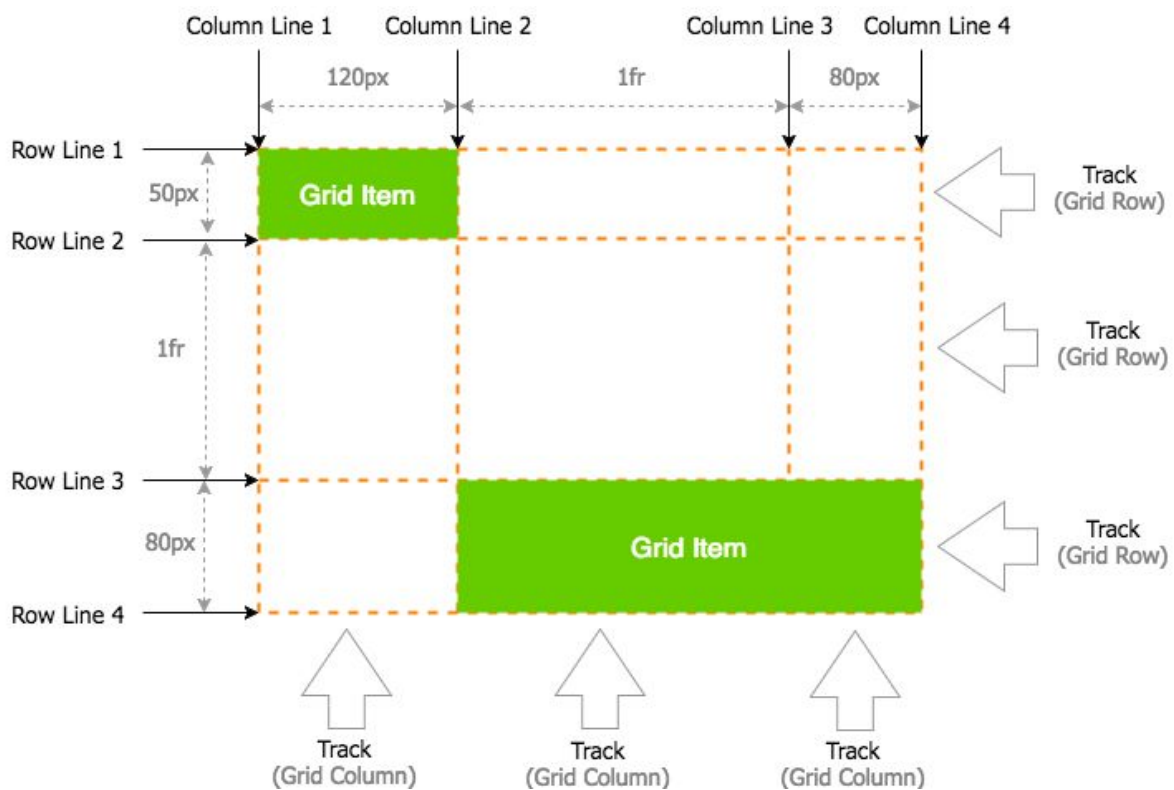
Heureusement, ce n'est pas un concept difficile à saisir. Fondamentalement, tout se résume aux lignes de la grille .





*Les lignes de la grille sont représentées ici par les lignes orange en pointillés. Cet exemple concerne un mode d'écriture de gauche à droite . Pour un mode d'écriture de droite à gauche, tel que l'arabe, les lignes de colonne commencent par le côté droit du conteneur de grille.*

Les lignes de la grille sont les lignes de division horizontale et verticale de la grille. Chaque ligne et colonne comporte une ligne de grille de chaque côté (ces lignes peuvent également être appelées lignes et colonnes ). Chaque ligne de la grille a un index numérique auquel vous pouvez vous référer lorsque vous placez un élément sur la grille.



*Les éléments de la grille sont placés dans une grille.*

Dans l'exemple ci-dessus, le premier élément de la grille commence à la ligne 1 et à la colonne 1 et se termine à la ligne 2 et à la colonne 2. Le deuxième élément de la grille commence aux lignes 3 et colonne 2 et se termine à la ligne 4 et ligne de colonne 4.

Voici ce que cela doit rendre.



2

Cela peut être fait avec le code suivant:

```
.container {
  display: grid;
  grid-template-rows: 50px 1fr 80px;
  grid-template-columns: 120px 1fr 80px;
  grid-gap: 5px;
  height: 90vh;
}

.un {
  grid-row-start: 1;
  grid-column-start: 1;
  grid-row-end: 2;
  grid-column-end: 2;
}

.deux {
  grid-row-start: 3;
  grid-column-start: 2;
  grid-row-end: 4;
  grid-column-end: 4;
}
```

Dans ce cas, nous n'avons pas vraiment besoin de spécifier les valeurs finales sur le premier élément de la grille, car il ne couvre de toute façon qu'une piste. On pourrait en dire autant de la dernière ligne de la deuxième ligne. Par défaut, si vous ne spécifiez pas de ligne de fond, elle ne couvrira qu'une piste.

En fait, nous n'avons pas eu besoin de spécifier de positionnement pour le premier élément de la grille, car il se trouve dans la position précise dans laquelle il aurait été. Mais si nous n'avons pas fait de positionnement sur le deuxième élément, il se serait positionné à côté de l'élément 1 sur la première rangée, couvrant seulement 1 piste.

# Lignes de grille nommées

Vous pouvez également créer votre propre nom pour ces lignes de grille pour faciliter la référence. Vous pouvez les nommer avec les propriétés `grid-template-rows` et `grid-template-columns`. Comme ça:

```
.container {
  display: grid;
  /* Set the tracks and name the lines */
  grid-template-rows: [row1-start] 50px [row2-start] 1fr [row3-start] 80px [row3-end];
  grid-template-columns: [col1-start] 120px [col2-start] 1fr [col3-start] 80px [col3-end];
  grid-gap: 5px;
  height: 90vh;
}

.deux {
  grid-row-start: row3-start;
  grid-column-start: col2-start;
  grid-row-end: row3-end;
  grid-column-end: col3-end;
}
```

Les lignes nommées peuvent être explicites ou implicites. Des lignes nommées implicites sont créées chaque fois que vous créez des zones de grille nommées à l'aide de la propriété `grid-template-areas`. Le nom provient de la zone de la grille, avec `-start` ou `-end` ajouté à la fin (selon qu'il s'agisse de la ligne de début ou de fin).

Ainsi, pour chaque zone de grille nommée “header”, quatre lignes nommées implicites sont créées. Deux nommés `header-start` nomment les début de ligne et de colonne de la zone de grille nommée, et deux nommés `header-end` les fin de ligne et de fin de colonne de la zone de grille nommée.

## Zones de grille nommées

Une zone de grille peut être nommée explicitement à l'aide de la propriété `grid-template-areas` du conteneur de grille.

Pour récapituler, cela ressemble à ceci:



```
.container{
  display: grid;
  /* Name the grid areas */
  grid-template-areas:
    "h h"
    "a b";
  ...
}

...


/* Now apply each grid item to a named grid area */
header {
  grid-area: h;
}

aside {
  grid-area: a;
}

main {
  grid-area: b;
}
```

Vous pouvez spécifier une cellule vide en utilisant un arrêt complet ( `.` ) ou une série d'arrêts complets sans espace entre eux.

Par exemple:



```
grid-template-areas:  
  grid-template-areas: "header header"  
    "... content";  
}
```

---

## Les propriétés de placement de grille

Il existe trois propriétés abrégées qui peuvent être utilisées à la place des propriétés de placement de grille longue expliquées dans les chapitres précédents.

`grid-area`

Cette propriété est un raccourci pour les propriétés suivantes:

`grid-column`

Cette propriété est un raccourci pour les propriétés suivantes:

`grid-column-start`

Spécifie la ligne de colonne sur laquelle commence un élément de la grille et le nombre de pistes qu'il couvre.

`grid-column-end`

Spécifie la ligne de colonne sur laquelle se termine un élément de la grille et le nombre de pistes qu'il recouvre.

`grid-row`

Cette propriété est un raccourci pour les propriétés suivantes:

`grid-row-start`

Spécifie la ligne de départ d'un élément de grille et le nombre de pistes qu'il couvre.

`grid-row-end`

Spécifie la ligne sur laquelle se termine un élément de la grille et le nombre de pistes qu'il couvre.

Vous pouvez également utiliser la propriété `grid-auto-flow` mentionnée lors du chapitre sur la grille explicite ou implicite . Ceci spécifie comment les éléments supplémentaires -implicites -doivent être placés sur la grille. Les éléments placés automatiquement sont ceux qui ne sont pas explicitement placés à l'aide de l'une des propriétés de placement ci-dessus.

## Créer une grille imbriquée

Une grille imbriquée où quand un élément de la grille devient une grille elle-même. Voici un bref aperçu et démonstration.

Les éléments de grille peuvent devenir eux-mêmes des grilles avec la disposition de la grille CSS. Vous pouvez ensuite placer des éléments de grille à l'intérieur de l'élément de grille, créant ainsi une grille imbriquée.

Pour créer une grille imbriquée, tout ce que vous avez à faire est d'appliquer `display:grid` ( ou `display:inline-grid`) à l'élément de grille et celui-ci devient une grille. Vous pourrez bientôt également appliquer `display:subgrid` pour créer une sous-grille mais cette propriété est encore à l'état de test.



```

.container {
  display: grid;
  grid-template-rows: 1fr 1fr;
  grid-template-columns: 1fr 1fr;
  grid-gap: 8px;
}

.sub-container {
  display: grid;
  grid-template-columns: 1fr 1fr;
  grid-gap: 8px;
}

.container div, .sub-container div {
  padding: 8px;
}

```

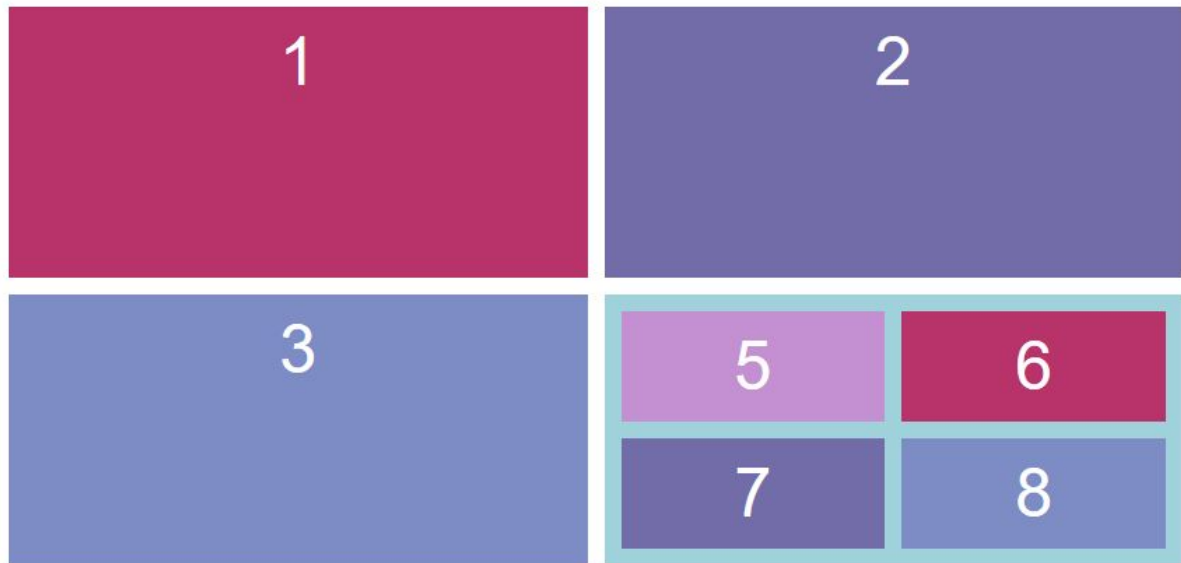
Voici un exemple:

```

<div class="container">
  <div class="un">1</div>
  <div class="deux">2</div>
  <div class="trois">3</div>
  <div class="sub-container quatre">
    <div class="cinq">5</div>
    <div class="six">6</div>
    <div class="sept">7</div>
    <div class="huit">8</div>
  </div>
</div>

```

Cela devrait ressembler à ceci dans le navigateur:



*Exemple de grille imbriquée.*

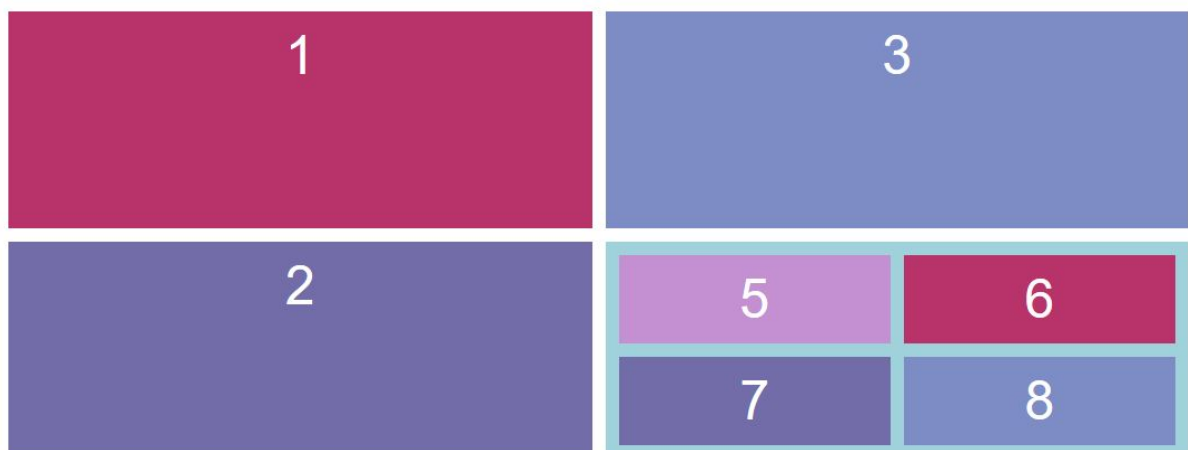
## Héritage

La plupart des propriétés de la grille ne sont pas héritées, ce qui signifie que votre grille imbriquée n'héritera pas des valeurs de sa grille parente. Cela vous permet d'apporter des modifications à la grille parente sans affecter par inadvertance la grille imbriquée.

Par exemple, supposons que vous avez `grid-auto-flow:column` sur la grille parente mais que vous n'avez pas défini cette propriété sur la grille imbriquée. Dans ce cas, la grille imbriquée sera définie sur `row` car c'est la valeur initiale de cette propriété.

Comme ça:

```
.container {  
  display: grid;  
  grid-template-rows: 1fr 1fr;  
  grid-template-columns: 1fr 1fr;  
  grid-gap: 8px;  
  grid-auto-flow: column;  
}  
  
.sub-container {  
  display: grid;  
  grid-template-columns: 1fr 1fr;  
  grid-gap: 8px;  
}
```



Notez que sur la grille parente, les nombres sont maintenant affichés verticalement dans les colonnes et non horizontalement sur les lignes, mais la grille imbriquée continue à être horizontale dans les lignes.

Cependant, de nombreuses autres propriétés CSS sont héritées, vous n'avez donc pas à réinitialiser toutes les propriétés.

## Alignement de grille CSS

Voici un aperçu des diverses propriétés d'alignement que vous pouvez appliquer aux conteneurs et aux éléments de la grille.

En règle générale, la plupart des propriétés d'alignement fonctionnent de la même manière sur les éléments de la grille que sur les autres éléments. Cependant, certaines propriétés d'alignement s'appliquent spécifiquement à la grille et à la flexbox. Voici un aperçu.

---

## La propriété `align-items`

La propriété `align-items` spécifie la valeur `align-self` par défaut pour tous les éléments de la grille participant au contexte de mise en forme du conteneur grid.



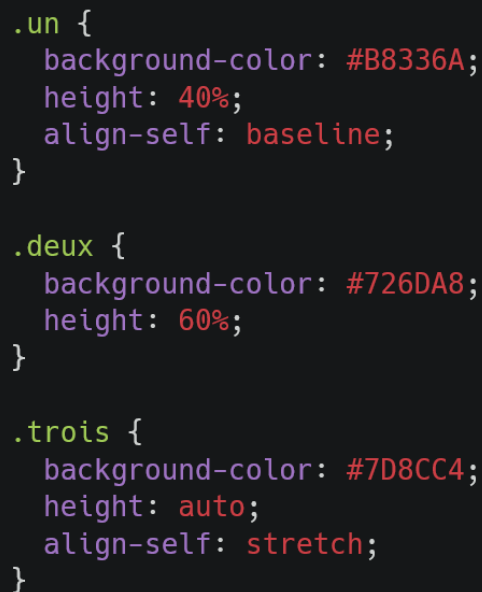
```
.container {  
  display: grid;  
  grid-template-columns: 1fr 1fr 1fr;  
  grid-template-rows: 100vh;  
  grid-gap: 10px;  
  align-items: center;  
}  
  
.un {  
  background-color: #B8336A;  
  height: 40%;  
}  
  
.deux {  
  background-color: #726DA8;  
  height: 60%;  
}  
  
.trois {  
  background-color: #7D8CC4;  
  height: auto;  
}
```



Dans cet exemple, nous appliquons `align-items:center` au conteneur de grille, par conséquent, tous les éléments de la grille sont alignés au centre de l'axe du bloc. Mais comme il s'agit d'un paramètre par défaut, l'un des éléments de la grille peut remplacer ce paramètre par la propriété `align-self`.

## La propriété `align-self`

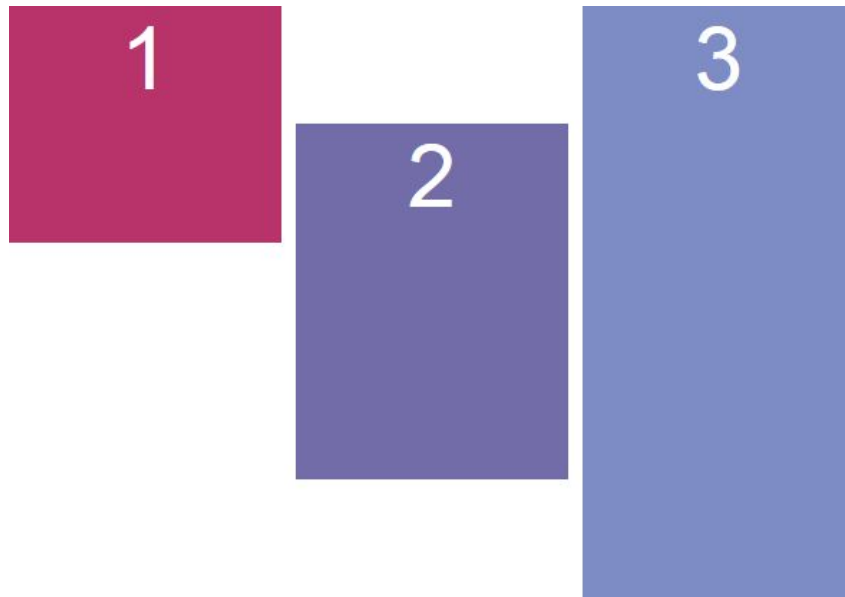
La propriété `align-self` aligne une case dans son bloc conteneur le long de l'axe bloc / colonne / cross-axis.



```
.un {
  background-color: #B8336A;
  height: 40%;
  align-self: baseline;
}

.deux {
  background-color: #726DA8;
  height: 60%;
}

.trois {
  background-color: #7D8CC4;
  height: auto;
  align-self: stretch;
}
```



Ici, l'élément de grille "1" a une valeur de `baseline` et l'élément "3" a une valeur de `stretch`. L'élément bleu a la hauteur auto, il s'étend donc sur toute la hauteur de sa grille.

Cependant, nous n'avons pas défini de valeur pour l'élément "2". Par conséquent, il utilise la valeur par défaut, qui dans ce cas est définie par le `align-items:center` du conteneur de la grille.

## La propriété `justify-items`

La propriété `justify-items` spécifie la valeur `justify-self` par défaut pour tous les éléments de la grille participant au contexte de mise en forme du conteneur de la grille.

```
.container {
  display: grid;
  grid-template-columns: 1fr;
  grid-template-rows: 1fr 1fr 1fr;
  grid-gap: 10px;
  justify-items: center;
}

.container > div {
  padding: 20px;
  width: 20%;
}

.un {
  background-color: #B8336A;
}

.deux {
  background-color: #726DA8;
}

.trois {
  background-color: #7D8CC4;
}
```

1

2

3



## La propriété `justify-self`

La `justify-self` propriété peut être utilisée pour aligner un élément de grille individuel le long de l'axe en ligne / row / principal.



```
.un {  
  background-color: #B8336A;  
  justify-self: end;  
}  
  
.deux {  
  background-color: #726DA8;  
}  
  
.trois {  
  background-color: #7D8CC4;  
  justify-self: start;  
}
```

1

2

3

## La propriété `justify-content`

La propriété `justify-content` aligne le contenu du conteneur de la grille dans son ensemble le long de l'axe principal / row.

Cela peut être utilisé pour aligner toute la grille dans le conteneur de grille, dans le cas où les pistes de la grille occupent moins d'espace que leur conteneur de grille. Cela peut arriver si vous définissez la taille de la piste avec une unité absolue (telle que les pixels), alors que le conteneur de grille prend plus de place que toutes les pistes combinées.

```
.container {
  display: grid;
  grid-template-columns: 100px;
  grid-template-rows: 1fr 1fr 1fr;
  grid-gap: 10px;
  justify-content: center;
}

.container > div {
  padding: 30px;
}

.un {
  background-color: #B8336A;
  width: 20px;
}

.deux {
  background-color: #726DA8;
}

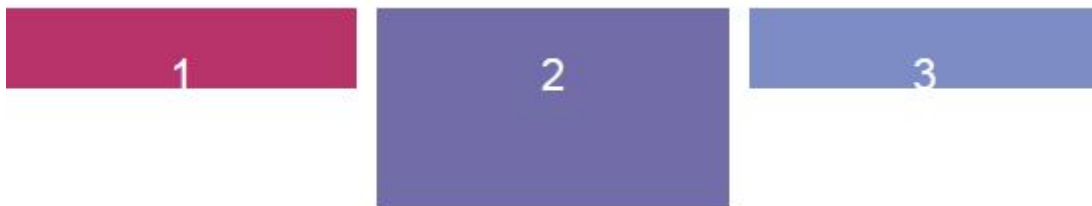
.trois {
  background-color: #7D8CC4;
  width: 40px;
}
```



## La propriété `align-content`

La propriété `align-content` aligne les éléments le long de l'axe `cross-axis` / bloc.

```
.container {  
  display: grid;  
  grid-template-columns: 1fr 1fr 1fr;  
  grid-template-rows: 100px;  
  grid-gap: 10px;  
  align-content: center;  
  height: 100vh;  
}  
  
.container > div {  
  padding: 20px;  
  font-size: 4vw;  
}  
  
.un {  
  background-color: #B8336A;  
  height: 20px;  
}  
  
.deux {  
  background-color: #726DA8;  
}  
  
.trois {  
  background-color: #7D8CC4;  
  height: 40px;  
}
```



## Propriétés abrégées

Au moment de la rédaction de ce document, la prise en charge de ces propriétés abrégées par le navigateur est limitée. Si cela ne fonctionne pas correctement, essayez Firefox.

## La propriété `place-content`

La propriété `place-content` est un raccourci pour `justify-content` et `align-content`.

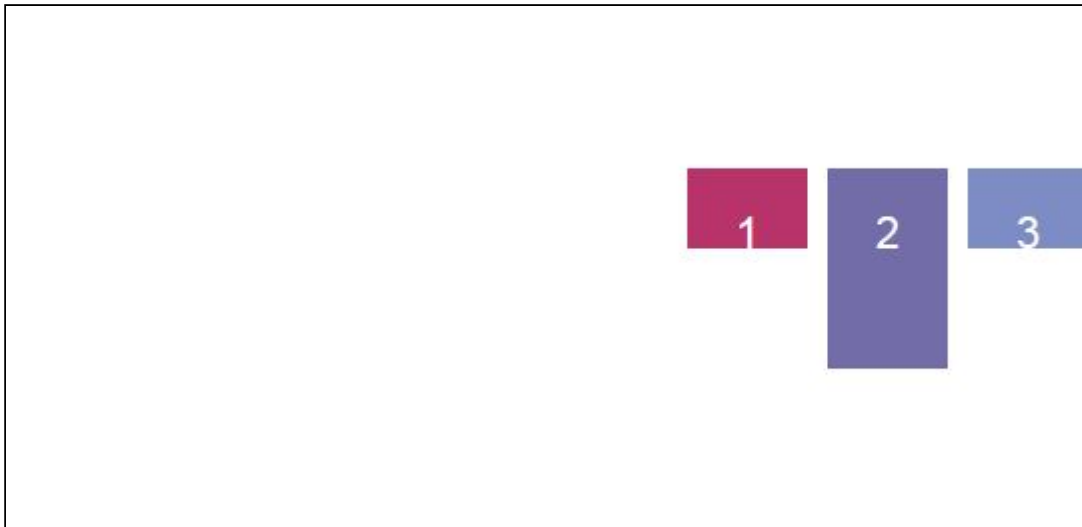
```
.container {
  display: grid;
  grid-template-columns: 60px 60px 60px;
  grid-template-rows: 100px;
  grid-gap: 10px;
  place-content: center end;
  height: 100vh;
}

.container > div {
  padding: 20px;
  font-size: 4vw;
}

.un {
  background-color: #B8336A;
  height: 20px;
}

.deux {
  background-color: #726DA8;
}

.trois {
  background-color: #7D8CC4;
  height: 40px;
}
```



## La propriété `place-items`

La propriété `place-items` est un raccourci pour `justify-items` et `align-items`.

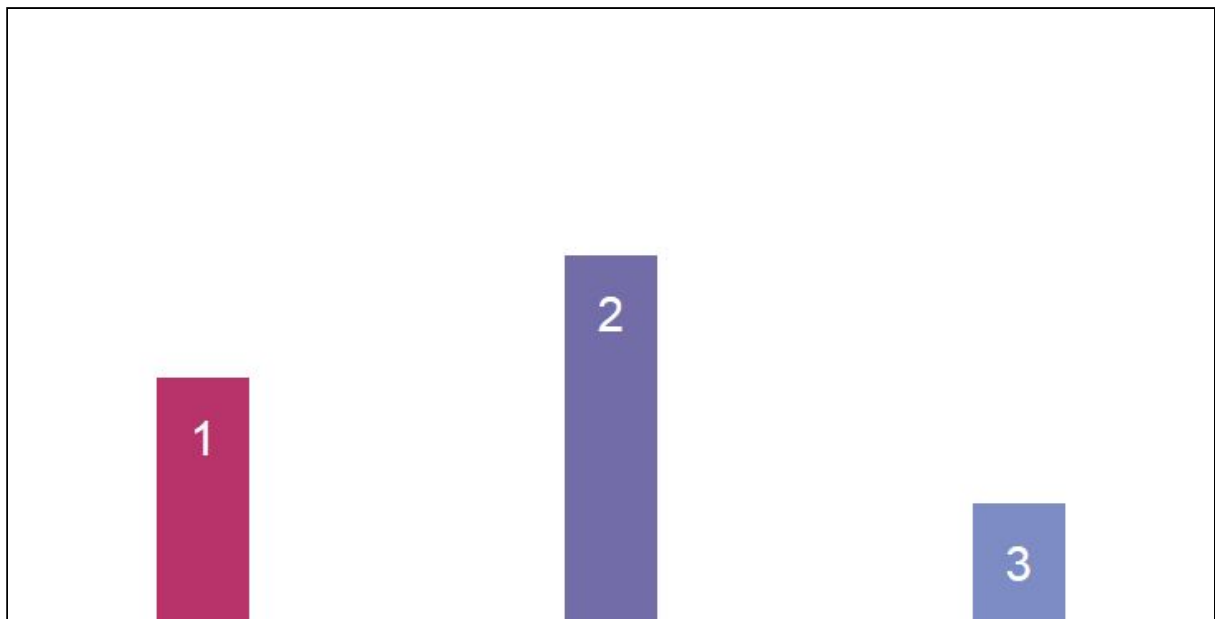
```
.container {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
  grid-template-rows: 100vh;
  grid-gap: 10px;
  place-items: end center;
}

.container > div {
  padding: 20px;
  font-size: 4vw;
}

.un {
  background-color: #B8336A;
  height: 40%;
}

.deux {
  background-color: #726DA8;
  height: 60%;
}

.trois {
  background-color: #7D8CC4;
  height: auto;
}
```





## La propriété `place-self`

La propriété `place-self` est un raccourci pour `justify-self` et `align-self`.

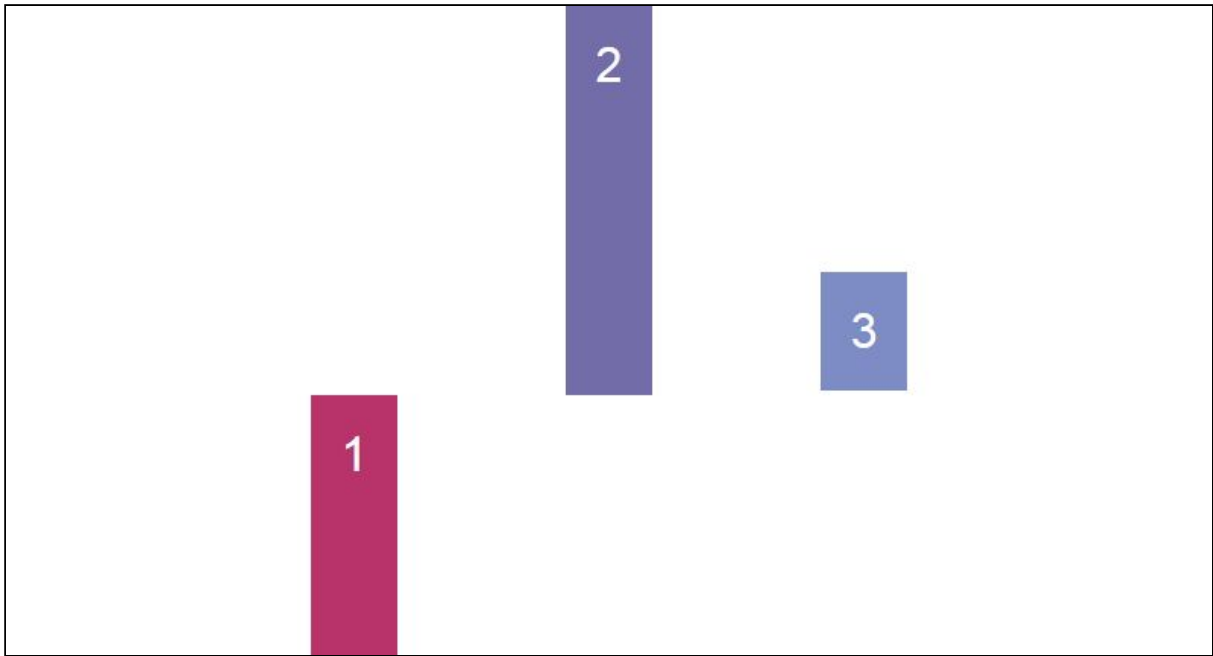
```
.container {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
  grid-template-rows: 100vh;
  grid-gap: 10px;
}

.container > div {
  padding: 20px;
  font-size: 4vw;
}

.un {
  background-color: #B8336A;
  height: 40%;
  place-self: end;
}

.deux {
  background-color: #726DA8;
  height: 60%;
  place-self: start center;
}

.trois {
  background-color: #7D8CC4;
  height: auto;
  place-self: center start;
}
```



## Superposition d'éléments de grille

Les éléments de la grille peuvent être superposés, de sorte que les zones de la grille qui se croisent se chevauchent.

Lors de l'utilisation de CSS Grid Layout, il est possible de superposer des éléments de la grille en raison de l'intersection de zones de la grille, de marges négatives ou d'autres techniques de positionnement. Lorsque cela se produit, les éléments sont positionnés en profondeur en fonction de l'ordre où les éléments apparaissent dans le HTML. On peut modifier cet ordre par la propriété `z-index`.

## Ordre par défaut des éléments

Si vous ne réorganisez pas ou n'appliquez pas spécifiquement des `z-index` aux éléments de la grille, les éléments qui se chevauchent

seront disposés en fonction de l'ordre des éléments. C'est-à-dire l'ordre dans lequel ils apparaissent dans le balisage source.

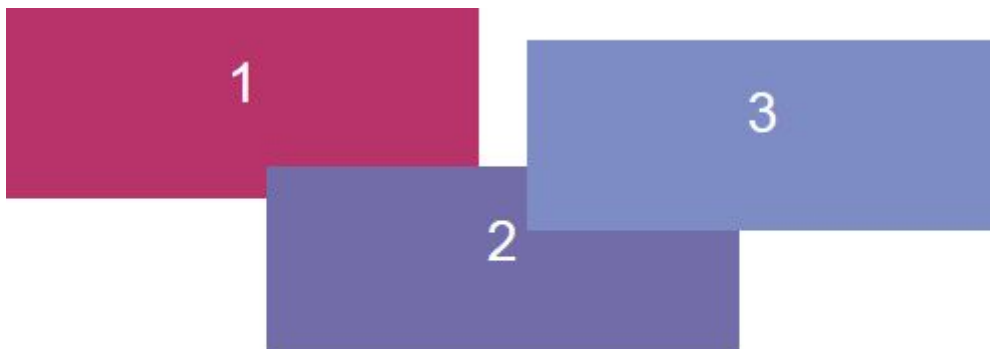
Exemple:

```
.container {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
  grid-template-rows: 1fr 1fr 1fr;
  height: 90vh;
}

.un {
  background-color: #B8336A;
}

.deux {
  background-color: #726DA8;
  position: relative;
  left: -15vw;
  top: 25vh;
}

.trois {
  background-color: #7D8CC4;
  position: relative;
  left: -30vw;
  top: 5vh;
}
```



# Modifié l'ordre des éléments

Ce sera l'ordre du document source html si vous n'avez rien spécifiquement réorganisé.

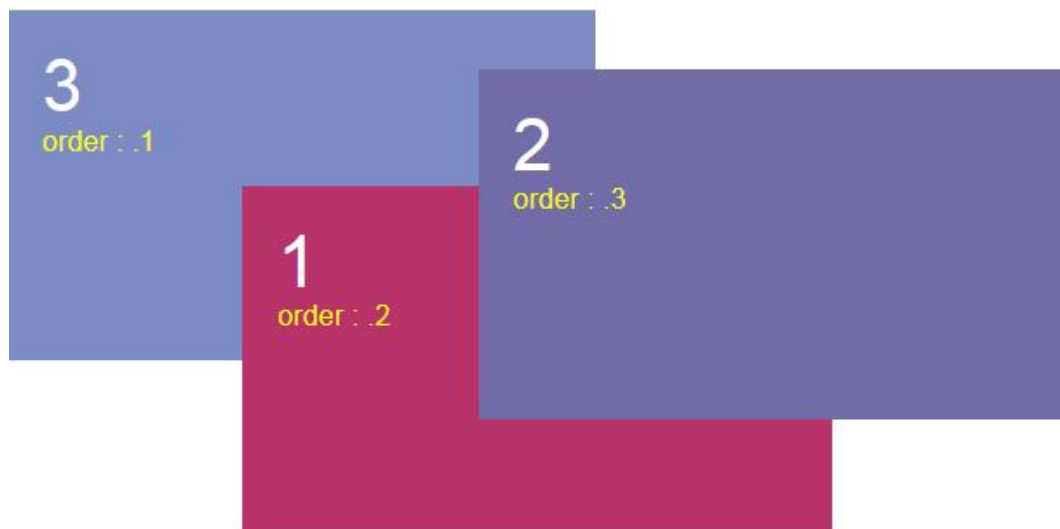
Toutefois, vous pouvez utiliser la propriété `order` pour réorganiser les éléments si nécessaire.

Lorsque vous utilisez cette technique, les éléments de la grille sont incrémentés par couches du numéro d'ordre le plus petit au plus grand. Les éléments avec le même groupe ordinal sont présentés dans l'ordre des éléments (c'est-à-dire leur ordre dans le code html source).

Voici un exemple:



```
.container {  
  display: grid;  
  grid-template-columns: 1fr 1fr 1fr;  
  grid-template-rows: 1fr 1fr 1fr;  
  height: 90vh;  
}  
  
.un {  
  background-color: #B8336A;  
  position: relative;  
  left: -20vw;  
  top: 20vh;  
  order: 2;  
}  
  
.deux {  
  background-color: #726DA8;  
  position: relative;  
  left: -40vw;  
  top: 10vh;  
  order: 3;  
}  
  
.trois {  
  background-color: #7D8CC4;  
  position: relative;  
  top: 5vh;  
  order: 1;  
}
```



## Utiliser la propriété `z-index`

Vous pouvez également utiliser la propriété `z-index` pour superposer les éléments. Cette propriété vous permet de contrôler l'ordre d'un élément le long de l'axe z *-profondeur-*.

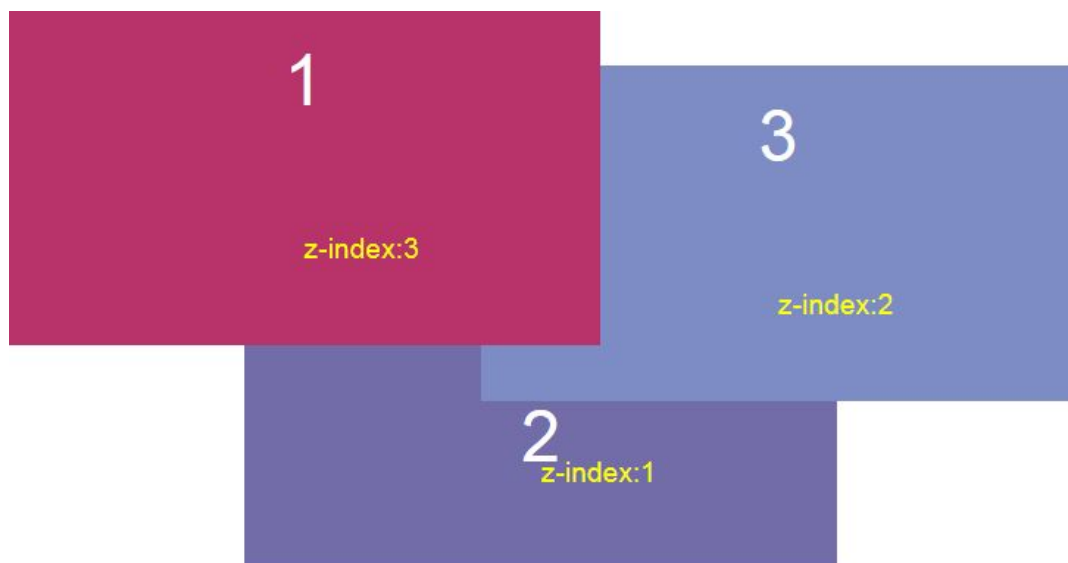
Voici un exemple:

```
.container {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
  grid-template-rows: 1fr 1fr 1fr;
  height: 90vh;
}

.un {
  background-color: #B8336A;
  position: relative;
  z-index: 3;
}

.deux {
  background-color: #726DA8;
  position: relative;
  left: -20vw;
  top: 20vh;
  z-index: 1;
  padding-top: 100px;
}

.trois {
  background-color: #7D8CC4;
  position: relative;
  left: -40vw;
  top: 5vh;
  z-index: 2;
}
```



## Combiner les propriétés `z-index` et `order`

Si les éléments de la grille ont été à la fois réorganisés et qu'un `z-index` `z` a été appliqué, les valeurs du `z-index` déterminent le contexte final de l'empilement.

Cela vous permet de réorganiser les éléments sans perdre le contrôle de la superposition.

Voici un exemple:

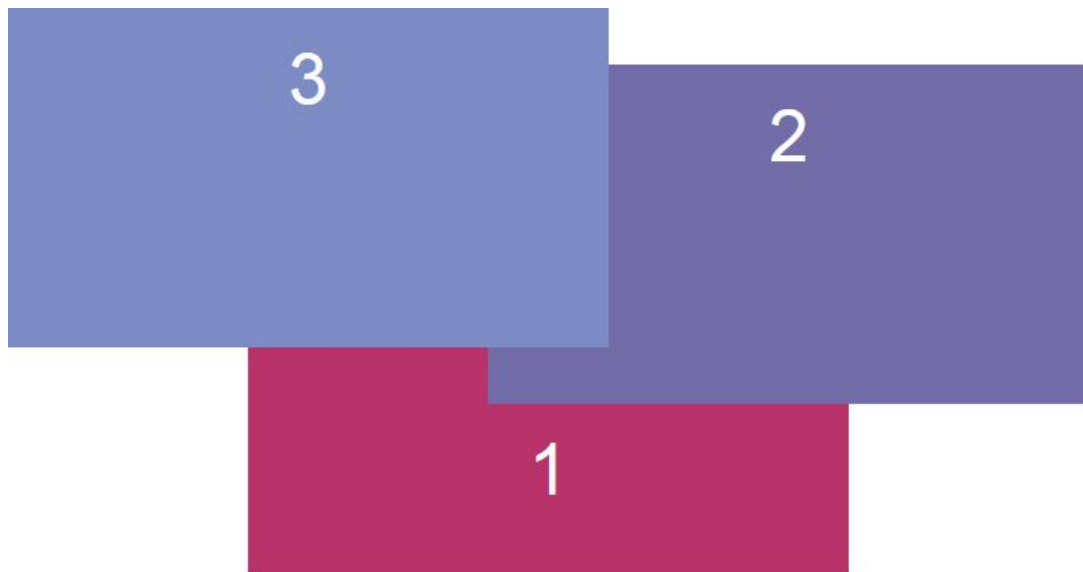
```
.container {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
  grid-template-rows: 1fr 1fr 1fr;
  height: 90vh;
}

.un {
  background-color: #B8336A;
  position: relative;
  left: -20vw;
  top: 20vh;
  order: 2;
  z-index: 1;
}

.deux {
  background-color: #726DA8;
  position: relative;
  left: -40vw;
  top: 5vh;
  order: 3;
  z-index: 2;
}

.trois {
  background-color: #7D8CC4;
  top: 5vh;
  order: 1;
  z-index: 3;
}
```





## Positionnement absolu avec grille

Des exemples montrant comment un positionnement absolu peut affecter le flux normal dans une grille.

Vous pouvez utiliser le positionnement absolu dans une grille, tout comme dans un bloc classique. Si vous définissez `position: relative` sur un élément de la grille, puis utilisez `position: absolute` sur l'un de ses enfants, tout positionnement que vous spécifiez s'appliquera à l'intérieur de cet élément de la grille.

Voici un exemple d'utilisation du positionnement absolu pour placer une icône dans le coin inférieur droit d'un élément de la grille:

```
body {
  display: grid;
  grid-template-areas:
    "header header header"
    "nav article ads"
    "nav footer footer";
  grid-template-rows: 50px 1fr 50px;
  grid-template-columns: 20% 1fr 15%;
  grid-gap: 10px;
  height: 100vh;
  margin: 0;
}

#pageHeader,
#pageFooter,
#mainArticle,
#mainNav,
#siteAds {
  padding: 1em;
  background: gold;
}

#pageHeader {
  grid-area: header;
}

#pageFooter {
  grid-area: footer;
}

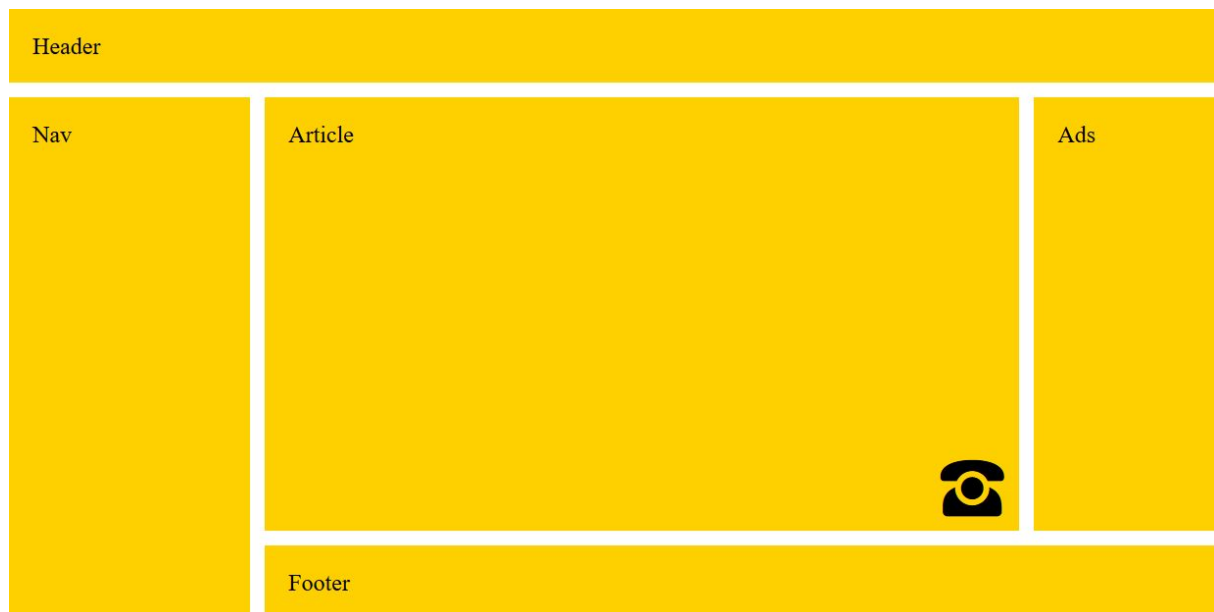
#mainArticle {
  grid-area: article;
  position: relative;
}

#mainNav {
  grid-area: nav;
}

#siteAds {
  grid-area: ads;
}

#mainArticle>.phone {
  position: absolute;
  bottom: 0;
  right: 1vw;
  font-size: 3em;
}
```

```
<body>
  <header id="pageHeader">Header</header>
  <article id="mainArticle">
    Article
    <div class="phone">&phone;</div>
  </article>
  <nav id="mainNav">Nav</nav>
  <div id="siteAds">Ads</div>
  <footer id="pageFooter">Footer</footer>
</body>
```



Toutefois, il convient de garder à l'esprit certains éléments lors du positionnement d'éléments dans une grille.

## Les éléments positionnés ne participent pas à la présentation de la grille

Les éléments positionnés de manière absolue ne participent pas à la disposition de la grille et ne prennent pas de place. Par conséquent, ils n'affectent pas le placement d'autres éléments de la grille.

Voici deux exemples pour démontrer cela.

## Sans positionnement absolu

Dans cet exemple, j'ai explicitement défini la case 4 dans la deuxième colonne de la première ligne et toutes les autres cases sont définies sur auto. La case 4 repousse efficacement la case 2 de son emplacement - les cases 2 et 3 sont automatiquement placées dans les zones de grille disponibles suivantes. Par conséquent, la case 4 affectait l'emplacement des éléments de la grille.

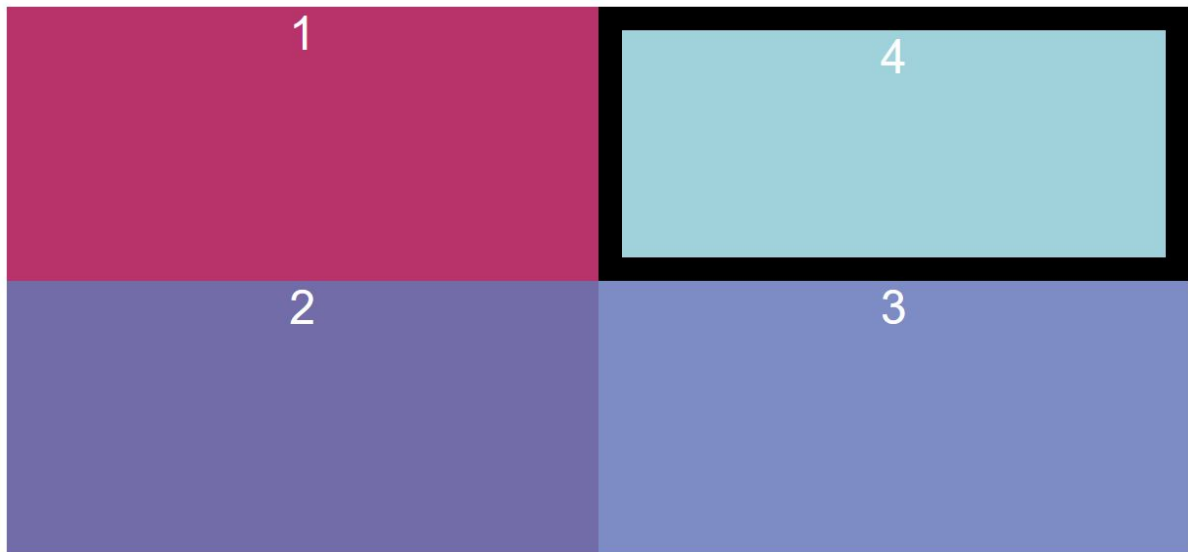
```
/* Grid */
.container {
  display: grid;
  position: relative;
  grid-template-columns: 1fr 1fr;
  grid-template-rows: 1fr 1fr;
  height: 90vh;
}

.un {
  background-color: #B8336A;
  grid-column: auto;
  grid-row: auto;
}

.deux {
  background-color: #726DA8;
  grid-column: auto;
  grid-row: auto;
}

.trois {
  background-color: #7D8CC4;
  grid-column: auto;
  grid-row: auto;
}

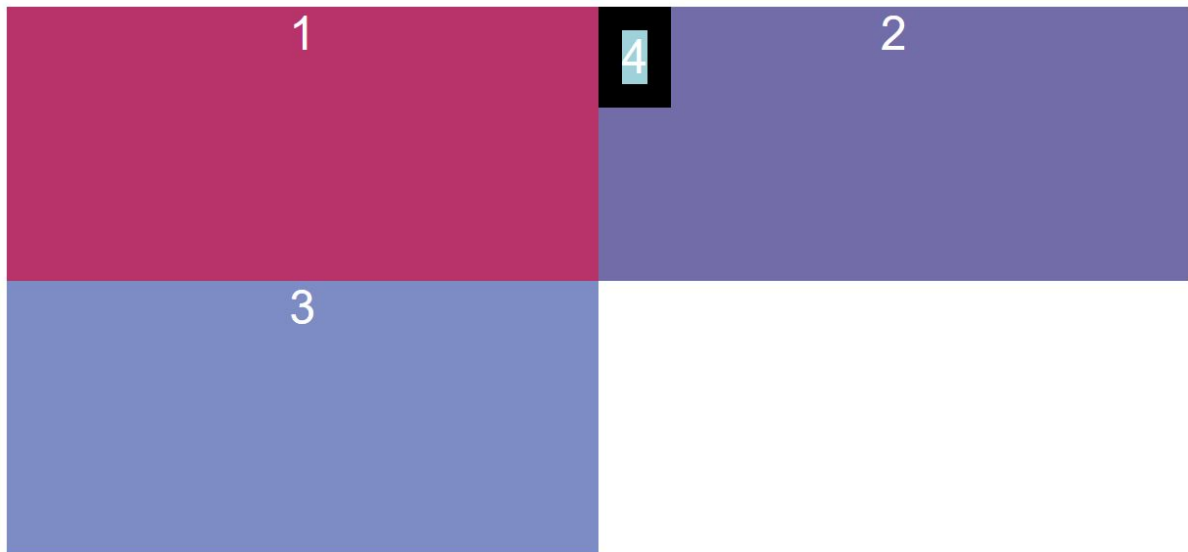
.quatre {
  background-color: #A0D2DB;
  border: 16px solid black;
  grid-column: 2 / 3;
  grid-row: 1 / 2;
}
```



## Avec positionnement absolu

En réglant la case 4 sur `position: absolute`, elle se trouve toujours dans la position de la case 2, mais la case 2 reste inchangée dans cette position. La case 3 n'est pas affectée non plus.

```
.quatre {  
  background-color: #A0D2DB;  
  border: 16px solid black;  
  grid-column: 2 / 3;  
  grid-row: 1 / 2;  
  position: absolute;  
}
```



## Les articles positionnés se rétractent pour s'adapter à leur contenu

Une autre chose à garder à l'esprit est que les éléments positionnés se réduisent pour s'adapter à leur contenu (vous pouvez voir cet effet dans l'exemple précédent).

Ceci est différent de ce que font les éléments de grille normaux. Les éléments de grille normaux s'étirent pour s'adapter à leur zone de grille. Par conséquent, si vous indiquez qu'un élément de grille s'étend sur deux colonnes et deux lignes, il le fera exactement.

## La valeur `auto` en éléments positionnés

Sur les éléments de grille normaux, la valeur `auto` d'une propriété de positionnement de grille est résolue à `span 1`, mais sur les éléments positionnés de manière absolue, elle est résolue sur le bord de remplissage du conteneur de grille.

Cela pourrait donner lieu à des événements apparemment étranges si vous ne comprenez pas comment cela fonctionne.

La valeur `auto` est la valeur initiale des propriétés de placement dans la grille. Cela se produira par défaut si vous ne spécifiez pas de valeur différente.

## Utilisation de l'inspecteur de grille CSS

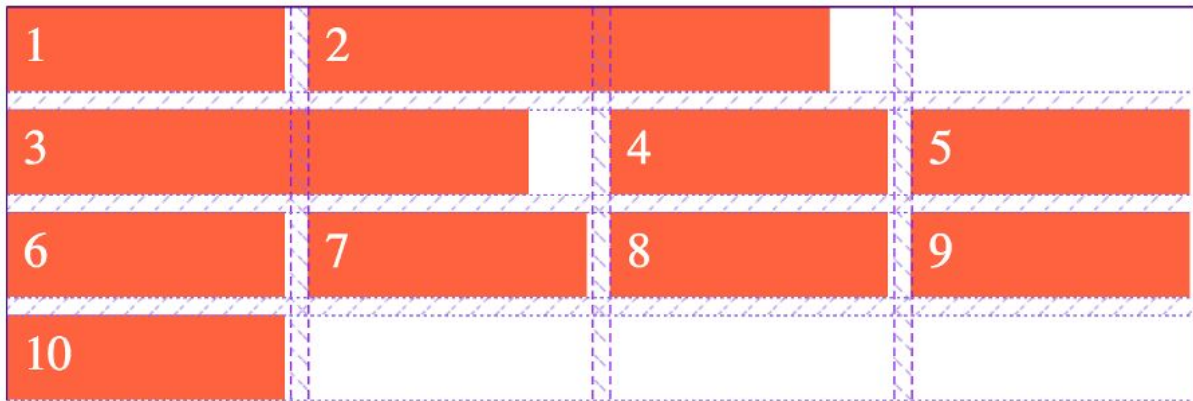
Firefox dispose d'un outil intégré qui vous permet d'inspecter une grille via le navigateur. Il existe également des outils tiers pour le faire dans Chrome.

Firefox DevTools permet d'inspecter la disposition de la grille que vous visualisez dans le navigateur. Cela peut être utile si vous avez une grille complexe avec des éléments de grille couvrant de nombreuses pistes ou une grille très peu peuplée. Dans de tels cas, il peut être difficile de visualiser la grille qui maintient les éléments en place.

L'inspecteur de grille de DevTools vous permet de voir les lignes de la grille et les gouttières, `grid-gap` qui les séparent.

Comme ça:





*Exemple de grille vue dans Firefox DevTools.*

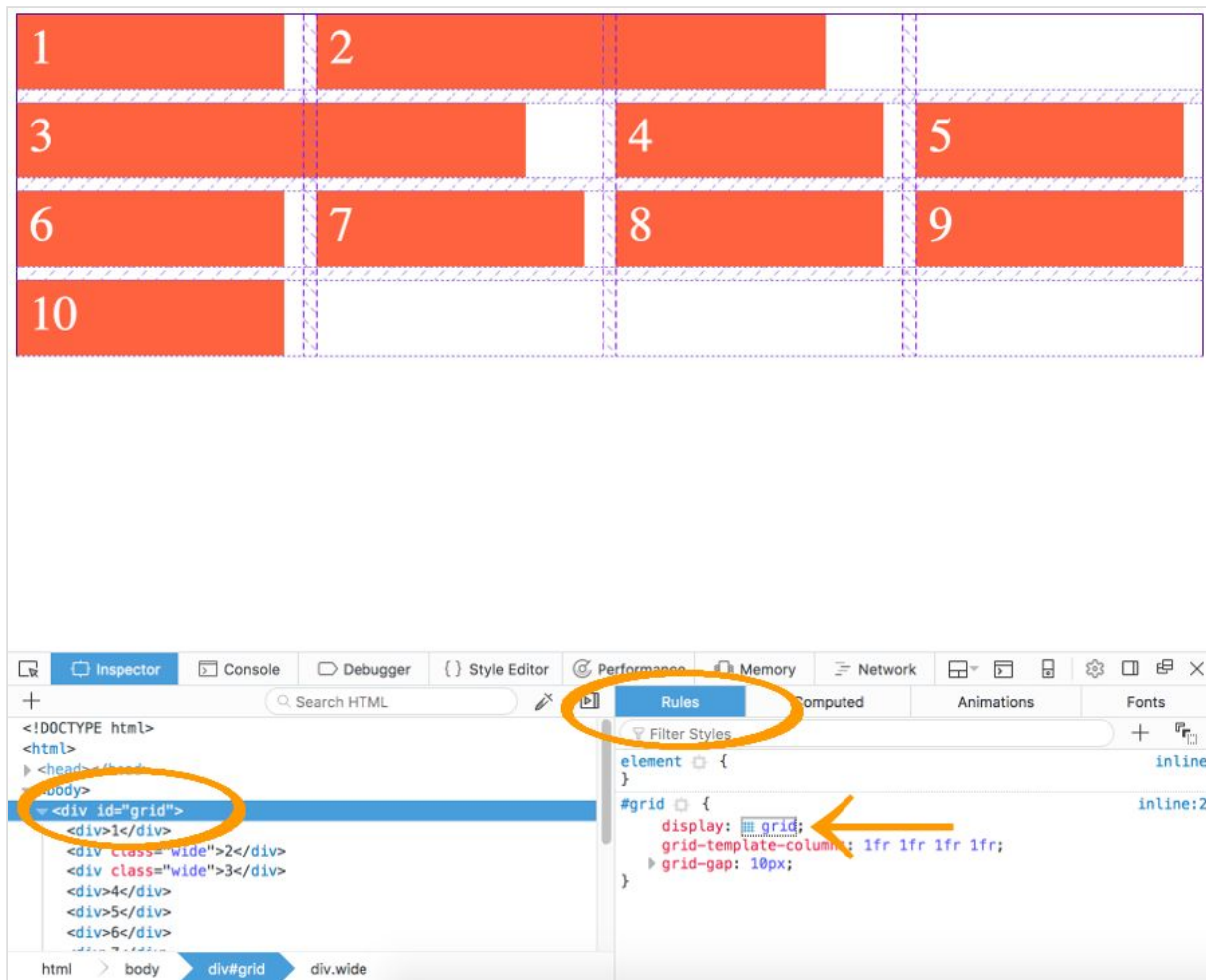
## Comment utiliser l'inspecteur de grille de Firefox

Voici comment utiliser Firefox DevTools pour afficher la disposition de la grille.



Ouvrez l'inspecteur

Lors de l'affichage d'une page Web dans votre navigateur, cliquez avec le bouton droit de la souris sur la grille à inspecter et sélectionnez “*Examiner l’élément*” dans le menu contextuel.



Sélectionnez l'inspecteur de grille

Assurez-vous que l'élément avec `display:grid` est sélectionné dans le volet avec le code source et que l'onglet Mise en Page est sélectionné dans l'autre volet, cliquez sur l'icône en forme de grille ( # ) située entre `display:grid`.

Pour enregistrer un clic supplémentaire ...

Lorsque vous cliquez avec le bouton droit sur la grille (étape 1), sélectionnez une zone vide sans aucun élément de la grille (telle qu'une gouttière ou une cellule vide). Cela ouvrira généralement l'inspecteur avec le conteneur de grille déjà sélectionné.

# Extensions pour Chrome

Au moment de la rédaction de ce document, Google Chrome ne dispose pas d'inspecteur de grille intégré. Cependant, quelques outils tiers ont été conçus dans ce but précis.

- CSS Grid Highlighter pour Chrome peut être téléchargé à partir de [GitHub](#).
- Nitty Griddy est une extension qui fournit un recouvrement pour les éléments stylés avec CSS Grid.