CSS: Le Display grid et les animations

Par Robert DIASSÉ

1. Introduction au Display Grid

Le Display Grid crée une grille bidimensionnelle composée de lignes et de colonnes, dans laquelle les éléments HTML peuvent être placés de manière précise. Voici les principaux concepts du Display Grid :

Déclaration du Conteneur Grid

Pour déclarer un conteneur comme une grille, utilisez la propriété `display: grid;` sur l'élément parent.

Exemple :

```
.container {
  display: grid;
}
```

Création de Lignes et de Colonnes

Vous pouvez définir des lignes et des colonnes dans la grille en utilisant les propriétés `grid-template-rows` et `grid-template-columns`.

Exemple :

```
.container {
  display: grid;
  grid-template-rows: 100px 200px; /* Crée deux lignes de hauteurs différentes */
  grid-template-columns: 1fr 2fr; /* Crée deux colonnes avec des largeurs
flexibles */
}
```

Placement des Éléments dans la Grille

Les éléments HTML peuvent être placés dans la grille en utilisant les propriétés `grid-row` et `grid-column`.

Exemple :

```css
.item {
  grid-row: 1 / 3; /* Place l'élément sur les lignes 1 à 3 */
  grid-column: 2 / 4; /* Place l'élément sur les colonnes 2 à 4 */
}
```

2. Propriétés Importantes du Display Grid

`grid-template-areas`

La propriété `grid-template-areas` permet de définir des zones nommées dans la grille, facilitant ainsi l'organisation visuelle des éléments.

Exemple :

```css
.container {
  display: grid;
  grid-template-areas:
    "header header header"
    "sidebar main main"
    "footer footer footer";
}
```

`grid-gap`

La propriété `grid-gap` définit l'espacement entre les lignes et les colonnes de la grille.

Exemple :

```css
.container {
  display: grid;
  grid-gap: 20px; /* Espacement de 20 pixels entre les lignes et les colonnes */
}
```

`justify-items` et `align-items`

Ces propriétés permettent d'aligner les éléments dans les cellules de la grille sur l'axe horizontal et vertical.

Exemple :

```css
.container {
  display: grid;
  justify-items: center; /* Centre les éléments horizontalement */
  align-items: center; /* Centre les éléments verticalement */
}
```

Ateliers Pratiques

Atelier : Créer une Grille

1. Créez un document HTML avec un conteneur div ayant la classe "grid-container".
2. Appliquez le style CSS suivant pour déclarer le conteneur comme une grille et définir deux lignes et trois colonnes.

```
.grid-container {
  display: grid;
  grid-template-rows: 100px 200px;
  grid-template-columns: 1fr 2fr 1fr;
}
```

3. Ajoutez quelques éléments enfants dans le conteneur et utilisez les propriétés `grid-row` et `grid-column` pour les placer dans la grille.

Utiliser `grid-template-areas`

1. **Modifier le Document HTML :**
   - Ouvrez votre document HTML existant.
   - Identifiez le conteneur grid-container où vous avez défini votre grille.
   - Remplacez les propriétés `grid-template-rows` et `grid-template-columns` par la propriété `grid-template-areas`.
2. **Définir les Zones Nommées :**
   - Utilisez des chaînes de caractères pour définir des zones nommées à l'aide de la propriété `grid-template-areas`.
   - Chaque ligne de la chaîne représente une ligne dans la grille et chaque caractère représente une cellule.
   - Utilisez des points (.) pour représenter des cellules vides et des lettres ou des noms pour identifier les zones.

Exemple :

```
.grid-container {
  display: grid;
  grid-template-areas:
    "header header header"
    "sidebar main main"
    "footer footer footer";
}
```

3. **Organiser les Éléments dans les Zones :**

- Utilisez la propriété `grid-area` sur chaque élément pour spécifier dans quelle zone il doit être placé.
- Utilisez les mêmes noms de zones que ceux définis dans `grid-template-areas`.

Exemple :

```
/*ces calsses ont été donné aux fils du container grid pour les repartir selon le grid-template-area */
.item-header {
  grid-area: header;
}

.item-sidebar {
  grid-area: sidebar;
}

.item-main {
  grid-area: main;
}

.item-footer {
  grid-area: footer;
}
```

Expérimenter avec `grid-gap`, `justify-items` et `align-items` pour le conteneur et `align-self` et `justify-self` pour les fils

1. **Ajouter des Espacements entre les Lignes et les Colonnes :**
   - Utilisez la propriété `grid-gap` pour spécifier l'espacement entre les lignes et les colonnes de la grille.
   - Vous pouvez utiliser des valeurs en pixels, en pourcentage ou d'autres unités de mesure.

Exemple :

```
.grid-container {
  display: grid;
  grid-gap: 20px; /* Espacement de 20 pixels entre les lignes et les colonnes */
}
```

2. **Aligner les Éléments dans la Grille :**
   - Utilisez les propriétés `justify-items` et `align-items` pour aligner les éléments dans la grille selon l'axe horizontal et vertical.
   - Vous pouvez utiliser les valeurs `start`, `end`, `center`, `stretch`, etc.

Exemple :

```
.grid-container {
  display: grid;
  justify-items: center; /* Centre les éléments horizontalement */
  align-items: center; /* Centre les éléments verticalement */
}
```

Suivez ces étapes simples et approfondissez les, vous pourrez facilement expérimenter avec les fonctionnalités avancées du Display Grid, y compris l'utilisation de `grid-template-areas` pour organiser visuellement les éléments dans des zones nommées, ainsi que l'ajout d'espacements et l'alignement des éléments dans la grille.

Les Animations CSS : Concepts de Base et Utilisation Pratique

1. Animations Simples sans Keyframes

Les animations simples en CSS peuvent être appliquées en utilisant les propriétés `transition` et `transform`. Voici quelques techniques courantes :

- **Transition** : La propriété `transition` permet de spécifier le changement progressif d'une propriété CSS sur une certaine durée.

  Exemple :

  ```
  .element {
    transition: transform 0.3s ease-in-out;
  }

  .element:hover {
    transform: scale(1.2);
  }
  ```

  Dans cet exemple, lorsqu'un élément est survolé, il sera agrandi avec une transition fluide de 0.3 seconde.

- **Transformations 2D et 3D** : Les transformations `translate`, `rotate`, `scale` et `skew` peuvent être utilisées pour animer les éléments dans l'espace 2D ou 3D.

  Exemple :

  ```
  .element {
    transition: transform 0.3s ease-in-out;
  }

  .element:hover {
    transform: rotate(45deg);
  }
  ```

Cet exemple fait tourner l'élément de 45 degrés lorsqu'il est survolé.

2. Utilisation de Keyframes pour des Animations Personnalisées

Les keyframes en CSS permettent de définir des étapes intermédiaires d'une animation en spécifiant explicitement les états initiaux et finaux ainsi que les états intermédiaires.

- **Définition de Keyframes** : Les keyframes sont définis avec l'at-rule `@keyframes`, suivi d'un nom personnalisé.

  Exemple :

```
@keyframes shake {
  0% { transform: translateX(0); }
  25% { transform: translateX(10px); }
  50% { transform: translateX(-10px); }
  75% { transform: translateX(10px); }
  100% { transform: translateX(0); }
}
```

- **Application des Keyframes** : Les keyframes définis peuvent ensuite être appliqués à un élément en utilisant la propriété `animation`.

  Exemple :

```
.element {
  animation: shake 0.5s infinite;
}
```

- **Compréhension des Valeurs de Timing** : Les valeurs comme `ease`, `ease-in`, `ease-out`, `ease-in-out`, `linear`, etc., définissent comment l'animation progresse dans le temps. Par exemple :
  - `ease` : L'animation commence lentement, accélère puis ralentit à la fin.
  - `ease-in` : L'animation commence lentement et accélère progressivement.
  - `ease-out` : L'animation commence rapidement et ralentit progressivement.
  - `ease-in-out` : L'animation commence lentement, accélère, puis ralentit à nouveau.
  - `linear` : L'animation progresse de manière linéaire à travers le temps.

Atelier : Création d'une Animation de Rotation

Étapes à Suivre :

1. **Définir les Keyframes :**
   - Utilisez l'at-rule `@keyframes` pour définir les étapes intermédiaires de votre animation.
   - Définissez les transformations nécessaires à chaque étape.

Exemple :

```
@keyframes rotate {
  0% { transform: rotate(0deg); }
  100% { transform: rotate(360deg); }
}
```

2. **Appliquer l'Animation à un Élément :**
   - Utilisez la propriété `animation` pour appliquer les keyframes à un élément spécifique.
   - Spécifiez la durée, le type de timing et d'autres options d'animation selon vos besoins.

   Exemple :

```
.element {
  animation: rotate 2s linear infinite;
}
```

les principales propriétés CSS flex et grid(conteneur et fils)

voici les principales propriétés CSS pour les conteneurs flex et les éléments enfants (items) flex, ainsi que pour les conteneurs grid et les éléments enfants (grilles) :

Pour les conteneurs Flex :

Propriétés pour le conteneur (flex container) :

1. `display`: Définit le type de boîte utilisé pour un élément. La valeur `flex` crée un conteneur flex.
2. `flex-direction`: Définit la direction principale dans laquelle les éléments flexibles sont affichés dans le conteneur. Les valeurs possibles sont `row`, `row-reverse`, `column`, `column-reverse`.
3. `flex-wrap`: Spécifie si les éléments flexibles doivent être enroulés dans plusieurs lignes (ou colonnes) ou non. Les valeurs possibles sont `nowrap`, `wrap`, `wrap-reverse`.
4. `justify-content`: Définit l'alignement le long de l'axe principal du conteneur flex. Les valeurs possibles sont `flex-start`, `flex-end`, `center`, `space-between`, `space-around`, `space-evenly`.
5. `align-items`: Définit l'alignement des éléments flexibles sur l'axe transversal du conteneur flex. Les valeurs possibles sont `stretch`, `flex-start`, `flex-end`, `center`, `baseline`.
6. `align-content`: Contrôle l'alignement des lignes flexibles dans le conteneur flex lorsque les espaces libres sont présents sur l'axe transversal. Les valeurs possibles sont `flex-start`, `flex-end`, `center`, `space-between`, `space-around`, `stretch`.

Propriétés pour les éléments enfants (items) :

1. `order`: Définit l'ordre des éléments flexibles à l'intérieur du conteneur flex.
2. `flex-grow`: Détermine la capacité de l'élément à s'étendre pour remplir l'espace disponible dans le conteneur flex.

3. `flex-shrink`: Détermine la capacité de l'élément à se rétrécir pour éviter le dépassement de l'espace disponible dans le conteneur flex.
4. `flex-basis`: Définit la taille de base de l'élément flexible avant tout espace restant est distribué.
5. `flex`: Réunit `flex-grow`, `flex-shrink`, et `flex-basis` en une seule propriété.
6. `align-self`: Permet à un élément de remplacer l'alignement défini par `align-items` pour son propre affichage.

Pour les conteneurs Grid :

Propriétés pour le conteneur (grid container) :

1. `display`: Définit le type de boîte utilisé pour un élément. La valeur `grid` crée un conteneur grid.
2. `grid-template-columns`: Définit la taille des colonnes dans la grille.
3. `grid-template-rows`: Définit la taille des lignes dans la grille.
4. `grid-template-areas`: Définit la disposition des zones nommées dans la grille.
5. `grid-template`: Regroupe `grid-template-rows`, `grid-template-columns`, et `grid-template-areas` en une seule propriété.
6. `grid-gap`: Définit l'espacement entre les rangées et les colonnes dans la grille.
7. `justify-items`: Aligne les éléments enfants (grilles) le long de l'axe de la colonne dans la grille.
8. `align-items`: Aligne les éléments enfants (grilles) le long de l'axe de la ligne dans la grille.
9. `justify-content`: Aligne le contenu de la grille le long de l'axe principal (horizontal).
10. `align-content`: Aligne le contenu de la grille le long de l'axe transversal (vertical).

Propriétés pour les éléments enfants (grid items) :

1. `grid-column-start`, `grid-column-end`, `grid-row-start`, `grid-row-end`: Spécifie les lignes et les colonnes de départ et de fin pour un élément grid.
2. `grid-column`, `grid-row`: Réunit `grid-column-start` et `grid-column-end`, ainsi que `grid-row-start` et `grid-row-end` respectivement en une seule propriété.
3. `grid-area`: Réunit `grid-row-start`, `grid-column-start`, `grid-row-end` et `grid-column-end` en une seule propriété(nommée).
4. `justify-self`: Aligne un élément grid le long de l'axe de la colonne.
5. `align-self`: Aligne un élément grid le long de l'axe de la ligne.

Maintenant que vous avez un aperçu de ces concepts clès approfondissez en ajoutant des propriété et en modifiant le comportement aussi de votre grille que de vos animations ces liens vont vous y aider :

pour les alignement dont le grid

MDN

LACONSOLE

JENSEIGN

JENSEIGN

pour les animations avec lesquelles vous pouvez vous inspirer, comprendre et approfondire

HUBSPOT