Comment CSS Grid a tout changé

Les grilles sont essentielles pour la conception de sites Web ; vous le saviez déjà. Mais vous êtes-vous arrêté pour vous demander comment vous avez conçu la grille que vous utilisez principalement ?

La plupart d'entre nous ne l'ont pas fait. Nous utilisons la grille à 12 colonnes qui est devenue une norme dans notre industrie.

- Mais pourquoi utilisons-nous la même grille ?
- Pourquoi les grilles sont-elles composées de 12 colonnes ?
- Pourquoi nos grilles sont-elles de taille égale ?

Voici une réponse possible à la raison pour laquelle nous utilisons la même grille : Nous ne voulons pas faire de calcul.

Dans le passé, avec les grilles basée sur “float”, pour créer une grille à trois colonnes, il fallait calculer la largeur de chaque colonne, la taille de chaque gouttière et comment positionner chaque élément de la grille. Ensuite, vous deviez créer des classes dans le HTML pour leur donner un style approprié. C'était assez compliqué.

Pour faciliter les choses, nous avons adopté des framework qui proposaient leurs propres grilles. Au début, des framework tels que 960gs et 1440px nous permettaient de choisir entre des grilles de 8, 9, 12 et même 16 colonnes. Plus tard, Bootstrap a gagné la guerre des framework. Parce que Bootstrap n'autorisait que 12 colonnes, et changer cela est un peu

pénible, nous avons finalement choisi 12 colonnes comme norme.

Mais nous ne devrions pas blâmer Bootstrap. C'était la meilleure approche à l'époque. Qui ne voudrait pas d'une bonne solution qui fonctionne avec un minimum d'efforts ? Une fois le problème de la grille réglé, nous avons tourné notre attention vers d'autres aspects de la conception, tels que la typographie, la couleur et l'accessibilité.

Maintenant, avec l'avènement de CSS Grid, les grilles sont devenues beaucoup plus simples. Nous n'avons plus à craindre les “maths de la grille”. C'est devenu si simple que je dirais qu'il est plus facile de créer une grille avec CSS que dans un outil de conception comme Sketch !

Pourquoi ?

Supposons que vous souhaitiez créer une grille à 4 colonnes, chaque colonne ayant une taille de 100 pixels. Avec CSS Grid, vous pouvez écrire 100px quatre fois dans la déclaration grid-template-columns, et une grille de 4 colonnes sera créée.

```
.grid {
  display: grid;
  grid-template-columns: 100px 100px 100px 100px;
  grid-column-gap: 20px;
}
```

Vous pouvez créer quatre colonnes de grille en spécifiant une largeur de colonne quatre fois avec `grid-template-columns`

Si vous voulez une grille de 12 colonnes, il vous suffit de répéter 100px fois 12 fois.

```
.grid {
  display: grid;
  grid-template-columns: 100px 100px 100px 100px 100px 100px 100px 100px 100px 100px 100px 100px;
  grid-column-gap: 20px;
}
```

Créer 12 colonnes avec CSS Grid

Oui, le code n'est pas beau, mais nous ne nous soucions pas d'optimiser la qualité du code (encore) - nous pensons toujours à la conception. CSS Grid permet à n'importe qui - même un concepteur sans connaissances en codage - de créer une grille sur le web.

Si vous voulez créer des colonnes de grille avec des largeurs différentes, vous n'avez qu'à spécifier la largeur désirée dans votre déclaration grille-modèle-colonnes, et c'est réglé.

```
.grid {
  display: grid;
  grid-template-columns: 100px 162px 262px;
  grid-column-gap: 20px;
}
```

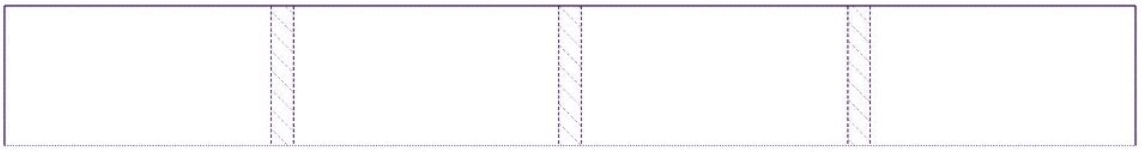
Créer des colonnes de différentes largeur est très simple.

RENDRE LES GRILLES RESPONSIVES

Aucune discussion sur la grille CSS n'est complète sans parler de l'aspect responsive. Il y a plusieurs façons de rendre la grille CSS responsive. Une façon (probablement la plus populaire) est d'utiliser l'unité *fr*. Une autre façon est de changer le nombre de colonnes avec des requêtes média.

fr est une longueur flexible qui représente une fraction. Lorsque vous utilisez l'unité *fr*, les navigateurs divisent l'espace disponible et attribuent les zones aux colonnes en fonction du multiple *fr*. Cela signifie que pour créer quatre colonnes de taille égale, vous devez écrire *1fr* quatre fois

```
.grid {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr 1fr;
  grid-column-gap: 20px;
}
```



Les grilles créées avec l'unité *fr* respectent la largeur maximale de la grille.

Faisons quelques calculs pour comprendre pourquoi quatre colonnes de taille égale sont créées.

Supposons d'abord que l'espace total disponible pour la grille est de *1260px*.

Avant d'allouer la largeur à chaque colonne, la grille CSS doit savoir combien d'espace est disponible (ou restant). Ici, il soustrait les déclarations de *grip-gap* de *1260px*. Depuis chaque espace *20px*, il nous reste *1200px* pour l'espace disponible. *(1260 - (20 * 3) = 1200)*.

Ensuite, il additionne les multiples de *fr*. Dans ce cas, nous avons quatre multiples de *1fr*, donc les navigateurs divisent *1200px* par quatre. Chaque colonne est donc *300px*. C'est pourquoi nous obtenons quatre colonnes égales.

Cependant, les grilles créées avec l'unité fr ne sont pas toujours égales !

Lorsque vous utilisez *fr*, vous devez être conscient que chaque unité *fr* est une fraction de l'espace disponible (ou restant).

Si vous avez un élément plus large que n'importe laquelle des colonnes créées avec l'unité *fr*, le calcul doit être fait différemment.

Par exemple, la grille ci-dessous a une grande colonne et trois petites colonnes (mais égales) même si elle a été créée avec des colonnes grille-modèle : *1fr 1fr 1fr 1fr 1fr 1fr 1fr*.

HTML

```
<div class="grid">
 <div class="grid-item">
   <img src="https://lorempixel.com/output/nature-q-c-1000-500-5.jpg" alt="">
 </div>
```

```
 <div class="grid-item"></div>
 <div class="grid-item"></div>
 <div class="grid-item"></div>
</div>
```

CSS

```
body {
 padding: 3em;
}

.grid {
 display: grid;
 grid-template-columns: repeat(4, 1fr);
 grid-gap: 1em;
}

.grid-item {
 background-color: orange;
}

img {
 display: block;
 width: 300px;
}
```

Résultat

Après avoir divisé 1200px en quatre et alloué 300px à chacune des colonnes 1fr, les navigateurs se rendent compte que le

premier élément de la grille contient une image qui est 1000px. Puisque 1000px est plus grand que 300px, les navigateurs choisissent d'allouer 1000px à la première colonne à la place.

Cela signifie que nous devons recalculer l'espace restant.

Le nouvel espace restant est de 1260px - 1000px - 20px * 3 = 200px ; ce 200px est ensuite divisé par trois en fonction de la quantité de fractions restantes. Chaque fraction est alors 66px. Espérons que cela explique pourquoi les unités fr ne créent pas toujours des colonnes de largeur égale.

Si vous voulez que l'unité fr crée des colonnes de même largeur à chaque fois, vous devez la forcer avec minmax(0, 1fr). Pour cet exemple spécifique, vous voudrez également définir la propriété max-width de l'image à 100%.

HTML voir exemple précédent.

CSS

```
body {
 padding: 3em;
}

.grid {
 display: grid;
 grid-template-columns: repeat(4, minmax(0, 1fr));
 grid-gap: 1em;
}

.grid-item {
 background-color: orange;
}
```

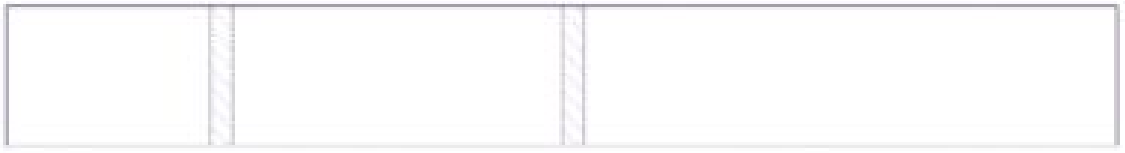
```
img {
 display: block;
 max-width: 100%;
}
```

Résultat

GRILLES DE LARGEUR INÉGALE

Pour créer des grilles avec des largeurs inégales, il suffit de varier le multiple *fr*. Ci-dessous se trouve une grille qui suit le nombre d'or, où la deuxième colonne est de 1,618 fois la première colonne et la troisième colonne est de 1,618 fois la deuxième colonne.

```
.grid {
  display: grid;
  grid-template-columns: 1fr 1.618fr 2.618fr;
  grid-column-gap: 1em;
}
```



Une grille à trois colonnes créée avec le nombre d'or.

MODIFICATION DES GRILLES À DIFFÉRENTS POINTS D'ARRÊT

Si vous souhaitez modifier la grille à différents points d'arrêt, vous pouvez déclarer une nouvelle grille dans une requête média.

```
.grid {
  display: grid;
  grid-template-columns: 1fr 1fr;
  grid-column-gap: 1em;
}

@media (min-width: 30em) {
  .grid {
    grid-template-columns: 1fr 1fr 1fr 1fr;
  }
}
```

N'est-il pas facile de créer des grilles avec CSS Grid ? Les concepteurs et les développeurs antérieurs auraient tué pour avoir une telle possibilité.

GRILLES BASEES SUR LA HAUTEUR

Il était impossible de faire des grilles basées sur la hauteur d'un site Web auparavant parce qu'il n'y avait aucun moyen pour nous de déterminer la hauteur de la fenêtre. Maintenant, avec les unités de viewport -vm,vh…-, CSS Calc, et CSS Grid, nous pouvons même faire des grilles basées sur la hauteur de la vue.
Dans la démo ci-dessous, j'ai créé des carrés de grille en fonction de la hauteur du navigateur.

HTML

```
<div class="grid">
 <div class="grid-item"></div>
 <div class="grid-item"></div>
 <div class="grid-item"></div>
 <div class="grid-item"></div>
 <div class="grid-item"></div>
 <div class="grid-item"></div>
 <div class="grid-item"></div>
 <div class="grid-item"></div>
 <div class="grid-item"></div>
 <div class="grid-item"></div>
 <div class="grid-item"></div>
 <div class="grid-item"></div>
 <div class="grid-item"></div>
 <div class="grid-item"></div>
 <div class="grid-item"></div>
 <div class="grid-item"></div>
 <div class="grid-item"></div>
 <div class="grid-item"></div>
 <div class="grid-item"></div>
 <div class="grid-item"></div>
 <div class="grid-item"></div>
 <div class="grid-item"></div>
 <div class="grid-item"></div>
 <div class="grid-item"></div>
 <div class="grid-item"></div>
 <div class="grid-item"></div>
 <div class="grid-item"></div>
 <div class="grid-item"></div>
 <div class="grid-item"></div>
 <div class="grid-item"></div>
 <div class="grid-item"></div>
 <div class="grid-item"></div>
 <div class="grid-item"></div>
 <div class="grid-item"></div>
```

```
 <div class="grid-item"></div>
 <div class="grid-item"></div>
</div>
```

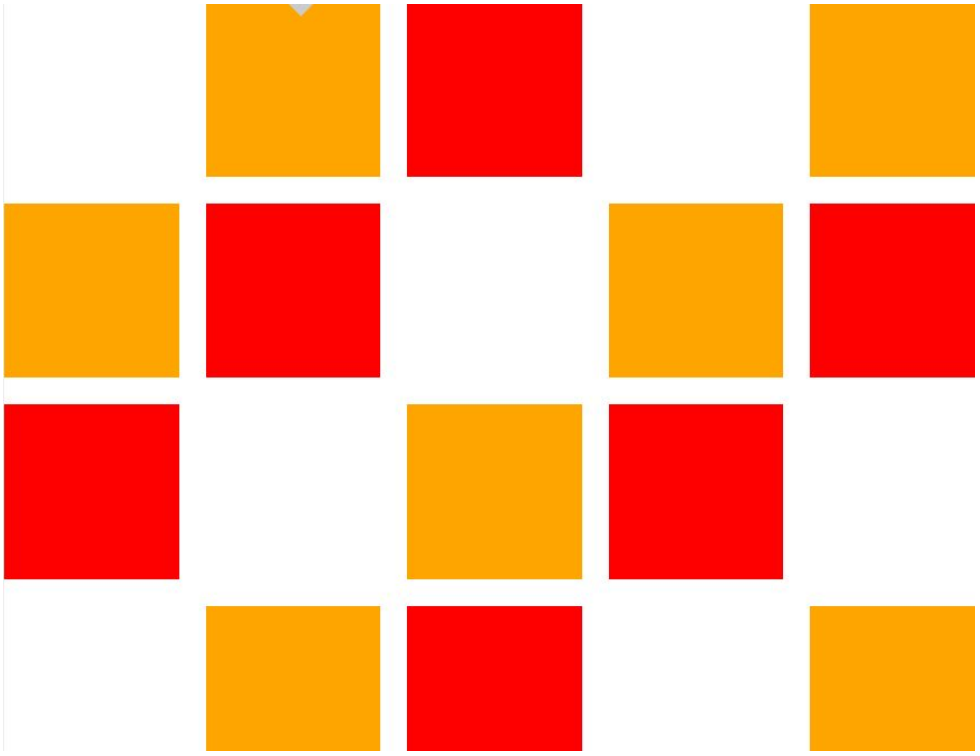
CSS

```
.grid {
 display: grid;
 grid-template-rows: repeat(4, 1fr);
 grid-auto-columns: calc((100vh - 3em) / 4);
 grid-auto-flow: column;
 grid-gap: 1em;
 height: 100vh;
}

.grid-item:nth-child(3n) {
 background-color: red;
}

.grid-item:nth-child(3n + 2) {
 background-color: orange;
}
```

Résultat



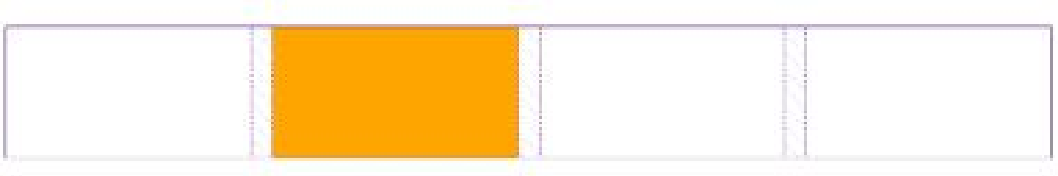
Consulter le résultat ici https://codepen.io/zellwk/pen/qoEYaL

PLACEMENT DES ÉLÉMENTS DE LA GRILLE

Le positionnement des éléments de la grille était très pénible dans le passé parce qu'il fallait calculer la propriété marge-gauche.

Maintenant, avec CSS Grid, vous pouvez placer les éléments de la grille directement avec CSS sans les calculs supplémentaires.

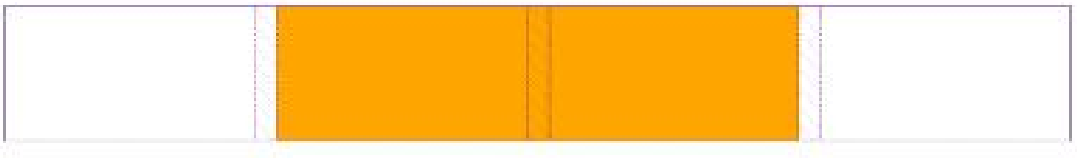
```
.grid-item {
  grid-column: 2; /* Mis sur le deuxième colonne */
}
```



Placer un élément sur la deuxième colonne.

Vous pouvez même dire à un élément de la grille combien de colonnes il doit prendre avec le mot-clé span.

```
.grid-item {
  /* Mis sur le deuxième colonne et occupe 2 colonne s */
  grid-column: 2 / span 2;
}
```



Vous pouvez indiquer aux éléments de la grille le nombre de colonnes (ou de lignes paires) qu'ils doivent utiliser avec le mot-clé span

Pour en savoir plus sur CSS Grid, consultez les ressources suivantes :

- Master CSS Grid, Rachel Andrew and Jen Simmons
- Les concepts de base
- https://cssgridgarden.com/#fr

Et beaucoup d’autres...