GRID VS FLEXBOX : QUE DEVRIEZ-VOUS CHOISIR ?

CSS Grid et CSS Flexbox sont des technologies de mise en page web complémentaires qui ont été très attendues pendant des années. Cependant, en dépit de quelques similitudes superficielles, ils sont en fait utilisés pour des tâches très différentes ; ils résolvent chacun un ensemble de problèmes très différents.

Dans un scénario idéal, il se peut que vous utilisiez les deux pour différentes tâches de mise en page.Nous examinerons leurs différences, nous verrons comment ils résolvent divers problèmes de mise en page et nous vous aiderons à choisir la bonne solution à vos problèmes.

Grid est basé sur le conteneur, Flexbox est basée sur le contenu.

Dans la structure flexbox, la taille d'une cellule (item flex) est définie à l'intérieur de l'élément flex lui-même, alors qu'avec grid, la taille d'une cellule (itemcgrid) est définie à l'intérieur du conteneur grid.

Prenons un exemple, voici le HTML pour créer une rangée d'éléments :

```
<div class="row">
<div>1</div>
<div>2</div>
<div>3</div>
<div>4</div>
</div>
```

Et nous le stylons en utilisant flexbox ainsi :

```css
.row {
    margin: 20px auto;
    max-width: 300px;
    display: flex;
}
.row > div {
    border: 1px dashed gray;
    flex: 1 1 auto; /* Size of items defined inside items */
    text-align: center;
    padding: 12px;
}
```

Nous avons défini la taille des cellules à l'intérieur de l'item flex en déclarant la propriété *flex : 1 1 auto*;. La propriété flex est une abréviation pour définir les propriétés flex-grow, flex-shrink et flex-basis en une seule instruction; sa valeur par défaut est 0 1 auto. Notez que la div de classe "row" est le conteneur flex, et nous ne définissons pas la taille des éléments là. Nous fixons la taille à l'intérieur de l'item flex.

Lorsque l'on prévisualise dans un navigateur, nous obtenons une rangée de boîtes, comme suis :

Voyons maintenant comment nous pouvons générer le même rendu en utilisant Grid :

```css
.row {
    margin: 20px auto;
    max-width: 300px;
    display: grid;
    grid-template-columns: 1fr 1fr 1fr 1fr;

    /* Size of items defined inside container */
}
.row div {
    border: 1px dashed gray;
    text-align: center;
    padding: 12px;
}
```

Le code ci-dessus nous donnera exactement le même résultat.

Notez que nous définissons maintenant la taille des cellules à l'aide de grid-template-columns dans le conteneur grid (.row), et non de l'élément de la grille.

C'est une différence importante. Il montre que la mise en page de la flexbox est calculée après le chargement de son contenu alors que la mise en page de grid est calculée indépendamment du contenu qu'elle contient.
Donc, si possible, évitez d'utiliser flexbox pour construire la mise en page globale de votre site web.

Grid a une propriété "Gap", Flexbox n'en a pas

Vous pouvez argumenter qu'une différence majeure entre flexbox et la grille est que dans cette dernière, nous pouvons créer des gouttières entre les éléments de grille en utilisant grid-column-gap, comme ça :

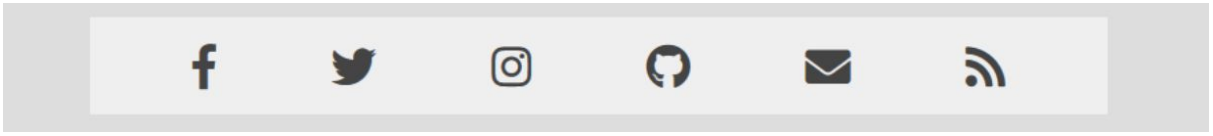
Pour obtenir le même résultat avec flexbox, il faudrait utiliser des padding et des conteneurs imbriqués, ou augmenter la largeur du flex-container et utiliser la propriété justify-content et l’appliquer sur les items flex.

Nous devons prendre un chemin détourné en flexbox parce qu'il n'existe pas de propriété permettant de gérer l'écart. Cependant, il est en cours d'élaboration.

Flexbox est unidimensionnel, Grid est bidimensionnel

Nous avons organisé les éléments sous forme de lignes et de colonnes sur le Web depuis que nous utilisons les tableaux pour faire la mise en page. Flexbox et grid sont tous deux basés sur ce concept. Flexbox est la meilleure solution pour disposer les éléments en une seule rangée ou en une seule colonne. La grille est idéale pour disposer les éléments en plusieurs rangées et colonnes.

En d'autres termes, Flexbox est unidimensionnel, et Grid est bidimensionnel. Examinons une disposition monodimensionnelle couramment utilisée - les boutons de partage social :



Tous les éléments sont dans une seule rangée. Nous pouvons l'implémenter en utilisant Flexbox comme ceci :

```
<ul class="social-icons">
  <li><a href="#"><i class="fab fa-facebook-f"></i></a></li>
  <li><a href="#"><i class="fab fa-twitter"></i></a></li>
  <li><a href="#"><i class="fab fa-instagram"></i></a></li>
  <li><a href="#"><i class="fab fa-github"></i></a></li>
  <li><a href="#"><i class="fas fa-envelope"></i></a></li>
  <li><a href="#"><i class="fas fa-rss"></i></a></li>
</ul>

.social-icons {
  display: flex;
  list-style: none;
  justify-content: space-around;
}
```

La propriété de justify-content détermine comment l'espace supplémentaire du conteneur flex est réparti entre les items flex. La

valeur space-around distribue l'espace de telle sorte que les items flex soient placés de manière égale avec la même quantité d'espace autour d'eux.

Ensuite, jetons un coup d'œil à une mise en page bidimensionnelle couramment utilisée :

Nous ne pouvons pas implémenter ce layout avec une seule ligne ou une seule colonne, nous avons besoin de plusieurs lignes et colonnes pour le faire, et c'est là que nous utilisons les grilles CSS. Faisons-le en utilisant CSS Grid :

```
<div class="container">
  <header>Header</header>
  <main>Main</main>
  <aside>Aside</aside>
  <footer>Footer</footer>
</div>
```

et le CSS :

```css
.container {
  max-width: 800px;
  margin: 2em auto;
  display: grid;
  grid-template-columns: 3fr 1fr;
  grid-template-rows: repeat(3,auto);
  grid-gap: 1rem;
}

.container header {
  grid-area: 1/1/2/3;
}

.container main {
  grid-area: 2/1/3/2;
}

.container aside {
  grid-area: 2/2/3/3;
}

.container footer {
  grid-area: 3/1/4/3;
}

.container > * {
  background-color: #ddd;
  padding: 1rem;
}
```

Nous créons deux colonnes en utilisant la propriété grid-template-columns et trois lignes en utilisant la propriété grid-template-row. La fonction repeat() crée 3 lignes avec hauteur automatique.

Ensuite, à l'intérieur des éléments de la grille (header, main, aside et footer), nous définissons la superficie que couvrira ces éléments sur la grille à l'aide de la propriété grid-area. La propriété grid-area spécifie la taille et l'emplacement d'un élément de la grille dans une disposition en grille et est une propriété abrégée pour les propriétés suivantes :

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

- grid-column-end

Flexbox wrap vs Grid wrap

Lorsque la largeur totale des éléments à l'intérieur du conteneur est supérieure à la largeur du conteneur, dans ce cas, les deux modèles de disposition ont la possibilité de mettre les éléments dans une nouvelle rangée. Cependant, la façon dont les deux manipulent cette création de nouvelle ligne est différente.

Examinons cette différence en construisant un exemple de mise en page. Créez 2 rangées et placez 6 div à l'intérieur de chaque rangée :

```
<h2>Flexbox</h2>
<div class="row-flex">
    <div>1 2 3  4 5 6 7 8 9 0</div>
    <div>2</div>
    <div>3</div>
    <div>4</div>
    <div>5</div>
    <div>6</div>
</div>

<h2>Grid</h2>
<div class="row-grid">
    <div>1 2 3  4 5 6 7 8 9 0</div>
    <div>2</div>
    <div>3</div>
    <div>4</div>
    <div>5</div>
    <div>6</div>
</div>
```

Maintenant, nous allons utiliser Flexbox pour mettre en page la première rangée et Grid pour la deuxième :

```css
/* Flexbox row styles */
.row-flex {
  margin: 40px auto;
  max-width: 600px;
  display: flex;
  flex-wrap: wrap;
}
.row-flex div {
  border: 1px dashed gray;
  flex: 1 1 100px;
  text-align: center;
  padding: 12px;
}
/* Grid row styles */
.row-grid {
  margin: 40px auto;
  max-width: 600px;
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(100px, 1fr));
}
.row-grid div {
  border: 1px dashed gray;
  text-align: center;
  padding: 12px;
}
```

Pour la première rangée, nous utilisons flex: 1 1 100px pour donner aux items flex une largeur de base de 100px et leur permettre de grandir et de se rétracter.

Nous permettons également l'emballage d'articles flexibles à l'intérieur du conteneur flex en réglant la propriété flex-wrap à wrap, sa valeur par défaut est nowrap.

Pour la deuxième ligne, nous utilisons la propriété grid-template-columns pour créer des colonnes avec une largeur minimale de 100px définie par la fonction minmax(). Nous utilisons la fonction repeat() pour créer des colonnes à répétition.

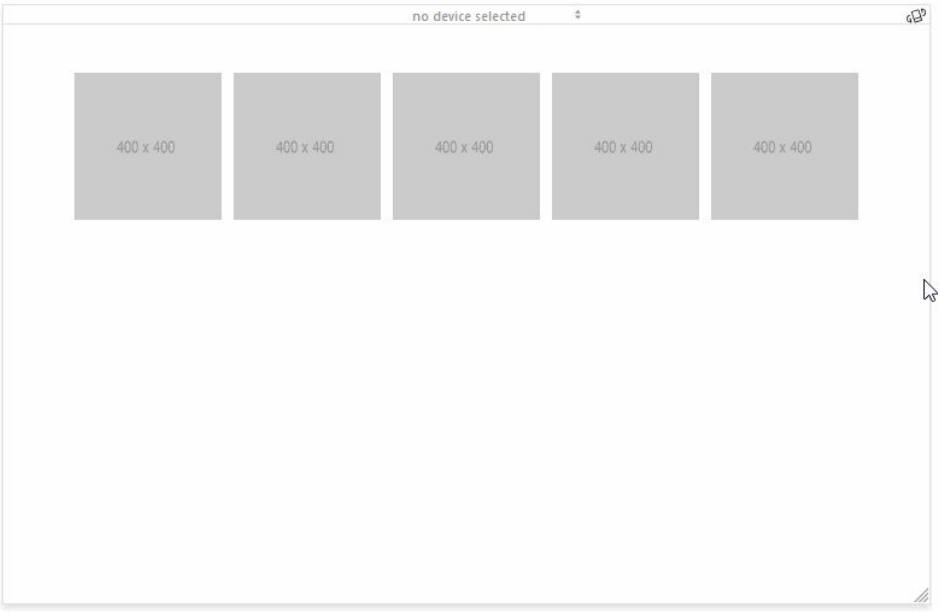
Vous pouvez voir la beauté de Grid et Flexbox réside dans la capacité d'étirer et de resserrer les éléments en fonction de la quantité d'espace disponible. Flexbox y parvient en utilisant les propriétés flex-grow et flex-shrink, et Grid y parvient en combinant les fonctions minmax et autofill à l'intérieur de la propriété grid-template-columns.

Cependant, examinez attentivement les cellules 5 et 6 au fur et à mesure qu'elles sont poussées vers le bas. Dans le cas de Flexbox, les cellules 5 et 6 ne sont pas de la même taille que les autres cellules lorsqu'elles sont poussées. Dans le cas de grid, elles conservent la même taille que toutes les autres cellules de la grille.

Cela se produit parce que lorsqu'un item flex est emballé et poussé dans une nouvelle rangée, l'algorithme de disposition Flexbox le traite comme une partie d'un conteneur flex différent. Par conséquent, l'élément poussé perd son contexte. La propriété flex est l'abréviation de trois propriétés : flex-grow, flex-shrink et flex-basis.

Flexbox surpasse Grid dans ce cas d'utilisation. Oui, vous pouvez utiliser un hack pour obtenir que Grid réplique ce comportement en utilisant la fonction minmax(), mais Flexbox est bien adapté pour ce type de mise en page unidimensionnelle.

Cependant, si vous souhaitez une mise en page multidimensionnelle avec les éléments conservant leurs largeurs, par exemple, une galerie d'images, alors grid est le meilleur choix :



Est-ce que CSS Grid rendra Flexbox obsolète à l'avenir ?

Absolument pas.

En fait, grid et Flexbox sont toutes deux conçues pour résoudre un ensemble différent de problèmes.

Actuellement, Flexbox couvre 99% de l'utilisation mondiale et Grid 93% de l'utilisation mondiale dans les navigateurs.

Bientôt, Grid bénéficiera également d'un très bon support parmi les navigateurs, et nous utiliserons un mélange de grid et de Flexbox pour créer des mises en page de sites Web incroyables qui n'étaient pas possibles auparavant.

Quelques ressources sur Flexbox et Grid

Flexbox

Les concepts de base pour flexbox

Comment fonctionne Flexbox - expliqué avec de grands gifs colorés et animés + ***Flexbox tutoriel animé***

https://codepen.io/enxaneta/full/adLPwv
https://codepen.io/justd/full/yydezN

https://codepip.com/games/flexbox-froggy/

Css grid

Les concepts de base des grilles CSS

CSS Grid layout + ***A simple visual cheatsheet*** + ***Grid par l'exemple***

https://codepen.io/anthonydugois/full/RpYBmy
https://cssgrid-generator.netlify.com/

https://cssgridgarden.com/#fr