CSS: La responsivité et les bonnes pratiques

Par Robert DIASSÉ

La Responsivité en Développement Web

La responsivité en développement web désigne la capacité d'un site web ou d'une application web à s'adapter et à fonctionner correctement sur différents appareils et tailles d'écrans, tels que les ordinateurs de bureau, les tablettes et les smartphones. Elle implique la création de mises en page flexibles, de grilles adaptables et de styles qui s'ajustent en fonction de la taille de l'écran.

Bonnes Pratiques pour la Responsivité

Voici quelques bonnes pratiques à suivre pour assurer la responsivité d'un site web :

1. **Utilisation de Mises en Page Flexibles :** Utilisez les propriétés CSS flexbox pour créer des mises en page flexibles qui s'adaptent facilement à différentes tailles d'écrans.

   Exemple :

   ```
   .container {
       display: flex;
       flex-wrap: wrap; /* Permet aux éléments de passer à la ligne si
   nécessaire */
       justify-content: space-between; /* Espacement égal entre les éléments */
   }
   ```

2. **Utilisation de Grilles Adaptables :** Utilisez des systèmes de grille CSS comme CSS Grid ou Bootstrap Grid pour créer des mises en page qui s'ajustent automatiquement en fonction de la taille de l'écran.

   Exemple :

   ```
   .grid-container {
       display: grid;
       grid-template-columns: repeat(auto-fit, minmax(250px, 1fr)); /* Colonnes
   adaptatives */
       gap: 20px; /* Espacement entre les éléments */
   }
   ```

3. **Utilisation d'Unités de Mesure Relatives :** Privilégiez l'utilisation d'unités de mesure relatives comme les pourcentages (%), les vh (viewport height), les vw (viewport width), les em et les rem plutôt que les unités absolues comme les pixels (px). Cela permettra à vos éléments de s'adapter en fonction de la taille de l'écran.

   Exemple :

```
.element {
    width: 50%; /* Largeur de 50% par rapport à son conteneur */
    font-size: 1.2em; /* Taille de police relative */
}
```

Bien sûr ! Voici une explication détaillée des fonctions avancées CSS mentionnées :

4. **Utilisation de Fonctions Avancées CSS :**

   Les fonctions avancées CSS telles que `calc()`, `min()`, `max()`, et `clamp()` sont très utiles pour créer des mises en page flexibles et dynamiques en ajustant les valeurs des propriétés CSS en fonction de différents facteurs.

   - `calc()` **:** Cette fonction permet d'effectuer des calculs simples pour définir les valeurs des propriétés CSS. Elle peut être utilisée pour combiner différentes unités de mesure, effectuer des opérations mathématiques, ou définir des valeurs en pourcentage.

     Exemple :

```
.element {
    width: calc(50% - 20px); /* Calcul de la largeur avec un espace de
20px */
}
```

   - `min()` **:** Cette fonction retourne la plus petite valeur parmi les valeurs fournies. Elle est utile pour définir une valeur minimale pour une propriété CSS.

     Exemple :

```
.element {
    width: min(50%, 300px); /* La largeur sera de 50% de la largeur
parente ou de 300px, selon la plus petite valeur */
}
```

   - `max()` **:** Cette fonction retourne la plus grande valeur parmi les valeurs fournies. Elle est utile pour définir une valeur maximale pour une propriété CSS.

     Exemple :

```
.element {
    width: max(50%, 300px); /* La largeur sera de 50% de la largeur
parente ou de 300px, selon la plus grande valeur */
}
```

- `clamp()` **:** Cette fonction permet de définir une valeur comprise entre une valeur minimale et une valeur maximale. Elle est utile pour définir des valeurs de propriétés CSS qui doivent être limitées à une plage spécifique.

  Exemple :

```
.element {
    width: clamp(200px, 50%, 500px); /* La largeur sera comprise entre
200px et 500px, mais pas en dehors de cette plage */
}
```

Dans cet exemple, la largeur de l'élément sera définie selon les règles suivantes :

- Si la largeur de l'élément parent est inférieure à `200px`, alors la largeur de l'élément sera `200px`.
- Si la largeur de l'élément parent est comprise entre `200px` et `500px`, alors la largeur de l'élément sera égale à `50%` de la largeur de son parent.
- Si la largeur de l'élément parent est supérieure à `500px`, alors la largeur de l'élément sera `500px`.

Ainsi, dans le contexte de la fonction `clamp()`, le `50%` représente une valeur cible ou préférée pour la largeur de l'élément lorsque les conditions le permettent. Si la largeur de l'élément parent est comprise dans la plage définie par `200px` et `500px`, l'élément prendra `50%` de la largeur de son parent, ce qui peut être considéré comme une valeur optimale dans certaines situations.

Utilisation des Media Queries pour la Responsivité

Les media queries sont des règles CSS qui permettent d'appliquer des styles différents en fonction des caractéristiques du périphérique, telles que la largeur de l'écran, la résolution, l'orientation, etc. Elles sont essentielles pour rendre un site web réactif et adaptatif à différents appareils. Voici comment les utiliser avec des exemples détaillés :

1. Media Queries pour les Tailles d'Écran Courantes :

Les tailles d'écran les plus courantes pour lesquelles vous voudrez définir des styles différents sont celles des appareils mobiles, des tablettes et des ordinateurs de bureau. Voici des exemples de media queries pour ces tailles d'écran :

- **Mobiles (inférieur à 768px)** :

```
@media screen and (max-width: 767px) {
    /* Styles à appliquer pour les appareils mobiles */
}
```

- **Tablettes (768px - 1023px)** :

```
@media screen and (min-width: 768px) and (max-width: 1023px) {
    /* Styles à appliquer pour les tablettes */
}
```

- **Ordinateurs de Bureau (1024px et plus)** :

```
@media screen and (min-width: 1024px) {
    /* Styles à appliquer pour les ordinateurs de bureau */
}
```

2. Media Queries pour l'Orientation de l'Écran :

Vous pouvez également utiliser les media queries pour ajuster le style en fonction de l'orientation de l'écran, c'est-à-dire si l'appareil est en mode paysage ou portrait. Voici un exemple :

- **Portrait** :

```
@media screen and (orientation: portrait) {
    /* Styles à appliquer pour l'orientation portrait */
}
```

- **Paysage** :

```
@media screen and (orientation: landscape) {
    /* Styles à appliquer pour l'orientation paysage */
}
```

3. Media Queries pour les Résolutions d'Écran :

Parfois, vous voudrez appliquer des styles différents en fonction de la résolution de l'écran. Voici comment le faire avec des exemples :

- **Résolution Retina (densité de pixels élevée)** :

```
@media screen and (-webkit-min-device-pixel-ratio: 2), (min-resolution:
192dpi) {
    /* Styles à appliquer pour les écrans à haute résolution (Retina) */
}
```

- **Résolution Standard (densité de pixels normale)** :

```
@media screen and (-webkit-min-device-pixel-ratio: 1), (min-resolution: 96dpi) {
    /* Styles à appliquer pour les écrans à résolution standard */
}
```

Exemple :

Voici un exemple complet montrant comment utiliser les media queries pour créer une mise en page réactive :

```
/* Styles de base pour tous les appareils */
.element {
    width: 100%;
}

/* Media Queries pour les appareils mobiles */
@media screen and (max-width: 767px) {
    .element {
        font-size: 14px;
    }
}

/* Media Queries pour les tablettes */
@media screen and (min-width: 768px) and (max-width: 1023px) {
    .element {
        font-size: 16px;
    }
}

/* Media Queries pour les ordinateurs de bureau */
@media screen and (min-width: 1024px) {
    .element {
        font-size: 18px;
    }
}
```

En utilisant des media queries de cette manière, vous pouvez créer des mises en page et des styles qui s'adaptent de manière fluide à différentes tailles d'écrans, offrant ainsi une expérience utilisateur optimale sur tous les appareils.

Exercices

1. **Création d'une Mise en Page Flexible avec Flexbox :**
    - Utilisez les propriétés flexbox pour créer une mise en page flexible qui s'adapte à différentes tailles d'écrans.
    - Atelier : Créez une grille de produits avec Flexbox et assurez-vous qu'elle s'ajuste correctement sur les appareils mobiles et les tablettes.

Exemple : Grille de Produits avec Flexbox

```
<div class="grid-container">
    <div class="product">Produit 1</div>
    <div class="product">Produit 2</div>
    <div class="product">Produit 3</div>
    <!-- Ajoutez plus de produits ici -->
</div>
```

```
.grid-container {
    display: flex;
    flex-wrap: wrap;
    justify-content: space-between;
}

.product {
    width: calc(33.33% - 20px); /* 3 colonnes avec un espacement de 20px */
    margin-bottom: 20px;
    background-color: #f2f2f2;
    padding: 20px;
    box-sizing: border-box;
}
```

Dans cet exemple, nous utilisons Flexbox pour créer une grille de produits. La classe `.grid-container` utilise `display: flex` pour créer un conteneur flexible qui dispose les éléments sur une ligne. La propriété `flex-wrap: wrap` permet aux éléments de passer à la ligne suivante lorsque l'espace est insuffisant. En ajustant la largeur des éléments avec `width`, nous obtenons une disposition flexible qui s'adapte automatiquement à différentes tailles d'écrans.

2. **Utilisation d'une Grille Adaptable avec CSS Grid :**
    - Utilisez CSS Grid pour créer une grille adaptable qui réorganise les éléments en fonction de la taille de l'écran.
    - Atelier : Créez une mise en page de blog avec CSS Grid et assurez-vous qu'elle s'adapte aux différentes tailles d'écrans.
      Exemple : Mise en Page de Blog avec CSS Grid

```
<div class="grid-container">
    <div class="post">Article 1</div>
    <div class="post">Article 2</div>
    <div class="post">Article 3</div>
    <!-- Ajoutez plus d'articles ici -->
</div>
```

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

```
    grid-template-columns: repeat(auto-fill, minmax(300px, 1fr));
    grid-gap: 20px;
}

.post {
    background-color: #f2f2f2;
    padding: 20px;
    box-sizing: border-box;
}
```

Dans cet exemple, nous utilisons CSS Grid pour créer une mise en page de blog. La classe `.grid-container` utilise `display: grid` pour créer un conteneur de grille. Avec `grid-template-columns: repeat(auto-fill, minmax(300px, 1fr))`, nous définissons des colonnes de taille flexible qui s'adaptent à la taille de l'écran. La propriété `grid-gap` ajoute un espacement entre les éléments de la grille.

3. **Utilisation des Media Queries pour la Responsivité :**
    - Utilisez les media queries pour appliquer des styles différents en fonction de la taille de l'écran.
    - Atelier : Définissez des media queries pour ajuster la taille du texte et la disposition des éléments sur les appareils mobiles et les tablettes.

```
/* Styles de base pour tous les appareils */
.element {
    width: 100%;
}

/* Media Queries pour les appareils mobiles */
@media screen and (max-width: 767px) {
    .element {
        font-size: 14px;
        text-align: center;
    }
}

/* Media Queries pour les tablettes */
@media screen and (min-width: 768px) and (max-width: 1023px) {
    .element {
        font-size: 16px;
        text-align: left;
    }
}

/* Media Queries pour les ordinateurs de bureau */
@media screen and (min-width: 1024px) {
    .element {
        font-size: 18px;
        text-align: right;
    }
}
```

Dans cet exemple, nous utilisons les media queries pour ajuster la taille du texte et la disposition des éléments en fonction de la taille de l'écran. Sur les appareils mobiles, le texte est centré, sur les tablettes il est aligné à gauche, et sur les ordinateurs de bureau il est aligné à droite. Cela garantit une lisibilité optimale et une présentation cohérente sur tous les appareils.

Suivez ces bonnes pratiques et en utilisant les outils CSS appropriés , vous pouvez créer des sites web responsifs qui offrent une expérience utilisateur optimale sur tous les appareils.
Dans l'approfondissement de ces concepts vous pouvez vous aider des sites suivants:

OPENCLASSROOM

Pierre Giraud

LE HOLLANDAIS VOLANT