Webdesigner

Une formation de Bruxelles Formation CEPEGRA

RWD

Responsive WebDesign

Bruxelles Formation | CEPEGRA | Olivier Céréssia | 2013

webdesigner.cepegra-labs.be

PRÉAMBULE

Vous n'êtes pas sans savoir que depuis l'apparition des périphériques mobiles entre nos mains, nos habitudes de surf ont changé.

Les périphériques qui nous servaient auparavant pour parcourir Internet étaient plutôt des stations fixes ou des ordinateurs portable mais voilà, depuis quelques années, le marché est envahi par ces tablettes et smartphones qui nous amènent à revoir complètement notre façon de vivre nos expériences de surf mais aussi, et surtout, pour nous Webdesigner, notre façon de réfléchir les produits que nous mettons entre les mains de nos utilisateurs.

Selon certains chiffres avancés par Adobe, il est prévu qu'en 2014 10% du trafic web total mondial passera par l'utilisation d'une tablette.

Ces nouvelles technologies ont aussi un grand impact sur notre façon de concevoir nos sites web : il est évident qu'on ne peut offrir la même expérience de surf à quelqu'un qui a un écran 27 pouces ou quelqu'un qui tient une tablette 10 pouces sur ses genoux.

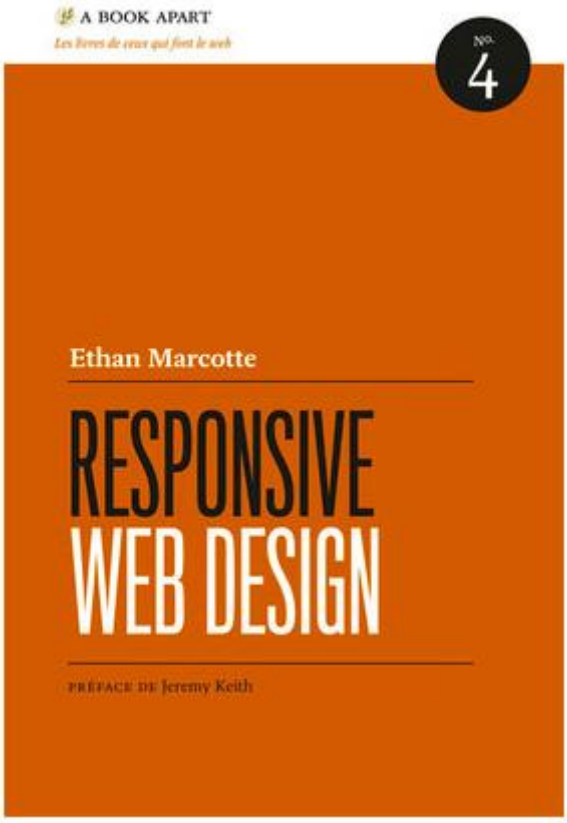
De plus, les façons de parcourir nos pages sont également totalement différentes : on laisse de plus en plus la souris de côté pour utiliser nos gros doigts ! Et cela doit avoir un impact sur les éléments que vous allez créer et qui, par exemple, devraient être cliquables !

Le responsive web design est une philosophie à part qui regroupe l'ensemble des approches et des techniques mises en œuvre pour offrir à nos utilisateur un design et un contenu adaptatif qui va varier selon le support sur lequel votre site va être consulté. Il demande une remise en question totale de la manière que nous avions, jusqu'à aujourd'hui, de concevoir un produit web.

TABLE DES MATIÈRES

INTRODUCTION

Ce cours a été réalisé en recoupant un bon paquet de sources dont la source principale est cet ouvrage très intéressant de Ethan Marcotte aux éditions Eyrolles et de la collection « A book apart » (et qui ne coûte que 12€ !) :



• D'une approche fixe à flexible

Jusqu'ici, généralement, notre approche d'un site ou, plus généralement, d'un produit web était essentiellement exprimée en valeurs fixes :

- Les valeurs de nos propriétés `font-size` étaient exprimées en px
- La majorité de nos conteneurs étaient bloqués par une valeur fixe exprimée en px sur leur propriété css `width` (généralement 960 pour le conteneur principal – inspiré de la fameuse grille de 960 qui va revenir assez fréquemment dans ce chapitre)
- Les images étaient exprimées en px
- Les `margin` et `padding` étaient exprimés en px également
- Etc.

C'est justement là que le bât blesse ! Dans une approche qui est sensée s'adapter à chaque support, il serait fou de créer X versions de votre site où X est le nombre de résolutions différentes présentes sur le marché des tablettes et des smartphones (et elles sont très nombreuses).

Nous allons donc devoir revenir aux bases du Web et des langages d'intégration.

Pendant longtemps, nous sommes allés à contre-nature du comportement de nos chères balises HTML puisque, par nature, les balises qui permettent de regrouper des contenus inline (càd les balises block) s'adaptent pusique leur `width` est de 100% de l'endroit où elles ont été placées !

Le web, à la base, était donc déjà responsive et nous allons devoir y revenir !

Néanmoins, il y a 2 façons d'approcher le responsive design qui sont diamétralement opposées :

- La dégradation élégante
- L'amélioration progressive (et nous tendrons à appliquer cette vision qui nous mènera à une approche en vogue : mobile first)

• La dégradation élégante

Des deux concepts, la dégradation élégante est celui qui, par le passé, a le plus été utilisé.

Cette appellation se retrouve d'ailleurs dans d'autres domaines que la conception Web.

Pour le web, elle consisterait à construire un site web tout d'abord pour les personnes disposant des machines/écrans/résolutions les plus avant-gardistes et ensuite de gérer les systèmes les moins performants.

En gros, ça veut dire qu'on travaille d'abord pour les versions écran et ensuite, on se penche sur le cas des périphériques mobiles … ce qui est juste l'inverse de la tendance actuelle. De plus, si on prend par exemple les problèmes liés à Flash sur tablette/smartphone, cela voudrait dire qu'on va d'abord développer une application flash qui fonctionnera parfaitement sur écran mais qui, une fois passée sur mobile, ne sera même plus visible …

Néanmoins, cette approche reste applicable dans des cas très précis : un client qui vous demande de faire une version responsive de son vieux site sans vous demander une refonte complète …

Un site qui se dégrade élégamment vaut bien mieux qu'un site totalement inutilisable pour une partie de la population, mais ce n'est pas la meilleure approche. Comme pour l'approche visuelle de la conception, il est souvent difficile de démarrer avec toutes les fonctionnalités puis de les retirer une à une, sans que tout s'effondre.

• L'amélioration progressive (ou approche mobile first)

L'amélioration progressive repose sur un principe radicalement différent de la dégradation élégante : commencer par la version de base, puis ajouter des améliorations pour ceux qui peuvent les gérer.

On commence par le balisage et on ajoute des styles par-dessus, ce qui est une amélioration progressive.

Cela veut dire qu'on va d'abord se préoccuper des configurations les plus basses pour améliorer ce que nous avions fait pour les configurations les plus costaudes.

Pour la majorité des spécialistes, il faut appliquer l'amélioration progressive à tous les niveaux : contenu, structure, images, …

Par exemple :

- Il faut d'abord utiliser la version la plus petite des images (destinée au mobile)
- Il ne faut pas appliquer de mise en forme de type colonage à la page
- Il faut mettre en avant le contenu le plus prioritaire

Avec les 3 points ci-dessus, vous avez déjà une bonne base à laquelle vous pourrez démarrer l'amélioration progressive :

- Afficher le chargement d'images plus grandes si la taille de l'écran le permet
- Appliquer une grille fluide à la page
- Faire apparaître les contenus moins prioritaires pour les résolutions qui le permettent (avec les media queries ou de l'ajax)
- Etc.

Dans cette approche, c'est **TOUJOURS** le contenu qui doit être au centre de votre réflexion, et pas le fait de travailler pour mobile ou pour écran … Il est important également que vous vous mettiez dans la peau de vos utilisateurs pour savoir sur quels éléments mettre vos priorités. (Que viennent chercher vos utilisateurs sur votre site ?)

• Les ingrédients d'un site web responsive

Du côté mise en page, pour réaliser un site web responsive (ou adaptatif), vous n'avez finalement besoin que de maitriser 3 éléments :

1. Une grille de mise en pageflexible,
2. des images et des médias flexibles,
3. et les media queries, un module de la spécification CSS3.

Nous allons voir, point par point, comment passer une maquette photoshop (qui à priori est plutôt fixe puisqu'on ne peut travailler en pourcentages sur photoshop) d'un statut fixe à flexible.

Nous veillerons à rendre la structure de vos pages (conteneurs, marges internes & externes, font-size relatifs) flexible, à faire en sorte que tous les médias que vous chargez sur vos pages (images, vidéo, etc.) s'adaptent à cette grille flexible et, finalement, nous verrons comment créer, avec media queries, des étapes où certaines règles vont changer car le design flexible a certaines limites qu'il faudra contrecarrer.

UNE GRILLE DE MISE EN PAGE FLEXIBLE

• Des variantes à nos width et nos height

Avec le passage au responsive, de nouvelles propriétés CSS vont voir le jour.

Celles-ci sont des variantes à deux propriétés que vous connaissez par cœur : width et height.

Voici ces nouvelles propriétés :

- min-width et max-width
- min-height et max-height

Soit 4 propriétés dont les plus utiles pour nous seront surtout celles qui concernent la largeur, car c'est la largeur d'un support qui va déterminer son affichage, et non sa hauteur.

Ces nouvelles propriétés sont faciles à comprendre.

Les indices « min » veulent littéralement dire « au minimum » tandis que les « max » veulent dire « au maximum ».

Prenons un exemple concret.

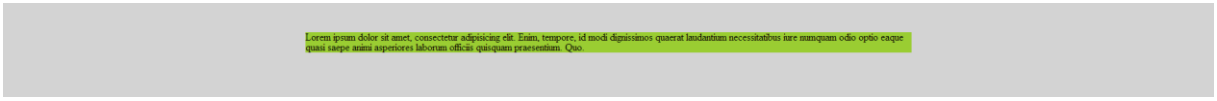
Imaginons, côté HTML, un div simple avec du texte de remplissage :

```
<div>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Enim, tempore, id modi dignissimos quaerat laudantium necessitatibus iure numquam odio optio eaque quasi saepe animi asperiores laborum officiis quisquam praesentium. Quo.</div>
```

Nous lui donnons une couleur de fond pour pouvoir facilement le distinguer, et nous le centrons en lui donnant une largeur de 960px :

```
div {background:yellowgreen; margin:0 auto; width:960px;}
```

Le résultat est, sans surprise le suivant :



En revanche, si on écrit :

```
div {background:yellowgreen; margin:0 auto; max-width:960px;}
```

Voyez le résultat par vous-même … !

Le conteneur prendra toujours au maximum, si la fenêtre de votre navigateur le lui permet, 960 pixels de large mais, s'il a moins de place que 960 pixels, alors il prendra toute la place qu'on lui offre, c'est-à-dire : 100% ! On a donc un conteneur mi-fluide, mi-fixe et ça ouvre les portes pour beaucoup de choses !

On aurait même pu ajouter un min-width en plus du max-width pour lui dire « tu feras toujours au minimum 500 pixels (par exemple) et au maximum 960 pixels ! ». Dans ce cas-ci, une fois que la

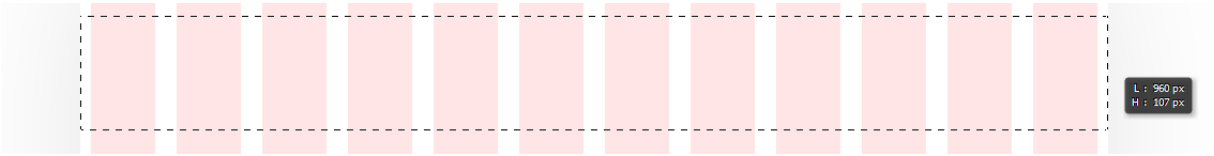
fenêtre de votre navigateur sera inférieure à 500 pixels de large, vous verrez apparaître un ascenseur horizontal, preuve qu'on a bloqué la dimension minimale.

• Passer d'une mise en page fixe à flexible

Jusqu'ici, je vous avais conseillé de réaliser les maquettes de vos sites pour une résolution de 1024 pixels et donc, en général, d'utiliser une grille de mise en page de 960px de large pour vous faciliter la vie.

Par défaut, ces grilles comportent 12 colonnes et des marges toutes identiques pour faciliter l'aération et la création de vos templates.

Ces colonnes mesurent chacune 60 pixels combinées à des marges de 10px à gauche et à droite de chaque colonne, ce qui, bout à bout, donne les fameux 960px de votre grille.



Pour rappel, on table sur du 960 pour une résolution de 1024 pour pallier aux problèmes de largeur définies des éléments de vos navigateurs : bordures, ascenseurs, etc.

Dans un premier temps, nous allons voir comment envisager le passage d'une mise en forme statique vers une mise en forme fluide.

Pour y parvenir, nous allons donc laisser de côté beaucoup des éléments que nous utilisions auparavant : marges fixes, padding fixes, largeurs fixes (on est beaucoup moins regardant au niveau de la hauteur puisque c'est la largeur d'un périphérique qui va servir de paramètre pour déterminer son affichage de telle ou telle façon), taille de fontes fixes, etc.

Partons d'un exemple concret :

Cette maquette a été réalisée à partir d'une grille de 960px de large mais nous aimerions qu'elle soit fluide et que ses éléments s'adaptent selon la résolution du périphérique qui est utilisé pour visionner cette page.

Toute la difficulté va être de traduire les éléments qui, à l'heure actuelle, sous photoshop, sont exprimés en pixels vers une unité proportionnelle (les %) qui va permettre à votre mise en page d'être élastique et de s'adapter en tout circonstance, peu importe la résolution employée pour la visionner.

Généralement, pour traduire cette interface, jusqu'ici, en valeur fixe, vous créiez un `wrapper` avec les propriétés suivantes :

```
width :960px;
margin :0 auto;
```

Pour centrer les éléments, ça reste exactement pareil, on garde le `margin :0 auto` (ouf !). En revanche, il va falloir donner une autre largeur à votre conteneur principal et trouver un moyen de l'exprimer en pourcentage.

Puisqu'on travaille sur une grille pour une résolution de 1024px et que notre conteneur fait 960px, définissons que dans cette résolution, j'aimerais que mon conteneur prenne 90% de la largeur de la page (c'est la seule décision non mathématique que vous devrez effectuer, elle sera le résultat de vos envies de design pour votre page).

Remplaçons donc les 960px de largeur de notre `wrapper` par 90% et nous aurons déjà une bonne base pour faire évoluer les différentes parties de ma page dans un conteneur élastique.

Le seul problème, c'est que dans les grandes résolutions, ce conteneur de 90% sera beaucoup trop large (et nous devrons donc utiliser une nouvelle propriété CSS pour le « cadrer » un peu et lui dire « prends 90% de la page mais ne va pas au-delà de XXX pixels (par exemple) ». Mais on y reviendra plus tard …

La règle universelle pour traduire une interface fixe vers une structure fluide est la suivante :

Largeur relative = Largeur fixe de ceconteneur / Largeur fixe du conteneur parent

Qu'est-ce que ça veut dire ?

Que la largeur relative (exprimée en poucentage) d'un conteneur sera égale à sa largeur fixe divisée par la largeur du conteneur dans lequel il se trouve.

Compliqué ?

Pas tant que ça.

Prenons un exemple concret.

Sur ma maquette des Heat, l'encart réservé aux joueurs (player roster) ne mesure pas toute la largeur de mon conteneur (qui mesure lui 960px de large et qu'on a pour le moment ramené à 90% de façon un peu arbitraire).

En réalité, il mesure exactement 820px.

Si on applique notre règle explicitée précédemment, cela nous donnerait le calcul suivant :

Largeur du conteneur « player roster » = Largeur fixe de ce conteneur (820px) / Largeur fixe du conteneur parent (ici c'est 960px).

.. ce qui nous donne un très beau résultat de : 0,8541666666666667 …

Il vous suffit ensuite :

- de déplacer la virgule de deux chiffres vers la droite : 085,41666666666667
- de supprimer le 0 qui sert à rien avant le 85 : 85,41666666666667
- d'ajouter le % à la fin : 85,41666666666667%
- de changer la virgule en point : 85.41666666666667% (pour pouvoir l'écrire ensuite en CSS)
- et finalement de l'ajouter dans votre feuille de style sous le sélecteur de votre conteneur player roster qui pourrait être :

```
.player-roster {width : 85.41666666666667%}
```

Et c'est tout ! On n'arrondit rien ! On garde bien les décimales telles quelles pour avoir un résultat le plus fidèle possible à votre maquette originale ! Youpie !

Il ne nous reste plus qu'à faire pareil pour chacune des items sur lesquelles nous avions placé une largeur fixe et le tour sera joué pour passer d'une interface fixe à une interface fluide !

Ce genre de calcul est valable pour les largeurs … mais aussi pour les espacements internes (padding) et externes (margin).

Le fait de travailler avec une grille n'a pas encore une grande importance à ce niveau-ci, sauf peut-être de vous aider à avoir les mêmes espacements entre vos éléments. Cette grille aura + d'importance quand on entrera dans la mise en pratique d'un framework CSS de type Twitter Bootstrap.

• Ainsi fontes, fontes, fontes …

Et nos fontes alors dans tout ça ?

Jusqu'ici, nous exprimions leur hauteur très précisément (en pixels) et nous n'allons pas oublier cette façon de faire, nous allons simplement en voir une nouvelle qui pourrait nous être plus précieuse et plus adaptée pour des écrans plus petits.

L'avantage du pixel ? Sur tous les supports, quoiqu'il arrive, en toute circonstance, il sera fidèle à lui-même et rendra de la même façon partout, c'est cool non ? Alors pourquoi aurait-on besoin des em ?

Simplement parce que les em vont nous faciliter la vie quand il va s'agir d'adapter les fontes de notre site pour un autre média.

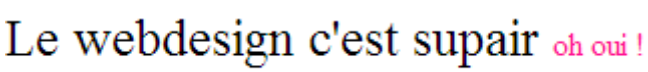
Par défaut, 1em équivaut exactement à 16 pixels, tout rond !

Question qui tue : combien de em vallent 32 pixels ? … 2em évidemment !

La grosse différence des em par rapport aux pixels, c'est que les em cherchent toujours un référent qui est leur parent pour afficher une taille de fonte tandis que les pixels sont toujours indépendants.

C'est comme ça que, si vous n'indiquez pas taille de fonte sur un document, la fonte prendra par sa valeur par défaut : 100%. Et une fonte de 100% se réfère toujours à la valeur éditer sur le body qui est de 100% également (soit 16 pixels – c'est le paramètre par défaut de tous les navigateurs … ouf ils se sont mis d'accord !).

Petit cas pratique, comment arriver à ce résutlat-ci en utilisant des em ?



Pour lecode HTML nous n'aurons rien d'autre que ceci :

```
<body>
<h1>Le webdesign c'est supair <span>oh oui !</span></h1>
</body>
```

Imaginons que, sur notre maquette, nous avons décidé d'attribuer la hauteur de 26px à notre h1, en CSS, si nous devons faire la traduction en em, voici ce que ça donnerait :

```
body {font-size:100%;}
h1 {font-size:1.625em; /* soit 26px divisé par 16px (les 100%) */}
```

On initialise, pour en être sûr, la valeur de la fonte de notre balise de référence (body) à 100% soit 16px. Ensuite on dit que, par rapport à cette référence, le H1, lui mesure 1.625em soit la division de

la valeur en pixels de notre H1 (26px) par la valeur du référent dans lequel est contenu le H1 (16px – attribué par la balise body).

Ce qui nous donne le résultat suivant :

Le webdesign c'est supair oh oui !

Malheureusement, ce n'est pas encore notre résultat final puisque la balise span n'a pas encore été mise en rose et sa taille n'a pas encore été revue à la baisse.

Dans notre maquette, nous avions attribué à ce span, un « font-size » de 16px.

Quel calcul, vais-je dès-lors devoir faire pour arriver à retrouver sa valeur relative en em ?

16px (valeur du span) / 26px (valeur du référent le plus proche sur lequel on a attribué un font-size càd la balise h1 (et plus la balise body)). Le résultat de cette division est : 0.6153846153846154em que nous allons indiquer tel quel pour respecter au mieux la taille et les proportions de notre span par rapport au H1.

```
h1 span {color:deeppink;/* ☺ */ font-size:0.6153846153846154em;}
```

Mais alors, quel intérêt de se prendre la tête comme ça avec des em, pour avoir exactement le même résultat qu'avec des pixels ?

C'est simple ! Imaginons que vous déclinez ce site pour une résolution inférieure (ou supérieure) et qu'à un moment donné vous aimeriez que toutes vos fontes soient un peu plus grandes ou un peu plus petites…

Avec des pixels, puisqu'ils sont tous indépendants, vous auriez dû contredire chacune des règles pour avoir le résultat désiré alors qu'avec les em il va vous suffire de changer la valeur initiale à laquelle tous les éléments font référence : le font-size de 100% appliqué sur le body !

Si, par exemple, vous indiquez que la fonte par défaut ne fait plus 100% de 16px mais plutôt 75%, vous auriez le résultat visuel suivant :

Le webdesign c'est supair oh oui !

Alors que, au niveau CSS, vous n'aurez changé qu'une règle :

```
body {font-size:75%;}
```

Les em vont donc vous offrir une grande facilité pour modifier, sur tout votre site d'un coup, la proportion des fontes les unes par rapport aux autres et ça, c'est puissant, non ?! ☺

• Les images flexibles

Jusqu'ici, nous avons vu comment adapter un design de son état statique à un état « élastique » ou « fluide » ce qui est le premier pas vers un site 100% responsive.

La deuxième chose dont nous allons devoir nous soucier sont les images et divers éléments multimédias placés sur notre document.

Pour mieux comprendre le problème, mettons-nous en situation.

Nous avons un conteneur (.container) de 35% de large par exemple avec une couleur de fond pour mieux voir ce qu'on fait et qui commence et se finit par un paragraphe rempli de Lorem Ipsum.

Entre ces 2 paragraphes, une image (que j'ai délibérément choisie beaucoup trop grande).

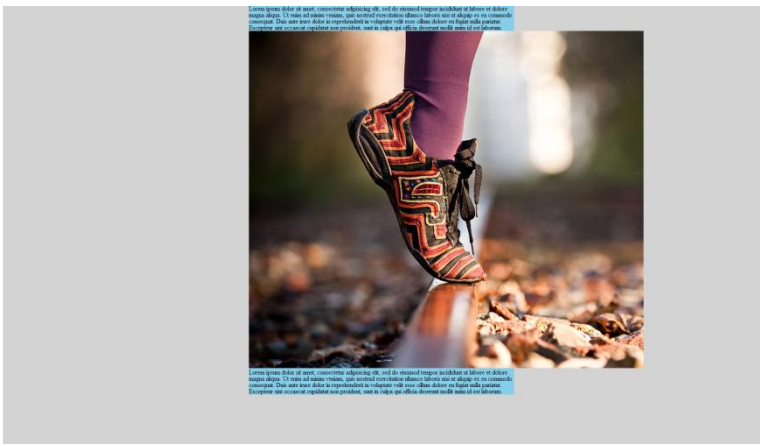
Au niveau HTML :

```
<div class="container">

      <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do
eiusmod            tempor incididunt ut labore et dolore magna aliqua. Ut
enim ad minim veniam,          quis nostrud exercitation ullamco laboris
nisi ut aliquip ex ea commodo        consequat. Duis aute irure dolor in
reprehenderit in voluptate velit esse            cillum dolore eu fugiat
nulla pariatur. Excepteur sint occaecat cupidatat non        proident, sunt
in culpa qui officia deserunt mollit anim id est laborum.</p>
      <img src="shoes.jpg" alt="Chaussure sur rail (en automne) !">

      <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do
eiusmod            tempor incididunt ut labore et dolore magna aliqua. Ut
enim ad minim veniam,          quis nostrud exercitation ullamco laboris
nisi ut aliquip ex ea commodo        consequat. Duis aute irure dolor in
reprehenderit in voluptate velit esse            cillum dolore eu fugiat
nulla pariatur. Excepteur sint occaecat cupidatat non        proident, sunt
in culpa qui officia deserunt mollit anim id est laborum.</p>

</div>
```

En css :

```
.container {width:35%; background:skyblue; margin:0 auto;}
```

Si on ajoute simplement l'image sans rien faire d'autre, voici le résultat :



Hmm, pas terrible hein :-\

Et pourtant le solution est toute simple et plutôt universelle !

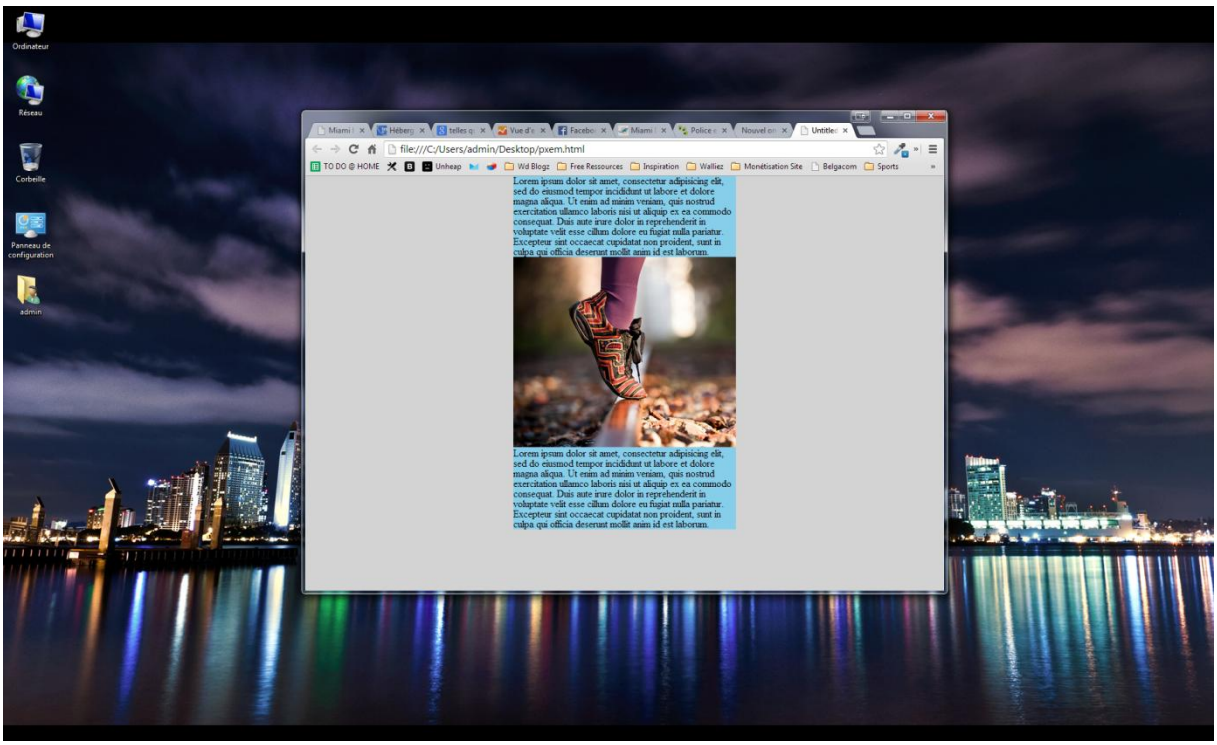
Il vous suffit de cibler toutes les images de votre document que vous désirez voir devenir « fluide » et de leur dire ceci en CSS :

```
img {max-width:100%;}
```

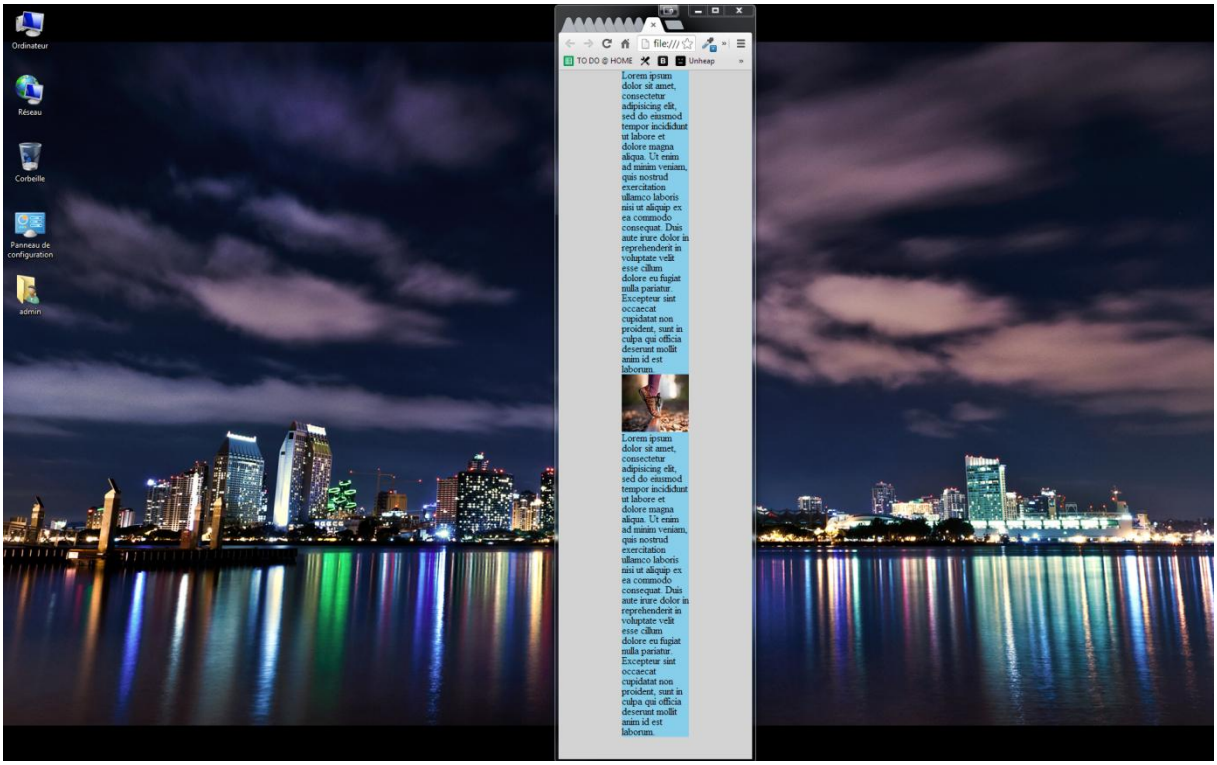
Si je devais énoncer cette propriété CSS en français, je dirais : Image, pour autant que tu sois suffisamment large pour remplir toute la largeur de là où tu te trouves, hé bien ne prends exactement que la largeur de ce conteneur !

Résultat (sur une grande résolution – 1920*1200) :

Si je redimensionne ma fenêtre en 1024*768 :



Ou dans un format « smartphone » :



Bref, cette solution est carrément par-fait-euh ! ☺

Là où ça devient carrément puissant, c'est quand vous associez cette propriété CSS sur les images avec un overflow : hidden sur le conteneur qui entoure cette image, vous pouvez de cette façon avoir une image qui va ne montrer que certaines parties d'elle-même selon la résolution tandis que les autres bouts de l'image seront cachés par l'overflow : hidden.

Un exemple simple ?

Côté HTML :

```
<div class="picture>
<img src="shoes.jpg" alt="Chaussure sur rail (en automne) !">
</div>
```

Côté CSS :

```
img {max-width:100%; display:block;}
.picture {width:35%;;height:150px; overflow:hidden;margin:0 auto;}
```

Càd en HTML on crée un conteneur et on place dans ce conteneur une image.
Ce conteneur mesure 35% de large et a une hauteur de 150px (oui oui on peut combiner les 2, c'est même fréquent et recommande). Le tout est overflow : hidden et du coup, on ne voit jamais que 150px de haut de l'image et pas plus … ce qui peut parfois être utile si on n'exagère pas trop ☺

Sur une grande résolution ça donne ceci :

Sur une résolution de 1024, ceci :

Et sur une résolution Smartphone (320 de large) ceci :

Utilisé à bon escient, sur des images qui ne sont pas forcément capitales pour la compréhension de votre contenu (ou alors avec d'autres propriétés CSS pour ne pas trop dénaturer l'apparition de l'image), ça peut vraiment le faire ☺ Retenez-le !

• Les media queries

Jusqu'ici, nous avons vu comment adapter un design de son état statique à un état « élastique » ou « fluide » ce qui est le premier pas vers un site 100% responsive.

Ce qu'il reste à faire maintenant, c'est, pour les résolutions trop grandes ou trop petites, quand les éléments sont soit trop les uns sur les autres (sur les petites résolutions) ou quand les conteneurs sont gigantesques (sur les grandes résolutions), il faut définir des règles qui vont permettre de changer l'affichage à partir de ces résolutions problématiques ou d'autres règles qui permettront au site de ne plus s'étirer au-delà d'une certaine résolution.