

Mettre en page avec CSS GRID

Beaucoup de développeurs manipulent les mises en page de différentes façons parce qu'il y a tellement de façons différentes de positionner les éléments. Ces dernières années, cependant, l'accent a été mis sur la simplification de la rédaction des feuilles de style en cascade.

L'un de ces développements récents en CSS est appelé CSS Grid. Il s'agit d'un ensemble de propriétés CSS qui permettent de créer des mises en page bidimensionnelles définies par des colonnes et des lignes.

Au lieu d'avoir à définir manuellement la position d'un élément à l'aide de distances de pixels et de calculs mathématiques compliqués (comme c'était le cas auparavant !), vous pouvez simplement définir le nombre de colonnes et de lignes que vous voulez et laisser Grid les positionner pour vous.

La plupart des sites Web sont, après tout, basés sur des grilles. Ce n'est peut-être pas évident à première vue, mais les sites Web ont souvent des colonnes et des lignes qui déterminent comment les éléments sont distribués sur la page.

CSS Grid permet de créer ce même type de mise en page structurée dans vos propres projets. Pour avoir une idée de ce que CSS Grid vous permet de créer, jetez un coup d'oeil aux mises en page suivantes, qui ont toutes été écrites avec quelques lignes de code simples :

1	2	3
4	5	6
7	8	

Éléments en colonnes et lignes simples

1	2	3
4	5	6
7	8	

Éléments en colonnes et en lignes avec des espaces entre chaque élément

1		
2	3	4
	5	
6		7
8		

Éléments de différentes tailles en colonnes et en lignes

Comme vous pouvez le voir, CSS Grid est très puissant. Il peut également être utilisé en combinaison avec les concepts que vous avez déjà vus dans ce cours, comme les marges, flexbox, positionnement....

Compatibilité navigateur

La seule raison pour laquelle quelqu'un pourrait hésiter à utiliser CSS Grid est que, comme beaucoup de nouvelles fonctionnalités, il était incompatible avec certains navigateurs. Cependant, CSS Grid est maintenant intégré dans tous les navigateurs, vous pouvez donc l'utiliser avec certitude.

Il y a beaucoup de bons conseils à ce sujet sur le [Mozilla Developer Network \(MDN\)](#).

Voici un excellent extrait de l'article de MDN :

Comme pour tout choix de technologie front-end, la décision d'utiliser la mise en page CSS Grid Layout dépendra des navigateurs que les visiteurs de votre site utilisent habituellement. S'ils ont tendance à utiliser des versions à jour de Firefox, Chrome, Opera et Safari, alors il serait logique de commencer à utiliser les grilles CSS une fois que ces navigateurs sont à jour. Si votre site dessert un secteur du marché qui est lié à des navigateurs plus anciens, cependant, cela n'a peut-être pas encore de sens.

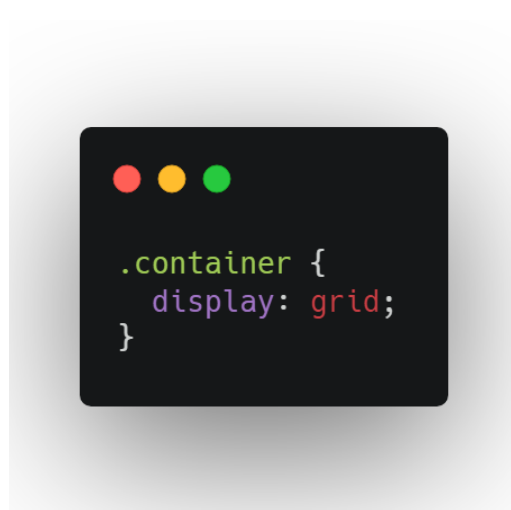
Je suis d'accord avec ce résumé sur la façon de prendre une décision concernant les mises en page en CSS. N'oubliez pas, cependant, que vous pouvez écrire du code CSS Grid et du code de compatibilité pour les anciens navigateurs dans la même base de code. Ils ne s'excluent pas mutuellement !

Sans plus attendre, nous verrons comment intégrer CSS Grid dans votre projet avec une ligne de code dans le chapitre suivant.

Mettre en place une grille de base

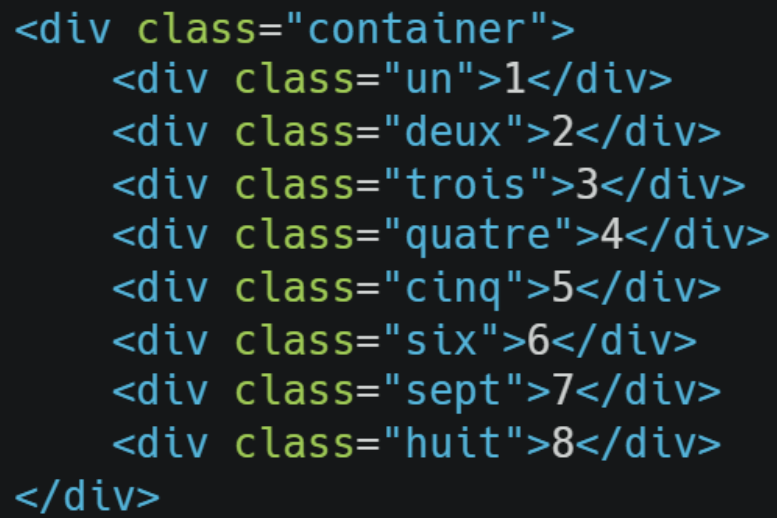
Pour créer des mises en page à l'aide de CSS Grid, il suffit d'ajouter une ligne de code à l'élément où vous voulez créer une grille !

Un type d'affichage de grille est généralement ajouté aux éléments de conteneur, c'est-à-dire un élément qui contient d'autres éléments. Il peut s'agir d'une section ou d'une division, par exemple. Imaginons que nous ayons un div avec une classe de "container" :



Dans les chapitres suivants, nous utiliserons ce même exemple pour explorer la puissance de la grille CSS. Disons que notre div, avec une classe de conteneur, contient huit autres divs, dont chacun contient un nombre :

HTML



```
<div class="container">
  <div class="un">1</div>
  <div class="deux">2</div>
  <div class="trois">3</div>
  <div class="quatre">4</div>
  <div class="cinq">5</div>
  <div class="six">6</div>
  <div class="sept">7</div>
  <div class="huit">8</div>
</div>
```

CSS



```
.container .un,  
.container .six {  
  background-color: #B8336A;  
}
```

```
.container .deux,  
.container .sept {  
  background-color: #726DA8;  
}
```

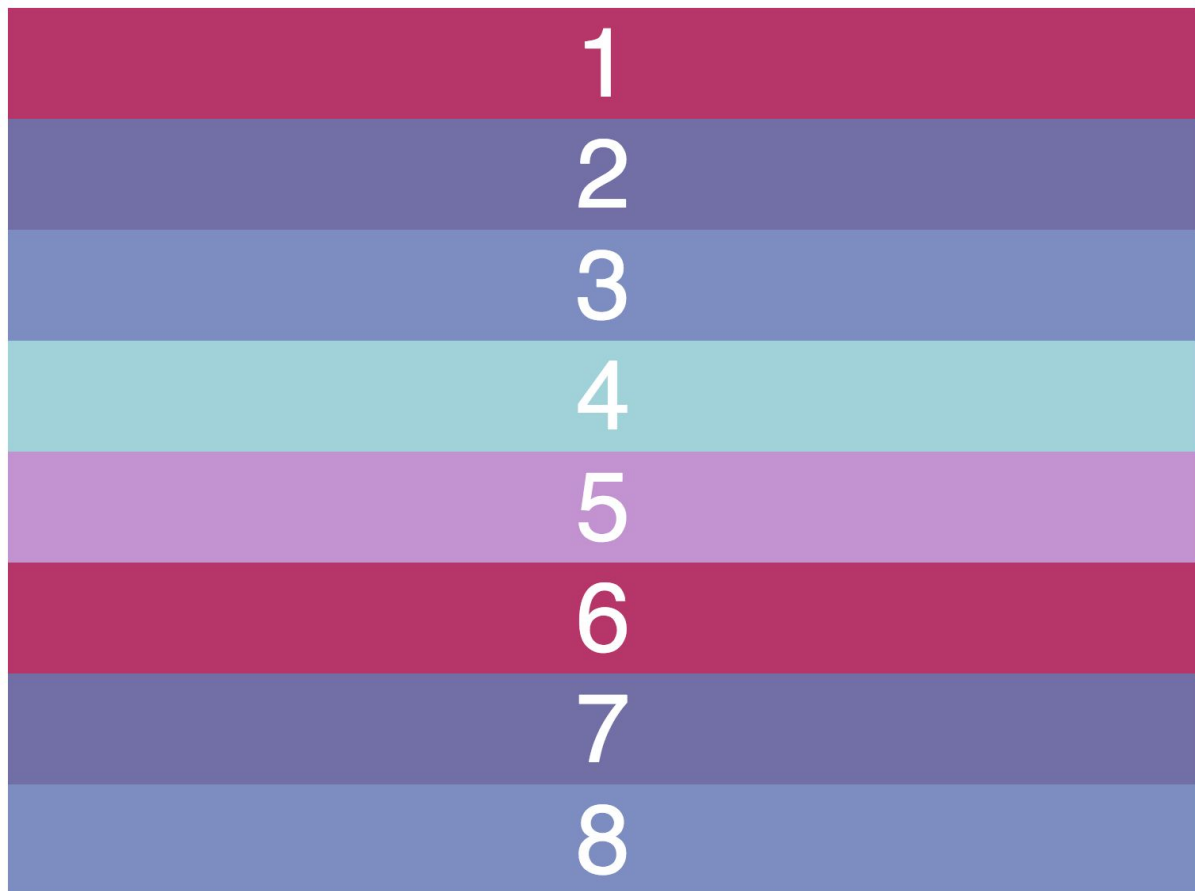
```
.container div.trois,  
div.huit {  
  background-color: #7D8CC4;  
}
```

```
.container .quatre {  
  background-color: #A0D2DB;  
}
```

```
.container div.cinq {  
  background-color: #C490D1;  
}
```

```
.container .extra {  
  background-color: #ff0000;  
}
```

Le résultat est un groupe de blocs qui permet de voir facilement la structure créée par les différentes configurations de grille :



Mise en place d'une grille de base

Comme nous l'avons dans l'introduction la première étape pour utiliser la grille CSS est de définir la propriété d'affichage d'un élément de conteneur sur grille `display: grid;`

Il n'y aura aucun changement immédiat avec cette ligne de code. Aucune grille magique n'apparaît ! Il est maintenant temps de définir nous-mêmes une grille personnalisée.

Pensez au moment où vous devez configurer une feuille de calcul ou un tableau. Vous pensez souvent au nombre de colonnes et de lignes dont vous aurez besoin pour vos données, comme "J'ai besoin de 2 colonnes, une pour les dates et une pour les montants. J'ai besoin de 25 rangées

parce qu'il y a 25 dépenses à ajouter." Vous devrez compter de la même manière ici en termes de colonnes et de lignes !.


Définissez les colonnes et les lignes de votre grille en utilisant deux propriétés :

- **grid-template-columns** vous permet de définir le nombre de colonnes et chacune de leurs largeurs.
- **grid-template-rows** vous permet de définir le nombre de lignes et chacune de leurs largeurs.

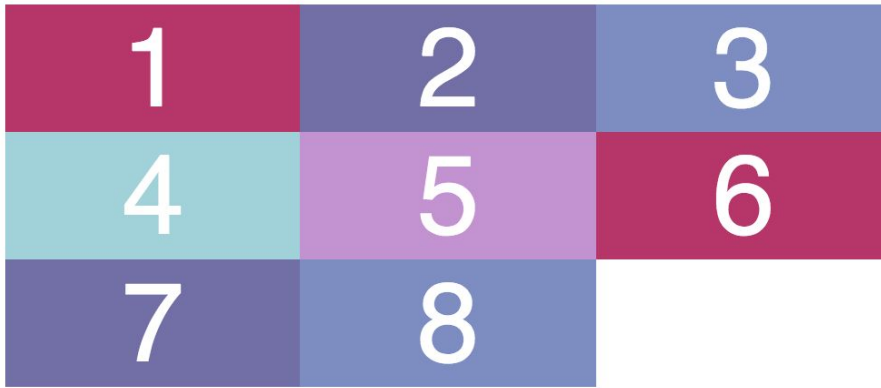
Commençons par les colonnes.

Définition des colonnes

Définissons 3 colonnes en utilisant la propriété `grid-template-columns`. Chaque colonne a une largeur de 20 %. Le 20% est relatif à la largeur de l'élément contenant : dans ce cas, le conteneur `div`.



```
.container {  
  display: grid;  
  grid-template-columns: 20% 20% 20%;  
}
```

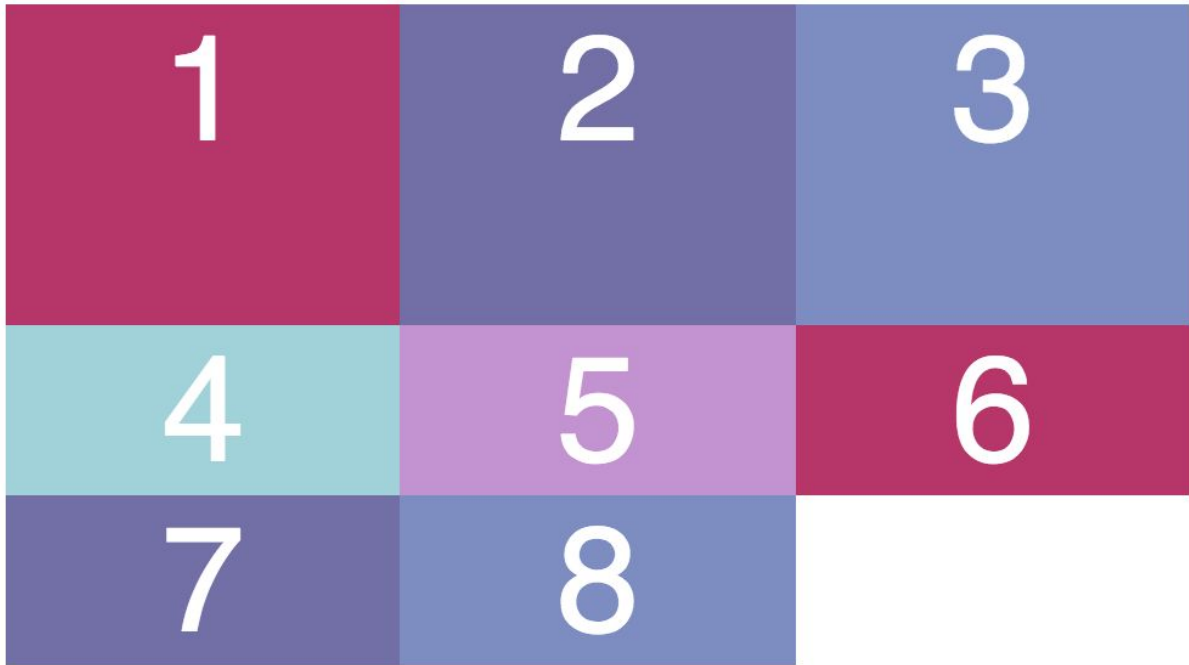



Remarquez qu'on ne dit pas explicitement "Je veux 3 colonnes !" Le nombre de colonnes est défini en fonction du nombre de largeurs que vous avez spécifié. Si vous spécifiez 3 largeurs de colonnes, vous obtiendrez 3 colonnes. Si vous spécifiez 10 largeurs de colonnes, vous obtiendrez 10 colonnes.

Définition des lignes

Ajoutons maintenant des lignes en utilisant la propriété `grid-template-rows` :

```
.container {  
  display: grid;  
  grid-template-columns: 20% 20% 20%;  
  grid-template-rows: 3em 1.6em 1.6em;  
}
```



Nous avons maintenant 3 rangées : la première est haute de 3em, la deuxième est haute de 1.6em, et la troisième est haute de 1.6em.

Et si on ajoutait d'autres blocs ?

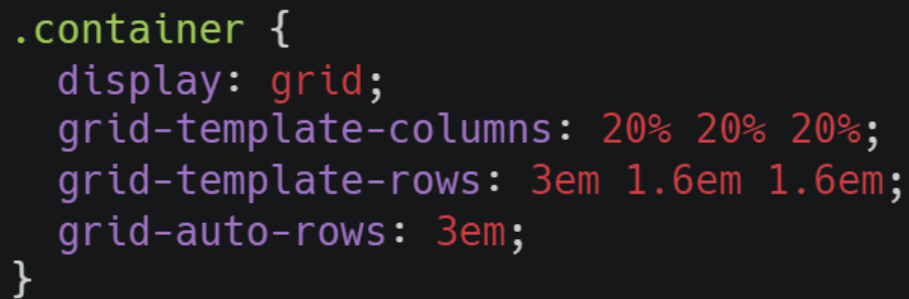
Une quatrième ligne serait créée, et vous auriez à modifier votre code CSS pour ajouter une hauteur pour cette nouvelle ligne ! Il est parfois difficile de savoir combien de rangées vous aurez à l'avance. Le réglage des valeurs ligne par ligne est très restrictif et inflexible.

C'est pourquoi vous pouvez définir une valeur de ligne par défaut qui s'appliquera aux numéros de ligne que vous n'avez pas spécifiés. Cette propriété s'appelle `grid-auto-rows`.

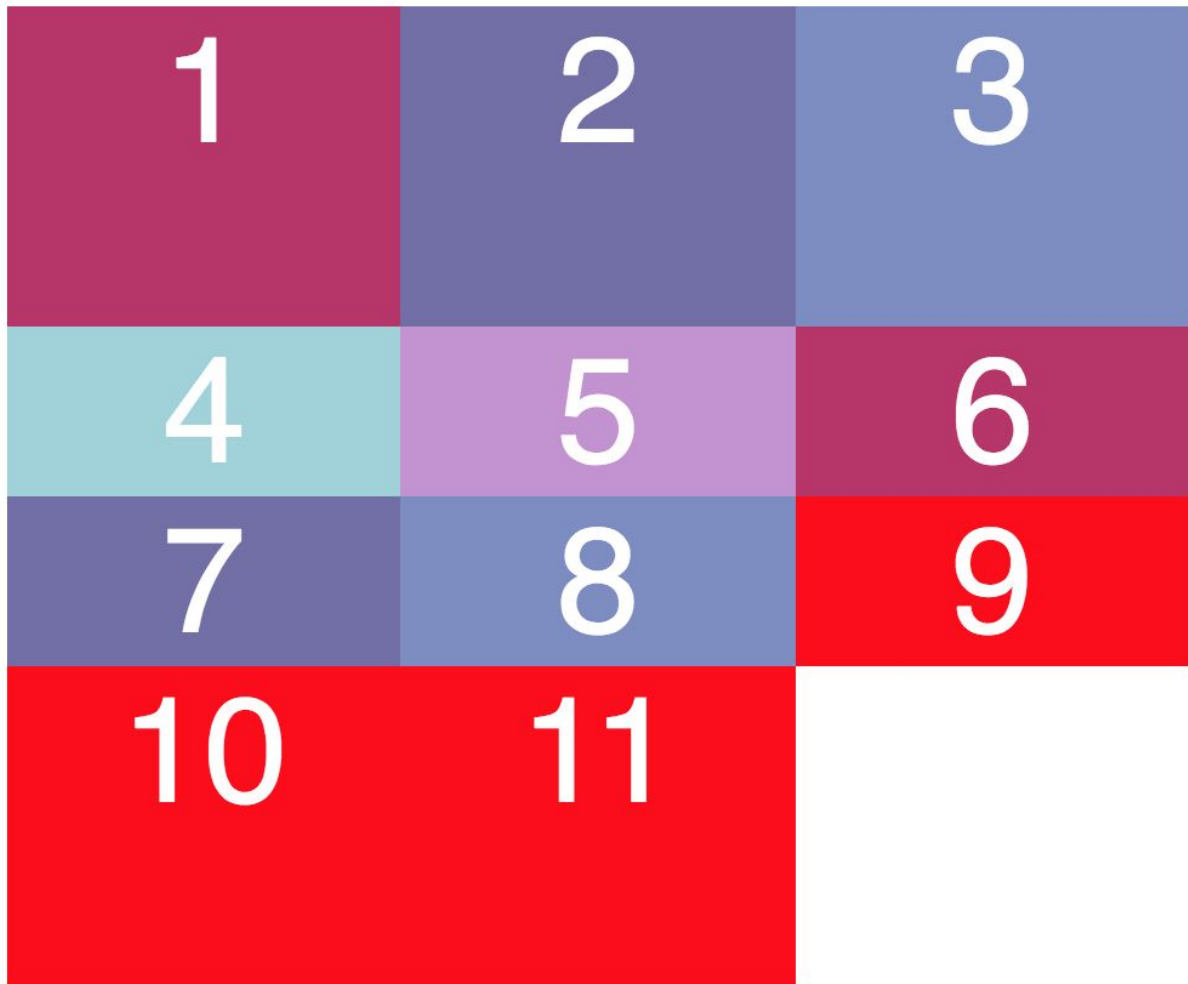
Vous pouvez définir la même chose pour les colonnes, mais comme les colonnes sont généralement liées aux largeurs de page (qui sont plus prévisibles et moins flexibles), vous n'aurez peut-être pas besoin de définir des largeurs de colonne par défaut supplémentaires très souvent.

Maintenant, quand j'ajoute 3 blocs supplémentaires (en rouge vif pour vous aider à voir la différence soit la couleur `#ff0000`), remarquez que la quatrième ligne qui est créée a une hauteur par défaut de 3em. Même si

j'ajoutais assez de contenu pour remplir 20 nouvelles lignes, chaque nouvelle ligne aurait une hauteur de 3em ! Les autres rangées conservent cependant leur hauteur spécifiée (1.6em).



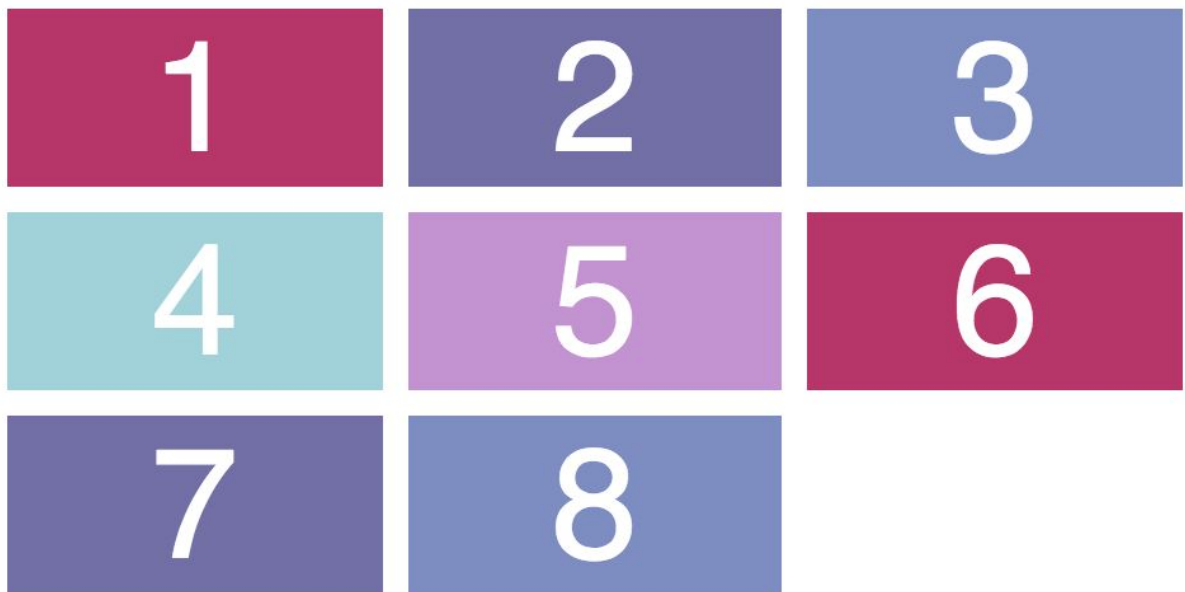
```
.container {  
  display: grid;  
  grid-template-columns: 20% 20% 20%;  
  grid-template-rows: 3em 1.6em 1.6em;  
  grid-auto-rows: 3em;  
}
```



Espace entre les éléments de grille

Afin de définir les espaces entre les éléments de votre grille, utilisez une propriété simple appelée *grid-gap*. Définissez une valeur si vous voulez que l'écart soit le même entre les lignes et les colonnes. Sinon, définissez deux valeurs (le premier pour l'écart entre les lignes, le second pour l'écart entre les colonnes).

```
.container {  
  display: grid;  
  grid-template-columns: 20% 20% 20%;  
  grid-auto-rows: 1.6em;  
  grid-gap: 10px;  
}
```



Mesures de colonnes et de lignes

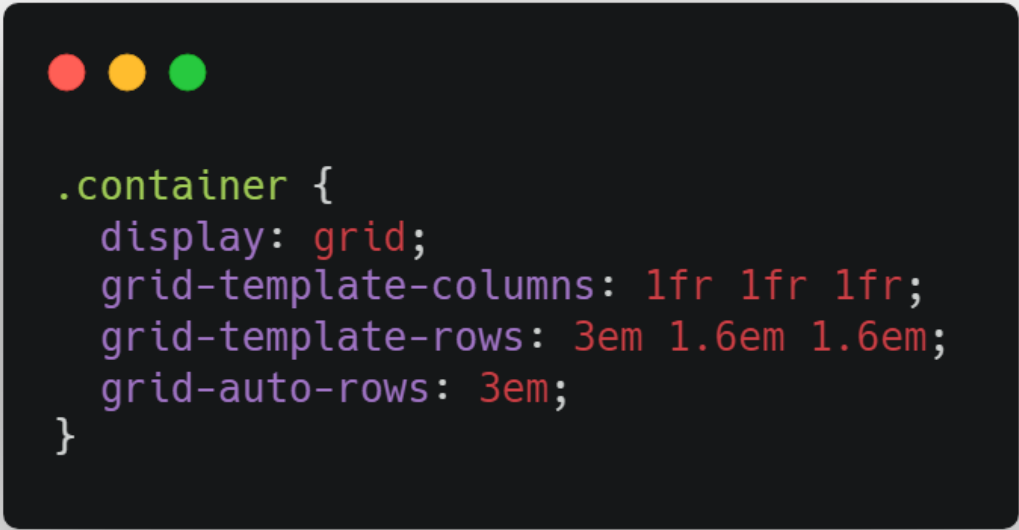
Vous pouvez définir des valeurs de colonnes et de lignes en utilisant les mêmes unités que vous avez vues jusqu'à présent dans ce cours comme em/rem, pourcentages et pixels.

Cependant, il y a une autre grande unité qui vous aidera à régler les mesures de la grille. On l'appelle "unité de fractionnement", ou `fr` pour faire court. C'est semblable aux pourcentages, mais plus souple.

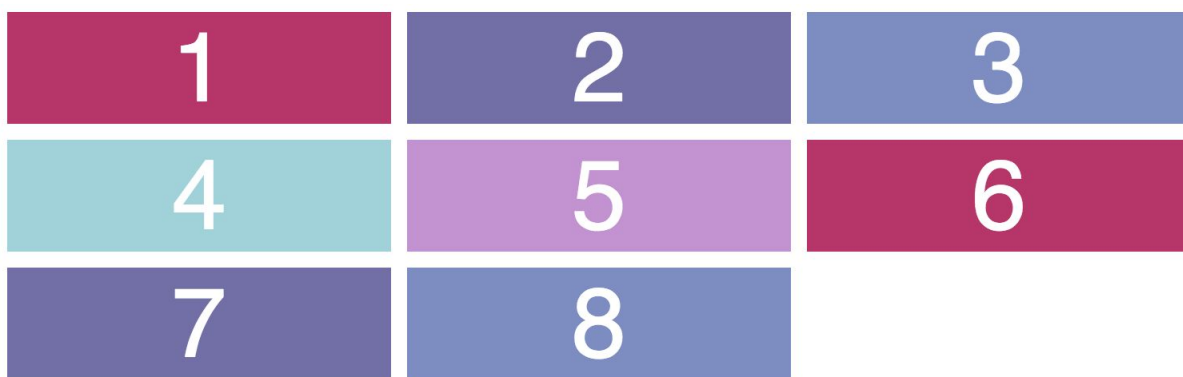
Les pourcentages peuvent se compliquer lorsqu'ils sont combinés avec les écarts de la grille - `grid-gap` -. Si vous avez 2 colonnes, chacune avec une largeur de 50%, vous vous attendriez à ce que cela occupe 100% de la fenêtre. Cependant, si vous ajoutez un espace de grille, votre conteneur s'étendra au-delà de la fenêtre d'observation. Les pourcentages ne sont pas recalculés pour tenir compte de l'écart.

Ceci peut être évité en utilisant des unités de fraction au lieu de pourcentages.

Prenons un exemple de colonne. La largeur d'une colonne "1fr" dépend de la largeur de l'élément contenant, comme pour les pourcentages. Vous n'avez pas besoin de faire beaucoup de maths, cependant ! Si vous déclarez 1fr 3 fois, vous aurez 3 colonnes de même largeur :



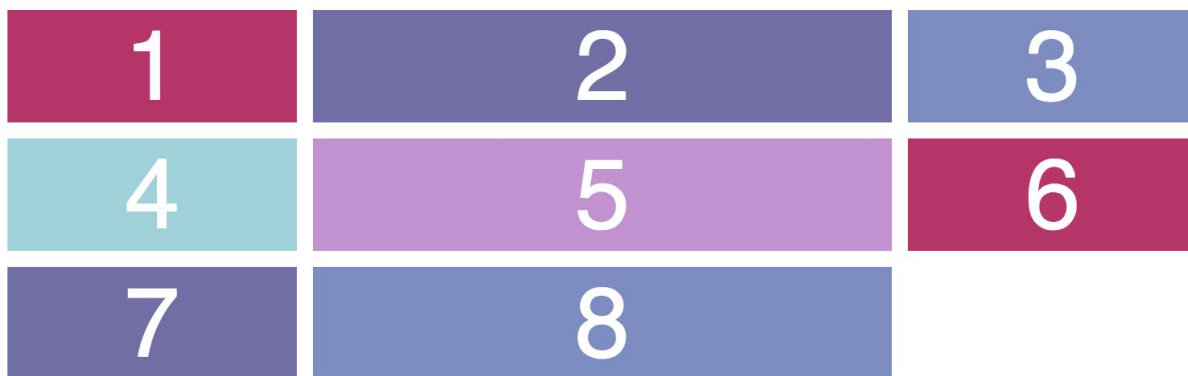
```
.container {  
  display: grid;  
  grid-template-columns: 1fr 1fr 1fr;  
  grid-template-rows: 3em 1.6em 1.6em;  
  grid-auto-rows: 3em;  
}
```



Vous pouvez même définir des mesures différentes pour chaque colonne. Supposons que vous vouliez que la colonne du milieu soit deux fois plus large que les deux autres. Laissez les unités de fraction manipuler les maths pour vous !



```
.container {  
  display: grid;  
  grid-template-columns: 1fr 2fr 1fr;  
  grid-template-rows: 3em 1.6em 1.6em;  
  grid-auto-rows: 3em;  
}
```

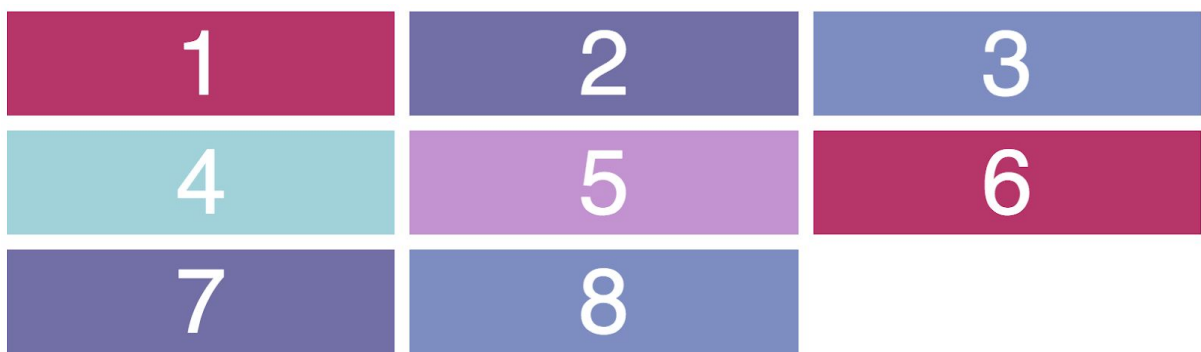


Les colonnes et les lignes sont la base de toute grille. Maintenant que vous savez comment régler leur largeur et leur hauteur.

Définir les colonnes et les lignes de façon simplifiée

Dans le dernier chapitre, vous avez appris comment définir la taille des colonnes et des lignes pour vos grilles. Bien que nous n'ayons travaillé jusqu'à présent qu'avec des carrés de couleurs différentes, les mêmes principes peuvent être appliqués à de vrais sites Web pour organiser le contenu !

Nous continuerons à travailler avec ces carrés afin de nous concentrer sur les concepts de grille eux-mêmes. Nous allons revenir à la grille suivante :



```
.container {  
  display: grid;  
  grid-template-columns: 1fr 1fr 1fr;  
  grid-auto-rows: 1.6em;  
  grid-gap: 10px;  
}
```

Nous allons voir deux façons abrégées d'écrire le code ci-dessus. Le résultat sera le même !

La fonction de répétition

Lorsque vous définissez des grilles, vous aurez souvent des valeurs qui se répètent. Dans notre exemple ci-dessus, il y a 3 colonnes qui ont chacune 1fr de largeur. Vous pouvez créer les 3 colonnes en spécifiant chacune de leurs largeurs une par une, comme vous l'avez déjà vu, ou vous pouvez utiliser une fonction CSS appelée `repeat` pour créer un motif de répétition.

Pour utiliser la fonction `repeat`, vous devez spécifier :

- *le nombre de fois que vous voulez que le motif de mesure se répète*
- *la ou les mesures en pixels, em, pourcentages, fr...quelle que soit l'unité que vous ressentez*

Dans notre exemple, nous voulons 3 colonnes, chacune d'une largeur de 1fr.

Par conséquent,

```
grid-template-columns: 1fr 1fr 1fr;
```

peut aussi s'écrire comme :

```
grid-template-columns: repeat (3, 1fr);
```

Traduit en langage courant, il se lit comme suit : "Répétez 3 fois une mesure de 1fr." Le résultat visuel est exactement le même.

En utilisant la répétition, vous pouvez même créer des motifs répétitifs avec plusieurs mesures ! Pour créer 2 colonnes, l'une d'une largeur de `0,5 fr` et l'autre d'une largeur de `1 fr`, puis pour répéter ce motif deux fois, vous pouvez écrire la ligne de code suivante :

```
grid-template-columns: repeat(2, 0.5fr 1fr);
```

Traduit en langage courant, il se lit comme suit : "Répéter 2 fois un motif avec 0.5fr et 1fr colonnes."

Le résultat est :

- La colonne 1 est large de 0.5fr.
- La colonne 2 est large de 1fr.
- La colonne 3 est large de 0.5fr.
- La colonne 4 est large de 1fr.

Notez qu'il n'y a qu'une virgule entre le nombre de fois qu'un motif doit se répéter et les mesures. Il n'y a pas de virgule entre les mesures elles-mêmes.

Ecrire 0.5fr 1fr **et surtout PAS** 0.5fr, 1fr

Caractéristiques des lignes et des colonnes en abrégé

Vous avez déjà défini des mesures de grille à l'aide de lignes de code distinctes pour décrire les lignes et les colonnes.

Voici notre exemple écrit de deux façons différentes (l'une avec la fonction de répétition et l'autre sans) :

```
.container {
  grid-template-columns: 1fr 1fr 1fr;
  grid-template-rows: 1.6em 1.6em 1.6em;
  /* ou en utilisant la fonction repeat:*/
  grid-template-columns: repeat(3, 1fr);
  grid-template-rows: repeat(3, 1.6em);
}
```

Les mesures séparées des colonnes et des lignes peuvent être combinées en une seule ligne de code. La nouvelle propriété s'appelle simplement `grid-template` *-les mêmes que les propriétés que vous avez déjà apprises, mais sans -column ou row- à la fin !-*

Dans la propriété `grid-template`, vous allez :

- spécifier vos mesures de ligne
- tapez une barre oblique
- spécifier les dimensions de votre colonne

Vous devez fournir les spécifications dans cet ordre !

Appliquons cette syntaxe abrégée à notre exemple de grille multicolore :



```
.container {  
  grid-template: 1.6em 1.6em 1.6em / 1fr 1fr 1fr;  
  /* ou en utilisant la fonction repeat:*/  
  grid-template: repeat(3, 1.6em) / repeat(3, 1fr);  
}
```

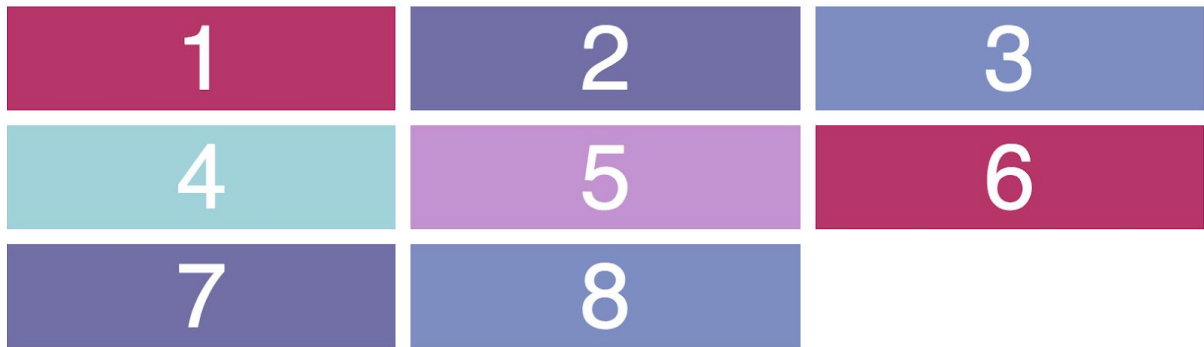
Le résultat est le même, sauf que vous avez utilisé une ligne de code au lieu de deux. Efficacité !

N'oubliez pas de créer une hauteur de ligne par défaut séparément à l'aide de la propriété `grid-auto-rows` si vous pensez que des lignes supplémentaires pourraient apparaître un jour. Ceci ne peut pas être combiné dans la notation abrégée du modèle de grille et doit être écrit sur une ligne séparée.

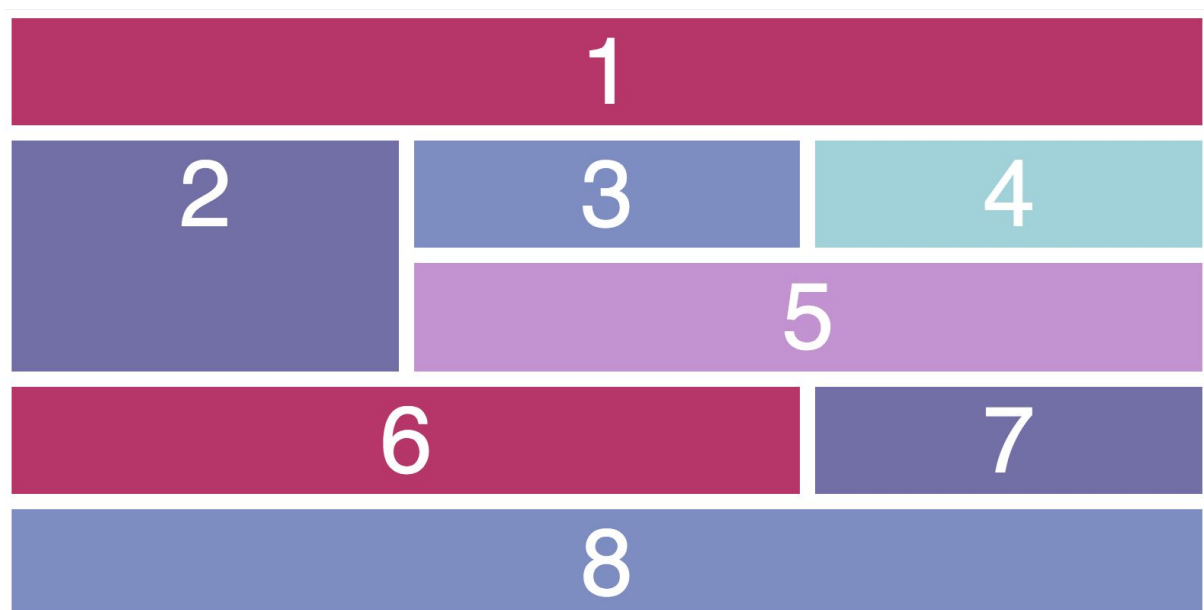
Définir la hauteur et la largeur des éléments de la grille

Jusqu'à présent, nous n'avons parlé que de l'établissement des mesures pour votre grille globale (comme la largeur de chaque colonne ou ligne).

Nous n'avons pas parlé de la façon de contrôler le dimensionnement des éléments individuels d'une grille. Nous n'avons travaillé qu'avec des éléments de taille égale :



Comment pourrions-nous utiliser CSS Grid pour obtenir un résultat où certains éléments prennent plus d'une colonne ou ligne, comme ceci ?



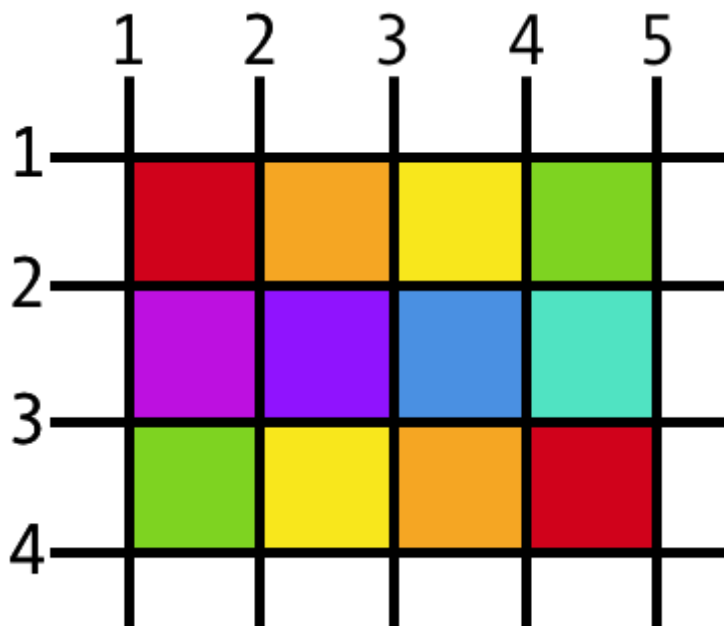
Remarquez comment la largeur de la grille est la même, mais le bloc "1" occupe 3 colonnes, le bloc "2" occupe 2 lignes et 2 colonnes, etc ?

Dans ce chapitre, nous verrons comment, en utilisant les propriétés `grid-column` et `grid-rows`.

Colonnes et lignes de la grille

Chaque grille contient des éléments. Cependant, il y a aussi des "lignes" entre chaque élément de la grille. Considérez le papier cadrillé: il y a des carrés qui sont marqués par des lignes verticales et horizontales.

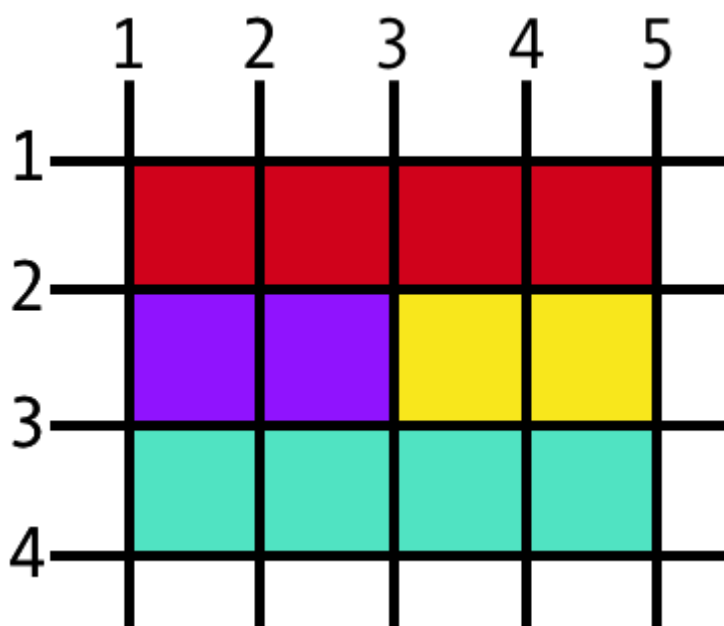
Dans l'illustration ci-dessous, vous pouvez voir ces lignes représentées (les carrés colorés sont les éléments eux-mêmes) :



Lignes de grille entre les éléments

C'est la représentation la plus simple d'une grille. C'est juste un tas de carrés entre des lignes régulièrement espacées, chacune d'elles étant marquée d'un chiffre.

Vous pourriez aussi avoir les mêmes lignes d'ensemble mais avec des éléments qui prennent plus de place. Dans l'image ci-dessous, voyez-vous la section rouge qui s'étend sur la ligne 1-5, ou la section jaune qui chevauche les lignes verticales 3-5 ?



Lignes de grille entre les éléments

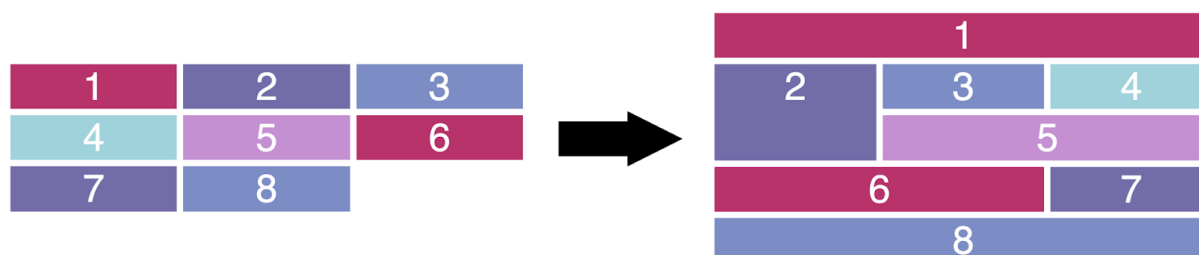
Les lignes de la grille restent les mêmes, mais les éléments occupent des espaces différents entre eux.

Régler les mesures des éléments

Pour définir sur quelles lignes de grille un élément doit commencer et se terminer, utilisez les propriétés suivantes :

- `grid-column-start`
- `grid-column-end`
- `grid-row-start`
- `grid-row-end`

Regardez l'image ci-dessous. Le code ci-dessus nous donne le résultat à gauche, et nous voulons le résultat à droite :



Résultat souhaité à droite

J'ai marqué le résultat du départ avec des numéros de ligne comme vous l'avez vu dans les images au début du chapitre. Ils nous aideront à définir où les éléments doivent commencer et se terminer :

	1	2	3	4
1	1	2	3	
2	4	5	6	
3	7	8		
4				

Dimensions des colonnes

Pour que le premier élément occupe la distance entre la ligne de colonne 1 et la ligne de colonne 4 (toute la largeur de la grille), nous devons spécifier qu'il doit commencer à la colonne 1 et se terminer à la colonne 4 :



Vous pouvez écrire ceci sur une ligne de code, avec la propriété `grid-column`, en séparant les lignes par une barre oblique.



Soustrayez le premier chiffre du deuxième et vous obtiendrez le nombre de colonnes que l'élément occupe, $4 - 1 = 3$, donc cet élément occupe 3 colonnes.

Le résultat serait le même ! Les autres éléments se réorganisent automatiquement autour de ces nouvelles mesures.

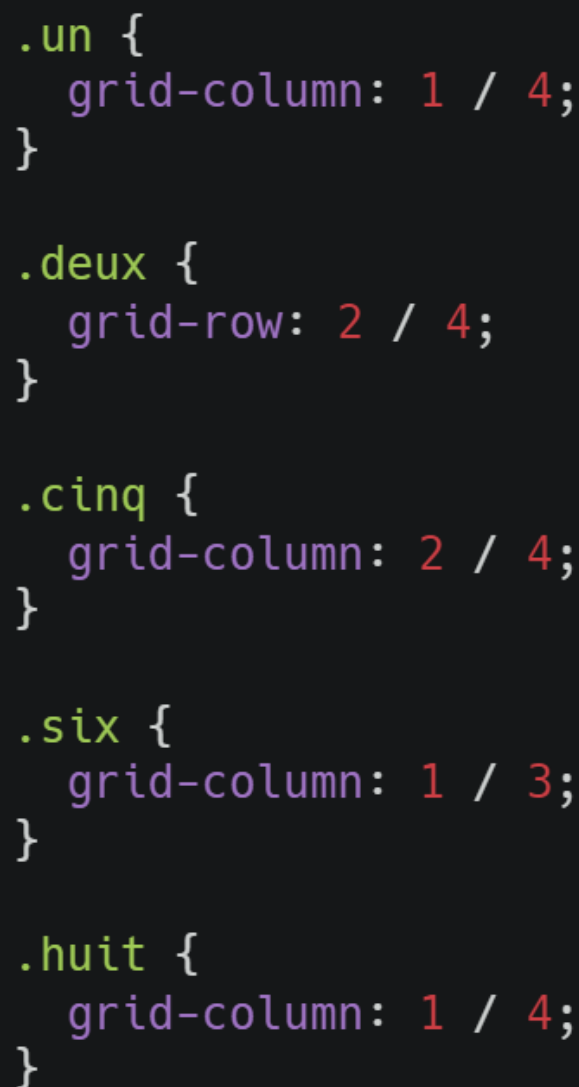
Mesures de rangées

Maintenant, nous voulons que le deuxième élément prenne deux rangées de hauteur. Puisqu'il commence maintenant à la ligne 2 (le premier élément l'a fait descendre), nous pouvons spécifier qu'il doit commencer à la ligne 2 et se terminer à la ligne 4 :



Soustrayez le premier chiffre du deuxième et vous obtiendrez le nombre de lignes que l'élément occupe. $4 - 2 = 2$, donc cet élément occupe 2 rangées.


Voici le code final de la grille et de la colonne qui produit le résultat que nous voulions obtenir plus haut. Prenez votre temps de parcourir le code et de dessiner, à la main, ce que vous pensez que le résultat sera ! Ça aide beaucoup de procéder ainsi.



```
.un {  
  grid-column: 1 / 4;  
}  
  
.deux {  
  grid-row: 2 / 4;  
}  
  
.cinq {  
  grid-column: 2 / 4;  
}  
  
.six {  
  grid-column: 1 / 3;  
}  
  
.huit {  
  grid-column: 1 / 4;  
}
```

Span et valeurs alternatives


Jusqu'à présent, nous n'avons utilisé que des valeurs numériques pour définir la taille des éléments le long des lignes de colonne et de ligne. Cependant vous pouvez également définir une plage de représentation simple :

A code editor window with a dark background and three colored window control buttons (red, yellow, green) in the top left corner. It contains CSS code for a class named .un.

```
.un {  
  grid-column: 1 / span 3;  
}
```

Ce code dit : "L'élément doit commencer à la ligne de colonne 1 et s'étendre sur 3 colonnes." C'est le même résultat que celui que vous avez vu ci-dessus - `grid-column : 1 / 4 -`.

Enfin, il est parfois difficile de savoir combien de colonnes ou de lignes vous aurez, surtout si votre contenu change tout le temps. Vous pouvez également utiliser des valeurs numériques négatives pour vous assurer que votre élément couvre toujours le nombre total de lignes, par exemple. La dernière ligne de ligne ou de colonne sera toujours la ligne "-1".

A code editor window with a dark background and three colored window control buttons (red, yellow, green) in the top left corner. It contains CSS code for a class named .un.

```
.un {  
  grid-column: 1 / -1;  
}
```

Ce code dit : "L'élément doit commencer à la ligne de colonne 1 et se terminer à la dernière ligne de colonne." Pour notre exemple, c'est le même résultat que celui que vous avez vu ci-dessus avec `grid-column: 1 / 4;` ou `grid-column: 1 / span 3;`.

Créer une zone de gabarit de grille

Dans ce chapitre, nous examinerons une autre façon de définir le nombre de colonnes ou de lignes qu'un élément occupe.

Cette nouvelle méthode sera plus visuelle et prendra en compte l'ensemble d'une mise en page à la fois (au lieu de travailler élément par élément).

Prenez cette disposition très générale :



Voici la mise en page balisée de façon beaucoup plus détaillée, y compris les lignes de colonnes et les lettres blanches qui représentent quelle section est laquelle.

	1	2	3	4	5
1	h	h	h	h	
2	a	a	b	b	
3	f	f	f	f	
4					

Considérez cela :

- h = header
- a = section A
- b = section B
- f = footer


Si la grille (4 colonnes, 3 lignes) était écrite en texte, avec chaque lettre représentant l'élément qui prend une position particulière, elle ressemblerait à ceci :

```
h h h h
a a b b
f f f f
```

Et en fait, c'est EXACTEMENT ce que nous allons mettre dans notre code !

Zones de gabarits de grille

Voici le HTML de début pour l'exemple de ce chapitre :



```
<div class="container">
  <header>Header</header>
  <section id="a">Section A</section>
  <section id="b">Section B</section>
  <footer>Footer</footer>
</div>
```

Avant de commencer, voici les étapes générales que nous allons suivre :

- Définissez le nombre de colonnes et de lignes, ainsi que leurs mesures, avec `grid-template` (ou séparément avec `grid-template-column` et `grid-template-row`).
- Définissez un modèle à l'aide de lettres, de mots ou de chiffres de votre choix dans les zones de la grille, en mettant un espace entre chaque ligne et en mettant chaque ligne entre guillemets.
- Associez les éléments à ces lettres, mots ou chiffres de votre choix en définissant une grille `grid-area` sur chaque élément.
- Tout d'abord, dans la feuille de style CSS ci-jointe, définissez un `grid-template` avec le nombre de colonnes et de lignes que vous voulez et leurs mesures.



```
.container {
  display: grid;
  grid-template: repeat(3, 1fr) / repeat(4, 1fr);
}
```

Ensuite, créez une mise en page à l'aide de zones de grille, qui contiendra une représentation codée de l'emplacement de vos éléments dans une grille, en mettant chaque ligne entre guillemets dans un ensemble de guillemets :

```
.container {  
  display: grid;  
  grid-template: repeat(3, 1fr) / repeat(4, 1fr);  
  grid-template-areas:  
    "h h h h"  
    "a a b b"  
    "f f f f"  
  ;  
}
```

Ensuite, affectez chaque lettre de votre modèle de zone de grille à un élément de votre grille à l'aide de la propriété `grid-area`. Par exemple, pour affecter l'élément d'en-tête à toutes les instances de la lettre "h" dans votre modèle :

```
header {  
  grid-area: h;  
}
```

Voici le CSS complet (incluant les couleurs et les couleurs de fond) pour obtenir le résultat final sous le code :



```
body {
  font-family: Helvetica;
  font-size: 1.4em;
  color: #fff;
  text-align: center;
}

.container {
  display: grid;
  grid-template: repeat(3, 1fr) / repeat(4, 1fr);
  grid-template-areas:
    "h h h h"
    "a a b b"
    "f f f f"
  ;
}

header {
  grid-area: h;
  background-color: #D0021B;
}

section#a {
  grid-area: a;
  background-color: #9013FE;
}

section#b {
  grid-area: b;
  background-color: #F8E71C;
}

footer {
  grid-area: f;
  background-color: #50E3C2;
}
```



En résumé :

- Définissez le nombre de colonnes et de lignes, ainsi que leurs mesures, dans le gabarit de grille (ou séparément dans le gabarit de grille - colonnes et gabarit - lignes).
- Définissez un modèle à l'aide de lettres, de mots ou de chiffres de votre choix dans les zones de la grille, en mettant un espace entre chaque ligne et en mettant chaque ligne entre guillemets.
- Associez les éléments à ces lettres, mots ou chiffres de votre choix en définissant une grille sur chaque élément.

Comparaison avec d'autres mesures

Dans le chapitre précédent du cours, vous avez appris comment régler les mesures en utilisant :

- `grid-column-start`
- `grid-column-end`
- `grid-row-start`
- `grid-row-end`

Cette façon de régler les mesures nécessite des calculs mathématiques légers.

L'utilisation d'une grille, d'un gabarit ou d'un gabarit est plutôt une façon plus abstraite et plus visuelle d'obtenir les mêmes résultats. Dans un cas comme dans l'autre, vous définissez une grille et vous déterminez ensuite la quantité d'espace que les éléments doivent occuper à l'intérieur de cette grille.

Définir les colonnes en fonction de la taille de l'écran

Jusqu'à présent, nous avons parlé de définitions rigides des grilles. Par exemple, définir le code suivant signifie qu'il y aura TOUJOURS 6 colonnes de contenu, quelle que soit la taille de l'écran de l'appareil :

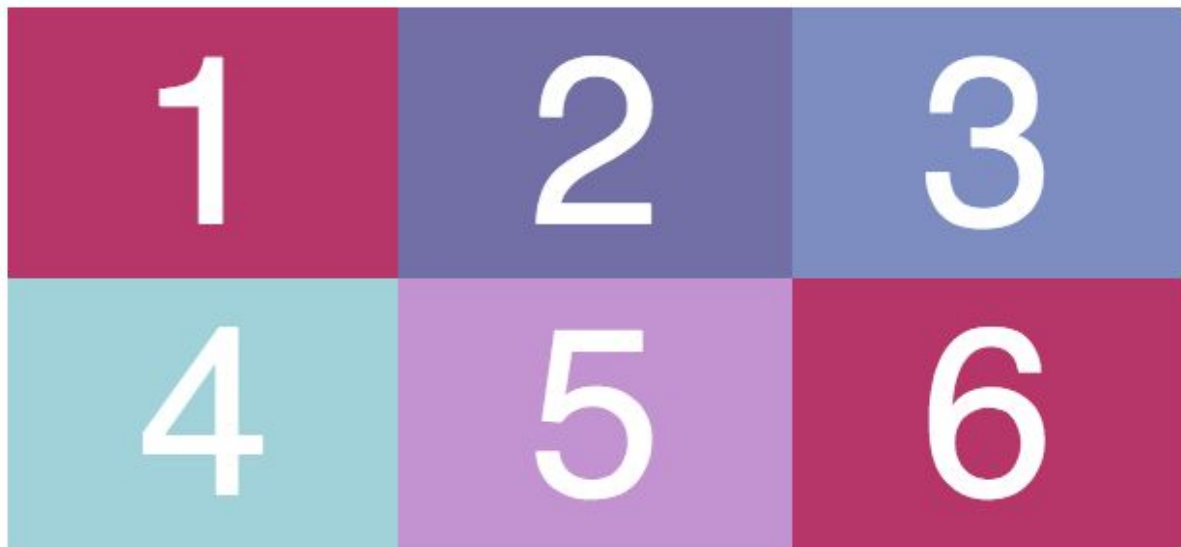
```
.container {  
  display: grid;  
  grid-template-columns: repeat(6, 1fr);  
}
```



Résultat sur un écran de bureau (plus large)



Résultat sur un écran de type portable (plus écrasé ensemble)



Résultat sur tablette

De cette façon, les colonnes ne peuvent pas s'écraser l'une contre l'autre juste pour obtenir 6 colonnes de suite ! CSS Grid offre des solutions de flexibilité pour différentes tailles d'écran qui n'impliquent pas de requêtes médias du tout !

Nous allons apprendre deux nouveaux mots-clés :

- `auto-fit`
- `minmax`

Les deux sont souvent combinés ensemble, alors nous les explorerons en tandem.

Auto-fit et minmax

Regardez la ligne de code ci-dessous. De gauche à droite, on peut lire :

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

Répétez autant de colonnes que vous le souhaitez pour un écran. Chaque colonne doit avoir une largeur minimale de 100px et une largeur maximale de 1fr.

```
<div class="container">
  <div class="un">1</div>
  <div class="deux">2</div>
  <div class="trois">3</div>
  <div class="quatre">4</div>
  <div class="cinq">5</div>
  <div class="six">6</div>
</div>
```

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

.container .un, .six {
  background-color: #B8336A;
}

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

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

.container .quatre {
  background-color: #A0D2DB;
}

.container .cinq {
  background-color: #C490D1;
}
```

Le résultat de ce code est beaucoup plus apparent lorsque vous visualisez le code dans le navigateur et que vous redimensionnez la fenêtre.

Dans l'ensemble, vous remarquerez que la grille se réajustera d'elle-même pour ajouter une autre colonne de 100px lorsqu'il y aura assez d'espace

disponible pour cela. Au fur et à mesure que vous redimensionnez la fenêtre, les colonnes continuent de s'agrandir jusqu'au moment même où il y aurait de la place pour une autre colonne de 100px.

Lors de la réduction de la taille de la fenêtre, les colonnes sont supprimées une par une car le contenu est poussé vers le bas sur les lignes à la place. Cependant, les colonnes ne seront jamais inférieures à 100px. Cela permet d'éviter les colonnes écrasées que vous avez vues au début du chapitre !

La même combinaison d'auto-fit et de minmax peut être utilisée pour les rangées.

C'est stimulant de pouvoir créer une mise en page réactive sans avoir besoin d'interrogations sur les médias !

Exercice : créez votre propre mise en page avec CSS Grid

Testez les nouveaux concepts que vous avez appris dans cette partie en créant un calendrier avec CSS Grid. Un calendrier est un moyen classique d'organiser le contenu à l'aide de lignes et de colonnes.

Avril

1		2	3	4	5
		6	7	8	9
10	11	12	13	14	15
16	17	18	19	20	21
22	23	24	25	26	27
28	29	30			

Exemple de calendrier quadrillé

Choisissez le mois qui vous convient, ainsi que la couleur du fond, de la police et de la couleur de la police. Les seules règles sont les suivantes :

- Vous devez avoir tous les éléments de date dans une grille en utilisant la grille CSS.
- Vous devez avoir une date plus grande que les autres (un jour férié, par exemple). Cet élément aura une mesure personnalisée à l'aide de la grille CSS.
- Vous devez avoir une vedette pour le mois qui ne fait pas partie de la grille.
- Vous devez utiliser aussi padding et border.