2

Media queries : gérer différentes zones de visualisation

Comme nous l'avons vu au chapitre précédent, les CSS3 sont constituées de modules. Media queries est simplement l'un d'eux. Ce module permet d'allouer des styles CSS en fonction de l'écran de l'appareil. Il suffit, par exemple, de quelques lignes de CSS pour modifier la manière dont le contenu s'affiche selon la largeur de la zone de visualisation, le ratio de l'écran, son orientation (paysage ou portrait), etc.

Dans ce chapitre, nous pourrons :

- Apprendre pourquoi les media queries sont nécessaires pour un design réactif
- Apprendre à construire une media query CSS3
- Comprendre quelles sont les fonctionnalités d'un appareil pouvant être testées
- Écrire notre première media query CSS3
- Allouer des règles de style CSS à des zones de visualisation particulières
- Apprendre à faire fonctionner les media queries sur les appareils iOS et Android

Les media queries sont utilisables aujourd'hui

Les media queries sont déjà largement utilisées et sont bien gérées par un grand nombre de navigateurs (Firefox 3.6+, Safari 4+, Chrome 4+, Opera 9.5+, iOS Safari 3.2+, Opera Mobile 10+, Android 2.1+ et Internet Explorer 9+). De plus, elles sont faciles à implémenter (bien que fondées sur JavaScript) et il existe des **rustines** pour les anciens navigateurs comme les versions 6, 7 et 8 d'Internet Explorer. Si vous avez besoin de vous les procurer immédiatement, allez au Chapitre 9, "Résoudre les défis liés à la compatibilité des sites réactifs avec tous les navigateurs". En bref, il n'y a aucune raison de ne pas les utiliser dès aujourd'hui !

NOTE

Les spécifications du W3C sont soumises à une procédure de ratification en quatre étapes (si vous avez une journée libre, assommez-vous avec l'explication officielle de cette procédure, à l'adresse **http://www.w3.org/2005/10/Process-20051014/tr**), de ***Working Draft*** (**WD**) à ***Candidate Recommendation*** (**CR**), à ***Proposed Recommendation*** (**PR**) avant d'arriver, des années plus tard, à l'état de ***W3C Recommendation*** (**REC**). Les modules ayant un niveau de maturité plus élevé sont généralement plus sûrs à utiliser. Par exemple, le module *CSS Transforms Module Level 3* (**http://www.w3.org/TR/css3-3d-transforms/**) en est au statut WD depuis mars 2009 et le support des navigateurs est loin d'atteindre le niveau des modules CR comme celui des media queries.

Un design réactif a besoin de media queries

Sans le module CSS3 media query, il est impossible d'allouer des styles CSS liés à des capacités d'appareils donnés, comme la largeur de la zone de visualisation. Si vous consultez les spécifications du W3C le concernant (**http://www.w3.org/TR/css3-mediaqueries/**), vous verrez qu'il s'agit de l'explication officielle de ce que sont les media queries :

"HTML 4 et CSS2 gèrent des feuilles de style dépendantes des médias et mises au point pour différents types de médias. Par exemple, un document peut utiliser des fontes sans-sérif pour l'affichage à l'écran et des fontes avec sérif pour l'impression. "screen" et "print" sont deux types de médias qui sont définis. Les media queries élargissent la fonctionnalité des types de médias en permettant de donner des styles plus précis dans les CSS."

"Une media query est constituée d'un type de média et de zéro ou plusieurs expressions qui vérifient les conditions des fonctionnalités d'un média donné. Les fonctionnalités utilisables sont "width", "height" et "color". Les présentations des pages peuvent donc être personnalisées pour une gamme précise d'appareils sans modifier le contenu lui-même."

Syntaxe des media queries

À quoi ressemble une media query et, plus important, quel est son fonctionnement ?

Saisissez ce code au bas d'un fichier CSS et visualisez la page web qui s'y rapporte :

```
body {
  background-color: grey;
}
@media screen and (max-width: 960px) {
  body {
    background-color: red;
  }
}
```

```
@media screen and (max-width: 768px) {
  body {
    background-color: orange;
  }
}
@media screen and (max-width: 550px) {
  body {
    background-color: yellow;
  }
}
@media screen and (max-width: 320px) {
  body {
    background-color: green;
  }
}
```

Ouvrez le fichier dans un navigateur moderne (au minimum IE 9, si vous utilisez IE) et redimensionnez-en la fenêtre. La couleur de l'arrière-plan de la page variera en fonction de la taille de la zone de visualisation. J'ai nommé les couleurs pour plus de clarté, mais utilisez plutôt leur code Hexa, par exemple `#ffffff`.

Continuons et décortiquons ces media queries pour comprendre comment les utiliser au mieux.

Si vous êtes habitué aux feuilles de style CSS2, vous savez qu'il est possible d'appeler une feuille de style par type d'appareil (par exemple `screen` ou `print`) grâce à l'attribut `media` de la balise `<link>`. Regardez l'extrait de code ci-après, le lien a été placé dans la balise `<head>` du fichier HTML :

```
<link rel="stylesheet" type="text/css" media="screen"
href="screen-styles.css">
```

Les media queries fournissent, essentiellement, la possibilité d'allouer des styles en fonction de la *capacité* ou des *fonctionnalités* d'un appareil, et non simplement selon le *type* d'appareil. Pensez-y comme une question posée au navigateur. Si la réponse de celui-ci est "vrai", les styles inclus sont appliqués. Si sa réponse est "faux", ils ne le sont pas. Les media queries ne se contentent pas de demander au navigateur "Êtes-vous un écran ?", ce qu'il est déjà possible de faire avec CSS2. Elles posent plus de questions, comme : "Êtes-vous un écran et êtes-vous orienté en mode portrait ?" Regardons cet exemple :

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

La media query demande d'abord le type (êtes-vous un écran ?), puis la fonctionnalité (l'écran est-il en mode portrait ?). La feuille de style `portrait-screen.css` sera chargée pour tout appareil à écran dont l'orientation est en mode portrait et ignorée par tous les autres. Il est possible d'inverser la logique d'une expression de media query en y ajoutant `not` au début. Concrètement, ce code annule le résultat

de l'exemple précédent et charge le fichier pour tout ce qui n'est pas un écran avec orientation en mode portrait :

```
<link rel="stylesheet" media="not screen and (orientation: portrait)"
href="portrait-screen.css" />
```

Il est aussi possible d'assembler plusieurs expressions. Par exemple, élargissons la première media query et limitons le fichier aux appareils dont la zone de visualisation est supérieure à 800 pixels :

```
<link rel="stylesheet" media="screen and (orientation: portrait)
and (min-width: 800px)" href="800wide-portrait-screen.css" />
```

On peut même avoir une série de media queries. Si l'une des requêtes est vraie, le fichier est chargé. Si aucune n'est vraie, il ne le sera pas. Voici un exemple :

```
<link rel="stylesheet" media="screen and (orientation: portrait)
and (min-width: 800px), projetion" href="800wide-portrait-screen.css" />
```

Notez deux choses. D'abord, une virgule sépare chaque requête, ensuite, après `projetion`, il n'y a aucun caractère de fin ni de fonctionnalité/valeur entre parenthèses. En l'absence de valeurs, la media query s'applique à tous les types. Ici, les styles s'appliqueront à tous les projecteurs.

Les media queries, comme les règles existantes des CSS, peuvent charger conditionnellement des styles de plusieurs manières. Jusqu'ici, ce sont des liens vers des fichiers CSS placés dans la section `<head></head>` du code HTML. On peut cependant les utiliser dans les feuilles de style elles-mêmes. Si nous ajoutons le code suivant dans une feuille de styles tous les éléments `h1` seront en vert si la largeur de l'écran de l'appareil est de 400 pixels ou moins :

```
@media screen and (max-device-width: 400px) {
  h1 { color: green }
}
```

On peut aussi utiliser la règle CSS `@import` pour charger conditionnellement des feuilles de style dans la feuille de style principale. Le code suivant importe la feuille de style `phone.css` si l'appareil est basé sur un écran et que sa zone de visualisation maximale est de 360 pixels :

```
@import url("phone.css") screen and (max-width:360px);
```

La fonctionnalité CSS `@import` ajoute une requête HTTP (ce qui influe sur la vitesse de chargement) ; utilisez donc cette méthode avec parcimonie.

Que testent les media queries ?

Avec un design réactif, les media queries les plus fréquentes concernent la largeur de la zone de visualisation des appareils (`width`) et celle de leurs écrans (`device-width`). J'ai personnellement trouvé peu d'appels à tester d'autres capacités. Cependant, si vous en avez besoin, voici une liste de ce que les media queries peuvent tester. Avec un peu de chance, certaines vous intéresseront :

- `width`. Largeur de la zone de visualisation.
- `height`. Hauteur de la zone de visualisation.
- `device-width`. Largeur de la surface visible (largeur de l'écran d'un appareil).
- `device-height`. Hauteur de la surface visible (hauteur de l'écran d'un appareil).
- `orientation`. Vérifie l'orientation de l'appareil (mode portrait ou paysage).
- `aspect-ratio`. Ratio entre la largeur et la hauteur basé sur la largeur et la hauteur de la zone de visualisation. Le ratio d'un écran large en 16:9 peut ainsi s'écrire `aspect-ratio: 16/9`.
- `device-aspect-ratio`. Semblable à `aspect-ratio:`, mais fondé sur la largeur et la hauteur de la surface de rendu de l'appareil et non sur sa zone de visualisation.
- `color`. Nombre de bits par composant de couleur. Par exemple, `min-color: 16` vérifie que l'appareil est en 16-bit.
- `color-index`. Le nombre d'entrées dans la table des couleurs de l'appareil. Les valeurs sont des chiffres qui ne peuvent être négatifs.
- `monochrome`. Teste le nombre de bits par pixel dans le tampon de trame monochrome. La valeur est un chiffre entier. Par exemple, `monochrome: 2` ne peut être négative.
- `resolution`. Teste la résolution de l'écran ou de l'impression. Par exemple `min-resolution: 300dpi`. Elle accepte aussi les mesures en points par centimètre ; par exemple `min-resolution: 118dpcm`.
- `scan`. Teste les fonctionnalités progressives ou entrelacées (plutôt réservé aux TV). Par exemple, une TV HD en 720p (p signifiant "progressif") peut être ciblée avec `scan: progressive` alors qu'une TV HD en 1080i (i signifiant "entrelacé") le sera avec `scan: interlace`.
- `grid`. Indique si l'appareil est basé sur une grille ou des images bitmap.

```
  width: 150px;
  height: 394px;
}
#sidebar {
  border-right: none;
  border-top: 2px solid #e8e8e8;
  padding-top: 20px;
  margin-bottom: 20px;
}
.sideBlock {
  width: 46%;
  float: left;
}
.overHyped {
  margin-top: 0px;
  margin-left: 50px;
}
```

Ces styles ne concernent que les appareils dont la zone de visualisation est de 768 pixels ou inférieure. Les zones de visualisation plus larges les ignorent. De plus, puisque ces styles sont placés après les styles existants, ils les outrepassent quand nécessaire. Résultat : les zones de visualisation les plus larges conservent le même design. Les appareils dont la largeur de la zone de visualisation est de 768 pixels ont l'aspect montré dans la capture d'écran suivante.

Il va sans dire que nous ne gagnerons aucun prix de design avec cette page, mais ces quelques lignes de code CSS dans une media query ont servi à créer une mise en page différente destinée à différentes zones de visualisation. Qu'avons-nous fait au juste ?

D'abord, nous donnons à toutes les zones de contenu la même largeur que celle de la media query, comme dans cet extrait de code :

```
#wrapper, #header, #footer, #navigation {
  width: 768px;
  margin: 0px;
}
```

Ensuite, il suffit d'ajouter des styles pour modifier la mise en page des éléments. Par exemple, cet extrait de code modifie la taille de la navigation, de la mise en page et de l'arrière-plan pour que les utilisateurs de tablettes (ou d'appareils avec une zone de visualisation de 768 pixels ou moins) puissent plus facilement sélectionner un lien de navigation :

```
#navigation {
  text-align: center;
  background-image: none;
  border-top-color: #bfbfbf;
  border-top-style: double;
  border-top-width: 4px;
  padding-top: 20px;
}
#navigation ul li a {
  background-color: #dedede;
  line-height: 60px;
  font-size: 40px;
}
```

À présent, le même contenu s'affiche dans une mise en page différente adaptée à la taille de la zone de visualisation. Les media queries sont bien, non ? Fêtons cela. Pendant que vous allez chercher le champagne, je vais regarder dans mon iPhone à quoi cela ressemble… Vous pouvez le voir sur la capture d'écran qui suit.

Les media queries ne sont qu'une partie de la solution

Ah… Remettez les glaçons dans le freezer ! Notre boulot est loin d'être terminé. La page est horrible dans la petite zone de visualisation de 320 pixels de large de mon iPhone. La media query fait exactement ce qu'elle doit faire : elle applique les styles selon les capacités de l'appareil. Le problème est que cette media query ne couvre qu'un spectre très étroit de zones de visualisation. Un appareil avec une zone de visualisation inférieure à 768 pixels de large verra un contenu tronqué, de même que tout appareil dont la zone de visualisation se situe entre 768 et 960 pixels puisqu'il obtiendra la version de la feuille de style sans la media query, qui, on le sait déjà, ne s'adapte pas lorsqu'elle mesure moins de 960 pixels de large (votre auteur se prend la tête entre ses mains et pousse un long soupir).

Une mise en page fluide est nécessaire

N'utiliser que les media queries pour modifier une conception convient si l'on vise un appareil précis. Nous venons de voir qu'adapter un design à un iPad est très facile. Mais cette stratégie présente de gros inconvénients : concrètement, elle n'est pas à l'épreuve du futur. À présent, la mise en page s'adapte et se modifie à chaque point de rupture indiqué dans les media queries dès que l'on redimensionne la zone de visualisation. Elle demeure toutefois statique, entre deux "points de rupture" de zones de visualisation. Il faut améliorer ceci. Écrire des styles CSS particuliers pour chaque zone de visualisation ne prend pas en compte les appareils du futur, et un vrai bon design doit pouvoir intégrer ce futur, du moins en partie. À ce stade, notre solution est incomplète. Il s'agit davantage d'un design adaptatif que du vrai design web réactif que nous souhaitons. La page doit passer par une phase intermédiaire avant de s'adapter. Pour l'obtenir, il est nécessaire de transformer la mise en page rigide et fixe en une mise en page fluide.

Synthèse

Dans ce chapitre, nous avons vu les media queries CSS3, comment les intégrer dans nos fichiers CSS et comment elles nous aident à créer un *responsive web design*. Nous avons aussi appris à obtenir des navigateurs mobiles le même rendu des pages que leurs équivalents pour ordinateurs et abordé le concept de type "le contenu d'abord" dans la structure des balises. Nous avons aussi vu les économies de données réalisées en utilisant les images avec modération.

Nous avons aussi appris, toutefois, que les media queries ne peuvent fournir qu'un design web adaptable et non un site réellement réactif. Elles sont un composant essentiel de cette technique, sachant qu'une mise en page fluide permet à nos conceptions d'évoluer entre les points de rupture gérés par les media queries. Nous traiterons au prochain chapitre la création d'une mise en page fluide pour amortir la transition entre ces divers points de rupture.