

# CSS3 et les Mobiles

---

PeiP2 : Introduction au Web  
Dino López, Benjamin Miraglio et Julien Philip

## 1 Introduction

En CSS2, la règle `@media` avait été introduit afin de mieux adapter le formatage d'une page web au dispositif de lecture. Par exemple, on pouvait définir facilement différentes règles CSS si jamais on se trouvait sur un écran (`@media screen`) ou si la page web allait être imprimée (`@media print`). CSS2 avait introduit différents types de media : `screen`, `print`, `tv`, `handheld`. Malheureusement, le pauvre support que cette règle a connu sur les dispositifs de lecture a été à l'origine de son échec.

L'idée derrière `@media` a été cependant reprise et améliorée par les règles media queries CSS3. Avec les media queries, au lieu de s'intéresser au type de dispositif (imprimante, tv, etc.) on s'intéresse surtout aux capacités du dispositif lecteur, ce qui peut-être plus facilement fourni par des logiciel comme les navigateurs. C'est de cette manière là qu'on arrive à avoir des sites qui s'adaptent à plusieurs types de dispositifs. Comme on dit dans le jargon du développement web, on rend la page *responsive*.

## 2 Les CSS3 Media queries

Un CSS3 media queries peut obtenir des infirmations à propos de

- l'hauteur et largeur de la fenêtre de navigation (viewport)
- l'hauteur et largeur du dispositif lecteur
- orientation (landscape ou portrait)
- résolution.

Une expression media query suit la syntaxe suivante :

```
@media not|only mediatype and (expressions) { CSS-Code; }
```

Si l'expression est évaluée à « vrai », alors les règles CSS contenues entre les accolades sont exécutées.

*Mediatype* peut être : **all**, **screen**, **print** ou **speech** (ce dernier est utilisé par les logiciels de lecture des page web).

*Expressions* est utilisé pour savoir si le mediatype fourni une certaine caractéristique. Les caractéristiques qui peuvent être utilisées sont : **width**, **height**, **device-width**, **device-height**, **orientation**, **aspect-ratio**, **device-aspect-ratio**, **color**, **color-index**, **monochrome**, **resolution**, **scan**, **grid**.

Notez que plusieurs caractéristiques mentionnées ci-dessus peuvent être précédé par le mot « **min-** » ou « **max-** ».

Voici donc un exemple de media query qui cherche à savoir si le dispositif de lecture est utilisé en mode portrait.

```
@media screen and (orientation: portrait) {...}
```

Vous vous demandez sans doute « mais où écrire cette media query et toutes les règles CSS ? ». La réponse est simple : la syntaxe que nous venons de voir doit être utilisée dans un fichier CSS. En conséquence, toutes les règles CSS du dit fichier, contenues dans les accolades de la media query en question, ne sont appliquées que si l'expression donnée est vraie.

Notez cependant qu'une media query peut également être introduite directement dans notre page HTML. Pour cela, on utilisera la balise *link* que vous utilisez déjà actuellement pour lier votre page web à un fichier CSS, de la manière suivante :

```
<link rel="stylesheet" media="screen and (orientation: portrait)"  
href="portrait.css"/>
```

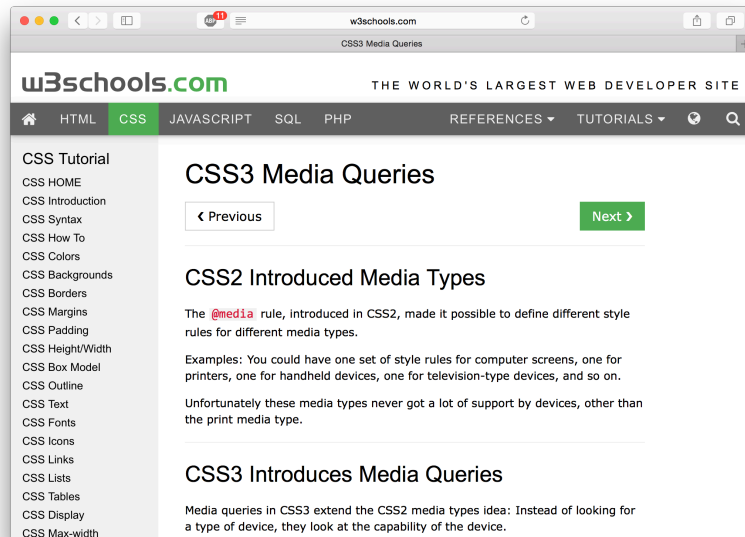
Dans l'exemple précédent, le fichier portrait.css est appliqué uniquement si la directive media query renvoie vraie.

Cette façon de procéder permet de séparer nos diverses mises en page selon la plateforme. Une organisation cohérente peut-être d'avoir un fichier CSS central contenant les informations communes à tous les supports (couleurs, police de texte ...) et de mettre les propriétés propres à chaque support dans un fichier séparé. Pour des exemples simples on peut également tout regrouper dans un seul fichier.

### **3 Quelques conseils pour une page web adaptée aux mobiles**


Pour commencer, vous devez donc appliquer certaines règles CSS uniquement si la fenêtre du navigateur est inférieure à une certaine largeur.

Pour continuer, comme vous l'avez sans doute déjà vu, la plupart des pages web dans un PC affichent au minimum 2 colonnes (une colonne pour un menu avec CSS et une colonne pour le contenu principal). On trouve aussi souvent des pages web avec un menu horizontal. Exemple :



La 2<sup>ème</sup> étape après avoir détecté un écran de petite taille consiste donc à linéariser notre page web (une seule colonne). Pour cela, si un menu horizontal existe, il devrait être minimisé à 2 ou 3 entrées, en réduisant considérablement le nombre de mots le composant. Si il n'est pas possible de réduire la taille du menu horizontal, on préfère s'en passer complètement ou bien, le rendre vertical avec une icône qui le fait apparaître. Pour ce faire on utilise javascript. Le comportement est alors le suivant : si on clique sur le bouton, notre menu vertical apparaît.

Un bouton menu peut être créé facilement en format vectoriel ou PNG ou même, en CSS. Exemple de bouton HTML/CSS :

| CSS  | HTML   | Rendu   |
|--|--|---|
| <pre>&lt;div&gt;&lt;/div&gt; &lt;div&gt;&lt;/div&gt; &lt;div&gt;&lt;/div&gt;</pre> | <pre>div {   width: 35px;   height: 5px;   background-color: black;   margin: 6px 0; }</pre> |  |

Conclusion : La double colonne doit donc laisser place à une simple colonne. Exemple :

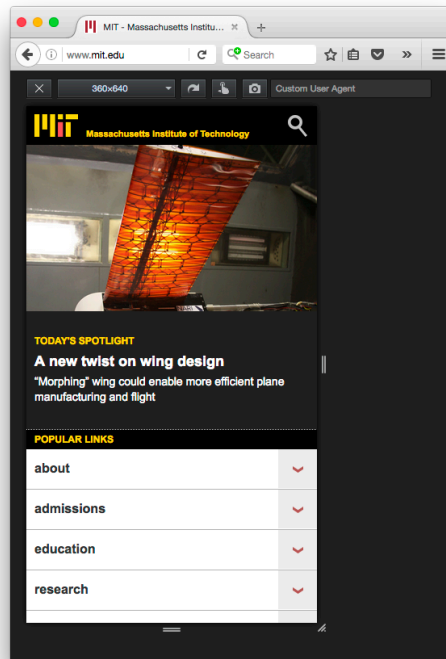


## 4 Quelques notes

### 4.1 Testez la *responsiveness* de votre page web

Vous pouvez tester l'adaptabilité des plusieurs sites avec firefox. Pour cela, vous devez activer le mode « Responsive Design ». Dans le menu Firefox, sélectionnez l'option « Web Developer », et ensuite, click sur « Responsive Design View ». En haut, à droite du bouton « close », vous trouverez un menu déroulant avec les plusieurs résolutions écrans que vous pouvez utiliser.

Vous pouvez maintenant vérifier si une page web s'adapte automatiquement aux différentes tailles (ce qui serait le signe de l'utilisation de Javascript), ou bien, si vous devez recharger la page web pour obtenir la page adapté à la taille demandé. Dans le cas où aucune des 2 options ne fonctionnerait, il se peut que le serveur soit configuré pour vérifier l' « User-Agent » du navigateur avant d'envoyer la page (votre smartphone arrive à lire la version mobile, mais pas votre navigateur), ou bien, la page n'est pas prévu pour les petits écrans tout simplement.



## 4.2 Les frameworks de développement

Actuellement, la plupart des frameworks pour le développement de sites web fournissent déjà un moyen de rendre le site responsive. Comme par exemple, Bootstrap, Foundation 3, Skeleton, etc.

L'objectif de ces frameworks est de rendre le développement web plus facile et rapide. Pour cela, ils s'appuient sur des organisation intelligentes des boîtes HTML et de preprocessors (logiciels qui reçoivent en entrée une ligne non-CSS, et qui après *parsing + substitutions*, deviennent des lignes compatibles CSS). Cependant, ils ne cherchent qu'à automatiser ce que nous avons vu précédemment et que pour bien comprendre, vous allez mettre en pratique manuellement pour l'instant (sachez que même lorsqu'on utilise un framework, on peut se voir obligé de modifier manuellement une petite partie du rendu, et sans connaître les bases d'une page responsive cela peut devenir presque mission impossible).

## 4.3 Petite note à propos de la balise « a »

Dans un site web, on peut utiliser la balise a pour créer un lien (i) vers une page web externe (eg. `href="www.google.com"`); (ii) qui ouvre un client email (eg. `href="mailto:youremail@example.com"`); (ii) mais aussi pour indiquer (dans le cas du téléphone) que le lien hypertext est en fait un numéro de téléphone auquel on peut appeler (e.g. `href="tel :+33123456789"`), entre autres possibilités.

## 5 Exercices

- 1) Comparez le site [www.lemonde.fr](http://www.lemonde.fr) et [mobile.lemonde.fr](http://mobile.lemonde.fr). Expliquez la différence architecturale entre les 2 pages du site.
- 2) Ouvrez avec firefox et avec votre éditeur HTML préféré le fichier `mapage.html`, afin d'explorer et comprendre le code.
- 3) Ajoutez à l'intérieur du fichier CSS fournit (`default.css`), des règles qui ne seront appliqué que si la largeur du navigateur est supérieure à 800px, mais également inférieure à 1024px (ce qui peut-être le cas de petites tablettes).
  - a. Quel media query devez vous utiliser ?
  - b. Faites en sorte que votre page n'ait que 2 colonnes (sans toucher au fichier HTML). On veut une colonne pour le menu vertical, plus une colonne pour le contenu.
- 4) Ajoutez dans le fichier HTML un media query qui utilisera les règles CSS d'un fichier `smartphone.css` si la taille de l'écran est inférieure à 800px.
  - a. Comment insère-t-on ce media query dans le HTML ?
  - b. Puis, sans toucher au fichier HTML, faites en sorte que votre page web passe à une seule colonne. Faites disparaître le menu vertical.
  - c. Modifier la page HTML pour ajouter un bouton « menu ». Placez le bouton en haut à gauche, comme dans l'exemple de la page W3Cschools en haut. Ensuite, faites en sorte que lorsque la souris passe sur l'icône, le menu vertical réapparaisse. Bien évidemment, l'icône de menu doit être visible uniquement pour les petits écrans. **Note :** ceci n'est qu'une preuve que vous savez gérer une page dans un petit écran, c'est un exemple jouet. Encore une fois, dans une page web à mettre sur Internet, vous devez utiliser un peu de javascript pour activer le menu avec un click (car il n'y a probablement pas de souris dans un petit écran ^\_^).
  - d. Insérez un footer avec un numéro de téléphone clickable.