Les fondamentaux du Responsive Web Design

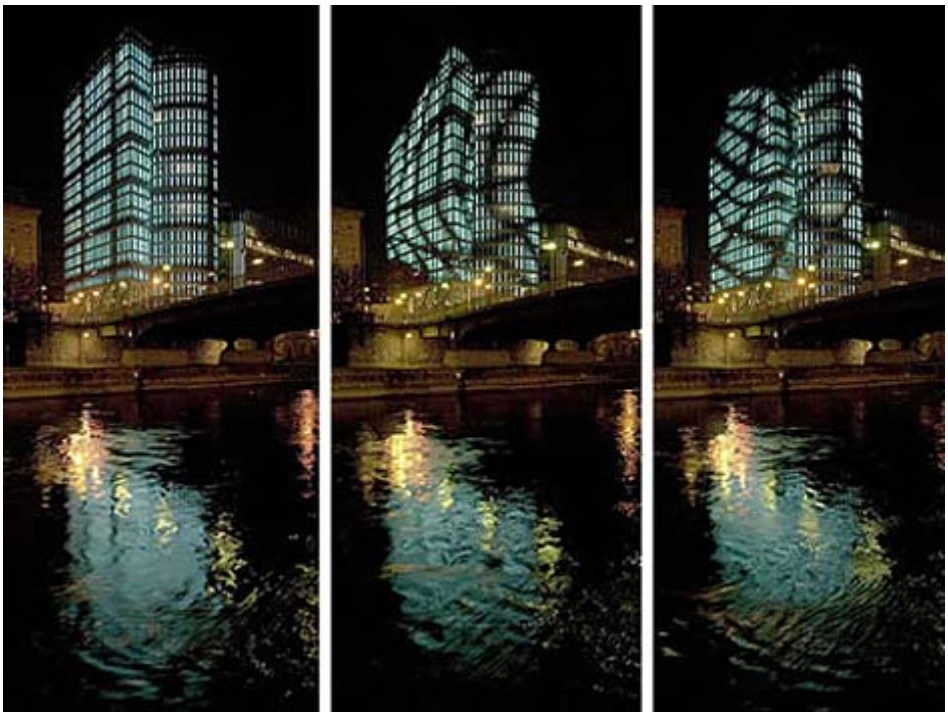
Par Julien Roche

Date de publication : 15 mars 2013

Le saviez-vous ? Le principe du *Responsive Web Design* nous vient de l'architecture. Cela se nomme le « *Responsive Architecture* ». Le but ? Que l'architecture et ses occupants s'influencent, voire s'informent réciproquement.

I - Introduction

Prenons deux exemples simples. Le premier pourrait être ces vitres qui s'opacifient en fonction de la luminosité extérieure. Ou encore ces murs-miroirs où plus nous nous approchons, plus ils nous semblent se déformer. Bref, ce que cherchent les architectes dans ce mouvement, c'est de dépasser/s'abstraire des contraintes inhérentes liées aux différents supports. Dans le monde de l'architecture, nous pouvons affirmer que ces supports sont vraiment multiples et nombreux.



Par simple jeu de lumière, cet immeuble semble avoir une structure différente.
Source : http://www.essential-architecture.com/STYLE/STY-075.htm

Le « *Responsive Web Design* » est né, car nous sommes de plus en plus confrontés aux mêmes problèmes que les architectes : s'adapter aux supports.

Voyez plutôt : autrefois, nous n'avions qu'un ou deux navigateurs et les tailles d'écrans variaient peu. Désormais, nous avons une multitude de navigateurs et un nombre important de résolutions (allant du 240x320 au 2515x1886, voire plus). Mais les supports également : nous pouvons accéder à Internet depuis nos classiques ordinateurs de bureau, mais aussi *via* les smartphones, tablettes, etc. Or nous savons par rapport à ces derniers que leur manière d'afficher un site Web diffère de celui d'un ordinateur de bureau.

Le Web à l'heure actuelle. Source : http://bradfrostweb.com/blog/post/this-is-the-web/

Et tout cela, c'est ce que nous avons maintenant. Nous commençons tout doucement à accéder à Internet *via* des télévisions (souvent par le biais de consoles, ou de *boxes*). Des prototypes commencent à apparaître, avec de l'Internet sur des autoradios, des montres, dans les voitures, sur les réfrigérateurs… Bref, un monde nouveau où nous devrons là-encore nous adapter.

Le Web de demain. Source : http://bradfrostweb.com/blog/post/this-is-the-web/

II - Les trois notions du Responsive Web Design

Pour faire une application Web qui soit *Responsive Web Design*, nous devons établir les trois points suivants :

- une grille d'affichage qui soit flexible :
  - autrement dit, un gabarit qui ne dépende pas d'une résolution minimale, maximale ;
- des médias flexibles :
  - à savoir, faire en sorte que mes images, vidéos (si besoin), ne débordent pas du cadre de notre grille d'affichage/gabarit ;
- un ensemble de règles CSS basé sur Media Queries :
  - le principe : mettre des conditions sur l'affichage afin d'afficher/masquer, voire changer le rendu de notre application Web.

Si nous faisons une analogie avec l'architecture, les trois derniers points représentent nos outils. En ce qui concerne nos matériaux, cela correspondra à nos navigateurs. Alors certains matériaux sont meilleurs que d'autres. Par exemple le bois miteux IE6 est moins robuste que le béton armé Chrome. Nos outils, aussi corrects qu'ils soient, ne vont pas toujours s'appliquer facilement. Car une idée fausse existe et résiste : il n'est pas possible de faire du « *Responsive Web Design* » sur les vieux navigateurs.

Il est heureusement possible de s'adapter pour y parvenir. Néanmoins, le coût sera (fatalement) plus important et l'utilisation de *frameworks*/*polyfills* JavaScript (code JavaScript permettant de combler un manque dans HTML5/CSS3) nous sera nécessaire. Nous allons en entrevoir quelques-uns tout le long de l'article.

III - Grille d'affichage flexible

Avant toute chose, il faut faire en sorte de créer un site dont la grille d'affichage soit flexible. Autrement dit, nous n'utilisons en aucun cas (ou tout du moins, le moins possible) de largeur/hauteur fixe.

Pour commencer, dimensionnons le site dans la résolution dite « optimale », à savoir la résolution pour laquelle l'affichage est la plus adaptée (par exemple 1280 * 800 pixels).

Une fois cela fait, nous allons modifier à la fois la taille des polices et la taille de nos éléments. Nous allons faire en sorte que tout soit exprimé en pourcentage (dans le cas de la police, nous utiliserons ce que nous appelons les « em », où 1em = 100 %).

En règle générale, nous estimons que la taille par défaut de notre texte (pour qu'il soit lisible) est de 16 pixels. Nous admettons que 16 pixels correspondent à un 1em.

Si, dans notre feuille de style, nous avons un titre dont la taille de police fait 24 pixels, nous remplacerons cette valeur par 1.5em.

En fait, nous devons appliquer à chaque fois la formule suivante :

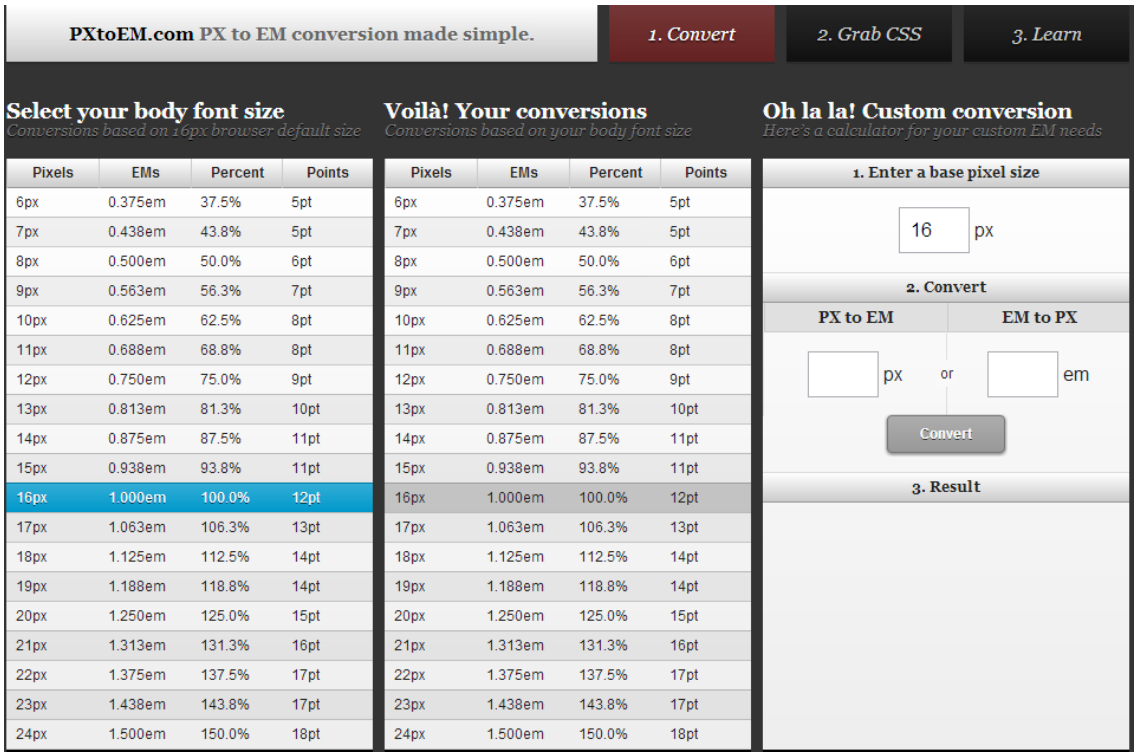
```
cible / contexte = résultat
```

La cible correspond à la taille souhaitée, le contexte à la taille « normale » et évidemment, le résultat est exprimé en « em ». Cela a pour intérêt que si le contexte change (de par la résolution), le texte reste proportionnel à l'affichage.

Une bonne pratique réside à mettre à côté du texte la formule afin de se souvenir du pourquoi du résultat :

```
h1 {
	font-size: 1.5em; /* 24 / 16 pixels */
}
```

Néanmoins, il n'est pas toujours facile de convertir des tailles de polices vu que nous pouvons manipuler des points, des pixels, des em ou des pourcentages. Du coup, je vous invite à aller sur le site **http://pxtoem.com/** qui vous donne un tableau de base ainsi qu'un convertisseur :



PXtoEM.com PX to EM conversion made simple.

1. Convert | 2. Grab CSS | 3. Learn

Select your body font size
Conversions based on 16px browser default size

Pixels	EMs	Percent	Points
6px	0.375em	37.5%	5pt
7px	0.438em	43.8%	5pt
8px	0.500em	50.0%	6pt
9px	0.563em	56.3%	7pt
10px	0.625em	62.5%	8pt
11px	0.688em	68.8%	8pt
12px	0.750em	75.0%	9pt
13px	0.813em	81.3%	10pt
14px	0.875em	87.5%	11pt
15px	0.938em	93.8%	11pt
16px	1.000em	100.0%	12pt
17px	1.063em	106.3%	13pt
18px	1.125em	112.5%	14pt
19px	1.188em	118.8%	14pt
20px	1.250em	125.0%	15pt
21px	1.313em	131.3%	16pt
22px	1.375em	137.5%	17pt
23px	1.438em	143.8%	17pt
24px	1.500em	150.0%	18pt

Voilà! Your conversions
Conversions based on your body font size

Pixels	EMs	Percent	Points
6px	0.375em	37.5%	5pt
7px	0.438em	43.8%	5pt
8px	0.500em	50.0%	6pt
9px	0.563em	56.3%	7pt
10px	0.625em	62.5%	8pt
11px	0.688em	68.8%	8pt
12px	0.750em	75.0%	9pt
13px	0.813em	81.3%	10pt
14px	0.875em	87.5%	11pt
15px	0.938em	93.8%	11pt
16px	1.000em	100.0%	12pt
17px	1.063em	106.3%	13pt
18px	1.125em	112.5%	14pt
19px	1.188em	118.8%	14pt
20px	1.250em	125.0%	15pt
21px	1.313em	131.3%	16pt
22px	1.375em	137.5%	17pt
23px	1.438em	143.8%	17pt
24px	1.500em	150.0%	18pt

Oh la la! Custom conversion
Here's a calculator for your custom EM needs

1. Enter a base pixel size

16 px

2. Convert

PX to EM | EM to PX

px or em

Convert

3. Result

Merci à http://pxtoem.com

Avant de voir nos éléments, je tiens à souligner un point important : parfois, nous aurons des résultats « farfelus », du style : 0.33333333333333… Bref, tomber sur un nombre infini. Nous aurons tendance à vouloir arrondir cela, à 0.34 par exemple. Et bien cela, en soi, est une erreur ! Il est conseillé de placer le plus possible de chiffres après la virgule (évidemment pas des centaines, mais une dizaine ou une quinzaine n'est pas choquant). Les navigateurs sont assez intelligents pour arrondir au mieux. Et ainsi, nous aurons un affichage optimisé !

Maintenant, pour nos éléments, nous allons appliquer exactement le même principe. Imaginons le cas suivant : nous avons notre site qui fait 1018 pixels de large dans le cas idéal. Nous avons alors une zone principale de 766 pixels et un menu de 252 pixels. Nous allons alors les déclarer de la façon suivante :

```
.menu {
	width: 24.7544204322%; /* 254 / 1018 pixels */
}

.main {
	width: 75.2455795678%; /* 766 / 1018 pixels */
}
```

Néanmoins, nous allons avoir une différence notable par rapport à la police de caractères : le contexte va varier ! En effet, la taille « normale » de la police sera la même partout dans notre page. Nous ne pouvons pas en dire autant de nos largeur et hauteur ! Autrement dit, nous devons réadapter notre contexte en fonction de notre parent.

Par exemple, si nous continuons sur notre cas, imaginons que la zone principale ait elle-même deux sous-zones. Leur contexte passera de 1018 pixels (largeur globale de la page) à… 766 pixels !

```
.main-left {
	width: 24.543080939%; /* 188 / 766 pixels */
}

.main-right {
	width: 75.456919061%; /* 578 / 766 pixels */
}
```

Et nous devrons appliquer ce principe sur toute notre page (autant que faire se peut), aussi bien sur les hauteurs, largeurs, que sur les marges (« margin », « padding ») et bordures.

Bien sûr, à partir d'une certaine résolution, soit il n'y aura pas assez de place pour afficher les images, le texte… soit il y aura trop de place et nous aurons beaucoup d'espace vide à combler. Les deux points suivants sont là pour compenser cela.

IV - Média flexible

Pour nous assurer que notre site Web puisse être *Responsive Design*, il faut pour cela nous assurer que les médias (les images, les vidéos, …) le soient également ! Rien de plus désagréable que de voir une image déborder de son cadre, enlaidissant ainsi notre site Web.

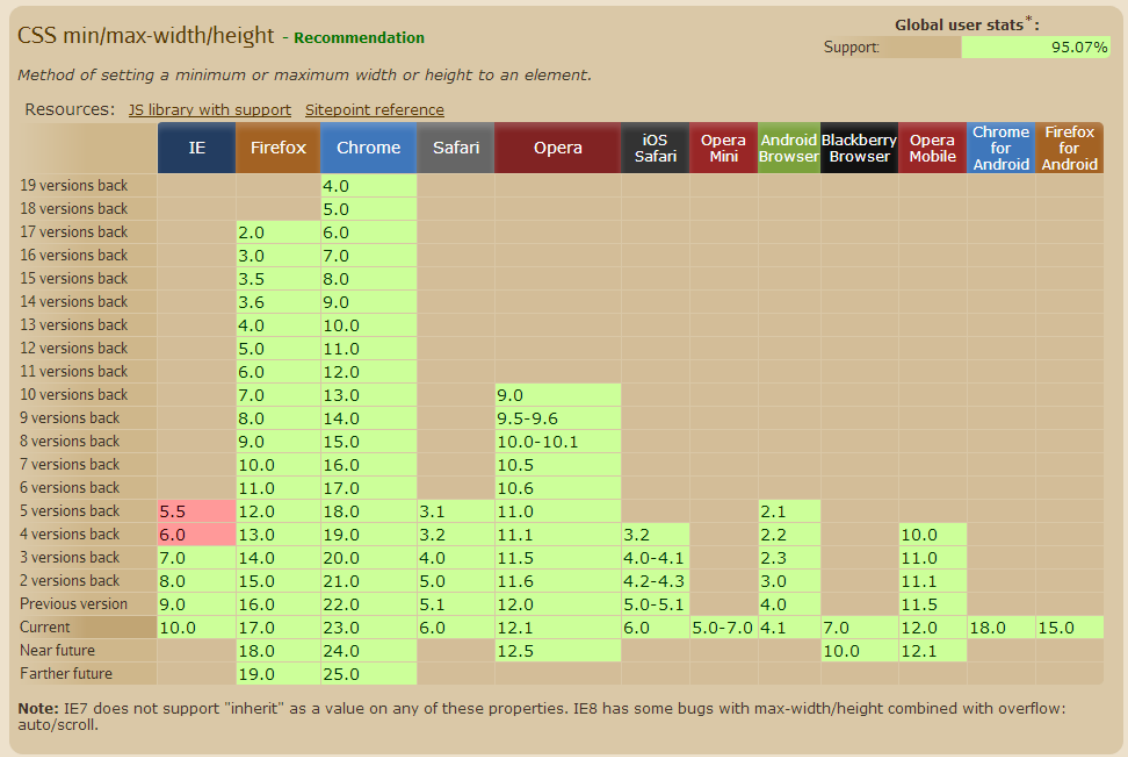
Afin de résoudre ce problème, CSS2 va nous venir en aide ! En effet, il existe une propriété se nommant « max-width » qui permet de spécifier la largeur maximum de l'élément, par rapport à son parent.

Du coup, il n'est pas rare que nous placions le CSS suivant dans nos fichiers :

```
img, object, embed, canvas, video, audio, picture {
	max-width: 100%;
	height: auto;
}
```

De cette façon, nous généralisons le « max-width » sur tous les médias de notre site Web. Même si nous spécifions sur une image précise une largeur déraisonnable, du moment que nous ne modifions pas la propriété « max-width », elle ne débordera jamais de son cadre. Il est à noter que l'utilisation de « height : auto » permet de conserver les bonnes proportions de nos médias (autrement, ils se verraient déformés).

Néanmoins, il y a un hic. En effet, même si cette propriété fait partie de spécification CSS2 et qu'elle existe depuis longtemps, IE6 (notre mortel ennemi) ne la reconnaît évidemment pas… Alors que les autres versions d'IE et les autres navigateurs la reconnaissent bien…



CSS min/max-width/height - Recommendation

Global user stats*: Support: 95.07%

Method of setting a minimum or maximum width or height to an element.

Resources: JS library with support Sitepoint reference

	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser	Opera Mobile	Chrome for Android	Firefox for Android
19 versions back			4.0									
18 versions back			5.0									
17 versions back		2.0	6.0									
16 versions back		3.0	7.0									
15 versions back		3.5	8.0									
14 versions back		3.6	9.0									
13 versions back		4.0	10.0									
12 versions back		5.0	11.0									
11 versions back		6.0	12.0									
10 versions back		7.0	13.0		9.0							
9 versions back		8.0	14.0		9.5-9.6							
8 versions back		9.0	15.0		10.0-10.1							
7 versions back		10.0	16.0		10.5							
6 versions back		11.0	17.0		10.6							
5 versions back	5.5	12.0	18.0	3.1	11.0			2.1				
4 versions back	6.0	13.0	19.0	3.2	11.1	3.2		2.2		10.0		
3 versions back	7.0	14.0	20.0	4.0	11.5	4.0-4.1		2.3		11.0		
2 versions back	8.0	15.0	21.0	5.0	11.6	4.2-4.3		3.0		11.1		
Previous version	9.0	16.0	22.0	5.1	12.0	5.0-5.1		4.0		11.5		
Current	10.0	17.0	23.0	6.0	12.1	6.0	5.0-7.0	4.1	7.0	12.0	18.0	15.0
Near future		18.0	24.0		12.5				10.0	12.1		
Farther future		19.0	25.0									

Note: IE7 does not support "inherit" as a value on any of these properties. IE8 has some bugs with max-width/height combined with overflow: auto/scroll.

Comptabilité de « max-width »

Du coup, pour pallier cela, nous spécifions un « *hack* » CSS qui va mettre la largeur à 100 % :

```
img, object, embed, canvas, video, audio, picture {
	max-width: 100%;
height: auto;
	_width: 100%; /* IE6 seulement */
}
```

Cette solution présente néanmoins des inconvénients : les médias prendront toujours toute la largeur et si nous en spécifions une, ils pourront déborder quand même. Bien que considérant IE6 comme mort (il n'est plus maintenu par Microsoft depuis presque un an), il est toujours utilisé dans des grands comptes et nous le retrouvons dans les spécifications de nos projets.

Dans ce cas, nous pouvons charger une librairie JavaScript qui se chargera de l'obliger à reconnaître cette propriété. C'est notamment le cas de ce projet OpenSource : **http://code.google.com/p/ie7-js/**.

Il contient des fichiers JavaScript pour IE8, IE9 et également pour les versions entre IE5 et IE7. Ils essaient de corriger certains manques (comme les problèmes de transparence sous IE6 et IE7). Il permet aussi de corriger le « max-width ».

Ainsi, grâce à cela, nous avons nos médias qui sont désormais *responsives* et pour une très large palette de navigateurs et de versions.

V - Media Queries

Si nous faisons un point, nous avons vu comment définir les gabarits de nos sites et faire en sorte qu'ils ne se basent pas sur des métriques figées, puis nous avons fait en sorte que nos médias ne perturbent pas l'affichage de nos éléments.

Maintenant, nous allons faire en sorte d'adapter l'affichage en fonction de la résolution. En effet, nous avons plus de choix de mise en place sur une résolution de 1280x800 pixels que sur une résolution de 320x240 pixels. Ainsi, nous

devons faire en sorte que nos éléments entrent dans ces différentes résolutions, quitte à ne pas en afficher certains, voire à les afficher à un autre endroit ou encore les afficher différemment.

Cela implique quelque part que nous devons poser des conditions sur les styles que nous devons appliquer sur nos sites Web. Et c'est très exactement ce que nous allons utiliser, par le biais des « *Media Queries* ».

C'est une fonctionnalité que nous apporte CSS3. Elle permet (comme son nom l'indique) de placer une condition sur :

- l'import d'un fichier CSS :
```
<link rel="stylesheet" media="screen and (color)" href="example.css" />
```
- un bloc CSS :
```
@media screen and (min-width:500px) {
    body {
        background-color: white;
    }
}
```
- ou en encore *via* JavaScript :
```
if(window.matchMedia("(min-width: 400px)").matches) {
    // We convert our menu to a collapsible menu
}
```

```
if(window.matchMedia("(min-width: 400px)").matches) {
    // We convert our menu to a collapsible menu
}
```

Il existe de nombreuses possibilités pour poser nos conditions. Nous pouvons le faire en fonction du type de média (screen, print, tv…), mais aussi en fonction des caractéristiques du support, comme le ratio, l'orientation, les couleurs…

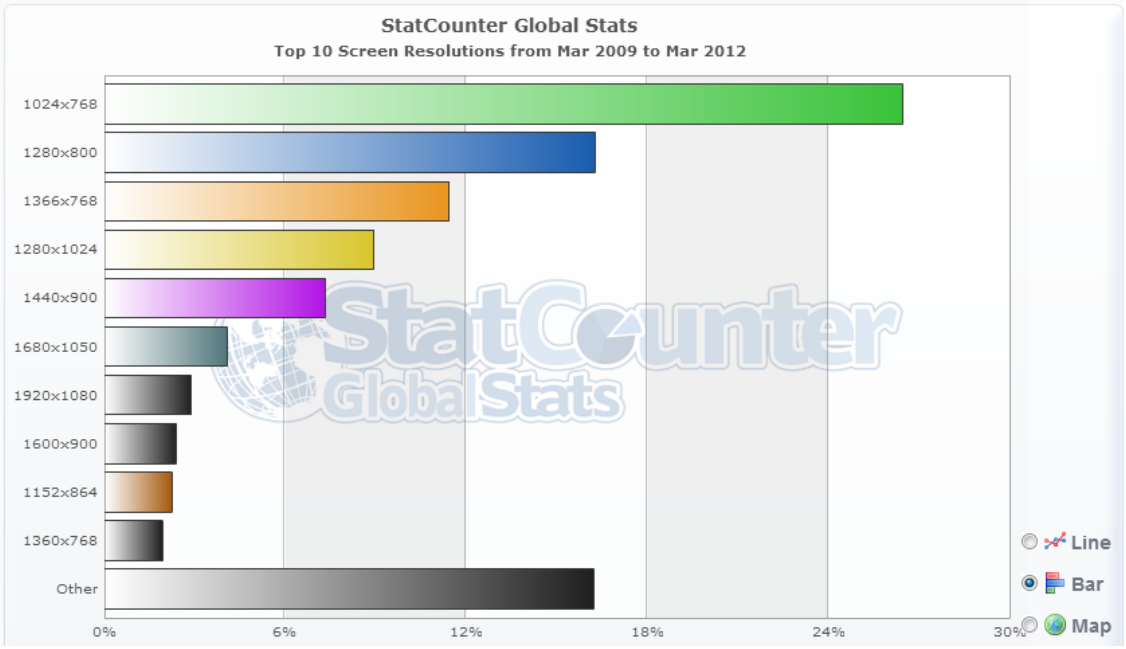
En règle générale, nous allons utiliser les conditions suivantes :

- orientation (pour savoir si nous sommes en mode portrait ou paysage) ;
- min-width, max-width (pour déterminer la largeur minimum/maximum) ;
- min-height, max-height (pour déterminer la hauteur minimum/maximum, leurs utilisations sont plus rares, nous préférons les deux propriétés précédentes).

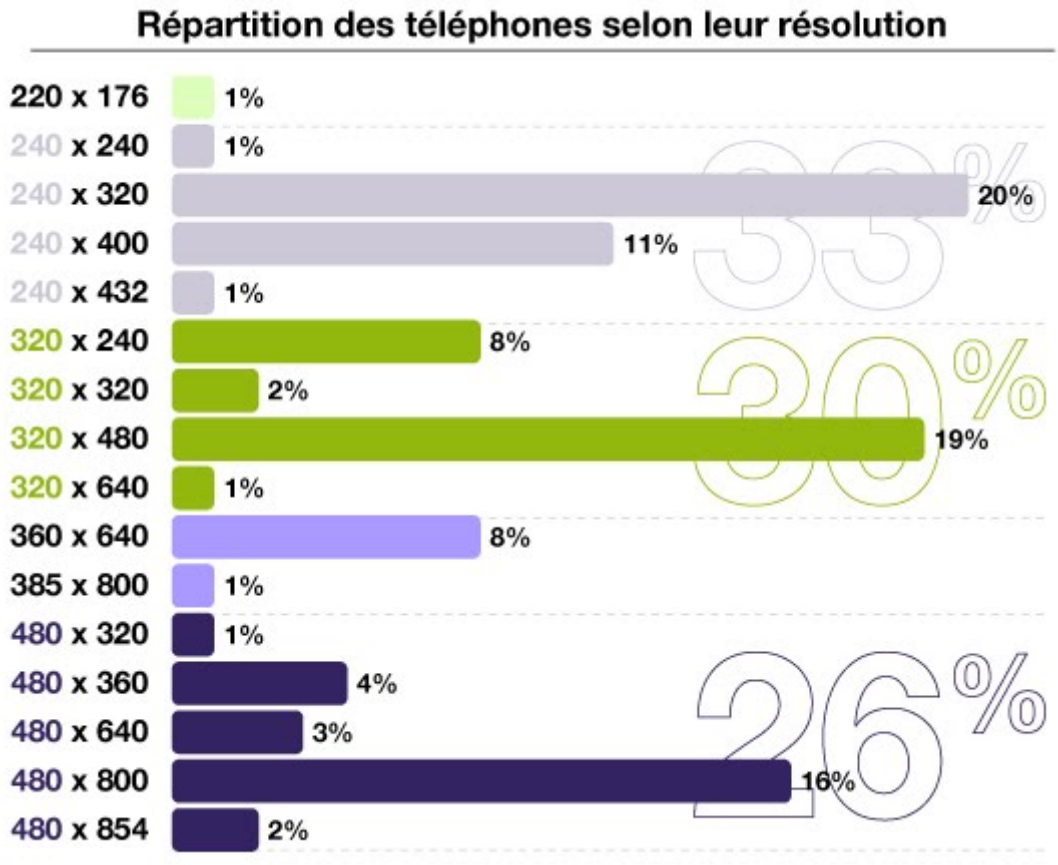
Avant d'aller plus loin, nous devons décider comment afficher nos éléments en fonction de l'orientation, mais également en fonction des différents tranches de résolution que nous allons gérer.

De manière courante, nous estimons, en mode portrait, que :

- une largeur d'écran de moins de 480px provient d'un smartphone ;
- une largeur d'écran entre 480px et 1024px provient de tablettes ;
- une largeur d'écran supérieure à 1024px provient d'ordinateurs de bureau ou de portables.



Résolution d'ordinateurs de bureau

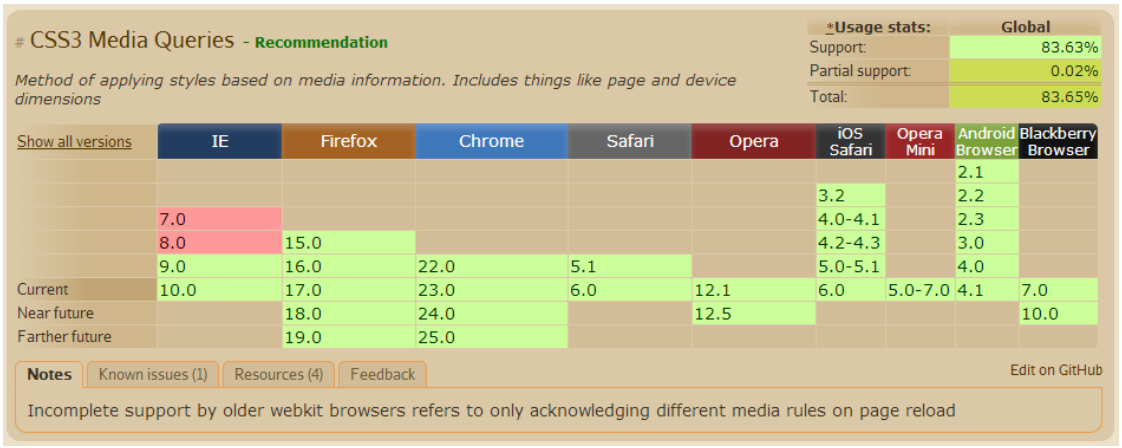


Résolution de smartphones

Ces règles ne sont pas figées dans le marbre, il existe bien évidemment des smartphones, tablettes et ordinateurs qui viendront les contredire. Il ne tient plus qu'à vous de décider sur quelles résolutions nous devons partir.

Une fois que nous avons décidé des tranches de résolution, de comment nous voulons afficher nos éléments par rapport à elles (et aussi par rapport à l'orientation), il suffit alors de placer le code CSS dans ces blocs afin d'obtenir l'effet escompté.

Certains vont dire : « C'est du CSS3, cela ne marche que sur des navigateurs récents ! ». Que nenni, les « *Medias Queries* » sont présents… sauf sur IE (en fait, cela commence à plus ou moins marcher sous IE8) !



CSS3 Media Queries - Recommendation

Method of applying styles based on media information. Includes things like page and device dimensions

*Usage stats:	Global
Support:	83.63%
Partial support:	0.02%
Total:	83.65%

Show all versions	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser
								2.1	
						3.2		2.2	
	7.0					4.0-4.1		2.3	
	8.0	15.0				4.2-4.3		3.0	
	9.0	16.0	22.0	5.1		5.0-5.1		4.0	
Current	10.0	17.0	23.0	6.0	12.1	6.0	5.0-7.0	4.1	7.0
Near future		18.0	24.0		12.5				10.0
Farther future		19.0	25.0						

Notes | Known issues (1) | Resources (4) | Feedback — Edit on GitHub

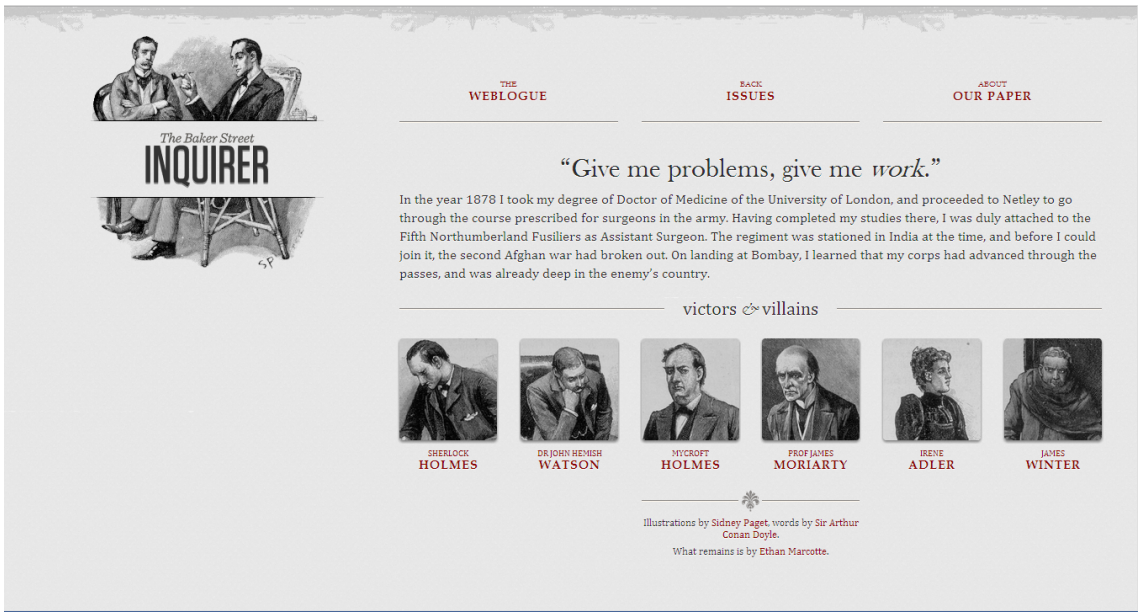
Incomplete support by older webkit browsers refers to only acknowledging different media rules on page reload

Compatibilité des « Media Queries »

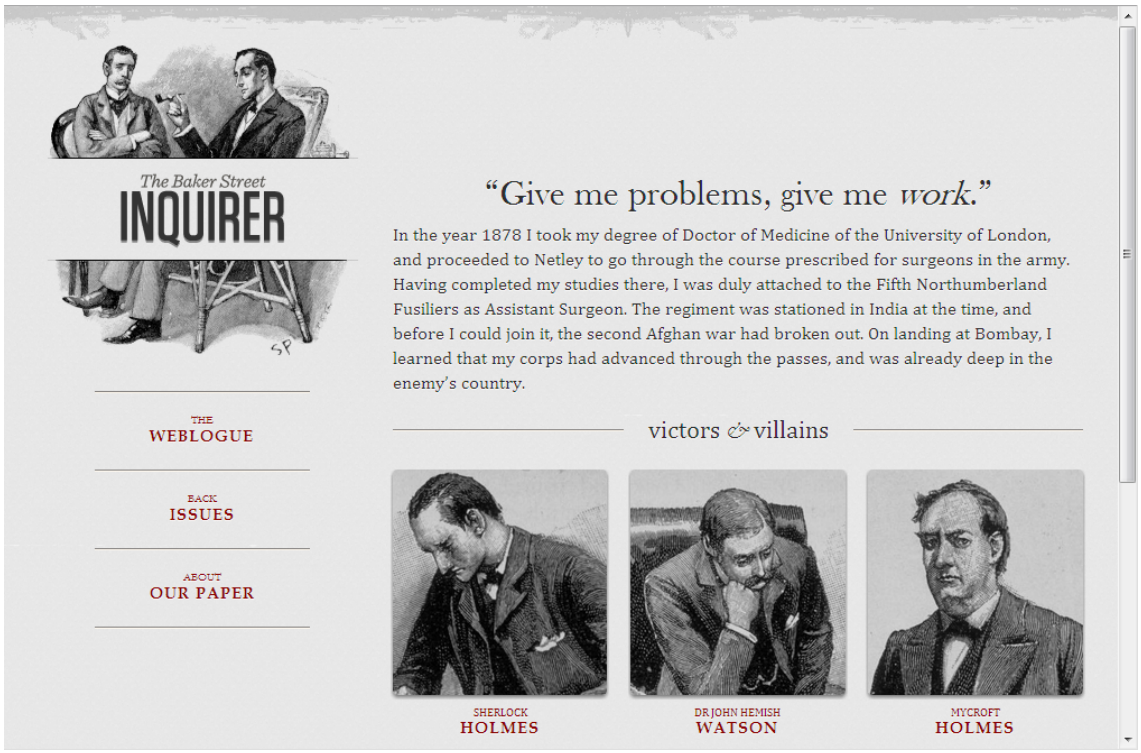
Heureusement pour nous, il existe un *polyfill* HTML5 célèbre, permettant de faire fonctionner les « *Medias Queries* » avec ces deux propriétés sur IE6 et IE7 ! Il se nomme « Respond.js » et nous pouvons le trouver sur lien GitHub suivant : **https://github.com/scottjehl/Respond**.

Il existe aussi « CSS3-mediaqueries » qui permet de rendre fonctionnel tous les « *Media Queries* » pour IE : **http://code.google.com/p/css3-mediaqueries-js/**.

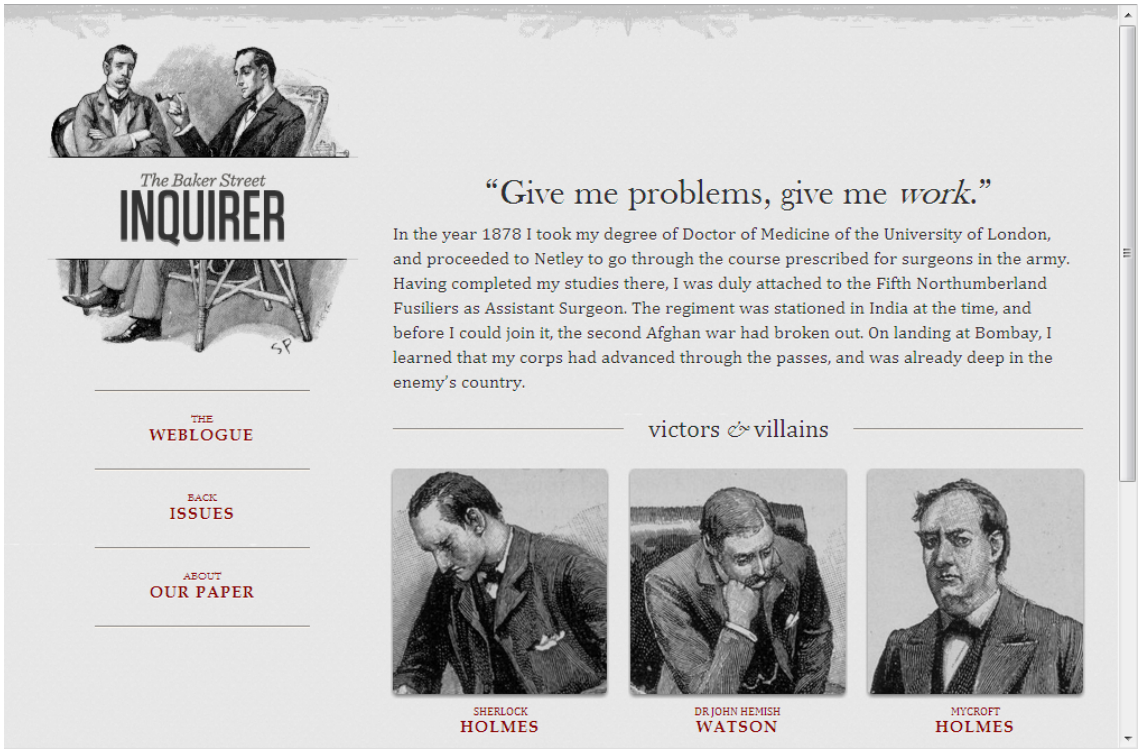
Voici un exemple de site *responsive* : le site Web d'Ethan Marcotte (grand référent du « *Responsive Design* ») sur Sherlock Holmes où en fonction de la résolution, l'affichage diffère :



*1360 * 768 pixels*



*1024 * 768 pixels*



*320 * 480 pixels*

VI - Faisons un exemple

Maintenant, nous allons essayer de mettre en pratique ce que nous avons vu. Pour cela, nous allons afficher un site sur trois colonnes :

- la colonne de gauche étant le menu ;

- la colonne de droite étant l'affichage des publicités ;
- la colonne centrale (la plus importante) affichant des listes d'articles.

Nous allons prendre comme résolution d'écran 1280x960 pixels.

Commençons par notre fichier HTML. Tapons :

```
<!DOCTYPE HTML>
<html lang="fr">
   <head>
      <title>Responsive design: par la pratique</title>
      <meta name="viewport" content="width=device-width, initial-scale=1, minimum-scale=1, maximum-
scale=1, user-scalable=no" />
      <meta charset="utf-8" />

      <!-- Styles -->
      <link type="text/
css" media="screen" title="no title" rel="stylesheet" href="responsive.design.css" />
   </head>

   <body>
      <!-- Structure de notre page: -->
      <div class="layout-main">
         <div class="main">
            <div class="layout-menu">
               <!-- Menu -->
               <div class="menu">
                  <h1>Menu</h1>
                  <ul>
                     <li><a href="#">Lien du menu 1</a></li>
                     <li><a href="#">Lien du menu 2</a></li>
                     <li><a href="#">Lien du menu 3</a></li>
                     <li><a href="#">Lien du menu 4</a></li>
                     <li><a href="#">Lien du menu 5</a></li>
                  </ul>

                  <select>
                     <option selected="selected">Choisir un menu</option>
                     <option>Lien du menu 1</option>
                     <option>Lien du menu 2</option>
                     <option>Lien du menu 3</option>
                     <option>Lien du menu 4</option>
                     <option>Lien du menu 5</option>
                  </select>
               </div>

            </div><div class="layout-content">
               <!-- Contenu -->
               <div class="content">
                  <div class="content-item">
                     <h1>Titre sujet 1</h1>
                     <p class="description">Une petite description</p>
                     <div class="article">
                        <img src="image.png" class="article-image" alt="Titre"/>

                        <p>Lorem ipsum dolor sit amet, quo ...</p>
                     </div>
                  </div>

                  <div class="content-item">
                     <h1>Titre sujet 2</h1>
                     <p class="description">Une petite description</p>
                     <div class="article">
                        <img src="image.png" class="article-image" alt="title" />

                        <p>Lorem ipsum dolor sit amet, quo ...</p>
                     </div>
                  </div>
               </div>
```

```
            </div><div class="layout-adv">
               <!-- Publicités -->
               <div class="adv">
                  <div class="adv-item">
                     <img src="image.png" alt="Pub 1" />
                     <br />
                     <a href="#">Lien pub 1</a>
                  </div>

                  <div class="adv-item">
                     <img src="image.png" alt="Pub 2" />
                     <br />
                     <a href="#">Lien pub 2</a>
                  </div>

                  <div class="adv-item">
                     <img src="image.png" alt="Pub 3" />
                     <br />
                     <a href="#">Lien pub 3</a>
                  </div>
               </div>
            </div>
         </div>
      </body>
</html>
```

Puis le CSS suivant :

```
@charset "UTF-8";

/**
 *    Ideal resolution: 1280 * 960 pixels
 */

html, body {
   height: 100%;
   background-color: #FFFFFF;
   border: 0px solid transparent;
   font-size: 16px;
   margin: 0px 0px 0px 0px;
   min-height: 100%;
   padding: 0px 0px 0px 0px;
   width: 100%;
}

.layout-main {
   height: 100%;
   margin: 0px auto 0px auto;
   min-height: 100%;
   width: 1024px;
}

.main {
   height: 100%;
   min-height: 100%;
}

.layout-menu, .layout-content, .layout-adv {
   display: inline-block;
   height: 100%;
   min-height: 100%;
   vertical-align: top;
}

.layout-menu {
   color: #FFFFFF;
   background-color: #CC0000;
   width: 214px;
}
```

```
.layout-content {
   color: #000000;
   background-color: #FFFFFF;
   width: 596px;
}

.layout-adv {
   color: #000000;
   background-color: #F9F7ED;
   width: 214px;
}

.menu, .content, .adv {
   padding: 5px 5px 5px 5px;
}

.menu h1 {
   font-size: 24px;
}

.menu li {
   list-style: none;
}

.menu li a {
   color: #FFFFFF;
}

.menu li a:before {
   content: "> ";
}

.menu li a:hover {
   color: #000000;
}

.content .content-item {
   border-top: 1px solid #36393D;
   clear: both;
   margin: 0px auto 0px auto;
   padding: 15px 0px 15px 0px;
   text-align: center;
   width: 542px;
}

.content .content-item:first-child {
   border: 0px solid transparent;
}

.content .content-item h1 {
   font-size: 24px;
}

.content .content-item p.description {
   font-size: 12px;
   font-style: italic;
}

.content .content-item div.article {
   text-align: justify;
}

.content .content-item img.article-image {
   float: left;
   padding: 0px 25px 25px 0px;
   width: 96px;
}

.adv .adv-item {
   border-top: 1px solid #36393D;
   margin: 0px auto 0px auto;
   padding: 15px 0px 15px 0px;
```

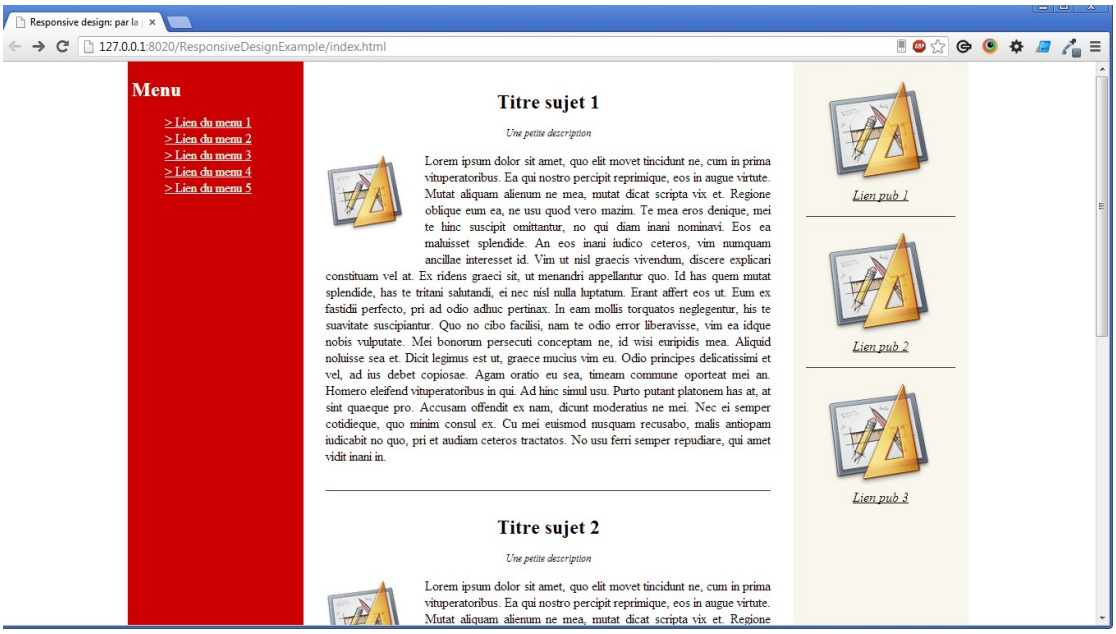
```
    text-align: center;
    width: 182px;
}

.adv .adv-item:first-child {
    border: 0px solid transparent;
}

.adv .adv-item a {
    color: #000000;
    font-style: italic;
}
```

Nous obtenons alors l'affichage suivant :

*1280 * 960 pixels*

Le rendu est bien celui qui est attendu, pour une résolution par 1280x960 pixels. Mais je suis consciencieux, et je veux essayer d'autres résolutions.

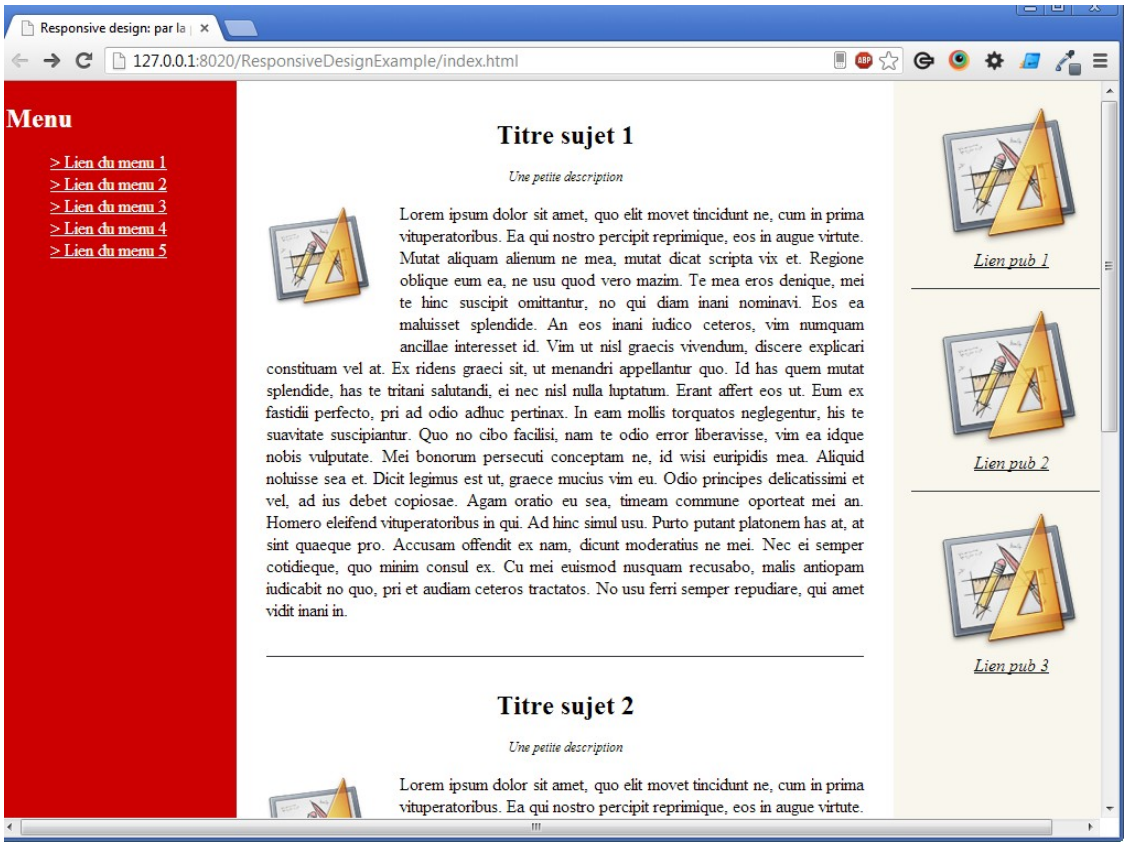
Je commence par tester 1360x768 pixels. J'obtiens cela :



*1360 * 768 pixels*

Je m'aperçois que j'ai de plus en plus d'espace blanc sur les bords de mon site, mais que mon gabarit n'est pas cassé. L'optimisation de l'affichage ne se fait pas, mais c'est encore lisible, c'est bon.

Maintenant, je prends la résolution 1024x768 pixels :



*1024 * 768 pixels*

Horreur ! Des barres de défilement apparaissent ! Et cela va être de pire en pire si nous baissons la résolution.

VI-A - Étape 1

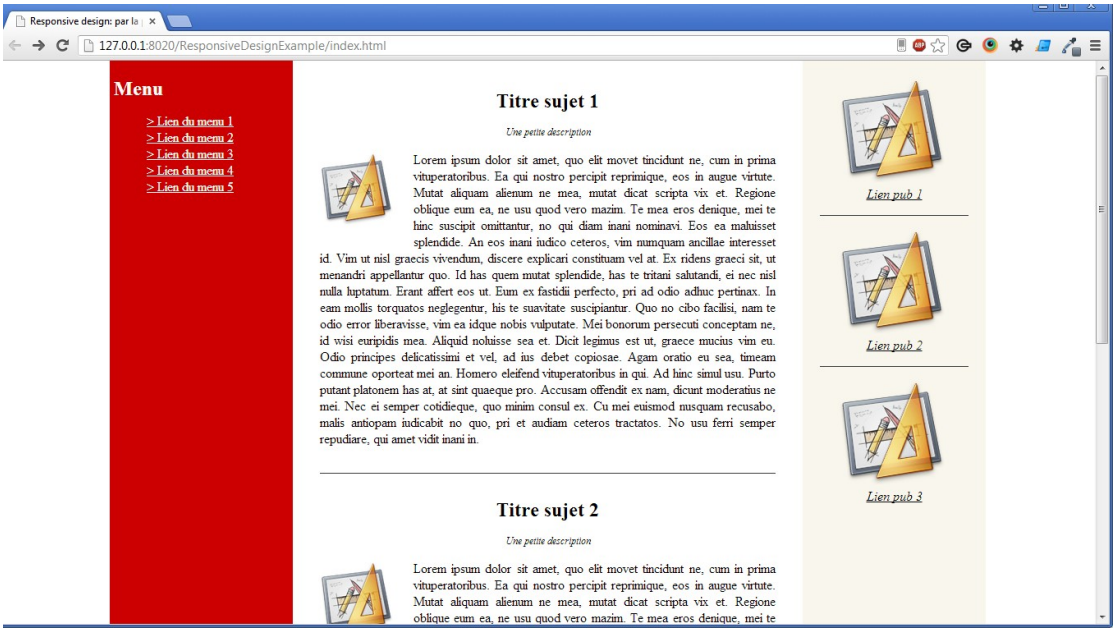
Commençons par les bases : plaçons le style CSS qui empêche les médias de prendre trop de place, de convertir les polices de caractères et de faire le ratio pour les largeurs. Nous obtenons le style suivant :

```
@charset "UTF-8";

/**
 *    Ideal resolution: 1280 * 960 pixels
 */

html, body {
   height: 100%;
   background-color: #FFFFFF;
   border: 0px solid transparent;
   font-size: 1em; /* Ideal: 16px */
   margin: 0px 0px 0px 0px;
   min-height: 100%;
   padding: 0px 0px 0px 0px;
   width: 100%;
}

img, object, embed, canvas, video, audio, picture {
   max-width: 100%;
   height: auto;
}

.layout-main {
   height: 100%;
   margin: 0px auto 0px auto;
   min-height: 100%;
   width: 80%; /* Ideal: 1024px -> 1024 / 1280 */
}

.main {
   height: 100%;
   min-height: 100%;
}

.layout-menu, .layout-content, .layout-adv {
   display: inline-block;
   height: 100%;
   min-height: 100%;
   vertical-align: top;
}

.layout-menu {
   color: #FFFFFF;
   background-color: #CC0000;
   width: 20.8984375%; /* Ideal: 214px -> 214 / 1024 */
}

.layout-content {
   color: #000000;
   background-color: #FFFFFF;
   width: 58.203125%; /* Ideal: 596px -> 596 / 1024 */
}

.layout-adv {
   color: #000000;
   background-color: #F9F7ED;
   width: 20.8984375%; /* Ideal: 214px -> 214 / 1024 */
}

.menu, .content, .adv {
   padding: 5px 5px 5px 5px;
}

.menu h1 {
   font-size: 1.5em; /* Ideal: 24px */
```

```
}

.menu li {
   list-style: none;
}

.menu li a {
   color: #FFFFFF;
}

.menu li a:before {
   content: "> ";
}

.menu li a:hover {
   color: #000000;
}

.content .content-item {
   border-top: 1px solid #36393D;
   clear: both;
   margin: 0px auto 0px auto;
   padding: 15px 0px 15px 0px;
   text-align: center;
   width: 90.939597315436241610738255033557%; /* Ideal: 542px -> 542 / 596 */
}

.content .content-item:first-child {
   border: 0px solid transparent;
}

.content .content-item h1 {
   font-size: 1.5em; /* Ideal: 24px */
}

.content .content-item p.description {
   font-size: 0.750em; /* Ideal: 12px */
   font-style: italic;
}

.content .content-item div.article {
   text-align: justify;
}

.content .content-item img.article-image {
   float: left;
   padding: 0px 25px 25px 0px;
   width: 16.107382550335570469798657718121%; /* Ideal: 96px -> 96 / 596 */
}

.adv .adv-item {
   border-top: 1px solid #36393D;
   margin: 0px auto 0px auto;
   padding: 15px 0px 15px 0px;
   text-align: center;
   width: 85.046728971962616822429906542056%; /* Ideal: 182px -> 182 / 214 */
}

.adv .adv-item:first-child {
   border: 0px solid transparent;
}

.adv .adv-item a {
   color: #000000;
   font-style: italic;
}
```

Si j'actualise ma page sur la résolution idéale, je m'aperçois que rien n'a bougé et si je grandis ou réduis la page, les éléments conservent une bonne proportion. Je viens de remplir une partie de mon contrat.

*1280 * 960 pixels*



*1360 * 768 pixels*

*1024 * 768 pixels*

VI-B - Étape 2

Il faut quand même l'admettre, plus la résolution diminue, moins l'affichage est exploitable. D'autant plus que j'ai comme ambition de faire fonctionner mon site sur les smartphones et les tablettes, impliquant des résolutions d'écran plutôt petites.

Je décide donc de la politique suivante :

- lorsque j'arrive sur une largeur inférieure à 1024 pixels, je réduis les bandeaux blancs sur les bords ;
- lorsque j'arrive à une largeur inférieure à 960 pixels, le menu se met au-dessus des deux autres colonnes ;
- lorsque j'arrive à une largeur inférieure à 760 pixels, il n'y a plus de bandeaux blancs sur les bords ;
- lorsque j'arrive à une largeur inférieure à 640 pixels, les publicités se mettent en-dessous des articles ;
- lorsque j'arrive à une largeur inférieure à 420 pixels, je masque les publicités et je modifie le menu pour le faire afficher dans une liste déroulante (ajout donc d'une balise « <select> » dans la page, au niveau des menus) ;
- lorsque j'arrive sur une largeur inférieure à 320 pixels, je n'affiche plus l'image qui se situe dans les articles.

Si nous devions représenter cela, cela donnerait :

```
<!-- Structure de notre page : -->
<!--
   Idéal :

    ------------------------------------------------
   |                                                |
   |       ------------------------------------     |
   |      |  M  |           C          |  P  |     |
   |      |  E  |           O          |  U  |     |
   |      |  N  |           N          |  B  |     |
   |      |  U  |           T          |  L  |     |
```

```
|       |     |           E           |  I  |       |
|       |     |           N           |  C  |       |
|       |     |           U           |  I  |       |
|       |     |                       |  T  |       |
|       |     |                       |  E  |       |
|       |     |                       |  S  |       |
|       ---------------------------------       |
|                                               |
 -----------------------------------------------
```

Reformaté 1 :

```
  -----------------------------------------------
|                                               |
|       ---------------------------------       |
|       |  MENU                         |       |
|       ---------------------------------       |
|       |                 C           |  P  |   |
|       |                 O           |  U  |   |
|       |                 N           |  B  |   |
|       |                 T           |  L  |   |
|       |                 E           |  I  |   |
|       |                 N           |  C  |   |
|       |                 U           |  I  |   |
|       |                             |  T  |   |
|       |                             |  E  |   |
|       |                             |  S  |   |
|       ---------------------------------       |
|                                               |
 -----------------------------------------------
```

Reformaté 2 :

```
  -----------------------------------------------
|                                               |
|       ---------------------------------       |
|       |  MENU                         |       |
|       ---------------------------------       |
|       |                 C             |       |
|       |                 O             |       |
|       |                 N             |       |
|       |                 T             |       |
|       |                 E             |       |
|       |                 N             |       |
|       |                 U             |       |
|       |                               |       |
|       ---------------------------------       |
|       |  PUBLICITES                   |       |
|       ---------------------------------       |
|                                               |
 -----------------------------------------------
```

Reformaté 3 :

```
  -----------------------------------------------
|                                               |
|       ---------------------------------       |
|       |  MENU SIMPLE                  |       |
|       ---------------------------------       |
|       |                 C             |       |
|       |                 O             |       |
|       |                 N             |       |
|       |                 T             |       |
|       |                 E             |       |
|       |                 N             |       |
|       |                 U             |       |
|       |                               |       |
|       |                               |       |
|       |                               |       |
|       ---------------------------------       |
|                                               |
 -----------------------------------------------
```

```
-->
```

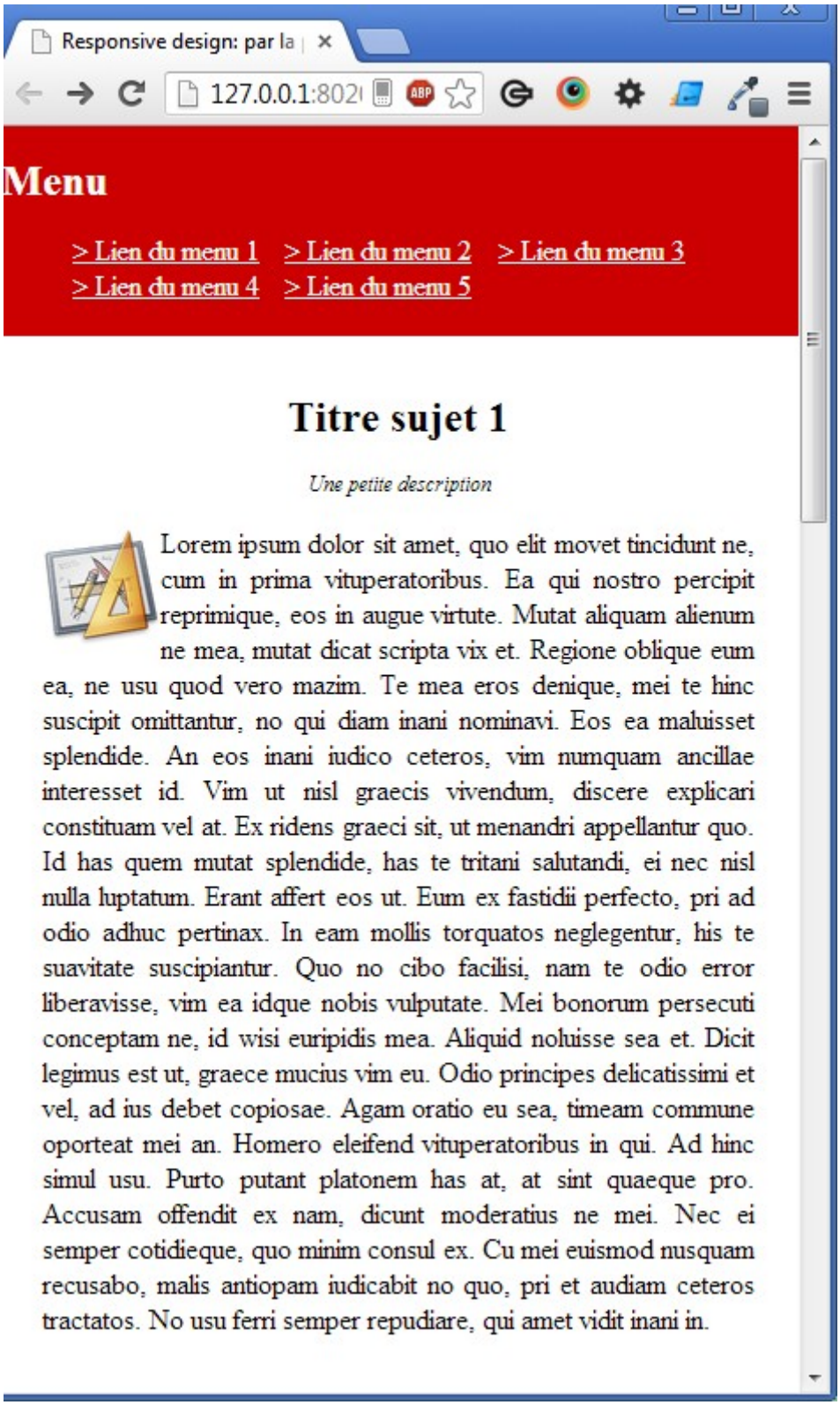
Il suffit juste d'ajouter dans notre CSS déjà fait les *Media Queries* suivantes :

```
@media screen and (min-width: 421px) {
   .menu select {
      display: none;
   }
}

@media screen and (max-width: 1024px) {
   .layout-main {
      width: 95%;
   }
}

@media screen and (max-width: 960px) {
   .layout-menu {
      display: block;
      height: auto;
      min-height: auto;
      vertical-align: top;
      width: 100%;
   }

   .layout-content {
      width: 79.1015625%;
   }

   .menu li {
      display: inline-block;
      margin-right: 2.604166666666666666666666666666667%; /* Ideal: 25px -> 25 / 960 */
   }
}

@media screen and (max-width: 760px) {
   .layout-main {
      margin: 0px 0px 0px 0px;
      width: 100%;
   }
}

@media screen and (max-width: 640px) {
   .layout-content {
      display: block;
      height: auto;
      min-height: auto;
      width: 100%;
   }

   .layout-adv {
      display: block;
      height: auto;
      min-height: auto;
      vertical-align: top;
      width: 100%;
   }

   .adv {
      text-align: center;
   }

   .adv .adv-item {
      display: inline-block;
   }
}

@media screen and (max-width: 420px) {
   .menu ul {
      display: none;
```

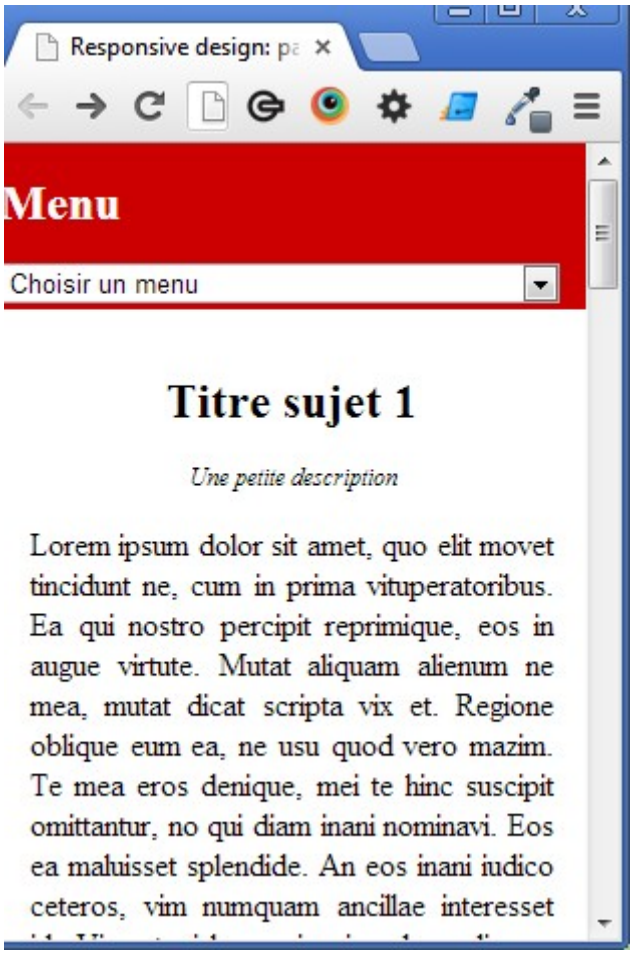
```
    }

    .menu select {
       display: inherit;
       width: 96%;
    }

    .layout-adv {
       display: none;
    }
}

@media screen and (max-width: 320px) {
    .content .content-item img.article-image {
       display: none;
    }
}
```

Cela donne les affichages suivants :

*960 * 640 pixels : mode tablette*



*800 * 400 pixels : mode smartphone large*



*480 * 320 pixels : mode smartphone*

VI-C - Étape 3

Désormais, il ne reste plus qu'à faire marcher cela sur notre ennemi endémique : IE. Pour ce faire, il suffit juste de rajouter dans le head de sa page HTML le polyfill « respond.js » :

```
<head>
   <title>Responsive design : par la pratique</title>
   <meta name="viewport" content="width=device-width, initial-scale=1, minimum-scale=1, maximum-
scale=1, user-scalable=no" />
  <meta charset="utf-8" />

   <!-- Styles -->
   <link type="text/
css" media="screen" title="no title" rel="stylesheet" href="responsive.design.css" />

   <!-- JavaScripts -->
   <script charset="utf-8" type="text/javascript" src="respond.min.js"></script>
</head>
```

Bon, je l'avoue, j'ai des surprises sous IE6 et IE7 : la propriété « display: inline-block » est mal reconnue. Qu'à cela se tienne, je fais faire un *hack* CSS (mineur) pour ces navigateurs :

```
/*
 * Hack IE6 CSS
 */
.content .content-item {
   _border-top:
 expression(this.previousSibling == null ? "0px solid transparent" : "1px solid #36393D");
}
```

```
.adv .adv-item {
   _border-top:
 expression(this.previousSibling == null ? "0px solid transparent" : "1px solid #36393D");
}

/**
 *    Hack IE6-7 CSS
 */
.layout-main {
   *clear: both;
}

.layout-menu, .layout-content, .layout-adv {
   *display: block;
   *float: left;
}

@media screen and (max-width: 960px) {
   .layout-menu {
      *float: none;
   }

   .menu li {
      *display: block;
      *float: left;
   }
}

@media screen and (max-width: 640px) {
   .layout-adv {
      *float: none;
   }

   .layout-content {
      *float: none;
      *width: 100%;
   }
}
```

Et nous obtenons l'effet escompté sous IE8, mais également sous IE7 et IE6 !

VII - Est-ce tout ?

En appliquant des règles simples, nous avons réussi à obtenir un site dit *responsive*, qui a l'avantage de fonctionner sur un ensemble important de navigateurs et de versions de navigateurs.

Néanmoins, être *responsive* ne se situe pas uniquement du côté de nos navigateurs ! En effet, si nous essayons d'afficher autant d'éléments, si nous offrons toutes les fonctionnalités et si nous essayons d'importer toutes les ressources, nous allons obtenir des sites très gourmands, gênant ainsi l'expérience utilisateur.

Un site Web *responsive*, c'est aussi un site Web qui ne va importer les ressources que lorsqu'il en a besoin (inutile d'importer les *polyfills* sur les smartphones et les tablettes : les fonctionnalités sont déjà présentes).

C'est également un site Web qui affichera des images adaptées (une image de résolution et/ou de compression différente peut être proposée pour les smartphones et tablettes, réduisant ainsi la bande passante).

C'est aussi n'offrir que les fonctionnalités essentielles en fonction de la place disponible (plus nous avons des choses à afficher, plus ce sera difficile de les atteindre/afficher avec une petite résolution ; cela permet également d'alléger la page).

Bref, faire du « *Responsive Design* », c'est offrir un site Web qui soit adapté en fonction du support (utiliser du bois pour construire la tour Eiffel aurait été laborieux et utiliser du zinc pour une porte d'entrée un peu ambitieux).

VIII - Chargement dynamique et conditionnel côté client

Être *responsive*, c'est faire en sorte que le site soit le plus rapide au chargement et dans son temps d'exécution. Pour cela nous n'allons charger les feuilles de styles et le JavaScript que lorsque que ce sera nécessaire.

Il faut savoir que selon Google, « *speed is a feature* ». Autrement dit, la rapidité est une fonctionnalité en tant que telle. De ce fait, ils ont pour objectif de charger (par exemple) Gmail en moins de deux secondes… même si votre boîte e-mail contient plusieurs Go de mails ! Bien évidemment, ils ne chargent pas tous les mails, en fait, ils ne chargent pas tout, tout court !

D'abord, ils ne vont récupérer les fichiers JavaScript que lorsque la page HTML est chargée : cela permet d'avoir son affichage tout de suite et le téléchargement/interprétation des fichiers JavaScript seront plus rapide. Il existe des *frameworks* pour nous aider à faire cela, comme « RequireJs », où nous déclarons des modules (et même des dépendances entre les modules !) : **http://requirejs.org/**.

Ensuite, Gmail n'importe que du JavaScript au strict minimum. La politique est très simple : à chaque première utilisation d'une fonctionnalité, une requête AJAX est lancée pour charger le JavaScript correspondant, l'interpréter et le rendre ensuite disponible. Cela permet à l'application de ne consommer de la mémoire que quand cela est nécessaire ! Et donc la rendre moins gourmande que si nous devions tout charger. Avec « RequireJS », cela se fait également simplement.

Ensuite, pour le CSS, nous le plaçons généralement en tête de notre page HTML, mais nous pouvons utiliser les « *Media Queries* » pour ne le charger que quand c'est nécessaire, comme l'illustre l'exemple suivant :

```
<link rel="stylesheet" media="screen" href="layout.css" />
<link rel="stylesheet" media="screen and (max-width: 640px)" href="small-screen-
layout.css" /><!-- Media Queries -->
<link rel="stylesheet" media="screen and (min-width: 1240px)" href="large-screen-
layout.css" /><!-- Media Queries -->
<link rel="stylesheet" media="print" href="print.css" />
```

Évidemment, les « *Media Queries* » ne sont pas présents sur tous les navigateurs. Nous pouvons alors placer les fichiers CSS « fondamentaux » en tête de notre page HTML et charger les autres *via* un *plugin* de « RequireJS » (en appliquant des conditions au préalable, bien sûr) : **https://github.com/guybedford/require-css**.

IX - Nos icônes et police de caractères indépendants de la résolution

Nous avons vu auparavant comment afficher les médias correctement et faire en sorte que la police ait toujours les meilleures proportions. Mais nous ne nous sommes pas intéressés aux images que nous utilisons (nous utilisons la propriété CSS « background-image »), ni à savoir quelles polices s'adaptent le mieux.

Pour nos background-image, cela est très important, car de plus en plus d'écrans dits « rétina » font leur apparition (surtout sur les tablettes). Cela veut dire que la densité de pixels (le dpi) change. Autrement dit, un pixel correspond parfois… à deux/trois pixels physiques ! Ce qui implique qu'une image avec un dpi normal sera affichée de manière plus petite sur un écran « rétina ». Nous avons deux techniques pour nous sortir de là.

Soit en utilisant les « *Media Queries* » qui, en fonction du dpi, utiliseront la meilleure image :

```
a.icon {
    background-image:url(images/icon.png);
}

@media only screen and (-webkit-min-device-pixel-ratio: 1.5),
    only screen and (-o-min-device-pixel-ratio: 3/2),
    only screen and (min--moz-device-pixel-ratio: 1.5),
    only screen and (min-device-pixel-ratio: 1.5) {

    a.icon {
```

```
        background-image:url(images/icon@2x.png);
    }
}
```

Soit en utilisant conjointement la propriété CSS3 background-size (permettant de redimensionner une image déclarée dans background-image) et en utilisant… un SVG ! De ce fait, nous aurons une image vectorielle qui s'adaptera correctement à tout type d'affichage. Rien ne nous empêche, tout comme pour nos images classiques, d'avoir des SVG contenant des « *sprites* » (un ensemble d'icônes réuni en une seule image, pour éviter de charger plusieurs fichiers).

```
a.icon {
    font-size: 1em;
    background-image: url('icon.svg');
    background-size: 1.875em 10em;
}
```

Bien sûr, les deux astuces se basent sur HTML5/CSS3. Pour les vieux navigateurs, nous pouvons utiliser le *framework* JavaScript « RetinaJS » : **http://retinajs.com/**.

Pour les polices de caractères, il est possible d'utiliser la propriété CSS3 « @font-face » permettant de déclarer des polices (et surtout les ressources liées) qui puissent s'adapter en fonction de la résolution (notamment en ayant des polices de caractères se basant sur SVG).

```
@font-face {
    font-family: 'Icon Font';
    src: url('icon-font.eot');
    src: local('...');
    url('icon-font.woff') format('woff'),
    url('icon-font.ttf') format('truetype'),
    url('icon-font.svg') format('svg'); }

.icon {
    font-family: 'Icon Font';
    font-size: 20px;
}
```

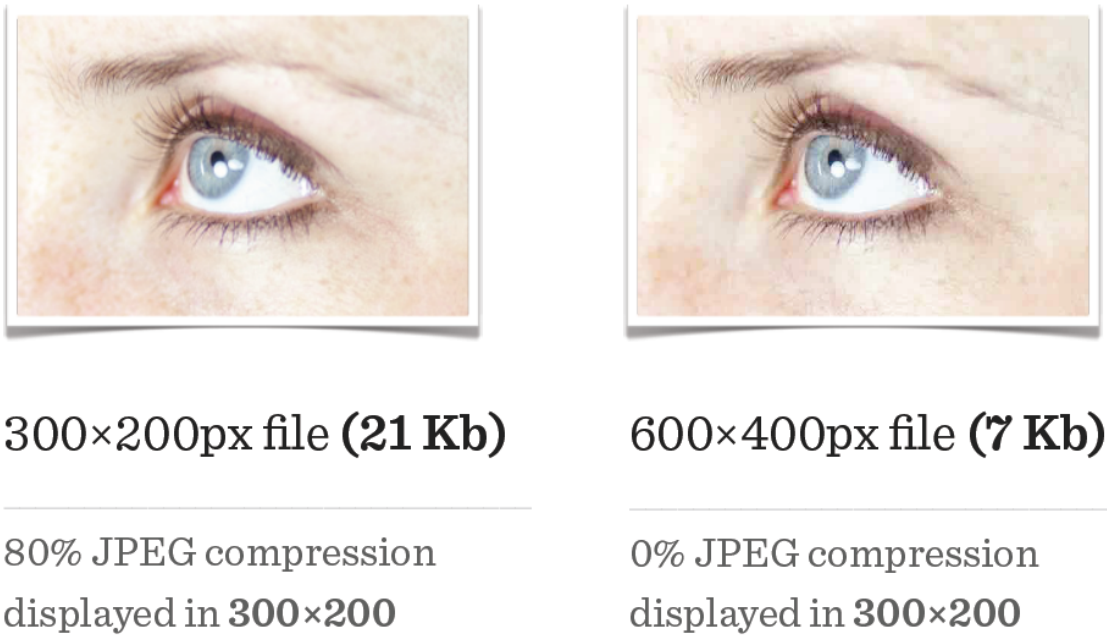
Vous pouvez voir un exemple de cela avec le *framework* « Fontello » : **http://fontello.com/**.

X - Images compressées

Il existe aussi des techniques pour avoir des images avec une très petite taille de fichier, mais qui puissent s'afficher correctement, sans effet de flou ou de créneaux de pixels.

Le principe est très simple : si nous avons une image avec une très grande résolution et un très fort taux de compression, sa taille sera plus petite qu'une même image deux fois plus petite avec une compression normale.

Comme l'illustre les *slides* de « Smashing Magazine » sur le « *Responsive Design* » (accessible sur le lien suivant : **https://speakerdeck.com/smashingmag/responsive-web-design-clever-tips-and-techniques**), nous pouvons avoir un gain de 70 % sur les tailles d'images :



Nous passons de 21 kb à 7 kb !

XI - RESS

Faire du « *Responsive Design* », c'est mettre en place une architecture nouvelle que nous n'avons pas forcément l'habitude de faire. Souvent, nous nous cantonnons à la partie navigateur, mais la partie serveur n'est pas à négliger. C'est le principe du « RESS » (*Responsive Server-Side*).

Il existe à notre disposition des *frameworks* pour cela. Nous pouvons citer « Wurfl » de Facebook, mais aussi le projet OpenSource « ua-parser » (**https://github.com/tobie/ua-parser**) permettant de détecter si nous sommes sur un support mobile ou non, voire détecter les capacités des navigateurs.

Ainsi, nous pouvons imaginer offrir une interface adaptée en fonction du support : un CSS différent, un widget dont l'affichage/l'utilisation s'adapte au support, générer différemment votre page en n'incluant pas les publicités, les commentaires de la page…

Attention, je ne dis pas que faire la détection côté navigateur est inutile, je dis qu'il ne faut pas se cantonner qu'à l'une ou l'autre : les deux sont complémentaires. Nous pouvons dans un premier temps n'utiliser qu'une des deux, mais nous arriverons très vite à des limites ou à des difficultés/avantages que l'autre solution nous pourrait nous fournir.

Prenons le cas des commentaires et des publicités : nous pourrions très bien les placer dans la page et les masquer en fonction de la taille de l'écran avec les *Media Queries* de CSS3. Mais nous allons à la fois générer du HTML/JavaScript pour rien et surtout générer du trafic réseau inutile.

Nous pouvons très bien imaginer que si nous voyons que notre support est mobile, nous ne générons pas le HTML correspondant, dans le cas contraire, nous le masquons, et le réaffichons *via* CSS3. Ou encore, nous générons des images qui sont adaptées aux mobiles (comme nous avons pu le voir sur le point précédent).

XII - Littérature, ressources et frameworks

J'en profite pour fournir une petite compilation de comptes Twitter à suivre et de livres à avoir, ainsi que de *frameworks* nous aidant à faire du Responsive Design.

Twitter :

- Ethan Marcotte : **https://twitter.com/beep** ;
- Brad Frost : **https://twitter.com/brad_frost** ;
- Future friendly : **https://twitter.com/future_friendly** ;
- Luke Wroblewski : **https://twitter.com/lukew**.

Livres :

- Responsive Web Design
  - **http://www.abookapart.com/products/responsive-web-design** ;
- Mobile First
  - **http://www.abookapart.com/products/mobile-first**.

Frameworks :

- Bootstrap : **http://twitter.github.com/bootstrap/** ;
- 960gs : **http://960.gs/** ;
- Foundation : **http://foundation.zurb.com/** ;
- Retinajs : **http://retinajs.com/** ;
- Devicejs : **https://github.com/borismus/device.js** ;
- Hammerjs : **http://eightmedia.github.com/hammer.js/** ;
- Pointerjs : **https://github.com/borismus/pointer.js/** ;
- IE7-js : **https://code.google.com/p/ie7-js/** ;
- Respond.js : **https://github.com/scottjehl/Respond** ;
- CSS3-mediaqueries.js : **https://code.google.com/p/css3-mediaqueries-js/** ;
- Ua-parser : **https://github.com/tobie/ua-parser** ;
- Wurfl : **http://wurfl.sourceforge.net/** ;
- Apache mobile filter : **http://www.apachemobilefilter.org/**.

XIII - Conclusion et remerciements

Vous pouvez voir **l'exemple de cet article en ligne** et Source **télécharger l'archive contenant les fichiers**.

Nous tenons à remercier **Torgar** et **_Max_** pour leur relecture attentive de cet article.

Et voici le forum associé à l'article :