Lorsque vous créez un site Web, une grande partie du travail consiste à placer les différents éléments. La mise en page doit être intéressante, mais en même temps être facilement compréhensible et claire. Les feuilles de style en cascade (CSS) constituent un outil intéressant pour rendre les sites Web attractifs. Alors que le HTML attribue des caractéristiques rudimentaires au contenu, les webmasters utilisent le CSS pour les conceptions complexes. La technologie du Web étant en constante évolution, le CSS3 comprend de nouvelles techniques qui rendent le langage de balisage utilisable pour la navigation mobile et le design responsif.

Sommaire

Pourquoi a-t-on besoin d'un layout de grille CSS ?

L'Internet mobile représente aujourd'hui un grand défi pour les concepteurs de sites Web : en raison de la multitude d'appareils mobiles, on ne peut en effet pas toujours savoir **quel est le format de l'écran** sur lequel le site Web doit être affiché. C'est pourquoi il est important que les différents objets (boîtes de texte, graphiques, éléments interactifs) soient répartis à fois de façon indépendante et clairement agencée, tout en tenant compte des particularités d'affichage des différents appareils mobiles.

Pendant plusieurs années il fallait se contenter de ce que l'on appelait les floats. Mais travailler avec cette technique était complexe et pouvait provoquer des erreurs. Depuis, il existe deux méthodes pour réaliser une **mise en page dynamique** : Outre la grille CSS, on peut également utiliser flexbox pour réaliser une conception intelligente. Mais les deux techniques diffèrent sur certains points.

La flexbox est unidimensionnelle, ce qui signifie qu'en principe, les éléments ne sont déplacés que sur un axe. Une grille CSS offre au concepteur deux dimensions pour le placement des objets. Au lieu d'un seul axe, la grille CSS permet une **grille avec des lignes et des colonnes**.

Grille CSS : tutoriel avec exemples

Ceux qui ont déjà eu une expérience avec le CSS n'auront aucun problème avec les grilles. Le tutoriel suivant vous présente les fonctions les plus importantes pour démarrer.

Note

Le tutoriel fonctionne avec un seul fichier : le code CSS est écrit directement dans l'en-tête du fichier HTML. Mais bien sûr, il est également possible de créer une feuille de style séparée et de l'appeler uniquement dans le fichier HTML .

Création de conteneurs et d'objets

La grille CSS comprend deux unités différentes : **les conteneurs et les objets**. Le conteneur constitue le niveau supérieur : vous y définissez des propriétés qui sont ensuite transmises à tous les éléments. Un objet se trouve à l'intérieur d'un conteneur. En outre, vous avez toujours besoin de HTML pour le layout de la grille CSS. Dans la partie HTML du code source, les différents objets (texte, graphiques, ...) sont créés puis repris dans la grille CSS et placés à la bonne position.

```
<!DOCTYPE html>
<html>
<head>
<style>
.grid-container {
  display: grid;
}



</style>
</head>
<body>


<div class="grid-container">
  <div class="grid-item1">1</div>
  <div class="grid-item2">2</div>
  <div class="grid-item3">3</div>
  <div class="grid-item4">4</div>
  <div class="grid-item5">5</div>
  <div class="grid-item6">6</div>
</div>


</body>
</html>
```

Nous avons maintenant créé six éléments, défini chacun comme des « objets de grille » et les avons tous placés dans le « conteneur de grille ». Pour rendre la grille CSS active, vous devez lancer la fonction dans le conteneur avec « display: grid ». Jusqu'à présent, le code ne génère que 6 chiffres, qui apparaissent les uns sous les autres. Une fois qu'ils ont été créés, vous pouvez placer ces éléments relativement librement sur l'écran.

Grilles, colonnes et lignes

Avec les grilles CSS, vous travaillez avec **des lignes et des colonnes pour créer une grille**. Les différents objets peuvent être disposés dans la grille. Cette grille peut ou doit être choisie librement : L'utilisateur décide lui-même de la taille des lignes et des colonnes. Pour ce faire, vous ajoutez deux déclarations au conteneur.

```
.grid-container {
  display: grid;
  grid-template-rows: 100px 100px;
  grid-template-columns: 100px 100px 100px;
}
```

Avec ces deux commandes, nous avons créé une grille de 2 par 3. La taille de **chaque ligne et de chaque colonne peut être ajustée en fonction des besoins spécifiques**. Les entrées sont simplement saisies l'une après l'autre (séparées par un espace). En plus des informations exactes sur les pixels, vous pouvez également utiliser un pourcentage ou d'autres tailles qui sont courantes dans le CSS. Avec les spécifications « max-content » et « min-content », vous pouvez également créer la grille en fonction du contenu.

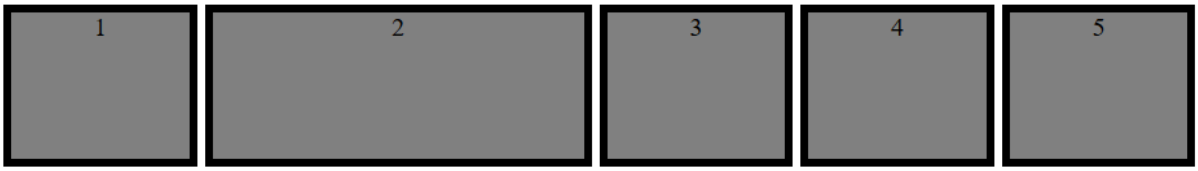
La formule « grid-gap » crée également un **espace**.

```
.grid-container {
  display: grid;
  grid-template-rows: 100px 100px;
  grid-template-columns: 100px 100px 100px;
  grid-gap: 5px;
}
```

Les cellules sont ainsi séparées les unes des autres de façon uniforme Si vous ne souhaitez pas un espacement spécifique, vous pouvez également utiliser les fonctions « grid-column-gap » et « grid-row-gap » pour concevoir les espaces selon vos besoins.

Les **fractions** sont une caractéristique particulière. Avec cette unité de mesure, l'écran peut être divisé de façon personnalisée, sous forme de pourcentages. Par exemple, pour diviser une zone en 7 unités horizontales, avec une seule colonne qui doit être deux fois plus grande que les autres, il faut utiliser le code suivant :

```
.grid-container {
  display: grid;
  grid-template-rows: 100px 100px;
  grid-template-columns: 1fr 2fr 1fr 1fr 1fr;
  grid-gap: 5px;
}
```



: Les colonnes (ou les lignes) ne doivent pas toutes avoir la même taille dans la grille CSS.

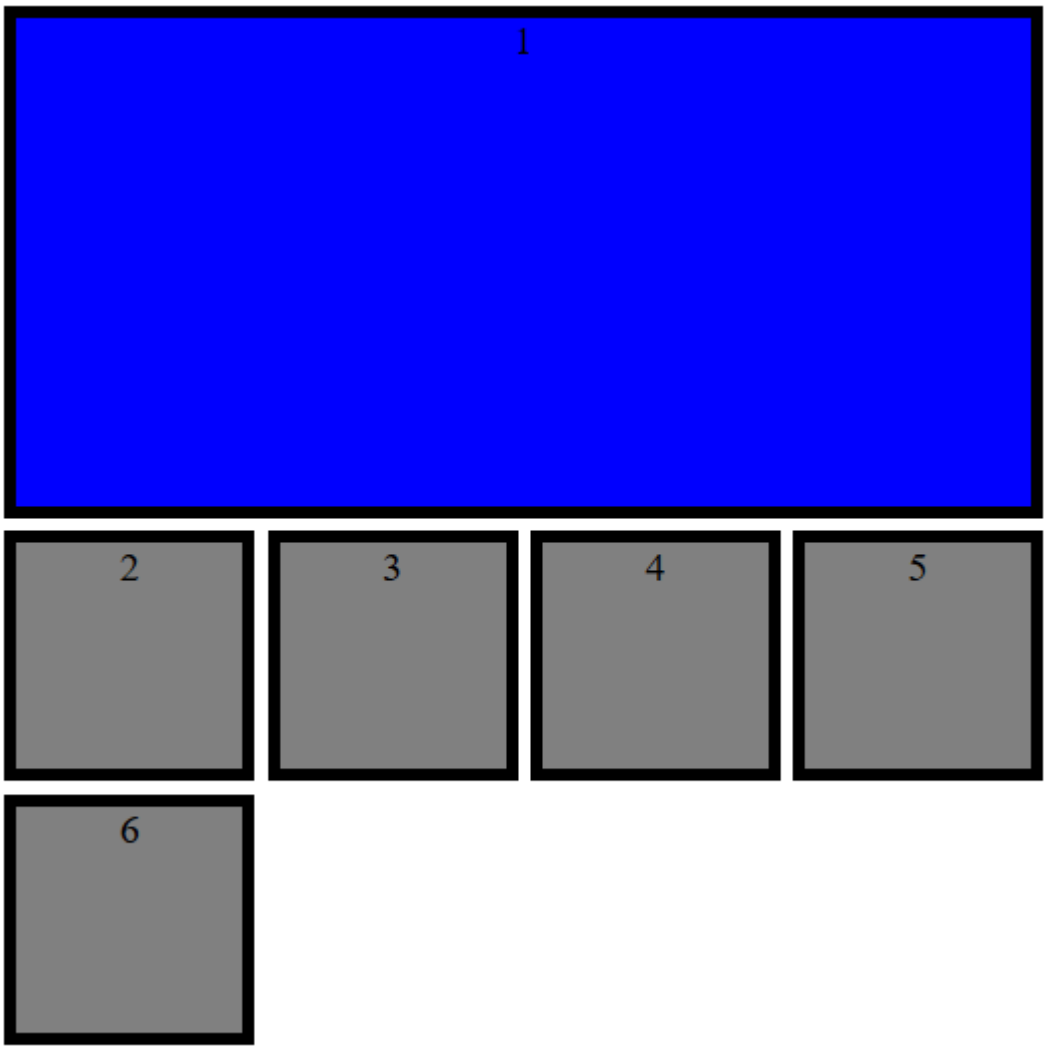
L'avantage de travailler avec des tailles relatives au lieu de spécifications statiques est le suivant : la grille peut **s'adapter automatiquement à la taille de l'écran**. Même si les colonnes doivent être plus petites, la deuxième colonne (dans notre exemple) reste toujours deux fois plus grande que les autres. Si vous souhaitez fixer une ligne, c'est-à-dire ne pas l'adapter à la taille de l'écran, il vous suffit de lui attribuer une valeur statique.

Placement des objets

Après avoir défini la grille, vous pouvez placer les différents objets. Pour ce faire, vous devez **créer des objets** et spécifier des valeurs de début et de fin. Mais chaque élément ne doit pas nécessairement occuper une seule cellule de la grille.

```
<!DOCTYPE html>

<html>

<head>

<style>

.grid-container {

  display: grid;

  grid-template-rows: 100px 100px 100px 100px;

  grid-template-columns: 100px 100px 100px 100px;

  grid-gap: 5px;

}


.grid-item1 {

  background: blue;

  text-align: center;

  border: black 5px solid;

  grid-column-start: 1;

  grid-column-end: 5;

  grid-row-start: 1;

  grid-row-end: 3;

}
```

```
.grid-item2 {
  background: grey;
  text-align: center;
  border: black 5px solid;
}

</style>
</head>
<body>

<div class="grid-container">
  <div class="grid-item1">1</div>
  <div class="grid-item2">2</div>
  <div class="grid-item2">3</div>
  <div class="grid-item2">4</div>
  <div class="grid-item2">5</div>
  <div class="grid-item2">6</div>
</div>

</body>
</html>
```



Le premier objet occupe quatre lignes et deux colonnes.

Nous avons maintenant introduit **deux types d'objets**. Alors que les six derniers éléments n'ont automatiquement que la taille d'une cellule, le premier s'étend sur quatre colonnes et deux lignes. (Pour une meilleure compréhension, nos exemples incluent également d'autres conceptions visuelles. Cependant, les couleurs, les bordures et le centrage du texte ne sont en fait pas définis par la grille CSS.

Les valeurs du début et de la fin des objets ne se réfèrent qu'indirectement aux lignes et aux colonnes. Elles font en fait référence à la **ligne de grille** correspondante. Dans l'exemple, la quatrième colonne se termine par la cinquième ligne. Cependant, vous disposez de plusieurs options pour spécifier les fourchettes.

- **Numérotation** : vous comptez les lignes de gauche à droite et de haut en bas.
- **Noms** : dans les « grid-template-rows » et les « grid-template-columns », vous pouvez donner des noms aux lignes (entre crochets) et vous référer ensuite à ces noms.
- **Span** : avec « span », vous pouvez spécifier combien de cellules l'objet doit contenir dans la direction correspondante.

Au lieu de définir les points de départ et d'arrivée dans une seule déclaration, les concepteurs de sites Web peuvent combiner les deux sous une seule commande. Le code suivant conduit au même résultat que l'exemple précédent.

```
<style>
.grid-container {
  display: grid;
  grid-template-rows: 100px [Line1] 100px [Line2] 100px [Line3] 100px [Line4];
  grid-template-columns: 100px 100px 100px 100px;
  grid-gap: 5px;
}

.grid-item1 {
  background: blue;
  text-align: center;
  border: black 5px solid;
  grid-column: 1 / span 4;
  grid-row: 1 / Line2;
}

.grid-item2 {
  background: grey;
  text-align: center;
  border: black 5px solid;
}

</style>
```

Répartition des zones

Avec une grille CSS, il est possible de **combiner des cellules en zones et de leur attribuer un nom**. De cette façon, il est beaucoup plus facile de répartir les différents éléments dans la grille. Les réglages sont effectués dans le conteneur. Pour ce faire, utilisez le champ « zones de grille » et inscrivez ensuite les noms des zones souhaitées dans les cellules ligne par ligne. Si vous ne voulez pas attribuer une cellule et la laisser vide, vous pouvez insérer un point. Chaque ligne est mise entre guillemets.

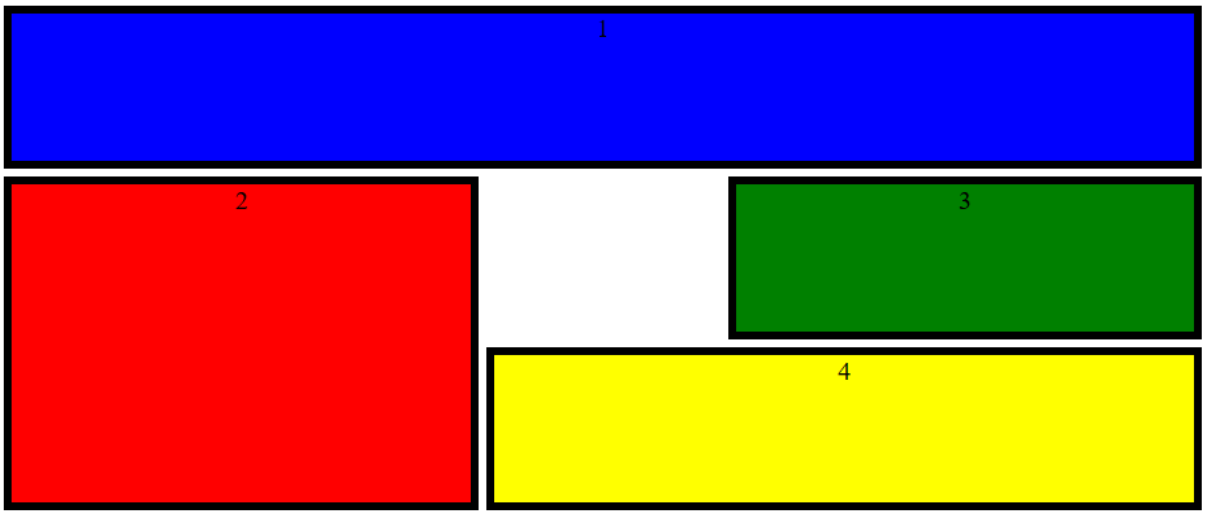
```
.grid-container {
```

```
  display: grid;
  grid-template-rows: 100px 100px 100px;
  grid-template-columns: 1fr 1fr 1fr 1fr 1fr;
  grid-gap: 5px;
  grid-template-areas:
  "area1 area1 area1 area1 area1"
  "area2 area2 . area3 area3"
  "area2 area2 area4 area4 area4";
}
```

Dans cet exemple de grille CSS, nous avons défini 4 zones différentes. Une cellule est laissée vide. Si vous définissez ensuite les éléments, il n'est plus nécessaire de spécifier les valeurs : il suffit de se référer à la zone correspondante.

```
.grid-item1 {
  background: blue;
  text-align: center;
  border: black 5px solid;
  grid-area: area1;
}

.grid-item2 {
  background: red;
  text-align: center;
  border: black 5px solid;
  grid-area: area2;
}

.grid-item3 {
  background: green;
  text-align: center;
  border: black 5px solid;
  grid-area: area3;
}
```

```
.grid-item4 {
  background: yellow;
  text-align: center;
  border: black 5px solid;
  grid-area: area4;
}
```



Le travail est plus facile si l'on utilise des zones pour affecter les objets.

Ajuster l'alignement des éléments

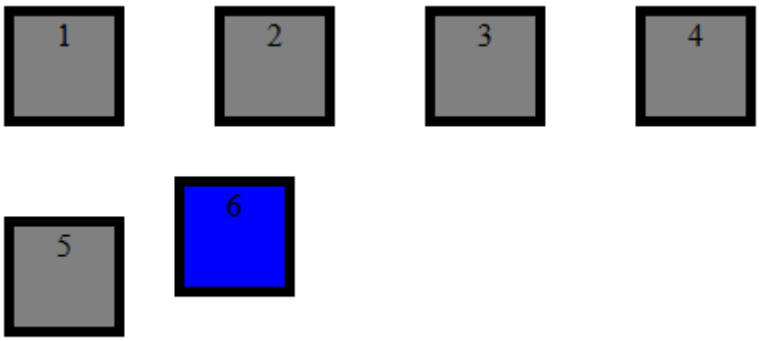
Comment les différents éléments de la grille CSS s'alignent-ils ? Le principe est le suivant : les éléments sont étirés de façon à s'insérer exactement dans la grille, ce qui signifie que **la taille de l'objet** n'a pas d'importance. Elle n'a pas été spécifiée dans les exemples jusqu'à présent, car nous avons plutôt renseigné les cellules sur lesquelles l'objet doit être réparti. Mais il est également possible d'attribuer une taille fixe à l'objet que vous souhaitez intégrer dans la grille.

Les Webdesigners ont le choix de définir l'alignement de tous les objets **de façon globale à travers le conteneur**, ou d'attribuer aux objets sélectionnés un alignement propre. Pour la variante globale, il faut utiliser les termes « justify-items » et « align-items ». Le premier contrôle l'alignement horizontal, le second l'alignement vertical. Si vous ne souhaitez aligner qu'un seul élément, utilisez les options « justify-self » et « align-self ». Toutes ces commandes contiennent les options ci-dessous.

- **Stretch** : la taille de l'objet est adaptée à la taille des cellules sélectionnées.
- **Start** : l'objet est aligné à gauche ou en haut.
- **End** : l'objet est aligné à droite ou en bas.
- **center** : l'objet est centré.

```
.grid-container {
  display: grid;
  grid-template-rows: 100px 100px 100px 100px;
```

```
  grid-template-columns: 100px 100px 100px 100px;
  grid-gap: 5px;
  justify-items: center;
  align-items: center;
}

.grid-item1 {
  background: grey;
  text-align: center;
  border: black 5px solid;
  width: 50px;
  height: 50px;
}

.grid-item2 {
  background: blue;
  text-align: center;
  border: black 5px solid;
  width: 50px;
  height: 50px;
  justify-self: start;
  align-self: start;
}
```



Il est possible de personnaliser l'alignement des éléments d'une grille CSS de façon globale ou individuelle.

Vous pouvez **également raccourcir les instructions** en utilisant les termes « place-items » ou « place-self ». Les deux informations (verticale et horizontale) peuvent être placées sur une seule ligne : place-items : <align-items> / justify-items>.

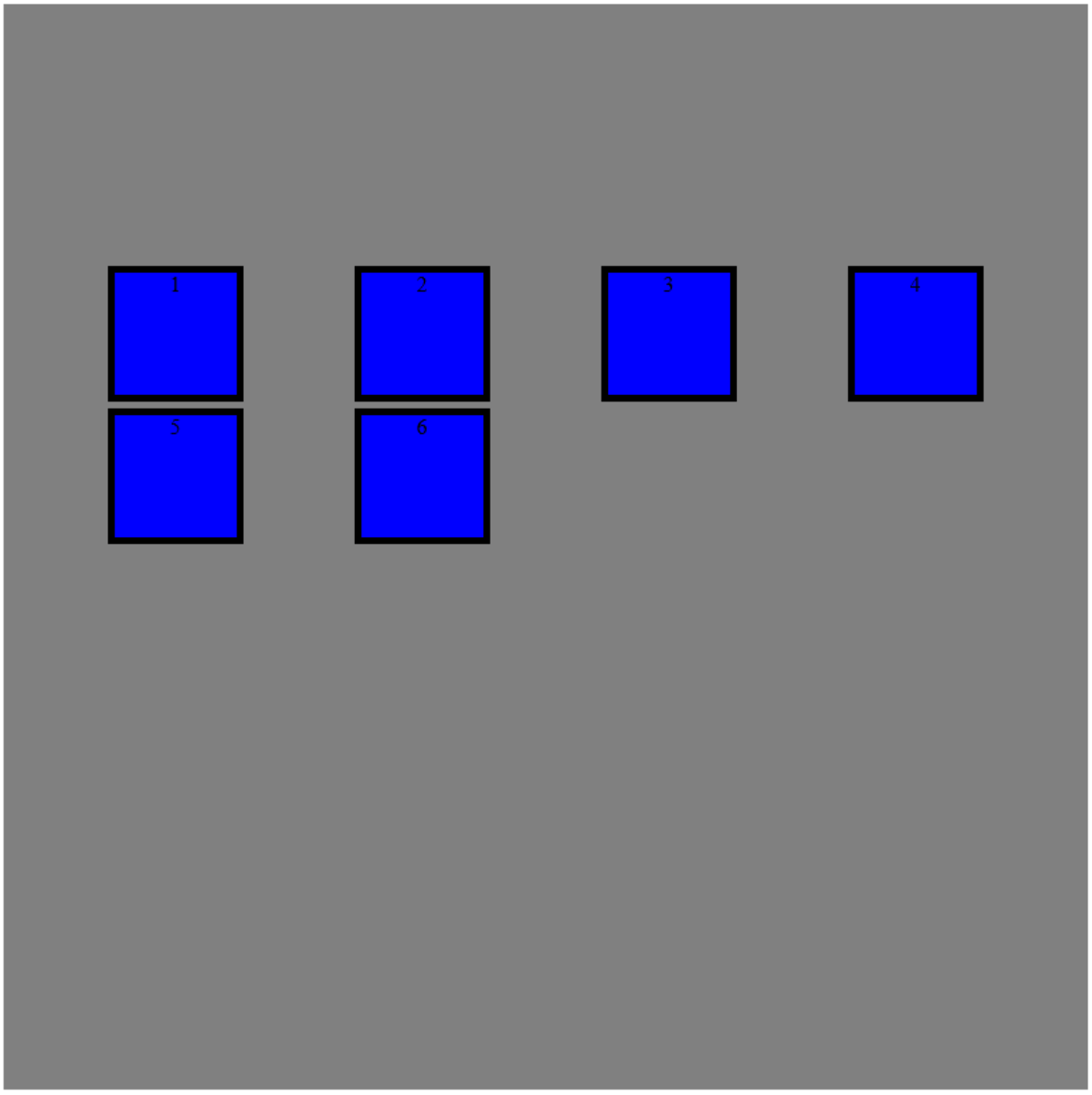
place-items: center / end;

Il est donc possible d'aligner les objets dans la grille, mais aussi de déplacer **l'ensemble de la grille** à l'intérieur du conteneur. Si vous travaillez avec des spécifications de taille statique pour la grille CSS, celle-ci peut ne pas avoir la même taille que le conteneur entier. Vous pouvez ensuite utiliser les fonctions « justify-content » et « align-content » pour aligner la grille dans le conteneur, donc dans l'affichage également. Ici aussi, il existe différentes possibilités d'alignement :

- **start** : aligné à gauche ou aligné vers le haut
- **end** : aligné à droite ou orienté vers le bas
- **center** : centré
- **stretch** : grille étirée
- **space-around** : répartition uniforme de l'espace autour des cellules
- **space-between** : répartition uniforme de l'espace entre les cellules
- **space-evenly** : répartition uniforme de l'espace autour des cellules, y compris la frontière

```
.grid-container {
  display: grid;
  width: 800px;
  height: 800px;
  background: yellow;
  grid-template-rows: 100px 100px 100px 100px;
  grid-template-columns: 100px 100px 100px 100px;
  grid-gap: 5px;
```

```
  justify-content: space-evenly;
  align-content: center;
}
```



Vous pouvez aligner des éléments de façon individuelle, mais aussi l'ensemble de la grille dans le conteneur.

Il existe également une version courte : place-content: <align-content> / <justify-content>.

place-content: center / space-evenly;

Ajustements automatiques pour le design responsif

L'un des principaux avantages de la grille CSS est sa flexibilité. Vous pouvez paramétrer la grille CSS pour qu'elle s'ajuste automatiquement. Ceci facilite non seulement l'affichage sur différents appareils, mais les **automatismes** facilitent également le travail avec le CSS.

« repeat() » est une fonction très utile : au lieu de définir chaque ligne ou colonne de façon individuelle, vous pouvez spécifier **la taille et la fréquence à laquelle ce schéma doit être répété**. C'est encore plus facile en combinaison avec les options « auto-fill » et « auto-fit ». Ici, la grille CSS est laissée pour créer des lignes et des colonnes. Avec la première option, la grille CSS insère autant de cellules que possible sans dépasser les limites du conteneur. Cependant, il peut arriver que l'espace reste inutilisé. L'option « auto-fit », en revanche, ajuste la taille des cellules de sorte que l'espace soit utilisé jusqu'au bord.

```
.grid-container {
  display: grid;
  width: 800px;
  height: 800px;
  grid-template-rows: repeat(auto-fit, 100px);
  grid-template-columns: repeat(auto-fit, 100px);
  grid-gap: 5px;
}
```

Les deux fonctions « grid-auto-columns » et « grid-auto-rows » sont également très pratiques. Grâce à ces deux instructions, vous avez **plus de liberté dans la création des objets**. Si, par exemple, une grille avait les dimensions de 4 x 4 cellules et que vous créiez maintenant un élément qui ne devrait commencer que dans la cinquième colonne, cela poserait des problèmes. En créant automatiquement un nombre suffisant de lignes et de colonnes, vous pouvez éviter ce genre de problèmes.

```
.grid-container {
  display: grid;
  grid-auto-rows: 100px;
  grid-auto-columns: 100px;
  grid-gap: 5px;
}
```

Même si la taille de la grille et des objets doit s'adapter à la taille de l'affichage, les concepteurs de sites Web aiment entrer des **valeurs minimales et maximales.** Vous pouvez utiliser la déclaration « minmax() » pour vous assurer qu'un objet ne devient pas trop petit ou trop grand. Entrez d'abord la plus petite valeur entre parenthèses, puis la plus grande.

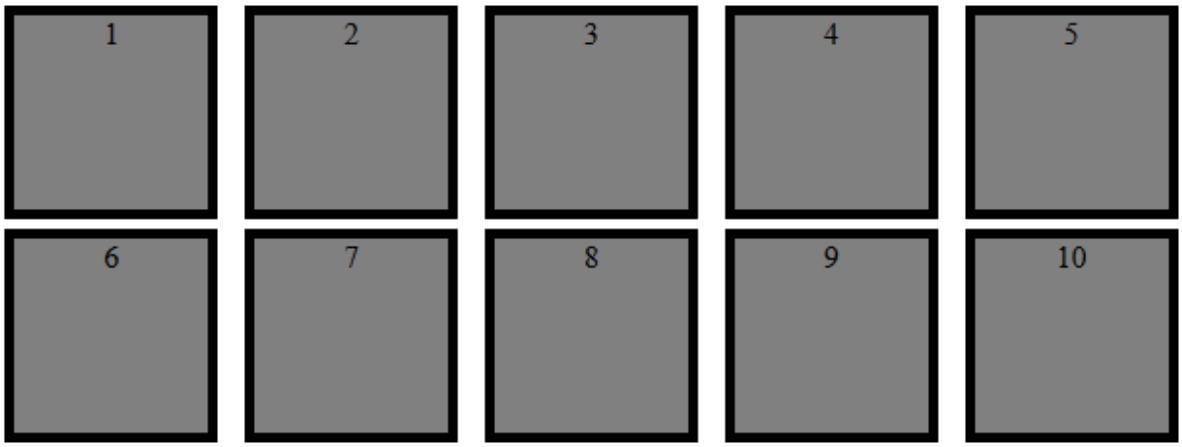
```
.grid-container {
  display: grid;
  grid-template-rows: 100px 100px 100px;
  grid-auto-columns: minmax(50px, 150px);
  grid-gap: 5px;
```

```
}
```

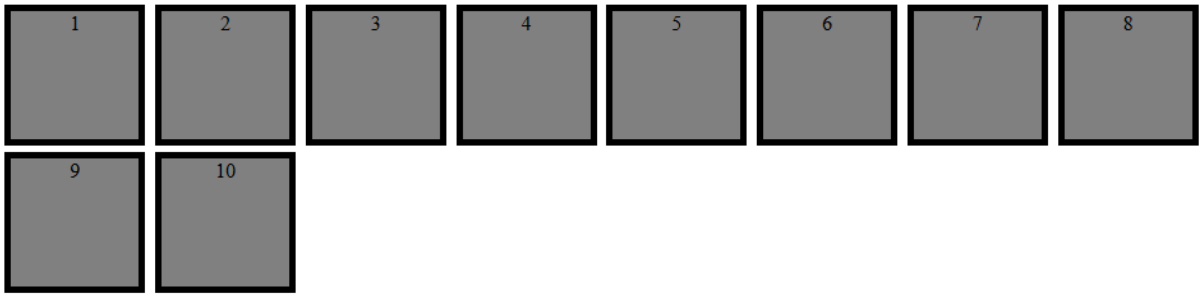
En combinant plusieurs de ces fonctions, vous pouvez facilement **créer un design responsif.**

```
<!DOCTYPE html>
<html>
<head>
<style>
.grid-container {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(100px, 1fr));
  grid-gap: 5px;
  justify-items: center;
  align-items: center;
}

.grid-item1 {
  background: grey;
  text-align: center;
  border: black 5px solid;
  width: 100px;
  height: 100px;
}



</style>
</head>
<body>

<div class="grid-container">
  <div class="grid-item1">1</div>
  <div class="grid-item1">2</div>
  <div class="grid-item1">3</div>
```

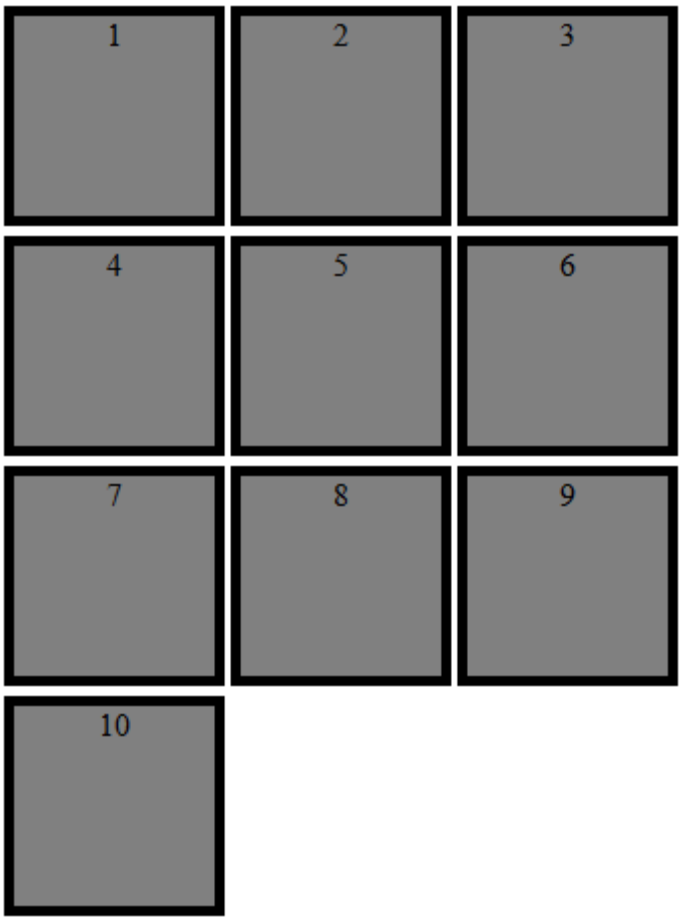
```
  <div class="grid-item1">4</div>
  <div class="grid-item1">5</div>
  <div class="grid-item1">6</div>
  <div class="grid-item1">7</div>
  <div class="grid-item1">8</div>
  <div class="grid-item1">9</div>
  <div class="grid-item1">10</div>

</div>

</body>
</html>
```



Si vous configurez correctement la grille CSS, vous pouvez créer très simplement un design responsif.



En agrandissant une zone d'affichage, il est possible de faire entrer plus d'éléments dans une ligne.



La grille CSS permet un affichage de qualité même sur de petits écrans.

En résumé

La grille CSS offre aux concepteurs de sites Web la possibilité de créer facilement des mises en page attrayantes. Grâce à la grille, vous avez toujours le contrôle sur le placement des objets, même avec un design responsif.

;,gfkb ;:v,blv,bcvjbnn