Le Responsive Web Design

1. Qu'est-ce que le Responsive Web Design ?

> Le Responsive Web Design (RWD) est une méthode de conception qui permet aux sites web de s'adapter à tous les types d'écrans (ordinateurs, tablettes, smartphones). Son objectif est d'offrir une expérience utilisateur fluide en ajustant la mise en page et les éléments en fonction de la taille du dispositif.

2. Les principes fondamentaux du Responsive Web Design

2.1. Le hack HTML avec 62.5%

Le hack HTML consiste à définir la taille de la police de base du site à `62.5%` dans le CSS :

```css
html {
  font-size: 62.5%; /* (62.5 * 16) / 100 = 10px */
}
```

Cela permet de simplifier les calculs d'unités relatives comme le `rem` .

- 1 rem correspond alors à 10px (au lieu des 16px par défaut).
- Cela facilite le calcul des tailles (`font-size` , `width` , `height` , `margin` , `padding`) dans le code css.

Si un utilisateur modifie la taille de police par défaut de son navigateur (par exemple, de 16px à 24px), toutes les dimensions définies en `rem` s'adapteront automatiquement.

Accessibilité améliorée :

- Les utilisateurs malvoyants qui augmentent la taille de police dans leur navigateur verront les dimensions de ton site (comme les marges, les paddings, les largeurs, etc.) s'ajuster proportionnellement, ce qui rendra ton site plus lisible et utilisable.

2.2. Fonctions responsives

> Pour ajuster les dimensions ou tailles de texte, deux approches efficaces sont utilisées.

2.2.1. Les rem dans les media queries

Les unités `rem` permettent de définir des tailles relatives à la taille de la police racine (`html`), ce qui apporte une flexibilité et facilite les ajustements en cascade.

Exemple :

```css
@media (max-width: 60rem) { /* Équivalent à 960px si le font-size est 62.5% */
  body {
    font-size: 1.4rem;
  }
}
```

2.2.2. Les clamp()

La fonction `clamp()` combine plusieurs tailles pour adapter les éléments de manière fluide entre des valeurs minimales et maximales.

- Syntaxe : `clamp(valeur_min, valeur_principale, valeur_max)`
- Exemple pour une taille de texte :

```css
font-size: clamp(1rem, 2.5vw, 2rem);
```

Ici, la taille du texte varie entre 1rem et 2rem, en fonction de la largeur de la fenêtre (`vw`).

3. Approches pour les media queries

3.1. Mobile first (*min-width*)

> L'approche "mobile first" commence par concevoir pour les petits écrans, puis affine le design pour les écrans plus grands à l'aide de media queries avec `min-width`.

Exemple :

```css
body {
  font-size: 1.4rem;
}

@media (min-width: 48em) { /* Équivalent à 768px */
  body {
    font-size: 1.6rem;
  }
}
```

Justification :

- Cette méthode est souvent recommandée car elle commence par des styles simples et légers pour les mobiles, adaptés aux contraintes de performance.

3.2. Desktop first (`max-width`)

> L'approche "desktop first" débute par concevoir pour les grands écrans, en appliquant des media queries avec `max-width` pour les écrans plus petits.

Exemple :

```css
body {
  font-size: 1.8rem;
}

@media (max-width: 48em) { /* Équivalent à 768px */
  body {
    font-size: 1.4rem;
  }
}
```

Justification :

- Cette méthode peut être utile si la majorité des utilisateurs accèdent au site depuis des écrans larges, ou si le contenu principal est plus adapté aux grands écrans.

4. L'utilisation des unités em dans les media queries

Utiliser les `em` au lieu des `px` dans les media queries améliore l'adaptabilité et évite des incohérences liées aux différentes résolutions.

Les media queries sont conçues pour être indépendantes du contenu de la page.

Elles se basent uniquement sur les dimensions du `viewport` (largeur et hauteur).

Dans les media queries, les unités `em` :

- ne se basent pas sur la taille de police de l'élément parent (comme c'est le cas pour les propriétés CSS classiques).
- se basent sur la taille de police de la racine, c'est-à-dire la taille de police définie sur la balise `<html>`.

Important ! :

- Sans modification de la taille de police sur `<html>` :
  - Les `em` dans les media queries se basent sur la taille par défaut du navigateur (16px).
- Avec une modification de la taille de police sur `<html>` :
  - Les `em` dans les media queries se basent sur la taille de police définie sur la balise `<html>` (par exemple avec `font-size: 62.5%`), cette nouvelle valeur devient la base pour les calculs des `em` dans les media queries.

- *Scénario 1 : aucune taille de police n'est définie sur la balise* `<html>`

Le navigateur utilise sa valeur par défaut (16px) alors dans les média-queries 1em = 16px.

Ainsi un point de rupture observé à 800px sur l'écran se traduira par en unité `em` par : 800 / 16 = 50em

```css
@media (min-width: 50em) {
    /* styles */
}
```

- Scénario 2 : une taille de police est définie sur la balise `<html>`

Imaginons que l'on définit `font-size: 62.5%` sur la balise `<html>`, la taille de police de la racine devient alors 62.5 x 16 / 100 = 10px

Ainsi un point de rupture observé à 800px sur l'écran se traduira par en unité `em` par : 800 / 10 = 80em

Dans ce cas, la même `media-query` s'écrira alors :

```
@media (min-width: 80em) {
    /* styles */
}
```

5. À RETENIR

- Hack HTML et 62.5% : Simplifie les calculs d'unités `rem` pour une meilleure flexibilité.
- Deux approches pour les fonctions responsives :
  - `rem` dans les media queries : pour des ajustements clairs et structurés.
  - `clamp()` : offre des adaptations fluides entre des valeurs limites.
- Deux philosophies responsives :
  - Mobile first (`min-width`) : Recommandée pour des styles légers et performants sur mobiles.
  - Desktop first (`max-width`) : Pratique lorsqu'un site est principalement utilisé sur des grands écrans.
- Utiliser l'unité `em` dans les media queries permet un design plus adaptable, fluide et accessible que les pixels.