TD4

Site web adaptatif

Objectif :

1. Créer une page web adaptatif (Responsive Web Design) (viewport, media queries);
2. Utiliser une grille "à la main" en flottant ;
3. Utiliser le système de grille CSS3 CSS Grid Layout.

Site web adaptatif / Responsive Web Design

De nos jours un site web peut se consulter sur de nombreux appareils : écrans d'ordinateurs fixes ou portables, tablettes, téléphones mobiles, liseuses, ... etc.
Définition à retenir : un site web responsif est un site permettant une bonne expérience de lecture et de navigation sur n'importe quel appareil.

Vous allez voir dans ce TP les principaux moyens de réaliser de tels sites avec HTML 5 et CSS 3.

Documentations

Documentation w3school : https://www.w3schools.com/css/css_rwd_intro.asp
Documentation MDN : https://developer.mozilla.org/fr/docs/D%C3%A9veloppement_Web/Design_web_Responsive

1. RWD Viewport

Définition : Le "**viewport**" (qu'on pourrait traduire en français par "fenêtre de visualisation") correspond à la région visible de l'utilisateur. Elle sera par exemple plus petite sur un smartphone que sur un écran d'ordinateur.

La balise HTML utilisée pour donner au navigateur des instructions de contrôle des dimensions et de l'échelle d'une page web, est la balise **<meta >** avec l'attribut ''name'' définit autant que ''viewport''

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

- La propriété **width=device-width** demande à ce que la page **s'adapte à la largeur** de l'écran de l'appareil de l'utilisateur.
- La propriété **initial-scale=1.0** fixe le niveau de zoom initial à l'ouverture de la page.

➔ Documentation W3school sur RWD Viewport

2. RWD Grid View : avec Positionnement float et taille en pourcentage

Afin de mettre en page le contenu des pages HTML et construire un gabarit, on utilisera dans un premier temps le modèle de grille avec **des éléments positionnés en float** comme déjà vu en TD précédent et des **tailles exprimées en pourcentage** pour s'adapter automatiquement à la taille de l'écran.

➔Documentation w3school sur RWD Grid

La majorité des pages HTML sont basé sur un modèle de grille avec 12 colonnes**, c**haque ligne du site est donc constituée de blocs occupant les 12 colonnes. Exemples :

- Une ligne avec un bloc occupant 4 colonnes et un bloc de 8 colonnes.
- Une ligne composée de 4 blocs de 3 colonnes chacun.
- Une ligne composée d'un bloc de 12 colonnes.

Chaque ligne devra occuper les 12 colonnes. On ne peut pas faire une ligne de 2 blocs de 4 colonnes par exemple.

Prenons l'exemple de ce site ci-dessous :

Ce site comporte 3 lignes :

- La première ligne (bloc titre en-tête "header" contenant "Mon site") est composée d'un bloc couvrant les 12 colonnes.
- La deuxième ligne est composée de 3 blocs :
  - Un premier bloc de 3 colonnes pour le menu.
  - Un deuxième bloc de 6 colonnes pour le contenu de la page.
  - Un troisième bloc de 3 colonnes pour le "blabla annexe".
- La troisième et dernière ligne (bloc bas de page "footer" contenant "Blabla bas de page") est composée d'un bloc couvrant les 12 colonnes.

Réalisons ce site :

Appliquons le style suivant (tout le code CSS suivant est dans le fichier siteGrillePourcentages.css, suivez ces instructions pour comprendre le style) :

- Pour commencer, il faut s'assurer que la marge intérieure, la marge extérieure ainsi que la bordure soient incluses dans la largeur et la hauteur des éléments (sinon les bordures des blocs se chevauchent).

```
*{
    box-sizing: border-box;
}
```

➔Documentation w3school sur la propriété box-sizing

- Créer des classes pour chaque colonne et préciser la largeur de cette dernière. Afin de découper la largeur de l'écran en 12 colonnes, il faut que chaque colonne ait une largeur de 100/12=8,33%, ce qui donne le style css suivant :

```css
.col-1 {width: 8.33%;}
.col-2 {width: 16.66%;}
.col-3 {width: 25%;}
.col-4 {width: 33.33%;}
.col-5 {width: 41.66%;}
.col-6 {width: 50%;}
.col-7 {width: 58.33%;}
.col-8 {width: 66.66%;}
.col-9 {width: 75%;}
.col-10 {width: 83.33%;}
.col-11 {width: 91.66%;}
.col-12 {width: 100%;}
```

- Mettre ces blocs en positionnement float à gauche, et ajouter une marge intérieure. Afin de sélectionner toutes ces classes en même temps nous utilisons un sélecteur d'attribut sélectionnant les classes dont le nom contient le texte 'col-''.

```css
[class*="col-"] {
  float: left;
  padding: 15px;
}
```

- Pour créer une ligne (row), il suffit d'affecter aux **blocs HTML** de chaque ligne les classes crées (col-1 .. col-12) en s'assurant que la somme des colonnes est 12 colonnes ç-à-d 100% de la largeur de la page.
  Exemple :
  Voici le code HTML d'une ligne composée de deux blocs :
  - le premier occupe 3 colonnes (25% de la largeur) ➔ class : col-3
  - et le deuxième occupe 9 colonnes (75% de la largeur)➔ class : col-9

```html
<div class="row">
    <div class="col-3">...</div> <!-- 25% -->
    <div class="col-9">...</div> <!-- 75% -->
</div>
```


- Pour que chaque ligne se finissent bien par un retour à la ligne (arrêt du positionnement flottant) et l'espace soit bien occupé (display:block) voici les propriétés à mettre pour la classe row :

```css
.row::after {
  content: "";
  clear: both;
  display: block;
}
```

- Et donc le code HTML permettant de créer le site à réaliser est dans le fichier html **siteGrillePourcentages.html.** Observez le code et les classes choisies pour les balises HTML
- Complétez le code CSS du fichier **siteGrillePourcentages.css**

3. RWD Media Queries

Pour mieux encore vous adapter aux différents écrans, vous pouvez modifier l'agencement des blocs, par exemple passer d'un menu horizontal sur grand écran à un menu vertical sur un petit écran, voire enlever du contenu. Ceci est possible grâce aux **media queries** avec la règle **@media**.

➔Documentation sur les media queries : du w3school et du MDN

La règle **@media** permet d'exécuter un bloc de propriétés CSS seulement si une certaine condition est vraie.

Exemples :

- Pour appliquer un style pour les tailles de fenêtre de navigateur **plus petites que 600px** :

```
@media only screen and (max-width: 600px) {

/* Règles CSS pour ce cas*/

}
```

- Pour appliquer un style pour les tailles de fenêtre de navigateur **plus grandes que 600px**

```
@media only screen and (min-width: 600px) {

/* Règles CSS pour ce cas*/

}
```

Il est conseillé de démarrer le design pour un petit écran puis d'ajouter des media queries avec l'augmentation de la taille de l'écran (approche dîtes "mobile first").

Par exemple, pour faire une adaptation aux petits écrans (exemple d'un téléphone), écrans moyens (exemple d'une tablette), et grand (exemple de l'écran d'un ordinateur) on fera :

```
/* Style 1 (sans media queries): écran plus petit que 600px; */

@media only screen and (min-width: 600px) {
/* Style 2 (avec media queries): Modification de style pour les plus de 600px*/
}

@media only screen and (min-width: 768px) {
/* Style 3 (avec media queries): Modification de design pour les plus de 768px*/
}
```

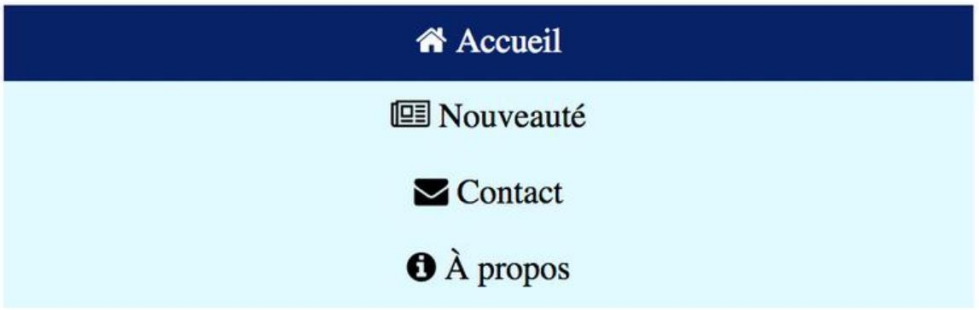
- Modifier le CSS du précédent site tel que si la taille de fenêtre est plus petite que 768px alors l'affichage soit celui illustré ci-dessous (méthode non "mobile first")
  Astuce : modifier la largeur des colonnes en 100%.
  Tester la page en redimensionnant la fenêtre du navigateur.

Menu horizontal Web responsive

- Reprenez maintenant votre menu horizontal réalisé au TD précédent
- Rendez ce site Web responsive pour les écrans plus petits que 500px.

➔Illustration du menu horizontal avec des icônes quand la taille de fenêtre est plus grande que 500px.

➔Illustration du menu horizontal avec des icônes quand la taille de fenêtre est plus petite que 500px.



N'oubliez pas d'ajouter à votre fichier HTML la balise meta viewport

4. CSS Grid Layout

Vous avez vu en début de TD comment faire "à la main" une grille de 12 colonnes avec un positionnement flottant et des tailles en pourcentage.

Depuis CSS a évolué et le concept de grille CSS : **CSS grid layout** a été créé.

Cette grille est facile à utiliser, il suffit de mettre la propriété **display: grid;** dans un conteneur parent en définissant le nombre de colonnes et de grilles ainsi que leurs dimensions avec par exemple **grid-template-columns: 100px 100px 100px;** (la grille est composée de 3 colonnes de 100px de large) et **grid-template-rows: 50px 50px;** (la grille est composée de 2 lignes de 50px de hauteur)

Les enfants (blocs contenus dans le parent) occuperont alors des cases de cette grille.

Pour étudier le fonctionnement et l'apprendre ! Consulter d'abord la documentation w3schools (grid intro / grid container / grid item)**,** ensuite faites ce petit tuto de 28 niveaux très bien fait pour bien assimiler les propriétés de base: http://cssgridgarden.com/#fr .

- Réaliser le même site (similaire) en utilisant le système de grille CSS3, que celui fait avec une grille "à la main" au début du TP.
  Pour pouvoir faire cet exercice, vous pouvez étudier le guide sur Grid Layout fait par Alsacréations disponible sur le lien suivant **https://www.alsacreations.com/article/lire/1388-css3-grid-layout.html .** De plus un mémento CSS Grid Layout se trouve ici**.**

Astuce :
Pour votre Grid vous pouvez définir 3 colonnes 1fr 2fr 1fr et 3 lignes 2fr 6fr 1fr
Vous pouvez ajouter de l'espace entre les cases avec **grid-gap : 10px** (propriété à spécifier

dans le bloc « conteneur » ou « wrapper » avec le **display : grid** et les **grid-template**).
Pour développer ce site et votre futur site beaucoup d'exemples se trouvent ici
https://gridbyexample.com/examples/