Cours CSS3 - Avancé
Responsive Web Design, Préprocesseurs, OOCSS

Responsive Web Design

Le responsive c'est quoi ?

C'est un mélange de techniques web permettant à vos internautes de lire votre site depuis leur tablette et leur mobile, sans avoir à zoomer.

Le principe général est que le design se réadapte en fonction de la largeur et la hauteur de l'écran.

Un positionnement en 4 colonnes se repositionnera en 3 colonnes sur une tablette par exemple, et 2 colonnes sur un mobile.

Le responsive est un mix de techniques et de technologies Web standard, qui seront comprises et compatible avec tout navigateur moderne.

Les composantes du Responsive Web Design

- Un design Fluide
- Les Media Queries

Premièrement, il convient de faire un site fluide, c'est à dire qui n'st pas fixé en hauteur et en largeur. Pour cela nous utiliserons des tailles en pourcentage sur les blocs, et en "em" sur les textes.

Viennent ensuite les média queries, qui vont permettre de changer le positionnement et le CSS selon certains critères, comme par exemple :
Si le navigateur

- est plus largue que cette taille
- moins large
- plus haut
- moins haut
- entre telle et telle taille

Plus de lecture à ce sujet :
http://www.abookapart.com/products/responsive-web-design/

Les prérequis

Avant de commencer un projet responsive, il convient d'appliquer certains styles de base pour simplifier le processus.

Afin d'être plus efficace à l'avenir, voici une base qui pourra servir pour tous vos projets :

Télécharger le .zip :
http://cl.ly/070a1F0X3q0A

Exemple en ligne :
http://codepen.io/smoothie-creative/pen/006ecbd75a86716b3b0e4c16683170be

Box Sizing

Box Sizing permet, si je définis un bloc de 600px, de prendre en compte dans le comptage le padding et les bordures. Ce qui n'est pas le cas par défaut.
Du coup un bloc de 25% de large, avec le padding et la bordure éventuelle, fera plus en réalité, et je ne pourrais pas rentrer 4 blocs en ligne.

Exemple :
http://codepen.io/smoothie-creative/pen/995e24e7d9d41608efa855ba95acba8d

Les images fluides

Il est nécessaire que les images puissent se redimensionner seules si jamais leur parent est plus petit que l'image elle même, afin d'éviter tout débordement.

On définit donc dans le css :

```
img {
    max-width: 100%;
    height: auto;
    width: auto; /* avoid ie8 bug */
}
```

De cette manière, les images seront automatiquement adaptées à leur parent.

Les tailles relatives

On utilisera donc les % pour les tailles de bloc, et les “em” pour les textes.

Principe :

on définit dans body la taille de texte par défaut (bien souvent 16px)
Le reste est défini comme un multiplieur de cette valeur

Par exemple :

```
h1 {
    font-size: 2em;
}
```

2em de 16px = 16*2 = 32 px

L’avantage est que sur un mobile, si on veut réduire tous les textes, il suffit de définir une nouvelle taille à body , par exemple 14px, et toutes les autres tailles suivront.

Notre H1 fera donc 2*14px = 28px au lieu de 32px sur un mobile

La Grille fluide

pour faire un système de colonnes, on crée un système de grille fluide pour simplifier les choses.
Des systèmes de grilles ont vu le jour, très complexes, mais on peut faire beaucoup plus simple.

Ces sytèmes comme http://www.profoundgrid.com/ ou http://960.gs/ ne sont plus trop utilisés.

A la place on met en place un système simple :

```
/* Structure de colonnes */

.col {
    float: left;
    padding: 20px;
}

.c1-1 { width: 100%; }
```

```
.c1-2 { width: 50%; }
.c1-4 { width: 25%; }
.c3-4 { width: 75%; }
.c1-3 { width: 33.33333333%; }
.c2-3 { width: 66.66666667%; }
```

Les Media Queries

Vous connaissez déjà les media queries, c'est ce qui permet de définir si le style est à destination du web ou d'un document imprimé. avec CSS3 elles ont été étendues pour détecter la taille du navigateur, l'orientation, la densité de pixels ...

Les média queries permettent de définir de nouvelles règles selon les tailles de l'écran, nous permettant de changer la disposition en fonction de la place disponible.

Exemple :
http://codepen.io/smoothie-creative/pen/4a04caef7072071782e276f8f7304ea3

Pour que cela fonctionne bien il faut appliquer la base :

- Le box-sizing
- la grille fluide

Plus d'infos sur les media queries :
https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Media_queries

On peut donc construire des sites complets et les réadapter de cette manière afin que l'utilisateur n'aie jamais d'ascenseur horizontal, ni le besoin de zoomer.

Attention !

Plus le design est complexe, plus long sera le responsive à mettre en place.

Exemples concrets

Exemples de sites Responsives :
http://www.wp-spread.com/blog/
http://www.wpinalps.com/

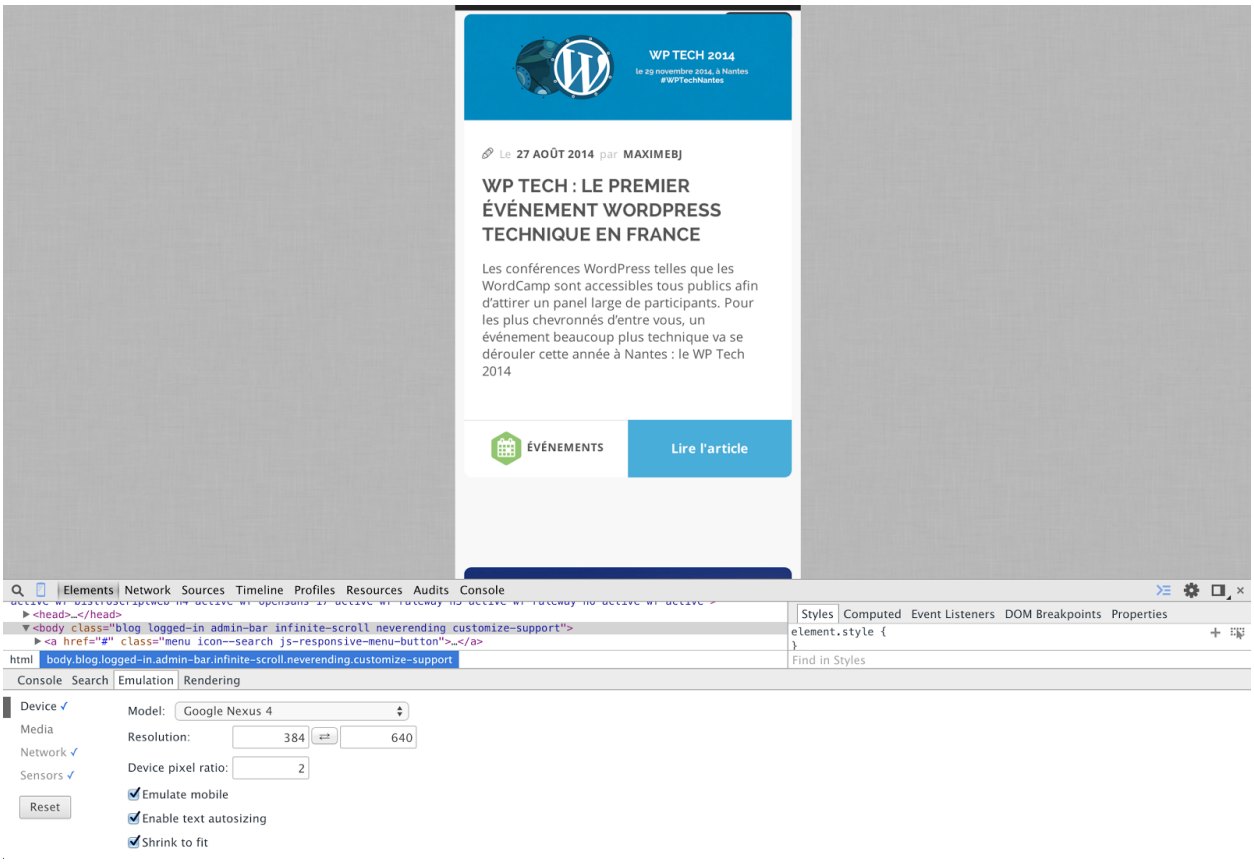
Outils pour tester le responsive d'un site

Sur Mac, vous pouvez utiliser Codekit qui compile vos fichiers scripts et CSS, et qui vous fourni un serveur virtuel accessible en local depuis vos périphériques.

Avantage : à chaque enregistrement de votre code, le navigateur recharge automatiquement la page sur votre ordinateur, vos tablettes et vos mobiles.

Très pratique !

Test virtuel avec Google chrome :
L'inspecteur Google Chrome vous permet de tester virtuellement différents périphériques afin de bien vous rendre compte du résultat.
Attention cependant car le rendu des polices et des tailles n'est pas exactement le même.



Les pré-processeurs CSS

Le problème, le principe

En écrivant du CSS, vous allez très souvent vous répéter. Les préprocesseurs sont des programmes qui vont compiler votre fichier avant de l'écrire en CSS, et de votre côté, vous allez pouvoir utiliser une syntaxe améliorée qui dispose de beaucoup d"améliorations.

Il faudra donc un logiciel qui nous permet de compiler automatiquement en CSS natif notre code "amélioré" à chaque enregistrement. Sur mac il y a l'excellent et incontournable Codekit https://incident57.com/codekit/

Sur Windows vous avez des apps spéciales, comme Sass qui permet de compiler le code sass.

Sass, Compass, Stylus et leurs amis

Il existe plusieurs préprocesseurs : Sass et compass, Stylus …

Voyons sass :
http://sass-lang.com/

On voit que l'on peut mettre sous forme de variable les valeurs redondantes :

```
$font-stack:    Helvetica, sans-serif;
$primary-color: #333;

body {
  font: 100% $font-stack;
  color: $primary-color;
}
```

Vous pouvez imbriquer des éléments au lieu de toujours les réecrire

```
nav {
  ul {
    margin: 0;
    padding: 0;
    list-style: none;
  }
```

```
  li { display: inline-block; }

  a {
    display: block;
    padding: 6px 12px;
    text-decoration: none;
  }
}
```

donc au lieu de mettre nav {} , puis nav ul {} , puis nav ul li {} , on imbrique !

Extensions pour éviter les répétitions :

```
.message {
  border: 1px solid #ccc;
  padding: 10px;
  color: #333;
}

.success {
  @extend .message;
  border-color: green;
}

.error {
  @extend .message;
  border-color: red;
}

.warning {
  @extend .message;
```

```
  border-color: yellow;
}
```

On peut également faire des opérations mathématiques :

```
.container { width: 100%; }
article[role="main"] {
  float: left;
  width: 600px / 960px * 100%;
}

aside[role="complimentary"] {
  float: right;
  width: 300px / 960px * 100%;
}
```

Point sur les techniques d'optimisation de code

Que pouvons-nous faire pour encore améliorer ça ?

Nous avons :

- Emmet, qui nous permet de transformer tac en text-align: center
- Les préprocesseurs qui nous permettent de gagner du temps

Stylus va encore plus loin dans la simplicité !

Amusons-nous avec Stylus

Stylus nous permet d'écrire en retirant tout ce qui n'est pas réellement nécessaire : les point-virgule, les deux-points et les accolades.

Voici un code avec Stylus :

```
border-radius()
  -webkit-border-radius arguments
```

```
  -moz-border-radius arguments
  border-radius arguments

body a
  font 12px/1.4 "Lucida Grande", Arial, sans-serif
  background black
  color #ccc

form input
  padding 5px
  border 1px solid
  border-radius 5px
```

Va compiler :

```
body a {
  font: 12px/1.4 "Lucida Grande", Arial, sans-serif;
  background: #000;
  color: #ccc;
}
form input {
  padding: 5px;
  border: 1px solid;
  -webkit-border-radius: 5px;
  -moz-border-radius: 5px;
  border-radius: 5px;
}
```

Propre non ?

CSS maintenables et orientés objets

Le CSS est un outil puissant, mais dans de gros projets devient très difficile à maintenir.

De bonnes lectures à ce sujet :

CSS maintenables : http://www.amazon.fr/CSS-maintenables-avec-Sass-Compass/dp/2212134177

SmaCSS
https://smacss.com/

Conventions ‘écritures efficaces :
http://cssguidelin.es/
Dont : http://cssguidelin.es/#naming-conventions

Le CSS Orienté objet

Il faut donc modulariser le CSS, afin de pouvoir le réutiliser facilement d’une page à l’autre, d’un cas de figure à l’autre et surtout d’un projet à l’autre.

L’objectif donc est de ne pas surimbriquer des styles et les rendre disponibles de partout.

Une des techniques est d’utiliser plusieurs classes par élément.

La notation BEM : faire du CSS modulaire

Ce n’est pas une technologie mais une convention de code.
on va noter un élément et ses enfants de la même manière
et on va aussi donner une classe différente pour quand il y a des modifications

Exemple :
http://codepen.io/smoothie-creative/pen/28a8baa816d17b918cbc509975c1fe5f

En résumé

- .element__enfant
    - Chaque enfant de l’élément porte le nom du parent, pour éviter les conflits avec d’autres modules
- .element--modifieur
    - Lorsqu’un élément peut changer de style en fonction du contexte, comme le prix lors d’une promotion qui serait affiché d’une autre couleur

De cette manière, chaque module est indépendant et réutilisable

Le bouton

Un nouvel exemple avec un bouton, dans lequel on va définir un style de base commun, mais la charte défini 3 ou 4 modèles de boutons différents sur le site.

Voici la meilleure manière d'écrire le CSS
http://codepen.io/smoothie-creative/pen/909ad59c0b3e7e00ee74a82f89b0ee41

Pour un gros bouton principal on utilisera alors les 3 classes : button, button--main et button--big

Astuce pour économiser une classe :
au lieu d'écrire la première classe en .button, on peut écrire [class^=button] (toute classe commençant par button). De cette manière on peut n'écrire que button--main (qui contient button)