TD 10 : Responsive Web Design

9 décembre 2015

Objectif: Dans ce TD, vous verrez comment concevoir un design web qui s'adaptera au terminal sur lequel il sera visualisé.

1 Introduction

Avec l'avènement des smartphones et tablettes, le web devient de plus en plus consultable n'importe où, n'importe quand. On surfe maintenant depuis son téléphone dans les transports en commun, à la pause café ou même dans son canapé. Seulement, vous l'aurez noté, les écrans de ces terminaux mobiles sont beaucoup plus petits que ceux des ordinateurs de bureau, pour qui les sites web ont été réfléchis à la base.

On a donc vu fleurir des « versions mobiles » de sites, avec une navigation plus adaptée aux petits écrans. En fonction du navigateur (user-agent) de l'utilisateur, la version mobile était activée (par exemple à l'aide de jQuery Mobile).

Cette technique, même si elle est effectivement plus *mobile-friendly*, pose tout de même des problèmes :

1. Et si l'utilisateur possède un écran de moyenne taille, comme par exemple une tablette, vers quelle version sera-t-il redirigé ? Une troisième ? Et pour un très grand écran, comme un téléviseur ?
2. La « version mobile » du site est dépendante du navigateur, et non pas de la taille de l'écran. Or, le même navigateur peut tourner sur différents terminaux ne possédant pas du tout la même résolution, donc les mêmes capacités.

Il fallait donc trouver une solution plus « agile » : s'adapter à la taille de l'écran du terminal, et non uniquement au navigateur car il existe aujourd'hui (et encore plus demain) plein de façons d'accéder au même site : smartphone, tablette, télévision, console de jeu, tondeuse à gazon... C'est là que le *Responsive Design* entre en piste : permettre une expérience utilisateur optimale, quel que soit le terminal utilisé.

À la création d'un site « responsive », il y a une réelle phase de réflexion qui doit avoir lieu en même temps que la démarche graphique. Comment va évoluer la page si l'écran est plus petit ? Plus grand ? En effet, il vaut mieux s'adapter à la structure du contenu pour que la mise en page soit idéale pour toutes les résolutions. La Figure 1 montre un site web appliquant ces principes à bon escient.

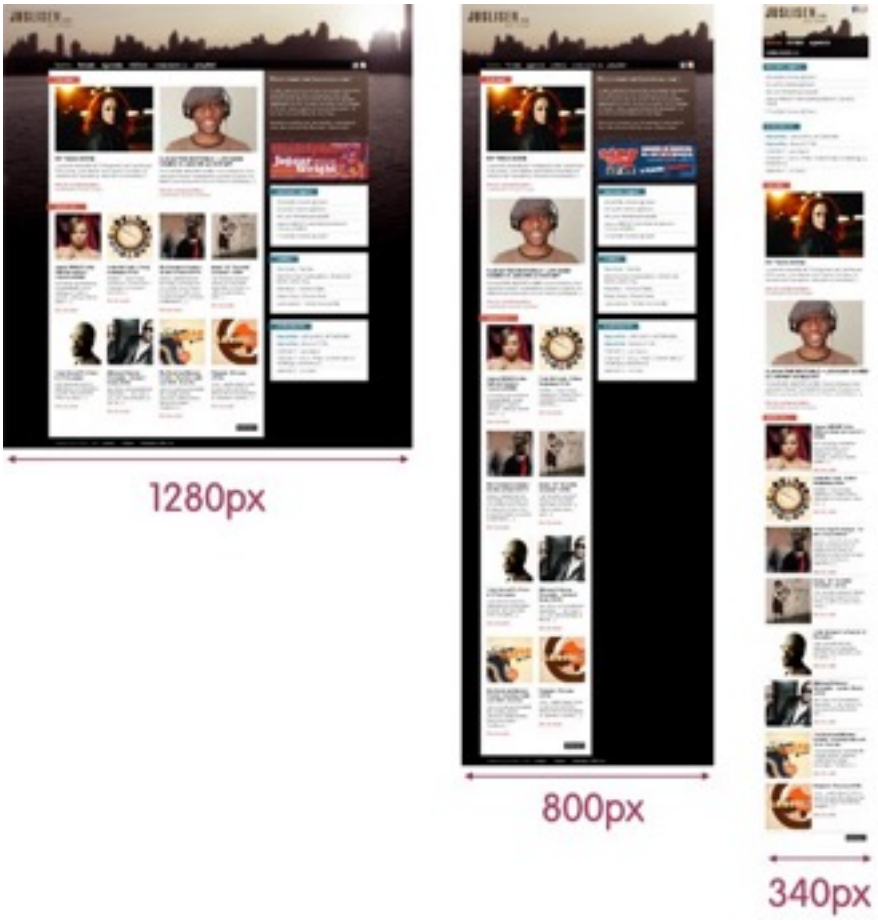

FIG. 1 – *Exemple d'une page web responsif.*

2 Testing

Dans ce TD, pour tester vos designs sur un vrai dispositif mobile, vous pouvez faire l'exercice « en ligne », c'est à dire, mettre votre code dans vôtre site personel de l'université.

Si vous ne l'avez pas encore fait, créez un dossier « WWW » à la racine de votre dossier personnel. Dans ce dossier, créez un fichier `index.html`, qui contient le squelette d'une page web. Avec ces préparations, vôtre site web s'affiche à l'adresse « `http://etudiant.u-pem.fr/~votrelogin` ».

Une autre solution est d'utiliser l'outil `Responsive Design Mode` de votre navigateur (pour acceder aux outils de développements dans Chrome or Firefox, par exemple, tapez « Ctrl + Maj + I »).

Pour la plupart des exercices, il suffit cépendant de changer la taille du navigateur pour pouvoir vérifier que le site est bien « responsive ».

3 Viewport

Le *Viewport* est l'espace d'une page web visible par l'utilisateur. Il varie en fonction de l'appareil, et est plus petite sur un téléphone mobile que sur un écran d'ordinateur.

Avant les tablettes et les téléphones mobiles, les pages web étaient conçus uniquement pour les écrans d'ordinateur, et il était courant pour les pages web d'avoir une conception statique et une taille fixe. Mais ces pages web de taille fixe sont trop volumineux pour être contenus dans le Viewport des tablettes et les téléphones mobiles. Pour résoudre ce problème, les navigateurs sur ces appareils réduisent la page Web entière pour l'adapter à l'écran.

Pour contrôler le Viewport, on utilise le balise `<meta>` avec l'attribut `name` ayant la valeur « `viewport` ».

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

Le « `width=device-width` » adapte le contenu du site à la largeur du dispositif.

▸ **Exercice 1:** Dans le fichier `index.html`, créez un site web simple contenant une image et un texte de quelques lignes. Maintenant surfez sur votre site avec un dispositif mobile [1]. Quelle est la difference entre vôtre site sans et avec le balise `<meta>`?

4 Grid-View

Pour faciliter la concéption des sites web, de nombreuses pages web sont basées sur une grille (*Grid-View*), ce qui signifie que la page est divisée en colonnes (voir Figure 2).

Pour faire ce design avec une grille, assurez vous d'abord que tous les éléments HTML ont la propriété `box-sizing` mis à `border-box`:

```
* {    // le "*" permet de choisir tous les éléments HTML dans vôtre code
  box-sizing: border-box;
}
```

Cela permet d'assurer que le *padding* et le *border* soient inclus dans la largeur et la hauteur totale des éléments.

Supposons qu'on veuille definir une grille à 10 colonnes [2] et une largeur totale de 100%. Cette grille se dilatera lorsque vous redimensionnerez la fenêtre du navigateur. Cela veut dire que chaque colonne a une largeur de 10%.

Un élément dans nôtre site web peut couvrir une ou plusieurs colonnes. Nous allons créer une classe pour chaque taille possible. Le nombre associé à chaque classe définit le nombre de colonnes qu'elle doit couvrir :

```
.col-1 {width: 10%;}
.col-2 {width: 20%;}
.col-3 {width: 30%;}
.col-4 {width: 40%;}
.col-5 {width: 50%;}
```

1. Vous pouvez aussi ajouter la balise `<meta>` dans la page que vous avez créée pour le TD 3. Ou si vous ne voulez ni modifier ni perdre celle-ci vous pouvez travailler dans un sous-répertoire `mobile/`.
2. Dans la réalité on tend à utiliser plus de colonnes : 12, 16 ou 24.

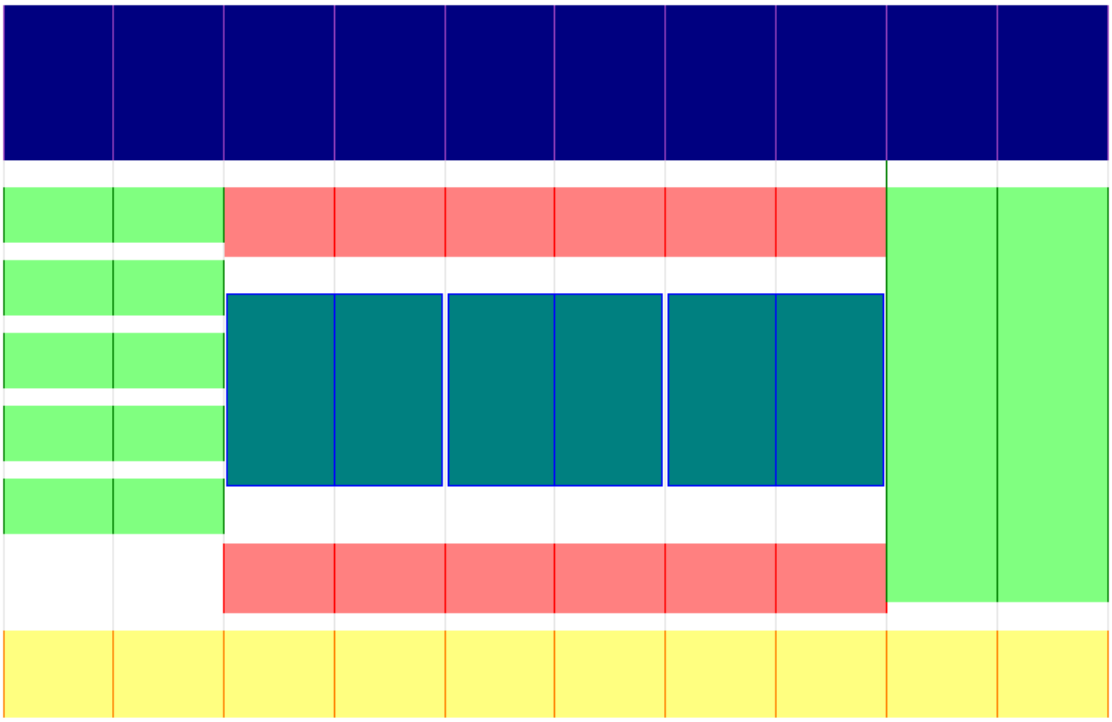

FIG. 2 – *Exemple d'un Grid-View à 10 colonnes.*

```
.col-6 {width: 60%;}
.col-7 {width: 70%;}
.col-8 {width: 80%;}
.col-9 {width: 90%;}
.col-10 {width: 100%;}
```

Le sélécteur de toutes les classes qui commencent en « `col-` » est « `[class*="col-"]` ». Par exemple, pour fixer le style de la bordure de toutes ces classes et les faire flotter à gauche, on écrira :

```
[class*="col-"] {
  border: 2px solid black;
  float: left;
}
```

Chaque ligne doit être enveloppé dans une balise `<div>`. Notez que le nombre total de colonnes d'une ligne doit toujours être 10.

Par exemple :

```
<div class="ligne">
  <div class="col-4">...</div>
  <div class="col-3">...</div>
  <div class="col-3">...</div>
</div>
```

Pour éviter les problèmes de "débordement" dûs à la présence d'éléments flottants, on prendra garde de gérer l'overflow.

Par exemple :

```
.ligne {
    overflow: auto;
}
```

▸ **Exercice 2:** En vous aidant de la feuille de style « `renaissance.css` », créez une page web ayant une grille de 10 colonnes. Le sujet de cette page sera "La renaissance italienne" et devra avoir les éléments suivants (voir Figure 2) :

- un en-tête de largeur maximale ;
- une barre de navigation latérale à gauche couvrant le 20% de la page ;
- une section centrale couvrant le 60% de la largeur de la page ;
- une section latérale à droite couvrant le 20% de la page ;
- un bas de page de largeur maximale.

1. La barre de navigation contiendra 5 éléments (pour le moment pas fonctionnels).
2. La partie centrale contiendra un titre avec un texte occupant la totalité de la section, suivi par trois images dans trois colonnes différentes (par exemple, les trois images dans le répertoire « img/33-33-33/ ») et par une vidéo (par exemple, `https://www.youtube.com/watch?v=IcaO92v9XK8`) qui occupera la totalité de la section. Comment faut-il inserer les trois images pour qu'elles aient la même taille ? Les classes « `col-1` », ..., « `col-10` » définies précédemment sont elles suffisantes pour partager en trois parties égales la section ? Ou faut il définir une nouvelle classe ?
3. Dans la section de droite on affichera un carousel d'images [3] (par exemple les images contenues dans le répertoire « img/carousel/ »).

Au final, votre page devra ressembler à celle montrée en Figure 3.

5 Media Queries

Les requêtes des médias (Media Queries) dans CSS sont un outil important pour implanter un Responsive Web Design. Elles sont utilisées pour verifier des aspects du dispositif comme par exemple :

1. la largeur et la hauteur de la fenêtre ;
2. la largeur et la hauteur du dispositif ;
3. l'orientation (distinguer entre le mode paysage et le mode portrait).

Le syntaxe est :

```
@media only|not typeDuMedia and (caractéristiqueDuMedia) {
  // Code CSS
}
```

où le type du média peut être « `all` », « `screen` », « `print` » ou « `speech` ». Dans nos applications, on utilisera seulement « `screen` ».

Comme caractéristique du média on peut, par exemple, considerer la largeur du Viewport « `width` », la largeur maximale de la fênetre de navigateur « `max-width` » ou encore la largeur minimale de la fênetre « `min-width` ».

3. Rappelez vous du TD 6.

FIG. 3 – *Exemple d'une page avec une grille à 10 colonnes.*

Par exemple, le code suivant va changer la couleur de l'arrière plan si la fênetre du navigateur est plus petite que 500px.

```
@media only screen and (max-width: 500px) {
  body {
    background-color: blue;
  }
}
```

De la même façon, on peut utiliser des Media Queries pour implenter une page qui s'adapte selon la taille de la fenêtre de navigateur.

Par exemple :

```
/* Pour les portables: */
[class*="col-"] {
    width: 100%;
```

```
}

/* Pour des ordinateurs avec un grand écran: */
@media only screen and (min-width: 750px) {
    .col-1 {width: 10%;}
    .col-2 {width: 20%;}
    .col-3 {width: 30%;}
    .col-4 {width: 40%;}
    .col-5 {width: 50%;}
    .col-6 {width: 60%;}
    .col-7 {width: 70%;}
    .col-8 {width: 80%;}
    .col-9 {width: 90%;}
    .col-10 {width: 100%;}
}
```

▸ **Exercice 3:** Modifiez le feuille de style de l'Exercice 2 pour obtenir trois différentes structures en grille selon la taille (largeur) de la fenêtre :

1. Si la taille de la fenêtre est plus grande que 750px, la structure sera la même que dans l'Exercice 2 (voir aussi Figure 2).
2. Si la taille de la fenêtre est plus grande que 500px (mais plus petite que 750px), on aura (voir aussi Figure 4):
   - un en-tête de largeur maximale ;
   - une barre de navigaion de largeur maximale (on a 5 éléments, donc chaque élément couvrira 20% de la barre) ;
   - une section centrale sur 70% de la largeur de la fenêtre ;
   - une section laterale à droite couvrant 30% de la largeur de la fenêtre ;
   - un bas de page de largeur maximale.
3. Si la taille de la fenêtre est plus petite que 500px on aura tous les éléments de largeur maximale, en faisant attention aux trois images (voir aussi Figure 5). De plus, modifiez l'arrière plan de l'en-tête, en utilisant l'image « `citta-ideale-m.jpg` » dans le cas où la taille de l'écran est plus petite que 500px.

Enfin, modifiez l'arrière plan de la page si la largeur de la fenêtre est plus large que la hauteur (*paysage*) ou plus petite (*portrait*). Pour cela, utilisez comme caractéristique du Media Query la propriété « `orientation` » avec la valeur « `landscape` » ou « `portrait` ».

FIG. 4 – *Exemple d'un Grid-View à 10 colonnes adapté pour des écrans de moyenne taille.*

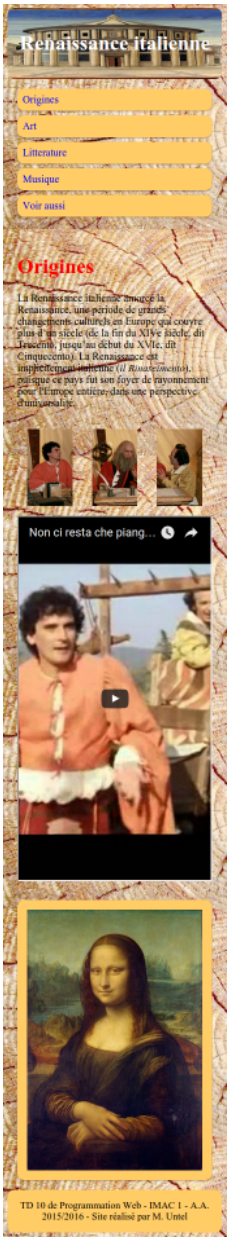

FIG. 5 – *Exemple d'un Grid-View à 10 colonnes pour des écrans de petite taille.*