

# Layout avec CSS Grid

# Table des matières

<b>I. Contexte</b>	<b>3</b>
<b>II. La notion de grille</b>	<b>3</b>
<b>III. Exercice : Appliquez la notion</b>	<b>8</b>
<b>IV. Les propriétés de l'élément parent</b>	<b>9</b>
<b>V. Exercice : Appliquez la notion</b>	<b>13</b>
<b>VI. Les propriétés des éléments enfants</b>	<b>15</b>
<b>VII. Exercice : Appliquez la notion</b>	<b>19</b>
<b>VIII. Aligner les éléments</b>	<b>21</b>
<b>IX. Exercice : Appliquez la notion</b>	<b>23</b>
<b>X. Auto-évaluation</b>	<b>24</b>
A. Exercice final .....	24
B. Exercice : Défi .....	25
<b>Solutions des exercices</b>	<b>33</b>

## I. Contexte

**Durée :** 1 h

**Environnement de travail :** Repl.it

**Pré-requis :** Bases de CSS

### Contexte

CSS propose plusieurs outils pour gérer la disposition des éléments sur la page. Nous avons déjà pu voir `float` et `flexbox`, qui fonctionnent sur une dimension à la fois. CSS Grid est une spécification CSS relativement récente qui permet de gérer le positionnement des éléments sur deux dimensions, en lignes et en colonnes.

## II. La notion de grille

### Objectifs

- Comprendre la notion de `grid-container` et de `grid-elements`
- Se familiariser avec les termes liés à CSS Grid : `line`, `cell`, `track` et `area`
- Voir un premier exemple de grille

### Mise en situation

CSS Grid résout le problème du placement vertical d'éléments, partiellement géré par `flexbox`. Les deux outils ont des usages différents et sont parfaitement complémentaires.

Grid permet de construire des dispositions de page complexes sur deux dimensions ; `flexbox` sera, lui, utilisé pour gérer le positionnement des éléments présents dans les grands blocs définis par la grille. Il ne faut pas voir `flexbox` et Grid comme deux outils concurrents, chacun brille dans le bon contexte d'utilisation. Grid est très bien supporté depuis 2017 sur tous les navigateurs principaux.

### Le fonctionnement de base

Grid, tout comme `flexbox`, définit des propriétés à utiliser sur l'élément parent, le `grid container`. Dans l'exemple suivant, il s'agit de l'élément `<div>` avec la classe `.container`.

Il existe également des propriétés à utiliser sur les enfants directs du `grid container`. Dans ce cas, ce sont les `<div>` avec les classes `.header`, `.content` et `.footer`. Ces éléments sont appelés des `grid items`.

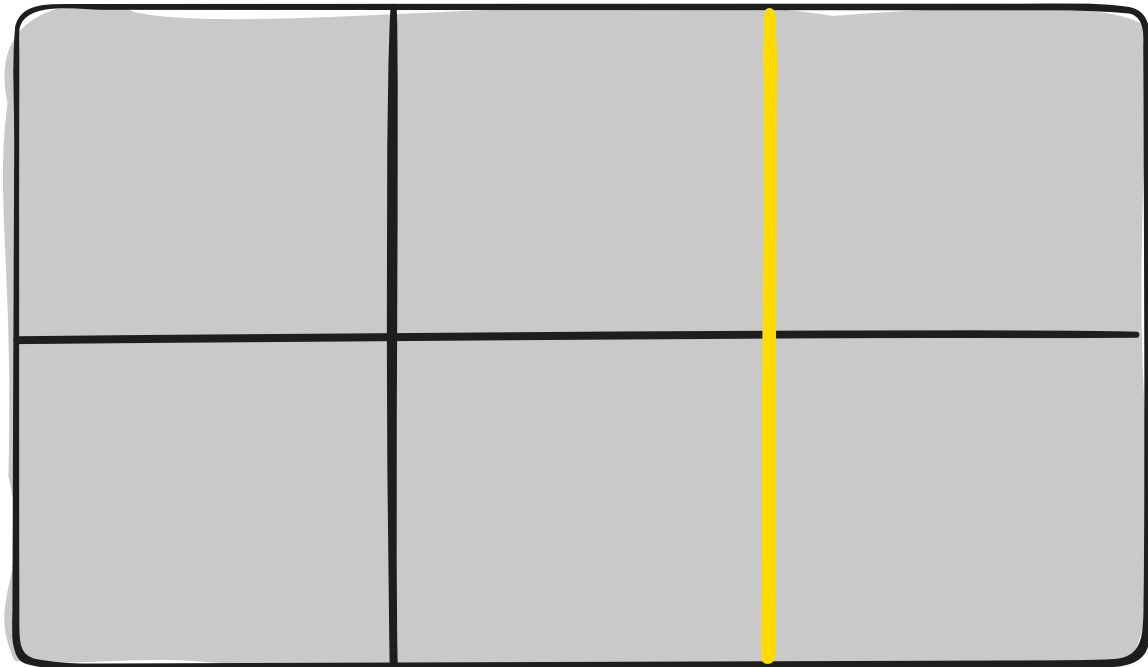
La `<div>` avec la classe `.sponsors` n'est pas concernée par la grille, puisque ce n'est pas un enfant direct.

```
1 <div class="container">
2   <div class="header"></div>
3   <div class="content"></div>
4   <div class="footer">
5     <div class="sponsors"></div>
6   </div>
7 </div>
8
```

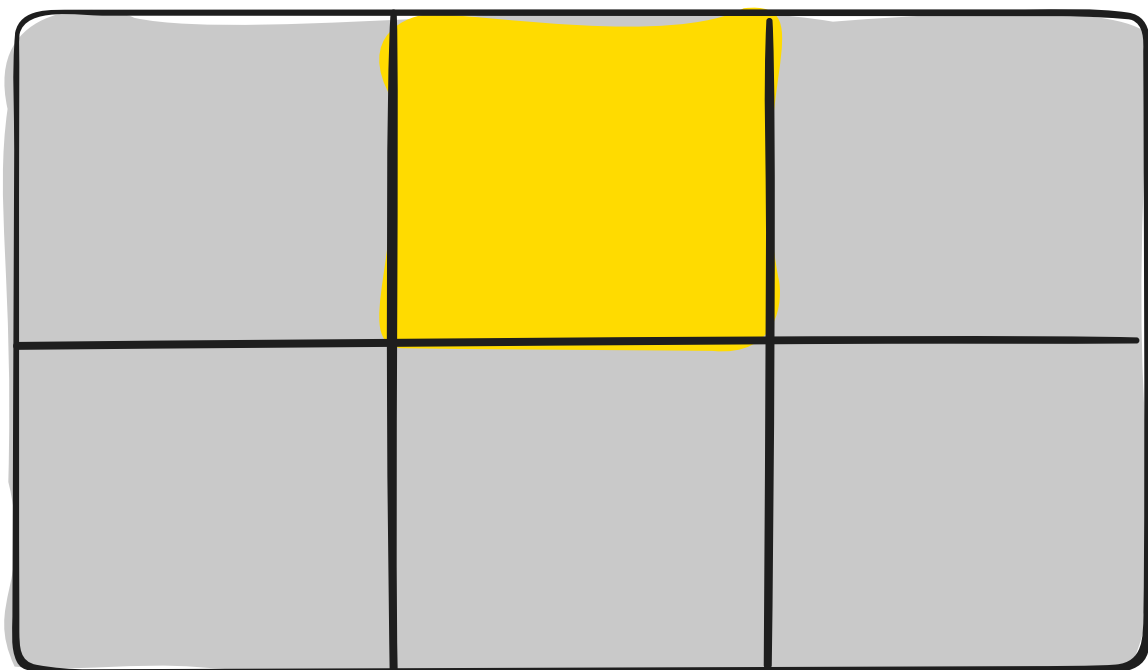
### La constitution d'une grille

Il est important d'être familier avec plusieurs termes quand on travaille avec CSS Grid.

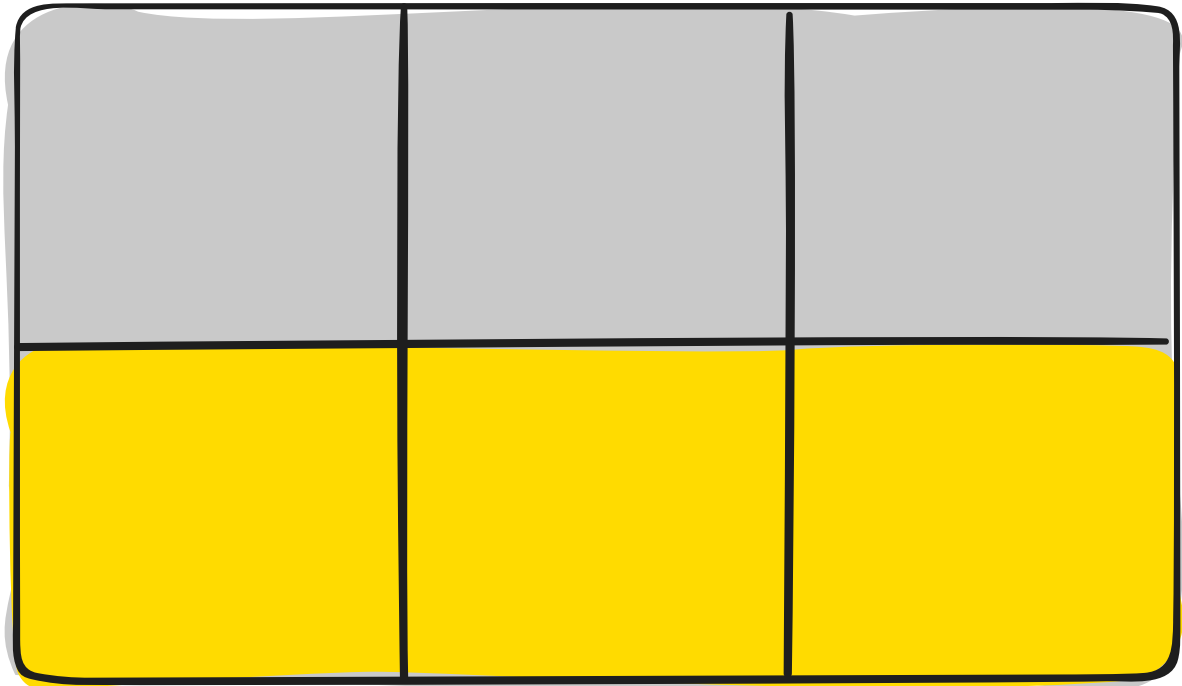
Une `grid line`, ou ligne, est la séparation entre deux lignes ou deux colonnes.



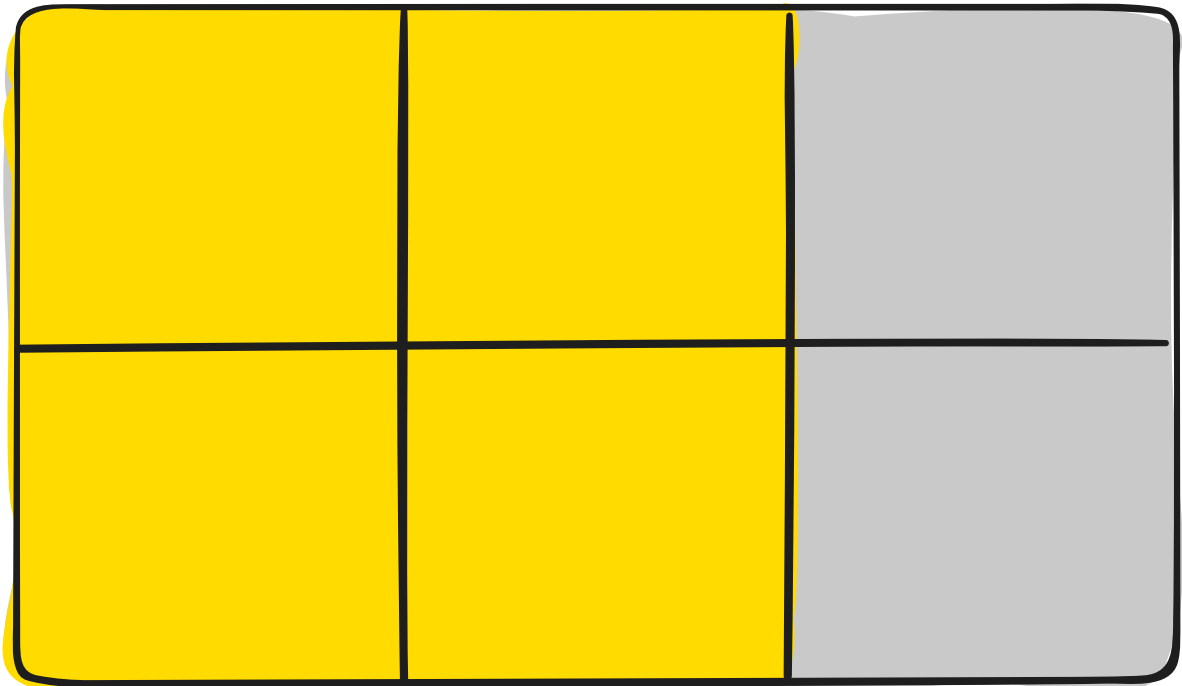
Une `grid cell`, ou cellule, est une case de la grille.



Une `grid track` est une ligne ou une colonne de la grille.



Une `grid area` est une zone composée de plusieurs cellules adjacentes.



### Exemple Une première grille

Prenons l'exemple d'une structure de page que l'on retrouve régulièrement sur un site web : une page avec un header, un footer, un contenu principal et un bloc de côté qui pourrait correspondre à un menu de navigation.

Pour mettre en place ce modèle, il nous suffit de placer ces quatre éléments dans un `grid container`. Ici, la grille en elle-même a été découpée en douze colonnes de 50 px chacune, et en trois lignes de 100 px, 220 px et 150 px.

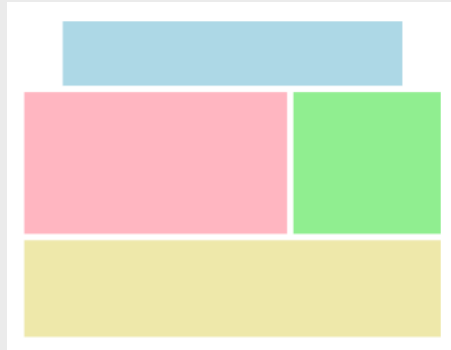
```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <meta name="viewport" content="width=device-width">
6     <title>repl.it</title>
7     <link href="style.css" rel="stylesheet" type="text/css" />
8   </head>
9   <body>
10    <div class="container">
11      <div class="header"></div>
12      <div class="content"></div>
13      <div class="aside"></div>
14      <div class="footer"></div>
15    </div>
16  </body>
17 </html>

1 html, body {
2   margin: 0;
3 }
4 .container {
5   display: grid;
6   grid-template-columns: 50px 50px 50px 50px 50px 50px 50px 50px 50px 50px 50px 50px;
7   grid-template-rows: 100px 220px 150px;
8   gap: 10px;
9   margin: 40px;
10 }
11
12 .header {
13   grid-column-start: 2;
14   grid-column-end: 11;
15   grid-row-start: 1;
16   background: lightblue;
17 }
18
19 .content {
20   grid-column-start: 1;
21   grid-column-end: 8;
22   grid-row-start: 2;
23   background: lightpink;
24 }
25
26 .aside {
27   grid-column-start: 8;
28   grid-column-end: 12;
29   grid-row-start: 2;
30   background: lightgreen;
31 }
32
33 .footer {
```

```

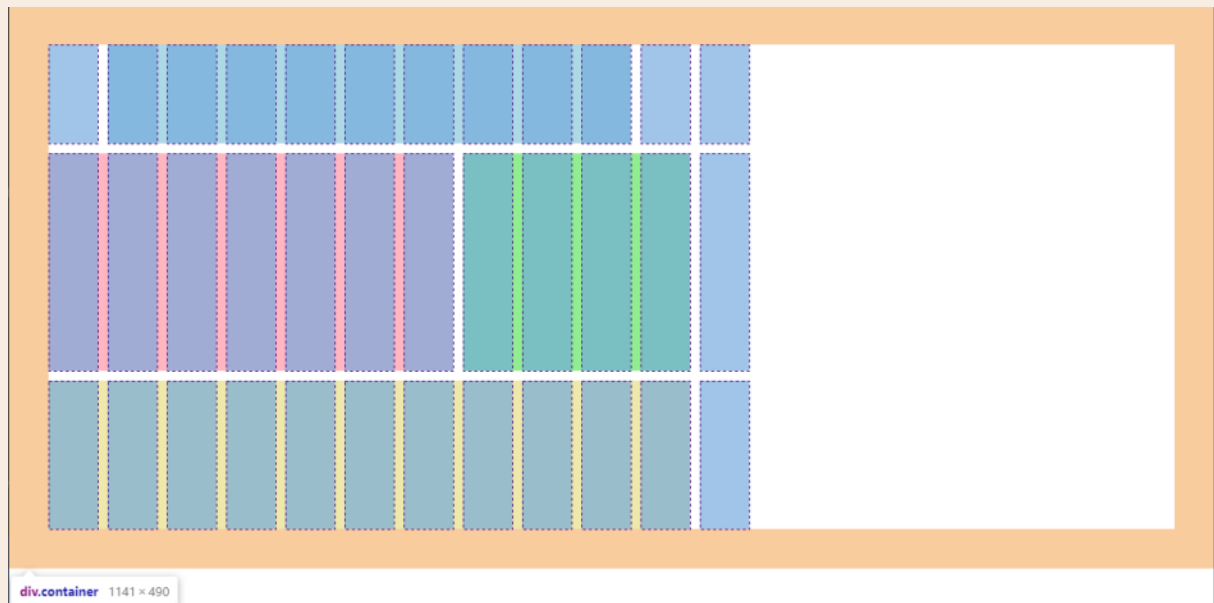
34  grid-column-start: 1;
35  grid-column-end: 12;
36  grid-row-start: 3;
37  background: palegoldenrod;
38  }

```



### Complément Inspecter l'élément

Si vous inspectez l'élément à l'aide des outils de développement de votre navigateur, vous pouvez distinguer facilement le template que l'on a créé, ainsi que les différents éléments qui le composent.



### Syntaxe À retenir

Pour utiliser CSS Grid, il est nécessaire d'appliquer les propriétés sur un élément parent : seuls ses enfants directs seront affectés par les changements.

Il faut être familier avec certains termes :

- `grid line` pour la séparation entre deux lignes ou colonnes
- `grid cell` pour représenter une case dans une grille
- `grid track` pour représenter une ligne ou colonne de la grille
- `grid area` pour représenter une zone de plusieurs cellules adjacentes

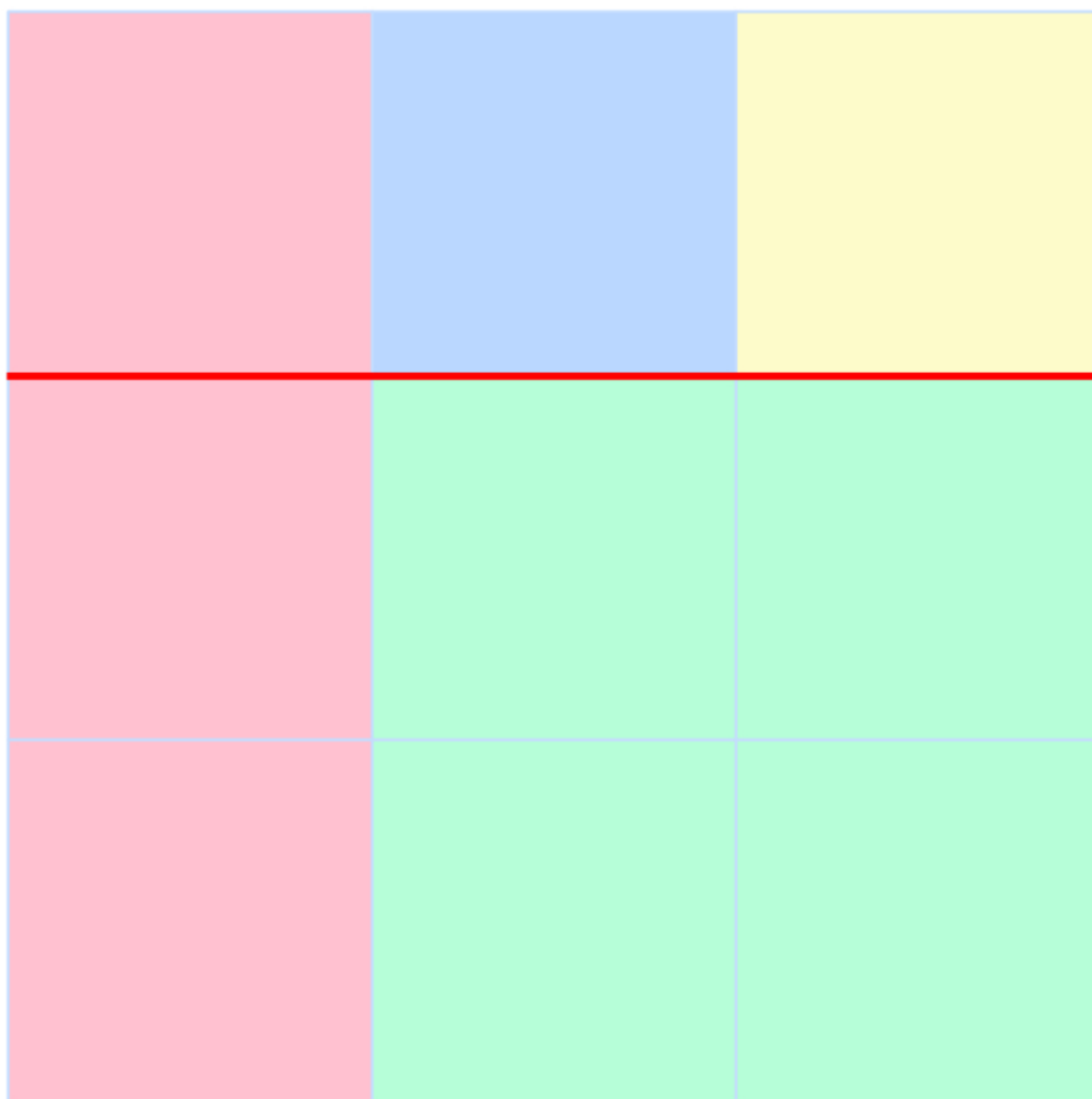
**Complément**

Documentation : CSS Grid<sup>1</sup>

**Exercice : Appliquez la notion**

[solution n°1 p.35]

Faites correspondre les termes aux couleurs représentées sur la grille.



grid-line

grid-track

grid-cell

grid-area

<sup>1</sup> [https://developer.mozilla.org/fr/docs/Web/CSS/CSS\\_Grid\\_Layout](https://developer.mozilla.org/fr/docs/Web/CSS/CSS_Grid_Layout)



Zone verte	Zone rose	Zone bleue / jaune	Séparateur rouge
------------	-----------	--------------------	------------------

## IV. Les propriétés de l'élément parent

### Objectifs

- Comprendre le paramétrage des lignes et des colonnes
- Savoir comment nommer les séparateurs de lignes/colonnes
- Savoir comment créer des zones pour regrouper plusieurs cellules
- Savoir créer un espacement entre les lignes et les colonnes d'une grille

### Mise en situation

Maintenant que nous avons identifié comment était constituée une grille, nous allons pouvoir entrer dans le vif du sujet et apprendre comment définir notre première grille. Certains éléments de syntaxe sont déjà apparus au cours du chapitre précédent, nous allons ici les voir en détails en nous concentrant sur l'élément parent et sur les différentes propriétés qu'il est possible d'y appliquer.

### Définir le mode d'affichage avec `display: grid`

La création d'une grille passe tout d'abord par l'utilisation de la propriété CSS `display` avec la valeur `grid`.

L'élément va alors hériter de certaines caractéristiques d'un élément en `display: block`. Il va réserver tout l'espace disponible en largeur et aucun élément ne pourra être placé à côté de la grille.

Il est également possible d'utiliser la valeur `inline-grid` : l'élément s'approchera alors du comportement d'un élément placé en `inline-block` en conservant le système de grille, c'est-à-dire que l'élément sera aussi grand que son contenu.

Une fois ce mode d'affichage défini, cet élément est maintenant considéré comme notre `grid container`.

### Constituer le template

Une fois le container déclaré, il est nécessaire de définir le corps de la grille, ou l'expression des `grid tracks`. Pour cela, CSS met à disposition deux propriétés : `grid-template-columns` et `grid-template-rows`.

#### Méthode Définir ses colonnes

```
1 grid-template-columns: 50px 50px 50px 50px;
```

`grid-template-columns` définit les colonnes de la grille. Nous avons ici défini quatre colonnes de 50 px chacune. La propriété accepte une liste de tailles qui peuvent être exprimées en pixels, mais également en pourcentage de la page, ou en "fraction" d'espace restant, exprimée sous la forme d'une nouvelle unité (`fr`).

## L'unité fr

On retrouve la même logique que pour la propriété CSS `flex` qui permet d'exprimer la taille d'un élément en fonction de la taille disponible restante dans le conteneur. C'est-à-dire que 1 "fr" représente une unité de taille restante.

Par exemple, si l'on définit les colonnes suivantes, une grille sera créée contenant quatre colonnes : la première de 50 px, et les trois autres se partageant respectivement 1/4 de la taille restante, 1/4 de la taille restante et 2/4 de la taille restante.

```
1 grid-template-columns: 50px 1fr 1fr 2fr;
```

### Complément Combiner les unités

Il est possible de panacher l'expression des tailles pour nos colonnes. Il est également possible d'utiliser la valeur "auto" qui fera en sorte de prendre 100 % de la taille restante après que les autres colonnes aient réservé leur taille.

### Complément Répéter la déclaration

Enfin, dans le cas d'un grand nombre de colonnes, il est possible d'utiliser l'expression `repeat` dans la définition du template pour exprimer une boucle. C'est une fonction qui prend deux paramètres : le nombre d'itérations et la taille de l'élément.

```
1 grid-template-columns: repeat(12, 1fr);
```

Cet exemple produira une grille similaire à celle proposée par Bootstrap, à savoir douze colonnes de tailles égales, 1 fr ou une unité de taille restante pour chaque colonne.

### Méthode Définir ses lignes

```
1 grid-template-rows: 100px 220px 150px;
```

La seconde propriété, `grid-template-rows`, permet cette fois-ci de définir les lignes de la grille. Comme pour la définition des colonnes, il est possible d'utiliser toutes les unités que nous avons vues (px, pourcentage, fr, auto) et il est également possible de définir plusieurs lignes de manière répétée avec la fonction `repeat`.

### Méthode Espacer les cellules

Dans le premier exemple, il y avait une propriété "gap" sur le `grid container`. Cette propriété va déterminer l'espacement entre les cellules d'une grille.

Par défaut, il n'y a pas d'espacement, mais il est possible d'utiliser cette propriété avec une ou deux valeurs. Dans le cas d'une utilisation avec deux valeurs, celles-ci font respectivement référence à l'espacement entre deux lignes de la grille et entre deux colonnes de la grille.

```
1 gap: 10px 15px;
```

### Exemple Structurer une page web

Reprenons l'exemple d'une page web classique et concentrons-nous sur la définition des propriétés de l'élément parent `.container`.

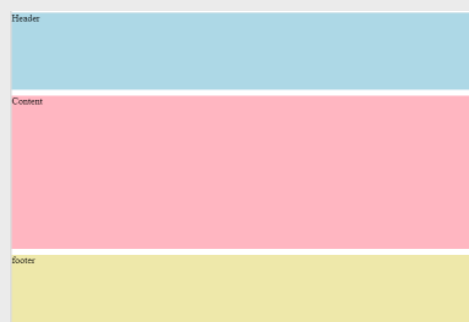
```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <meta name="viewport" content="width=device-width">
6     <title>repl.it</title>
7     <link href="style.css" rel="stylesheet" type="text/css" />
```

```
8   </head>
9   <body>
10    <div class="container">
11      <div class="header">Header</div>
12      <div class="content">Content</div>
13      <div class="footer">Footer</div>
14    </div>
15  </body>
16 </html>
```

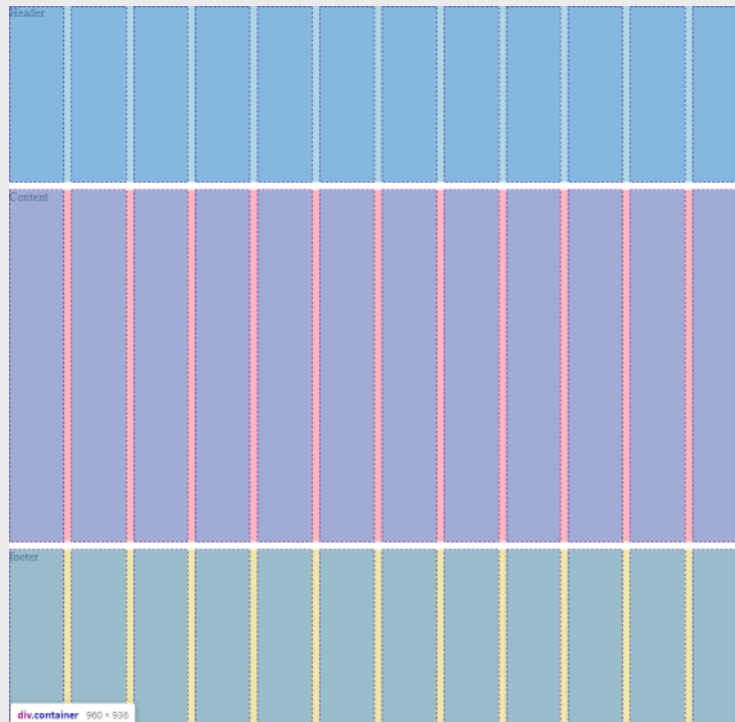
Notre élément `.container` prendra toute la hauteur de l'écran, sera séparé en 12 colonnes, qui prendront chacune une portion de l'espace disponible. Les 3 lignes prendront quant à elles respectivement une fraction de l'espace disponible, deux fractions et une fraction.

On indique que l'on souhaite également un espacement de 10 pixels entre nos lignes et colonnes.

```
1 html, body {
2   margin: 0;
3 }
4 .container {
5   display: grid;
6   grid-template-columns: repeat(12, 1fr);
7   grid-template-rows: 1fr 2fr 1fr;
8   gap: 10px;
9   height: 100vh;
10 }
11
12 .header {
13   grid-column-start: 1;
14   grid-column-end: 13;
15   grid-row-start: 1;
16   background: lightblue;
17 }
18
19 .content {
20   grid-column-start: 1;
21   grid-column-end: 13;
22   grid-row-start: 2;
23   background: lightpink;
24 }
25
26
27 .footer {
28   grid-column-start: 1;
29   grid-column-end: 13;
30   grid-row-start: 3;
31   background: palegoldenrod;
32 }
```



Si on inspecte notre élément, on constate que tous nos éléments sont bien là.



#### Complément Nommer les séparations

Il est possible de nommer les lignes de séparation (`grid lines`) entre deux colonnes ou entre deux lignes de la grille pour pouvoir placer les éléments plus facilement par la suite. Pour nommer ces séparateurs, il faut simplement ajouter entre crochets un nom entre deux colonnes ou entre deux lignes.

```
1 grid-template-rows: [start] 1fr [spacing-left] auto [spacing-right] 1fr [end];
```

#### Complément Nommer les zones

Il est possible de nommer dans les propriétés du `grid container` les zones, ou `grid area`, là encore pour faciliter le positionnement des éléments par la suite et donner un peu de sémantique aux lignes et aux colonnes.

```
1 grid-template-columns: 50px 50px 50px 50px;
2 grid-template-rows: 100px 220px 150px;
3 grid-template-areas:
4   "header header header header"
5   "main main . sidebar"
6   "footer footer footer footer";
7
```

La syntaxe est un peu particulière : elle consiste en quelque sorte à dessiner les zones souhaitées.

Ici, on a défini que la première ligne serait consacrée au header, que la seconde ligne portera la section principale sur les deux colonnes les plus à gauche, puis qu'une colonne sera vide pour créer un espacement (c'est le caractère "." qui représente cet espace vide), et enfin, dans la dernière colonne sur la droite, un élément de barre d'outil. Toute la ligne du bas contiendra le footer.

On abordera la manière de positionner un élément par rapport à ces zones dans la partie sur les propriétés des `grid elements`.

**Syntaxe**    **À retenir**

- La création d'une grille passe par l'application de la propriété `display` sur un élément. On y affecte la valeur `grid` ou `inline-grid` qui se rapprocheront respectivement des valeurs `block` et `inline-block`.
- On définit les lignes qui composeront la grille au moyen de la propriété `grid-template-rows` et les colonnes au moyen de la propriété `grid-template-columns`.
- L'unité `fr` permettra de représenter une fraction de l'espace restant.
- Si l'on souhaite répéter une configuration, on pourra utiliser la fonction `repeat`.
- Une séparation entre deux lignes ou deux colonnes peut être nommée en indiquant son nom entre crochets : `[nom_de_la_separation]`, tandis que les différentes zones peuvent être nommées grâce à la propriété `grid-template-areas`. Ces noms permettent de positionner plus facilement des éléments par la suite.
- Il est possible d'espacer les cellules grâce à la propriété `gap`.

## V. Exercice : Appliquez la notion

Pour réaliser cet exercice, vous aurez besoin de travailler sur l'environnement de travail :



---

1 <https://repl.it/>

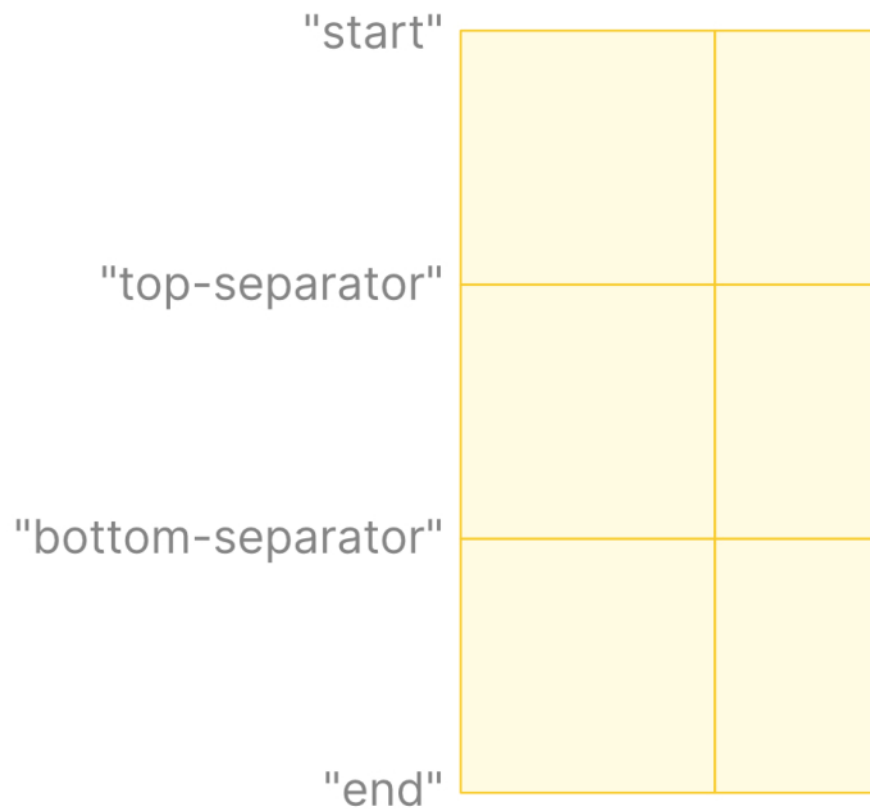
### Question 1

[solution n°2 p.36]

Définissez dans l'élément `.wrapper` une grille de douze colonnes de taille égale et de 3 lignes. Les lignes tout en haut et tout en bas doivent mesurer 100px, et celle du centre doit prendre la taille restante.

Les colonnes devront être générées automatiquement avec un système de boucle.

Les séparateurs entre les lignes devront être nommés de la manière suivante :



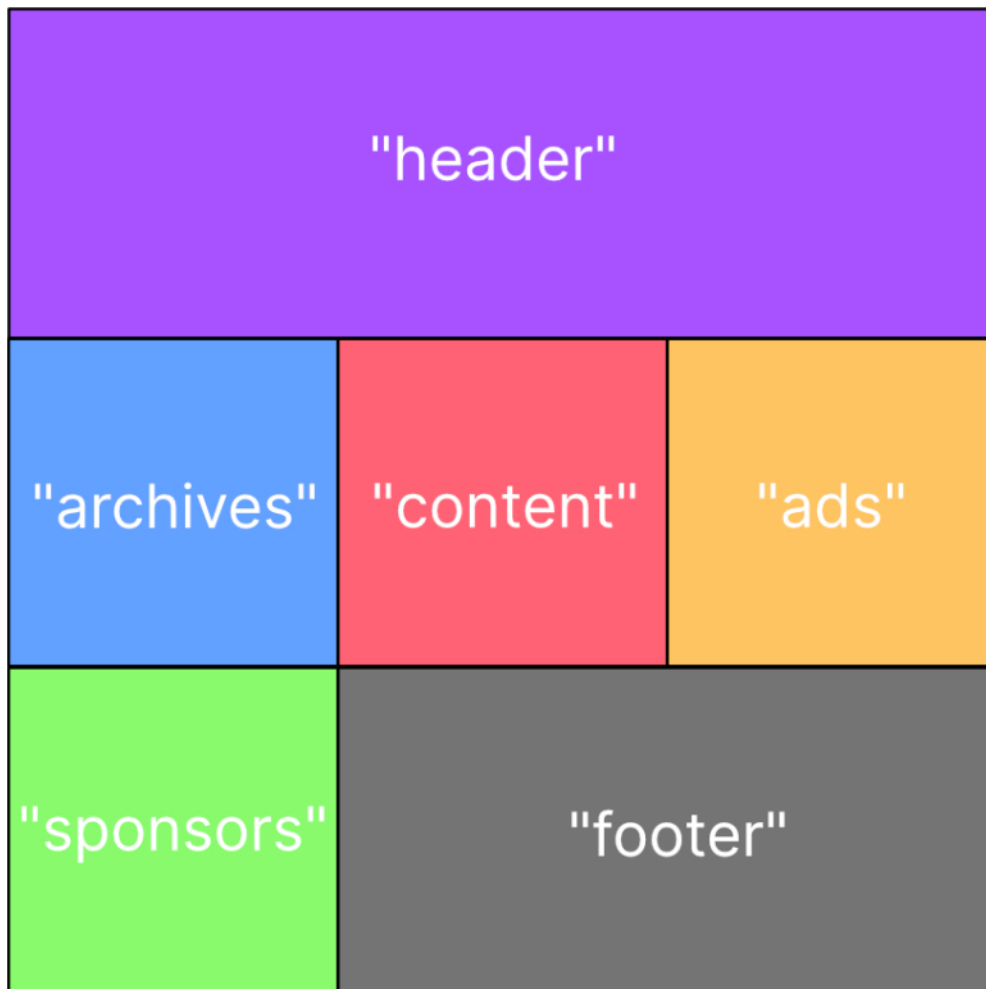
```
1 .wrapper {
2   //
3 }
```

**Question 2**

[solution n°3 p.36]

Définissez dans l'élément `.wrapper` une grille de trois colonnes par trois lignes. Chaque cellule de la grille doit mesurer 200 pixels par 200 pixels.

Créez des `grid-areas` qui correspondent au schéma suivant :

**VI. Les propriétés des éléments enfants****Objectifs**

- Placer des éléments sur la grille avec `grid-row-x` et `grid-column-x`
- Placer des éléments avec les séparateurs nommés
- Placer des éléments sur des `grid-area`

## Mise en situation

Nous venons de voir le paramétrage global d'une grille CSS avec les propriétés du `grid container`. Voyons maintenant le positionnement des éléments sur la grille.

Contrairement au `grid-container`, il est inutile d'explicitement définir un élément comme `grid item`: il sera automatiquement considéré comme tel par le navigateur s'il s'agit d'un enfant direct du `grid-container`.

Il est également à noter que, de ce fait, les propriétés `float`, `display` et `vertical-align` n'auront aucun effet sur un `grid-item`. Son positionnement sera dicté par la grille.

Pour les exemples suivants, considérons ce `grid-container` : une grille de 4 par 4, chaque cellule fait 100 px par 100 px, et elles sont espacées de 10 px verticalement et horizontalement.

```
1 .container {
2   display: inline-grid;
3   grid-template-columns: repeat(4, 100px);
4   grid-template-rows: repeat(4, 100px);
5   gap: 10px;
6   background: lightgray;
7 }
```

### Syntaxe Positionner un élément

Les premières propriétés qui permettent de positionner un élément enfant sur la grille sont `grid-column-start` et `grid-row-start`. Ces propriétés permettent d'indiquer respectivement le séparateur (`grid-line`) de départ pour les colonnes et les lignes de la grille.

Pour placer un élément sur la première ligne, deuxième colonne, on indiquera :

```
1 .item1 {
2   grid-row-start: 1;
3   grid-column-start: 2;
4   background: #0984e3;
5 }
```



### Méthode Gérer les dimensions

Par défaut, un élément placé sur la grille va mesurer une colonne de largeur et une ligne de hauteur, mais il est possible de paramétrer la ligne/colonne de fin, en reprenant le code précédent.

En ajoutant les propriétés `grid-column-end` et `grid-row-end`, il est possible de définir la taille de l'élément sur la grille.



```

1 .item1 {
2   grid-column-start: 2;
3   grid-row-start: 1;
4   grid-column-end: 4;
5   grid-row-end: 3;
6   background: #0984e3;
7 }

```



Pour chacune de ces propriétés, il est possible de spécifier deux valeurs séparées par un slash ( / ) : l'élément s'étendra alors sur plusieurs lignes ou plusieurs colonnes.

grid-row est une propriété raccourcie pour :

- grid-row-start : numéro de la ligne où commencera l'élément.
- grid-row-end : numéro de la ligne contre laquelle s'arrêtera l'élément, ou avec le mot span, le nombre de lignes sur lequel s'étendra l'élément.

grid-column est une propriété raccourcie pour :

- grid-column-start : numéro de la colonne où commencera l'élément.
- grid-column-end : numéro de la colonne contre laquelle s'arrêtera l'élément, ou avec le mot span le nombre de colonnes sur lequel s'étendra l'élément.

#### **Méthode** Positionner par rapport à des séparateurs nommés

Si la grille avait été définie avec des séparateurs nommés, par exemple :

```

1 .container {
2   display: inline-grid;
3   grid-template-columns: [start-col] 100px [middle-col] 100px [end-col];
4   grid-template-rows: auto;
5 }
6

```

Il aurait alors été possible d'indiquer le nom du séparateur comme valeur des propriétés grid-column-start, grid-row-start, grid-column-end ou grid-row-end à la place du numéro du séparateur.

```

1 .item1 {
2   grid-column-start: start-col;
3   grid-row-start: 1;
4   grid-column-end: end-col;
5 }

```

## Méthode Positionner par rapport à des zones

Il est aussi possible d'utiliser la propriété `grid-area`, qui va venir remplacer `grid-column-start`, `grid-row-start`, `grid-column-end` et `grid-row-end` pour indiquer cette fois-ci une zone, si la grille a été définie avec des zones (`grid-area`), comme cette grille-ci :

```
1 .container {
2   display: inline-grid;
3   grid-template-columns: repeat(4, 100px);
4   grid-template-rows: repeat(4, 100px);
5   gap: 10px;
6   grid-template-areas:
7     "header header header header"
8     "main main . aside"
9     "main main . aside"
10    "footer footer footer footer";
11 }
12
13 .page-header {
14   grid-area: header;
15   background: lightpink;
16 }
17
18 section.content {
19   grid-area: main;
20   background: lightgreen;
21 }
22
23 nav.toolbar {
24   grid-area: aside;
25   background: palegoldenrod;
26 }
27
28 footer {
29   grid-area: footer;
30   background: lightblue;
31 }
```



## Syntaxe À retenir

- Le positionnement d'un élément de la grille est dicté par la grille, les propriétés `float`, `display` et `vertical-align` ne l'impactent donc pas.
- Les propriétés `grid-column-start` et `grid-row-start` permettent d'indiquer la ligne et la colonne de départ d'un élément.
- On pourra gérer les dimensions d'un élément en indiquant sa ligne et sa colonne de fin avec les propriétés `grid-column-end` et `grid-row-end`.

- Les propriétés `grid-column` et `grid-row` sont des raccourcis des propriétés `grid-column-start`, `grid-column-end`, `grid-row-start` et `grid-row-end`.
- Par défaut, ces propriétés prennent comme valeur le numéro de la ligne/colonne. En revanche, si des séparateurs ont été nommés, on pourra utiliser leur nom.
- En utilisant la propriété `grid-area` sur un item, et si des zones ont été nommées, on pourra indiquer qu'un élément occupera toute cette zone.

## VII. Exercice : Appliquez la notion

Pour réaliser cet exercice, vous aurez besoin de travailler sur l'environnement de travail :



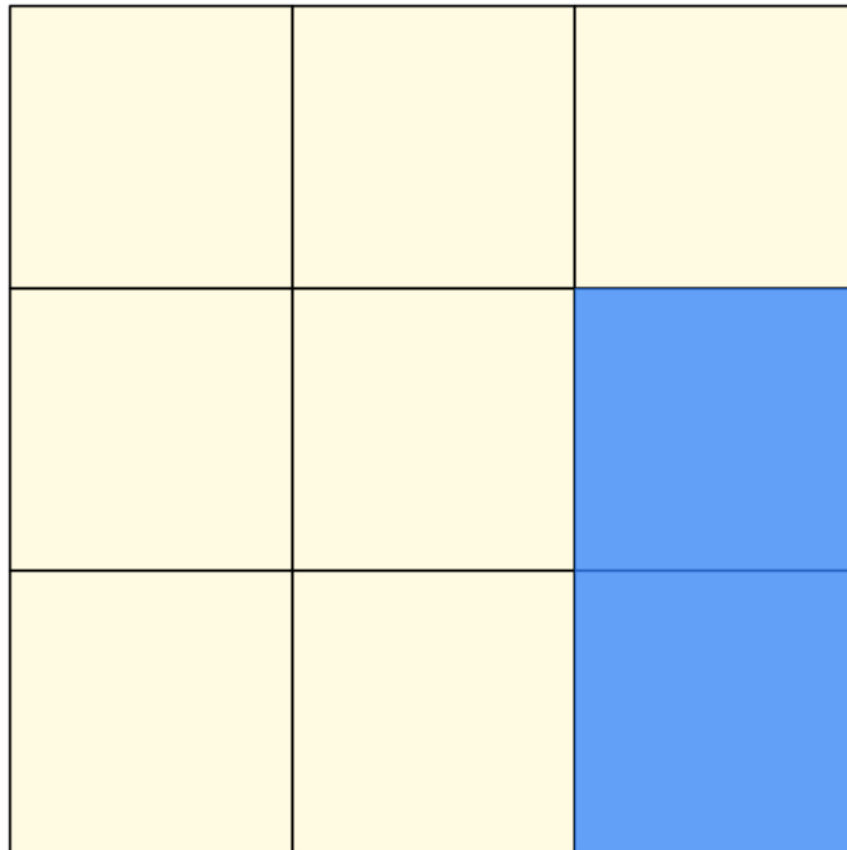
---

1 <https://repl.it/>

## Question 1

[solution n°4 p.36]

La grille suivante vous est fournie, placez l'élément `.content` comme représenté sur le schéma. Vous proposerez deux codes : dans le premier, l'élément sera placé avec des valeurs numériques ; dans le second, il sera placé par rapport aux séparateurs nommés.



```

1 .wrapper {
2   display: grid;
3   grid-template-columns: [col-start] 1fr [separator-left] 1fr [separator-right] 1fr [col-end];
4   grid-template-rows: [row-start] 1fr [separator-top] 1fr [separator-bottom] 1fr [row-end];
5 }
6
7 .content {
8   //
9 }

```

## Question 2

[solution n°5 p.36]

La grille suivante vous est fournie, placez les éléments `.header`, `.content` et `.footer` en les faisant correspondre à leurs zones respectives (elles portent le même nom).

```

1 .wrapper {
2   display: grid;
3   grid-template-columns: [col-start] 1fr [separator-left] 1fr [separator-right] 1fr [col-end];
4   grid-template-rows: [row-start] 1fr [separator-top] 1fr [separator-bottom] 1fr [row-end];
5   grid-template-areas:
6     ". header ."
7     "content . ads"
8     "footer footer footer";
9 }
10
11 .header {
12   //
13 }
14
15 .content {
16   //
17 }
18
19 .footer {
20   //
21 }
```

## VIII. Aligner les éléments

### Objectifs

- Aligner horizontalement un élément dans une `grid-area`
- Aligner verticalement un élément dans une `grid-area`

### Mise en situation

Jusqu'à présent, tous les `grid-items` que nous avons utilisés n'avaient pas de hauteur ou de largeur explicitement définie. Par conséquent, ces éléments vont par défaut s'étirer verticalement et horizontalement, jusqu'à remplir toute la zone qui leur était assignée.

Dans le cas où un `grid-item` serait explicitement plus petit que la taille de la zone dans lequel il est placé, il est possible de gérer le positionnement de cet élément à l'intérieur de sa `grid-area`.

#### Exemple

Imaginons, pour les exemples qui vont suivre, le `grid-container` suivant :

```

1 .container {
2   display: grid;
3   grid-template-columns: repeat(4, 100px);
4   grid-template-rows: repeat(4, 100px);
5   gap: 10px;
6 }
```

Soit une grille de 4 par 4, chaque cellule mesurant 100 px par 100 px.

Et des `grid-elements` possédant les propriétés suivantes :

```
1 .item {
2   width: 50%;
3   height: 50%;
4   background: #0984e3;
5 }
```

Les `grid-elements` en bleu sont placés chacun sur une cellule de la grille et, pour le moment, sont alignés en haut à gauche de leur cellule.

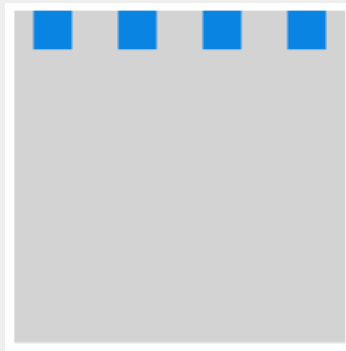


### Syntaxe Positionnement horizontal

La propriété `justify-items` permet de configurer le positionnement horizontal de l'élément. Les valeurs `start`, `center` et `end` permettent respectivement de placer les éléments à gauche, au centre, et à droite de la cellule.

Voici un exemple avec la valeur `center` :

```
1 .container {
2   display: grid;
3   grid-template-columns: repeat(4, 100px);
4   grid-template-rows: repeat(4, 100px);
5   gap: 10px;
6   justify-items: center;
7 }
```

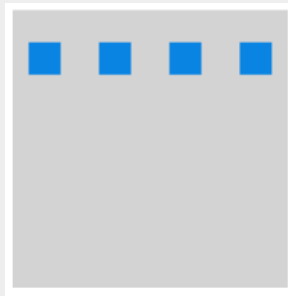


**Syntaxe**    **Positionnement vertical**

À l'inverse, la propriété `align-items` utilisée sur le `grid-container` permet de gérer le positionnement vertical des éléments dans les `grid-area`. Là encore, les valeurs possibles sont `start`, `center` et `end`.

Ici, un exemple avec `end` :

```
1
2 .container {
3   display: grid;
4   grid-template-columns: repeat(4, 100px);
5   grid-template-rows: repeat(4, 100px);
6   gap: 10px;
7   justify-items: center;
8   align-items: end;
9 }
```

**Syntaxe**    **À retenir**

- Pour gérer le positionnement vertical des éléments d'une grille, on utilisera la propriété `align-items`.
- Le positionnement horizontal sera quant à lui géré au moyen de la propriété `justify-items`.
- Ces deux propriétés acceptent les valeurs `start`, `end`, et `center`.

**IX. Exercice : Appliquez la notion**

Pour réaliser cet exercice, vous aurez besoin de travailler sur l'environnement de travail :

**Question**

[solution n°6 p.37]

La grille suivante vous est fournie, ainsi que le CSS correspondant à l'élément `.content`.

Cet élément est plus petit que la cellule qui le contient : faites en sorte d'aligner l'élément sur la droite de la cellule horizontalement, et au centre de la cellule verticalement.

```
1 .wrapper {
2   display: grid;
3   grid-template-columns: repeat(4, 100px);
4   grid-template-rows: repeat(4, 100px);
5 }
6
```

1 <https://repl.it/>

```
7 .content {
8   background: lightblue;
9   height: 30px;
10  width: 30px;
11 }
```

## X. Auto-évaluation

### A. Exercice final

#### Exercice 1

[solution n°7 p.37]

Exercice

CSS Grid est...

- ☐ Un outil qui permet de gérer le layout de la page
- ☐ Un système de tableau croisé dynamique porté pour le Web

Exercice

CSS Grid et flexbox...

- ☐ Sont redondants
- ☐ Sont deux outils qui n'ont pas du tout le même rôle
- ☐ Sont complémentaires

Exercice

L'élément de base sur lequel est définie la grille est appelé...

- ☐ Un grid-wrapper
- ☐ Un grid-container
- ☐ Un grid-base

Exercice

Combien de colonnes affichera la grille suivante ?

```
1 .wrapper {
2   display: grid;
3   grid-template-columns: repeat(12, 1fr);
4 }
```

- ☐ 12
- ☐ 1
- ☐ Ce code n'ajoute pas de colonnes

Exercice

L'unité `fr` correspond à...

- ☐ 100 px de la page, à peu près
- ☐ Un pourcentage de l'élément parent
- ☐ Une unité de taille restante



## Exercice

Dans la syntaxe suivante :

```
1 grid-template-columns: [start] 100px [end];
```

À quoi font référence [start] et [end] ?

- ☐ Ils font partie intégrante de la syntaxe pour décrire la colonne
- ☐ Ce sont des séparateurs nommés
- ☐ Ce sont des colonnes nommées, équivalentes à des colonnes mesurant 1 fr

## Exercice

La propriété `grid-area`...

- ☐ Permet de définir les zones dans la grille
- ☐ Permet de placer un élément sur une zone définie

## Exercice

Par défaut, un élément placé dans la grille...

- ☐ Mesure toute la largeur, mais pas toute la hauteur de sa zone
- ☐ Mesure toute la hauteur et toute la largeur de sa zone
- ☐ Mesure toute la hauteur, mais pas toute la largeur de sa zone

## Exercice

Par défaut, un élément placé en `display: grid`...

- ☐ Prend toute la largeur de son conteneur
- ☐ Prend toute la hauteur de son conteneur
- ☐ Empêche d'autres éléments de se placer à côté
- ☐ Permet de déclarer une nouvelle grille

## Exercice

La valeur correcte de la propriété `display` pour afficher une grille qui partage certains comportements d'un élément `inline-block` est...

- ☐ `inline-grid`
- ☐ `grid-inline`
- ☐ Il n'est pas possible d'afficher une grille en ligne

**B. Exercice : Défi**

Pour réaliser cet exercice, vous aurez besoin de travailler sur l'environnement de travail :



---

1 <https://repl.it/>

Un drame s'est produit sur le projet de blog gastronomique de votre client étoilé ! Votre stagiaire a commencé à refondre le design du site, mais a utilisé des `float` pour aligner les éléments ! Heureusement, maintenant que vous maîtrisez l'utilisation de Grid CSS, vous allez pouvoir lui montrer comment créer des layouts modernes sans effort.

Votre stagiaire a retiré les affreux `float` qu'il avait utilisés et n'a laissé qu'une base HTML et quelques classes CSS, qui pourront vous être utiles. Vous pouvez vous concentrer sur la mise en place du layout.

Voici le code :

```

1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Grid'Cuisine</title>
5     <meta charset="utf-8" />
6     <link href="/style.css" rel="stylesheet" type="text/css">
7     <link
8       href="https://fonts.googleapis.com/css?family=Lato&display=swap"
9       rel="stylesheet"
10    />
11  </head>
12  <body>
13    <header>
14      <h1>Grid'Cuisine</h1>
15    </header>
16    <nav>
17      <ul class="navbar">
18        <li>Entrées</li>
19        <li>Plats</li>
20        <li>Déserts</li>
21      </ul>
22    </nav>
23    <div class="content">
24      <div class="clickable">
25        <h2 class="img-title">
26          Saumon papilote
27        </h2>
28        
32      </div>
33      <div class="block advert clickable">
34        
38        <div style="margin: 2rem;">
39          <h2>Découvrez nos offres partenaires !</h2>
40          Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi
41          placerat vehicula mattis. Ut aliquet, sem vel rutrum posuere, augue
42          nunc porttitor elit, quis aliquam velit odio id dolor.
43        </div>
44      </div>
45      <div class="block clickable full-row">
46        
50        <div style="padding-left: 2rem;">

```

```

51     <h2>Les crêpes de Laurent</h2>
52     <h3>Ingrédients</h3>
53     <ul>
54         <li>300g de farine</li>
55         <li>3 oeufs entiers</li>
56         <li>3 cuillères à soupe de sucre</li>
57         <li>2 cuillères à soupe d'huile</li>
58         <li>50g de beurre fondu</li>
59         <li>60cl de lait</li>
60         <li>5cl de rhum</li>
61     </ul>
62 </div>
63 </div>
64 <div class="clickable">
65     <h2 class="img-title">Verrine fruits</h2>
66     
70 </div>
71
72 <div class="clickable">
73     <h2 class="img-title">Pizza italienne</h2>
74     
78 </div>
79
80 <div class="clickable">
81     <h2 class="img-title">Tiramisu fruité</h2>
82     
86 </div>
87 </div>
88 <footer>
89     Tous droits réservés et bon appétit !
90 </footer>
91 </body>
92 </html>
93

```

```

1  body {
2      font-family: lato;
3      margin: 0;
4  }
5  img {
6      max-width: 100%;
7  }
8  h1 {
9      margin: 2rem;
10 }
11 .clickable:hover {
12     cursor: pointer;
13     opacity: 0.5;
14 }

```

```

15     .img-title {
16         position: absolute;
17         margin-left: 2rem;
18         color: white;
19     }
20     .advert {
21     }
22     .navbar {
23         list-style: none;
24     }
25     header {
26     }
27     nav{
28         border-bottom: 1px solid;
29     }
30     .content {
31         margin: 3rem;
32     }
33     footer {
34         text-align: center;
35         border-top: 1px solid;
36     }
37     .block {
38         border-radius: 25px;
39     }
40     .full-row {
41     }

```

### Question 1

[solution n°8 p.38]

La première étape est de réparer la barre de navigation. Pour cela, il va falloir utiliser le `display grid`, pour mettre sur la même ligne le nom du site et le menu.

Vous commencerez donc par définir le layout général de la page, comme ceci :



Puis vous en profiterez pour définir la partie centrale de la page, qui contiendra les recettes dans la div `.content`. Cette partie doit être composée de trois lignes et de trois colonnes. Pour plus d'harmonie, les trois colonnes auront les mêmes dimensions, de même pour les lignes.

## Grid'Cuisine

Entrées

Plats

Desserts



Saumon papillote



### Découvrez nos offres partenaires !

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi placerat vehicula mattis. Ut aliquet, sem vel rutrum posuere, augue nunc porttitor elit, quis aliquam velit odio id dolor.



### Les crêpes de Laurent

#### Ingrédients

- 300g de farine
- 3 oeufs entiers
- 3 cuillères à soupe de sucre
- 2 cuillères à soupe d'huile
- 50g de beurre fondu
- 60cl de lait
- 5cl de rhum



Verrine fruits



Pizza italienne



Tiramisu fruité

Tous droits réservés et bon appétit !

## Question 2

[solution n°9 p.39]

Maintenant que les bases du layout sont en place, il faut s'occuper du contenu : la partie principale qui contient les recettes.

Pour le moment, les blocs sont tous collés les uns aux autres. Vous allez ajouter un peu d'espace entre ces derniers (30 px).

Ensuite, pour apporter un côté design à la page et mettre en avant certaines informations importantes, il faudra que le bloc « *Découvrez nos offres partenaires* » prennent deux colonnes de long, et que le bloc « *Les crêpes de Laurent* » en prenne trois.

Pour cela, vous pouvez utiliser les classes `.advert` et `.full-row` créées par votre stagiaire.

Grid'Cuisine

Entrées

Plats

Desserts



Les crêpes de Laurent

Ingrédients

- 300g de farine
- 3 oeufs entiers
- 3 cuillères à soupe de sucre
- 2 cuillères à soupe d'huile
- 50g de beurre fondu
- 60cl de lait
- 5cl de rhum



Tous droits réservés et bon appétit !

## Question 3

[solution n°10 p.40]

C'est un peu mieux, mais il reste quelques détails à régler. La recette de crêpe s'affiche très mal, vous avez donc l'idée de modifier l'affichage de ce bloc en utilisant `grid`, pour qu'il soit séparé en deux parties égales : à gauche, il y aura l'image de la crêpe, et à droite, la recette.


Grid'Cuisine


Entrées

Plats

Desserts


Saumon papillote





Découvrez nos offres partenaires !

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi placerat vehicula mattis. Ut aliquet, sem vel rutrum posuere, augue nunc porttitor elit, quis aliquam velit odio id dolor.




### Les crêpes de Laurent


**Ingrédients**

- 300g de farine
- 3 oeufs entiers
- 3 cuillères à soupe de sucre
- 2 cuillères à soupe d'huile
- 50g de beurre fondu
- 60cl de lait
- Sci de rhum


Verrine fruits



Pizza d'été



Tiramisu fruité



Tous droits réservés et bon appétit !

## Question 4

[solution n°11 p.40]

En ayant remis un peu d'ordre, vous vous rendez compte qu'il manque une recette ! Après vérification, c'est celle du bœuf, sous le saumon papillote.

En utilisant votre outil de versionning préféré, vous arrivez à récupérer le code du bloc que vous pouvez insérer sous celui du saumon, dans le document HTML :

```

1      <div class="clickable">
2          <h2 class="img-title">Boeuf romarin</h2>
3          
7      </div>

```

Maintenant, les deux recettes s'affichent au-dessus du bloc partenaire. C'est assez problématique, car ce n'est pas très esthétique.

Pour remédier à cela, vous avez l'idée d'englober ces deux éléments dans une div, et d'utiliser `grid` pour qu'elles s'affichent l'une au-dessus de l'autre, sur la même ligne que le bloc partenaire !



Grid'Cuisine

Entrées

Plats

Desserts



Saumon papillote



Boeuf romarin



Découvrez nos offres partenaires !

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi placerat vehicula mattis. Ut aliquet, sem vel rutrum posuere, augue nunc porttitor elit, quis aliquam velit odio id dolor.



Les crêpes de Laurent

Ingrédients

- 300g de farine
- 3 oeufs entiers
- 3 cuillères à soupe de sucre
- 2 cuillères à soupe d'huile
- 50g de beurre fondu
- 60cl de lait
- 5cl de rhum



Verrine fruits



Pizza italienne



Tiramisu fruité

Tous droits réservés et bon appétit !

Question 5

[solution n°12 p.40]

Pour finir, vous allez pouvoir ajouter la petite touche qui va rendre le style parfait. Vous alignez verticalement le contenu des éléments suivants :

- Le menu,
- Le footer,
- Les deux recettes "saumon" et "bœuf" pourraient être centrées par rapport à l'offre partenaire.

Vous n'oublierez pas d'augmenter l'espace entre ces deux recettes.



## Grid'Cuisine

Entrées

Plats

Déserts



Saumon papillote



Boeuf romarin



## Découvrez nos offres partenaires !

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi placerat vehicula mattis. Ut aliquet, sem vel rutrum posuere, augue nunc porttitor elit, quis aliquam velit odio id dolor.



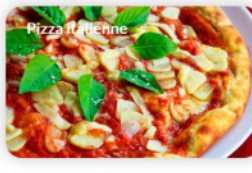
## Les crêpes de Laurent

## Ingrédients

- 300g de farine
- 3 oeufs entiers
- 3 cuillères à soupe de sucre
- 2 cuillères à soupe d'huile
- 50g de beurre fondu
- 60cl de lait
- 5cl de rhum



Verrine fruits



Pizza italieenne



Tiramisu fruité

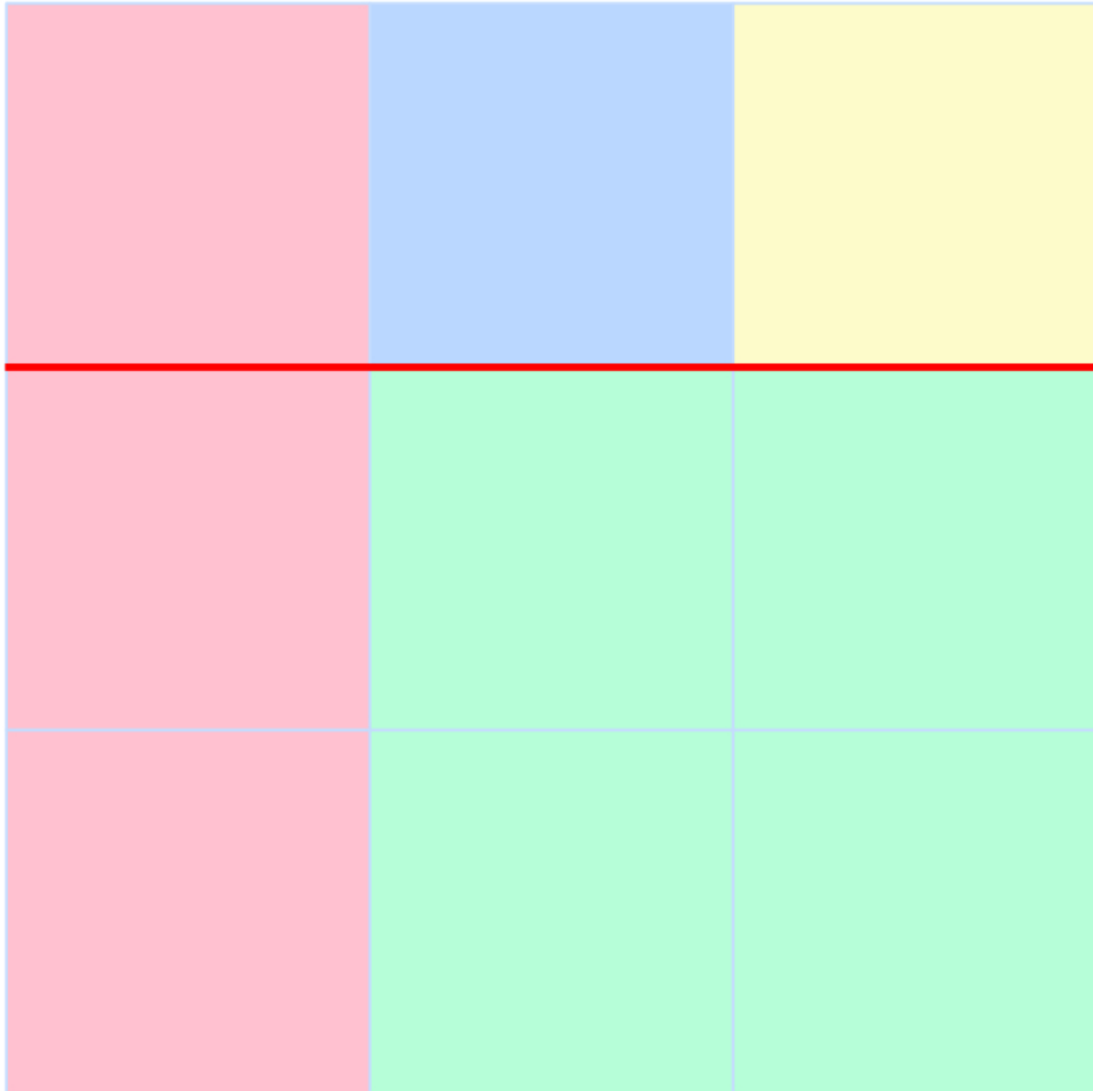
Tous droits réservés et bon appétit !

## Solutions des exercices



**Exercice p. 8 Solution n°1**

Faites correspondre les termes aux couleurs représentées sur la grille.



Zone verte	Zone rose	Zone bleue / jaune	Séparateur rouge
grid-area	grid-track	grid-cell	grid-line



- Le terme `grid-area` correspond à la zone en vert de la grille.
- Le terme `grid-track` correspond à la zone en rose de la grille, une track peut être verticale ou horizontale.
- Le terme `grid-line` correspond au séparateur horizontal en rouge.
- Le terme `grid-cell` correspond aux zones bleue et jaune de la grille.

**p. 14 Solution n°2**

```
1 .wrapper {
2   display: grid;
3   grid-template-columns: repeat(12, 1fr);
4   grid-template-rows: [start] 100px [top-separator] auto [bottom-separator] 100px [end];
5 }
```

**p. 15 Solution n°3**

```
1 .wrapper {
2   display: grid;
3   grid-template-columns: repeat(3, 200px);
4   grid-template-rows: repeat(3, 200px);
5   grid-template-areas:
6     "header header header"
7     "archives content ads"
8     "sponsors footer footer";
9 }
```

**p. 20 Solution n°4**

```
1 // Placement avec valeurs numériques
2 .content {
3   grid-column-start: 3;
4   grid-row-start: 2;
5   grid-row-end: 4;
6 }
7
8 // Placement par rapport aux séparateurs nommés
9 .content {
10  grid-column-start: separator-right;
11  grid-row-start: separator-top;
12  grid-row-end: row-end;
13 }
```

**p. 21 Solution n°5**

```
1 .header {
2   grid-area: header;
3 }
4
5 .content {
6   grid-area: content;
7 }
8
9 .footer {
10  grid-area: footer;
11 }
```

## p. 23 Solution n°6

```
1 .wrapper {  
2   display: grid;  
3   grid-template-columns: repeat(4, 100px);  
4   grid-template-rows: repeat(4, 100px);  
5   justify-items: end;  
6   align-items: center;  
7 }
```

## Exercice p. 24 Solution n°7

**Exercice**

CSS Grid est...

- ☒ Un outil qui permet de gérer le layout de la page
- ☐ Un système de tableau croisé dynamique porté pour le Web

**Exercice**

CSS Grid et flexbox...

- ☐ Sont redondants
- ☐ Sont deux outils qui n'ont pas du tout le même rôle
- ☒ Sont complémentaires

**Exercice**

L'élément de base sur lequel est définie la grille est appelé...

- ☐ Un grid-wrapper
- ☒ Un grid-container
- ☐ Un grid-base

**Exercice**

Combien de colonnes affichera la grille suivante ?

```
1 .wrapper {  
2   display: grid;  
3   grid-template-columns: repeat(12, 1fr);  
4 }
```

- ☒ 12
- ☐ 1
- ☐ Ce code n'ajoute pas de colonnes

**Exercice**L'unité `fr` correspond à...

- ☐ 100 px de la page, à peu près
- ☐ Un pourcentage de l'élément parent
- ☒ Une unité de taille restante

#### Exercice

Dans la syntaxe suivante :

```
1 grid-template-columns: [start] 100px [end];
```

À quoi font référence [start] et [end] ?

- ☐ Ils font partie intégrante de la syntaxe pour décrire la colonne
- ☒ Ce sont des séparateurs nommés
- ☐ Ce sont des colonnes nommées, équivalentes à des colonnes mesurant 1 fr

#### Exercice

La propriété `grid-area`...

- ☐ Permet de définir les zones dans la grille
- ☒ Permet de placer un élément sur une zone définie

#### Exercice

Par défaut, un élément placé dans la grille...

- ☐ Mesure toute la largeur, mais pas toute la hauteur de sa zone
- ☒ Mesure toute la hauteur et toute la largeur de sa zone
- ☐ Mesure toute la hauteur, mais pas toute la largeur de sa zone

#### Exercice

Par défaut, un élément placé en `display: grid`...

- ☒ Prend toute la largeur de son conteneur
- ☐ Prend toute la hauteur de son conteneur
- ☒ Empêche d'autres éléments de se placer à côté
- ☒ Permet de déclarer une nouvelle grille

#### Exercice

La valeur correcte de la propriété `display` pour afficher une grille qui partage certains comportements d'un élément `inline-block` est...

- ☒ `inline-grid`
- ☐ `grid-inline`
- ☐ Il n'est pas possible d'afficher une grille en ligne

```

1  body {
2      font-family: lato;
3      margin: 0;
4      display: grid;
5      grid-template-columns: 20vw 1fr;
6      grid-template-rows: 10vh 1fr 8vh;
7      grid-template-areas:
8          "header  navigation"
9          "content  content"
10         "footer   footer";
11  }
12  .navbar {
13      list-style: none;
14      display: grid;
15      grid-template-columns: 1fr 1fr 1fr;
16  }
17  header {
18      grid-area: header;
19  }
20  nav {
21      grid-area: navigation;
22      border-bottom: 1px solid;
23      display: grid;
24  }
25  .content {
26      margin: 3rem;
27      grid-area: content;
28      display: grid;
29      grid-template-columns: 1fr 1fr 1fr;
30      grid-template-rows: 0.5fr 0.5fr 0.5fr;
31  }
32  footer {
33      grid-area: footer;
34      text-align: center;
35      border-top: 1px solid;
36  }

```

## p. 29 Solution n°9

```

1  .content {
2      margin: 3rem;
3      grid-area: content;
4      display: grid;
5      gap: 30px;
6      grid-template-columns: 1fr 1fr 1fr;
7      grid-template-rows: 0.5fr 0.5fr 0.5fr;
8  }
9  .advert {
10     grid-column: 2/4;
11 }
12 .full-row {
13     grid-column: 1/4;
14 }

```

p. 31 Solution n°10

```
1 .full-row {
2   grid-column: 1/4;
3   display: grid;
4   grid-template-columns: 2fr 2fr;
5 }
```

p. 31 Solution n°11

```
1 .column {
2   display: grid;
3   grid-template-rows: 0.5fr 0.5fr;
4 }
```

p. 32 Solution n°12

Vous avez réussi à rendre le blog culinaire très moderne presque sans effort !

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Grid' Cuisine</title>
5     <meta charset="utf-8" />
6     <link
7       href="https://fonts.googleapis.com/css?family=Lato&display=swap"
8       rel="stylesheet"
9     />
10  </head>
11  <body>
12    <header>
13      <h1>Grid' Cuisine</h1>
14    </header>
15    <nav>
16      <ul class="navbar">
17        <li>Entrées</li>
18        <li>Plats</li>
19        <li>Desserts</li>
20      </ul>
21    </nav>
22    <div class="content">
23      <div class="column">
24        <div class="clickable">
25          <h2 class="img-title">
26            Saumon papillote
27          </h2>
28          
32        </div>
33
34        <div class="clickable">
35          <h2 class="img-title">Boeuf romarin</h2>
36          <img
37            src="https://cdn.pixabay.com/photo/2018/10/22/22/18/steak-3766548_1280.jpg"
```



```

38         class="block"
39     />
40 </div>
41 </div>
42 <div class="block advert clickable">
43     
47     <div style="margin: 2rem;">
48         <h2>Découvrez nos offres partenaires !</h2>
49         Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi
50         placerat vehicula mattis. Ut aliquet, sem vel rutrum posuere, augue
51         nunc porttitor elit, quis aliquam velit odio id dolor.
52     </div>
53 </div>
54 <div class="block clickable full-row">
55     
59     <div style="padding-left: 2rem;">
60         <h2>Les crêpes de Laurent</h2>
61         <h3>Ingrédients</h3>
62         <ul>
63             <li>300g de farine</li>
64             <li>3 oeufs entiers</li>
65             <li>3 cuillères à soupe de sucre</li>
66             <li>2 cuillères à soupe d'huile</li>
67             <li>50g de beurre fondu</li>
68             <li>60cl de lait</li>
69             <li>5cl de rhum</li>
70         </ul>
71     </div>
72 </div>
73 <div class="clickable">
74     <h2 class="img-title">Verrine fruits</h2>
75     
79 </div>
80
81 <div class="clickable">
82     <h2 class="img-title">Pizza italienne</h2>
83     
87 </div>
88
89 <div class="clickable">
90     <h2 class="img-title">Tiramisu fruité</h2>
91     

```

```

95     </div>
96 </div>
97 <footer>
98     Tous droits réservés et bon appétit !
99 </footer>
100 </body>
101 </html>
102
1     body {
2         font-family: lato;
3         margin: 0;
4         display: grid;
5         grid-template-columns: 20vw 1fr;
6         grid-template-rows: 10vh 1fr 8vh;
7         grid-template-areas:
8             "header  navigation"
9             "content  content"
10            "footer  footer";
11     }
12     img {
13         max-width: 100%;
14     }
15     h1 {
16         margin: 2rem;
17     }
18     .clickable:hover {
19         cursor: pointer;
20         opacity: 0.5;
21     }
22     .img-title {
23         position: absolute;
24         margin-left: 2rem;
25         color: white;
26     }
27     .advert {
28         grid-column: 2/4;
29     }
30     .navbar {
31         list-style: none;
32         display: grid;
33         grid-template-columns: 1fr 1fr 1fr;
34     }
35     header {
36         grid-area: header;
37     }
38     nav {
39         grid-area: navigation;
40         border-bottom: 1px solid;
41         display: grid;
42         align-items: center;
43     }
44     .content {
45         margin: 3rem;
46         grid-area: content;
47         display: grid;
48         gap: 30px;
49         grid-template-columns: 1fr 1fr 1fr;
50         grid-template-rows: 0.5fr 0.5fr 0.5fr;

```

```
51     }
52     footer {
53         grid-area: footer;
54         grid-column: 1/4;
55         grid-row: 3;
56
57         text-align: center;
58         border-top: 1px solid;
59         display: grid;
60         align-items: center;
61     }
62     .block {
63         border-radius: 25px;
64     }
65     .full-row {
66         grid-column: 1/4;
67         display: grid;
68         grid-template-columns: 2fr 2fr;
69     }
70     .column {
71         display: grid;
72         gap: 15px;
73         grid-template-rows: 0.5fr 0.5fr;
74         align-items: center;
75     }
```