Travaux dirigés n°10

IMAC 1

Responsive Web Design

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

Avant-propos

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 :

- 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 ?

- 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.

Une nouvelle méthodologie

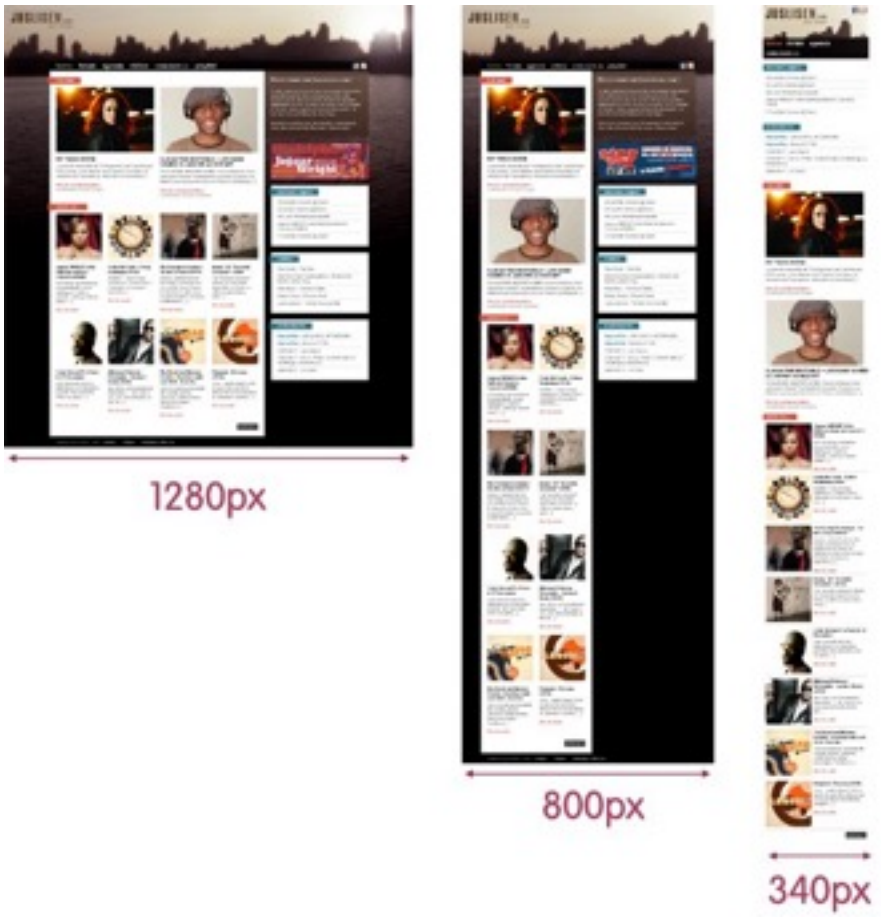
A la création d'un site « responsive », il y a une réelle phase de réflexion qui doit avoir lieu en même temps que la démarche graphique. Comment va évoluer la page si l'écran est plus petit ? Plus grand ?

Sans avoir forcément à faire 5 maquettes pour 5 versions différentes, il faut simplement réfléchir en amont du développement front. Vous pouvez envisager l'aspect de votre site pour les terminaux les plus utilisés (iPhone, Android, iPad, ...), mais pas que. En effet, il vaut mieux s'adapter à la structure du contenu pour que la mise en page soit idéale pour toutes les résolutions : on parle de « **content focused design** ».

Exemple[1] :

Voici un site appliquant ces principes à bon escient, dans le cadre d'une surface d'affichage réduite :

[1] Source : http://www.alsacreations.com/article/lire/930-css3-media-queries.html



Exercice :

- Sur la maquette fournie, repérez plusieurs points de rupture (quel bloc peut poser problème ? A partir de quelle taille ?) et réfléchissez à l’évolution des différents blocs en fonction de la résolution : plus petite, puis plus grande.

Une grille fluide

Maintenant, conservez dans un coin de votre tête vos évolutions de design. Vous allez pouvoir commencer l’intégration.

Exercice :

- **En n’utilisant que des unités relatives** (% et em), intégrez le design pour que sa largeur s’adapte entièrement à la largeur de l’écran. Notez que vous vous adressez à des terminaux mobiles récents, donc oubliez Internet Explorer et amusez-vous en CSS3 (pour cet exemple seulement, mais n’oubliez pas que les problématiques de rétrocompatibilité des navigateurs sont toujours d’actualité...). Votre page devra faire toute la largeur de votre fenêtre, quelle que soit la taille de celle-ci.

Des media queries

Vous devez maintenant avoir une page entièrement fluide. Seulement, vous remarquez que pour certaines tailles de fenêtre (les points de rupture que vous aviez sans-doute notés), le rendu n'est pas optimal... Vous allez donc pouvoir (enfin !) utiliser des media queries.

Les Media Queries sont des expressions dont la valeur est true (vrai) ou false(faux). Lorsqu'un Media Query a pour valeur true, les instructions situées entre les accolades qui le suivent sont exécutées. Elles sont ignorées dans le cas contraire.

Si nécessaire, vous pouvez associer plusieurs expressions à l'aide d'opérateurs logiques:

Opérateur	Signification
and	et
only	uniquement: masque la suite sur les navigateurs non compatibles avec les Media Queries
not	non
,	ou

Les media queries, supportées en CSS3, sont un mécanisme permettant d'identifier le type de média (écran, papier, etc.) mais aussi la résolution du terminal. Ainsi, il est possible de spécifier certaines règles CSS (ou même des feuilles à part) en fonction du terminal. Voici quelques-unes des syntaxes utilisables:

Syntaxe	Signification
`@media (max-width: largeur){...}`	Largeur de la fenêtre inférieure à la largeur spécifiée
`@media (max-device-width: largeur){...}`	Largeur du périphérique inférieure à la largeur spécifiée
`@media (min-width: largeur1) and (max- width: largeur2){...}`	Largeur de la fenêtre comprise entre les deux largeurs spécifiées
`@media (max-device-width: largeur) and (orientation: landscape){...}`	Largeur du périphérique inférieure à la largeur spécifiée et écran tenu horizontalement

Par exemple :

```
@media screen and(max-width: 1024px) {
    body { background: red; }
}
```

Vous pouvez également importer toute une feuille :

```
@import url("wide.css") screen and (min-width:1024px);
```

Voici un exemple de code. Ici quatre sont affichées. En fonction de la résolution du périphérique (max-device-width) ou de l'écran (min-width et max-width), l'une des voit son arrière-plan coloré en rouge.

```
<!DOCTYPE html>
      <html>
            <head>
                  <meta charset="utf-8">
                  <title>Media Queries</title>
                  <style>
                  div{
                        border : dotted 1px #666;
                        padding : 5px 10px ; margin: 40px;}
                  @media (max-device-width: 480px){
                        .iphone{ background : red; }
                  }
                  @media (max-width: 600px){
                        .inf600{ background : red; }
                  }
                  @media (min-width : 600px) and (max-width : 1200px){
                        .de600a1200{ background : red; }
                  }
                  @media (min-width : 1200px){
                        .sup1200{ background : red; }
                  }
                  @media (min-width : 1600px){
                        .sup1600{ background : red; }
                  }
                  </style>
            </head>
            <body>
                  <div class='iphone'> Cette div apparait en rouge si la
largeur maximale de l écran est de 480 pixels. </div>
                  <div class='inf600'> Cette div apparait en rouge si de la
fenêtre est inférieure a 600 pixels. </div>
                  <div class='de600a1200'> Cette div apparait en rouge si la
largeur de la fenêtre est compris entre 600 et 1200 pixels. </div>
                  <div class='sup1200'> Cette div apparait en rouge si la
largeur de la fenêtre est supérieure a 1200 pixels. </div>
                  <div class='sup1600'> Cette div apparait en rouge si la
largeur de la fenêtre est supérieure a 1600 pixels. </div>
            </body>
      </html>
```

Testez le code précédent sur des écrans différents (comme ceux d'un smartphone, une tablette ou un ordinateur).

Pour plus d'info:

- **https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Media_queries**
- **http://www.w3.org/TR/css3-mediaqueries/**
- **http://www.alsacreations.com/article/lire/930-css3-media-queries.html**

Exercice :

- En fonction de vos points de rupture précédemment définis, modifiez des règles CSS pour que la mise en page soit optimale sur des écrans plus petits.

- Faites de même pour des plus grandes résolutions.