Sites web réactifs

De nos jours, la consultation d'un site web peut se faire sur des appareils très différents, par exemple : ordinateur de bureau, ordinateur portable, smartphone, téléviseur, montre connectée…

Pour permettre un affichage optimal sur tous ces supports, il est possible de recourir à différents **principes de conception de sites réactifs** - aussi appelé "**responsive web design**".

Sites web réactifs

Media queries

Cette technique permet l'**application de styles différents** en fonction du **contexte** :

- taille d'écran,
- navigateur de l'utilisateur,
- version imprimable,
- ...

Sites web réactifs

Media queries

Elle repose sur l'utilisation de **règles conditionnelles** au sein d'une feuille de styles, via la requête `@media`

```
main {
    width: 1200px;
    padding: 20px;
}

@media only screen and (max-width: 720px) {
    main {
        width: 100%;
        padding: 10px;
    }
}
```

Sites web réactifs

Media queries

La syntaxe de ces requêtes repose sur des **opérateurs logiques** :

- `and` : les styles sont appliqués si toutes les conditions sont tenues,
- `,` : les styles sont appliqués si au moins une des conditions est tenue,
- `not` : les styles ne doivent *pas* être appliqués si la condition est rencontrée,
- `only` : les styles ne sont appliqués *que* si la condition est rencontrée.

Sites web réactifs

Media queries

Le type de média est optionnel, et peut prendre les valeur suivants :

- **screen** : styles destinés aux écrans,
- **print** : styles destinés à l'impression et aux documents paginés (ex. PDF),
- **all** : valeur par défaut, correspondant à tous les types.

Sites web réactifs

Media queries

Enfin, il existe un certain nombre de caractéristiques de médias, par exemple :

- `min-width / max-width` : sans doute la plus utilisée, elle permet d'appliquer des styles **en fonction de la largeur de l'écran**, et sa valeur est généralement exprimée en `px`, mais il est possible aussi d'utiliser des `%` ou l'unité `vw` - viewport width, qui va de 0 à 100,

- `min-height / max-height` : même principe, mais **en fonction de la hauteur d'écran**,

- `orientation` : permet d'appliquer des styles **en fonction de l'orientation de l'appareil**, ce qui peut être utile sur mobile et tablette. Elle peut prendre les valeurs `portrait` ou `landscape` (paysage).

Sites web réactifs

Media queries

D'autres caractéristiques correspondent à des **réglages utilisateurs**, et peuvent aider à améliorer l’expérience utilisateur et l'accessibilité du site, par exemple :

- `prefers-reduced-motion` qui indique la préférence de l'utilisateur pour une limitation des animations sur la page,

- `prefers-color-scheme` qui permet d'ajuster automatiquement le thème du site en fonction de la préférence de l'utilisateur pour un thème clair (`light`) ou sombre (`dark`).

Sites web réactifs

Flexbox

CSS Flexible Box Layout est un modèle de mise en page sur **une seule dimension**, c'est-à-dire que l'organisation des éléments se fait dans une direction à la fois - ligne ou colonne - mais pas les deux ensembles.

Afin de transformer un élément HTML en **conteneur flexible**, on l'indique dans sa propriété `display` :

```
.container {
    display: flex; /* ou inline-flex */
}
```

Sites web réactifs

Flexbox

Propriétés des conteneurs flex :

- `flex-direction` permet d'indiquer la direction de l'**axe principal** sur lequel les descendants seront organisés. Elle peut prendre les valeurs `row` (par défaut), `row-reverse, column, column-reverse.`

- `flex-wrap` permet, lorsqu'elle prend la valeur `wrap`, de créer un conteneur flexible **sur plusieurs lignes**.
  A contrario, la valeur `nowrap` force les éléments à se redimensionner en empêchant le passage à la ligne - s'il n'y a pas assez d'espace disponible, cela causera un dépassement.

Sites web réactifs

Flexbox

Propriétés des conteneurs flex (suite) :

- `justify-content` définit l'**alignement** des descendants directs suivant l'**axe principal**, et peut prendre les valeurs `flex-start, flex-end, center, space-between, space-around, space-evenly, stretch…`
- `align-items` définit l'alignement des éléments suivant l'**axe transversal**, qui est toujours perpendiculaire à l'axe principal et peut prendre les valeurs `flex-start, flex-end, center, stretch…`
- `align-self` a un comportement comparable à `align-items` (et accepte les mêmes valeurs), mais il ne s'applique que sur les éléments ciblés par le sélecteur CSS, si leur parent est un conteneur flex.

Sites web réactifs

Flexbox

Attention, **ces propriétés ne s'héritent pas** : elle ne s'applique qu'à l'élément cible et/ou à ses descendants.

Toutes ces explications étant un peu arides, un développeur a décidé de créer une application permettant d'expérimenter visuellement les effets des différentes valeurs des propriétés des éléments Flex : Flexbox Froggy - https://flexboxfroggy.com/#fr

En bonus, un petit pense-bête pour éviter d'avoir à retenir toutes les propriétés ainsi que leurs valeurs possibles d'un seul coup : https://github.com/alsacreations/kiwipedia/blob/main/resources/flexbox-cheatsheet.png

Sites web réactifs

Pour aller plus loin…

- **CSS Grid Layout** est un autre modèle de mise en page via la propriété `display`, cette fois-ci **sur deux dimensions**.
  Le site interactif Grid Garden https://cssgridgarden.com/#fr permet d'en comprendre les ressorts.
  Pense-bête :
  https://github.com/alsacreations/kiwipedia/blob/main/resources/grid-cheatsheet.png

- Les **Container Queries** sont basés sur les mêmes principes que les Media Queries, mais au lieu de prendre comme référence les dimensions de la fenêtre, ils interrogent les dimensions de l'ancêtre d'un élément pour déterminer si des styles doivent être appliqués.
  Ressource en français :
  https://www.alsacreations.com/article/lire/1915-Les-Container-Queries-en-CSS.html