EP2IC3IV - ECUE Introduction au web

Introduction

Le Responsive Web design est un ensemble de méthodes qui permettent de concevoir un site web capable d'adapter sa mise en page au type d'appareil de lecture détecté (ordinateur fixe, écran TV, téléphones mobiles, tablettes, liseuses,etc.). Le terme de "Responsive Web Design" a été introduit par Ethan Marcotte dans un article de *A List Apart* publié en mai 2010 et développe les aspects pratiques du web *responsive* dans son ouvrage **"Responsive Web Design"** publié en 2011.

En CSS2, la règle **@media** a été introduite afin d'associer des règles CSS spécifiquement à différents types de média de lecture. Par exemple, si l'utilisateur consulte la page web sur un écran on utilisera les règles CSS associées à *@media screen* alors que s'il envoie la page vers l'imprimante on utilisera les règles associées à *@media print*.

L'idée derrière @media a été reprise et améliorée par les règles ***media queries*** en CSS3. Au lieu de s'intéresser au type du dispositif, les *media queries* s'intéressent à une multitude de caractéristiques du média utilisé telles que, par exemple, la résolution de l'écran ou encore la largeur de la partie visible d'un écran (*viewport*). Ces caractéristiques sont facilement fournies par les navigateurs web.

Les CSS3 Media Queries

Les CSS3 Media Queries permettent de détecter différents **types de média** (liste non exhaustive) :
all : tous les appareils,
print : document en aperçu avant impression,
screen
et ses **caractéristiques** associées (liste non exhaustive) :
la taille de l'écran,
la taille de la fenêtre,
la résolution,
le niveau de luminosité,
l'orientation de l'écran (portrait ou paysage),
etc.
Pour une liste exhaustive, rendez-vous sur la page : https://developer.mozilla.org/fr/docs/Web/CSS/@media.

Syntaxe et principe :

Une règle *media query* suit la syntaxe suivante :

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

La condition entre parenthèses s'appelle une requête média. Il s'agit d'une caractéristique de media qui est évaluée à True ou False. Il est possible de combiner plusieurs requêtes media avec des connecteurs logiques (*e.g.* and, or, not). Le type de média (*mediatype*) désigne une catégorie d'appareil et fait également partie de la requête média. Il est cependant optionnel. Notez que les requêtes qui utilisent des types de médias non reconnus sont évaluées à False. Au sein des accolades se trouve une règle CSS dont la syntaxe a été décrite dans la section 1. Cette règle CSS est appliquée si la requête média est évaluée à True.

Les caractéristiques sont par exemple :
width,
height,
device-width,
device-height,
resolution,
etc.
Notez que plusieurs caractéristiques mentionnées ci-dessus peuvent être précédées par le terme *min-* ou *max-* afin de préciser une requête média.
Voici un exemple de *media query* qui vérifie que le dispositif de lecture est un écran et qu'il est utilisé en mode portrait :

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

Où écrire la *media query* ?

La syntaxe décrite plus haut doit être utilisée dans un fichier CSS qui déclare le style de votre page mais cette fois de manière conditionnelle.
Il est également possible d'introduire la *media query* directement dans le fichier HTML en utilisant la balise <link> que l'on utilise habituellement pour lier une page web à un fichier CSS. On procède de la manière suivante :

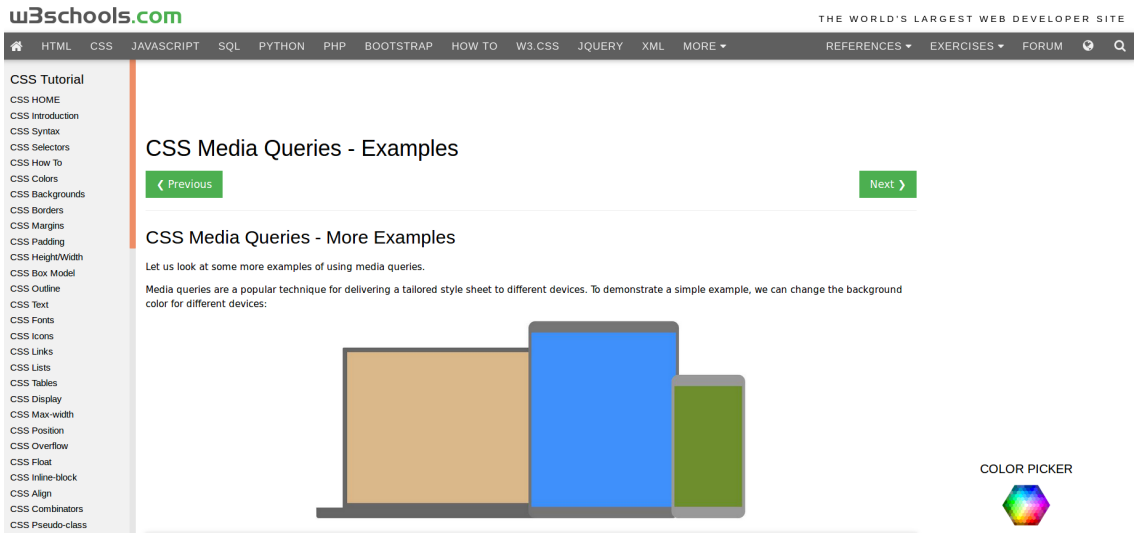
```
<link rel="stylesheet" media="screen and (orientation: portrait)" href="portrait.css">
```

Les règles CSS du fichier portrait.css sont appliquées uniquement si la requête média *"screen and (orientation: portrait)"* renvoie True. Cette manière de procéder permet de rassembler dans un fichier CSS les informations de styles communes à tous les types d'appareils utilisés (e.g. couleurs, police de texte, etc.) d'une part, et les propriétés spécifiques à chaque support dans un ou plusieurs fichiers séparés d'autre part.

Des pages web adaptées aux smartphones

De la version PC à la version mobile

Les mobiles ont la particularité de posséder une taille d'écran plus petite qu'un écran de PC. Le site web doit donc être capable de s'adapter au minimum à ces deux types d'appareils. La structure d'une page web va donc être impactée par ce critère d'adaptabilité. La plupart des pages web dans un PC affichent au *minimum* 2 colonnes : une pour un menu de navigation et une autre pour le contenu principal. Il est également courant de retrouver des pages web avec un menu horizontal comme abordé dans la section précédente.
Par exemple :



Les *media queries* sont utilisées pour adapter la mise en page aux mobiles après avoir détecté un écran de plus petite taille. Les règles CSS consisteront à "linéariser" la page web, en diminuant le nombre de colonnes. Ainsi, pour les plus petits écrans (e.g. des écrans de smartphones), il vaut mieux ne laisser qu'une seule colonne. S'il existe, le menu horizontal doit être limité à deux ou trois entrées au *maximum* avec une taille de police et le nombre de mot/entrée drastiquement réduit. Si le menu horizontal ne peut pas être réduit sous peine d'altérer le sens des champs, le menu horizontal doit être transformé en menu vertical, visible lorsqu'on *click* sur un bouton "menu".
Dans l'exemple suivant, nous avons un menu horizontal et un menu vertical. Ce dernier est visible en clickant sur le bouton "menu", en haut à gauche, sous forme de 3 petites barres horizontales :

Un bouton "menu" peut être créé facilement en format vectoriel ou PNG, voire même en HTML/CSS comme proposé ci-dessous :

HTML	CSS	Rendu

`<div></div>`
`<div></div>`
`<div></div>`
	`div {`
	`  width: 35px;`
	`  height: 5px;`
	`  background-color: black;`
	`  margin: 6px 0;`
	`}`
		☰

"Linéarisation" d'une page avec des règles CSS

Si la création d'une page web avec 2 ou plus de colonnes est basée sur la mise en place de boîtes sur un flux flottant, la linéarisation consiste simplement à remettre les boîtes flottantes sur le flux normal et, si nécessaire, en modifiant les largeurs des colonnes. Par exemple, pour passer une page web ayant 2 colonnes à une seule colonne, on supprime la propriété "float: left;" ou "float: rigth;" de l'une des colonnes, puis, on lui permet de prendre toute la largeur de la page web.

On supprime (ou "écrase") une propriété "float: left;" ou "float: rigth;" avec la règle "float: none;". Une largeur minimale est supprimée avec la règle "min-width: auto;" et une largeur maximale avec "max-width: none;". Une largeur fixe est supprimée avec la règle "width: auto;".

Nous réutiliserons enfin les propriétés "display:none;" et "display: block;", afin de montrer ou cacher un menu dans une page web responsive.

Quelques notes à usage pratique

Smartphone et *viewports* virtuels

Dans un dispositif mobile, les pages web sont souvent affichées *via* une fenêtre virtuelle nommée *viewport*. Ce dernier possède une taille supérieure à l'écran physique du dispositif et lorsqu'une page web *responsive* y est chargée, la *media query* cherchant un écran de taille inférieure ne pourrait jamais être évaluée à vraie. Pour éviter ce problème et raisonner en fonction de la taille de l'écran physique, il faut utiliser la balise *<meta>* de cette façon :

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

Tester la *responsiveness* d'une page web

En pratique pour tester l'adaptabilité d'un site web sur le navigateur web Firefox vous devez :

1. Sélectionner l'option *Web Developer* dans le menu de Firefox.
2. Cliquer sur *Responsive Design Mode*.
3. En haut dans le menu horizontal : sélectionner un champ de la première entrée pour visualiser différentes résolutions d'écran (smartphone, ipad, kindle, etc).

Exercices

Les ressources pour ce TP sont disponibles comme toujours en fin de page

Exercice 1. Page adaptée aux écrans de type « tablette »

1. Pour nos exercices, vous devez partir des fichiers données en ressources pour ce TP (qui n'est pas autre chose que la correction au TP 2 "Applications en HTML/CSS")

- séparez le code du TP précédent de ce TP avec un commentaire (e.g. "/* ==== regles exercice 3.1 ==== */").

2. Dessinez l'architecture de page par défaut (version faite lors du dernier TP => pour un écran supérieur à 800px) et l'architecture qu'aurait cette même page, si on devait l'adapter à un écran pour un dispositif plus petit, comme une tablette dont la largeur peut être inférieure à 800 pixels.
   - Voici une vidéo montrant à quoi ressemble cette adaptation.
   - Identifiez les boîtes qui devront changer de style d'affichage pour parvenir à la nouvelle architecture;
   - Précisez/expliquez quelles modifications ces boîtes doivent subir. Ne donnez aucun code pour répondre à cet exercice.
3. Maintenant, essayons d'implémenter cette adaptation "tablette". Ajoutez à l'intérieur du fichier CSS des règles qui ne seront appliquées que si la largeur du dispositif de lecture est inférieure ou égale à 800px (ce qui peut-être le cas de petites tablettes).
   - Quel *media query* devez-vous utiliser ?
   - Faites en sorte que votre page ne comporte que 2 colonnes (sans toucher au fichier HTML). On veut une colonne pour le flux twitter à droite, et une 2ème colonne avec le contenu principal à gauche;
   - La boîte du contenu principal ne possède qu'une seule colonne (i.e. les trois colonnes internes doivent être « linéarisées »).

Précisions :

il y a une distance de 10 pixels entre les bords droit et gauche de la page (balise <body>)
la section principale possède une largeur maximale de 65% de la page
le flux twitter possède une marge à gauche de 67%

Exercice 2. Page adaptée aux écrans de type « smartphone »

Ajoutez dans le fichier HTML de l'exercice 1, un *media query* qui appliquera les règles CSS d'un fichier *smartphone.css* si la taille de l'écran est inférieure ou égale à 600px. Vous devez créer le fichier *smartphone.css* à partir de rien.

1. Comment insère-t-on ce *media query* dans le fichier HTML ?
2. Dessinez l'architecture de la page adaptée aux tablettes et l'architecture de page adaptée aux petits écrans.
   - Voici une vidéo montrant à quoi ressemble cette adaptation.
   - Identifiez les boîtes qui devront changer de style d'affichage pour parvenir à la nouvelle architecture
   - Précisez/expliquez quelles modifications ces boîtes doivent subir. Ne donnez aucun code pour répondre à cet exercice.
3. Puis, sans toucher au fichier HTML, faites en sorte que votre page web passe à une seule colonne uniquement.
   - Faites disparaître le menu vertical.
   - Remplacez-le par un bouton « menu de navigation ». Le menu textuel n'apparaîtra que lorsque la souris sera placée sur la région de navigation. Note : dans une page web à mettre sur Internet, vous devez utiliser un peu de Javascript (cf. cours Application du Web du 2nd semestre) pour activer le menu avec un "*click*" (car il n'y a probablement pas de souris dans un petit écran :-)). Cependant, le code CSS que vous emploierez ici reste le même, à l'exception de la pseudo-classe « :hover » bien sûr.

Précisions item 3 :

le bord rouge du flux twitter a été déplacé vers le haut
le menu horizontal devient un menu vertical. Notez que
le menu vertical est caché par défaut et uniquement l'icône « menu de navigation » à 3 barres horizontales est visible par défaut (voir la balise dans le fichier HTML);
lorsque la souris se trouve sur la zone de navigation (i.e. sur la boîte dédiée aux menus de navigation), le bouton disparait et laisse la place au menu textuel;
la liste correspondant au menu principal n'est plus limitée en largeur;
la largeur et hauteur de chaque élément de la liste correspondant au menu principal est de 150 et 18 pixels respectivement.
En ce qui concerne le sous-menu :
lorsqu'il devient visible, sa position devient relative (code CSS : "position : relative;"), avec un décalage à gauche de 150 pixels et un décalage vers le haut de 42 pixels;
expliquez pourquoi avec un des décalages suggérés ci-dessus, *le sous-menu* vient se placer exactement à droite et s'aligne avec le menu principal. Quels seraient les décalages à appliquer pour obtenir la même chose mais cette fois si uniquement la 3ème entrée contenait *le sous-menu* ?
les balises « ancre » du sous-menu possèdent une largeur de 100%.

Exercice 3 (optionnel - à faire si la séance n'est pas encore finie). Test sur un vrai smartphone

Si vous souhaitez tester l'affichage de votre page web dans votre smartphone, c'est possible. Pour cela, suivez les étapes listées ci-dessous :

1. Uniquement si vous êtes aux Lucioles (sinon, passez directement au point 2 "sur votre machine Linux...") :
    1. Activez le partage de réseau sur votre téléphone portable
    2. Connectez votre ordinateur à votre téléphone portable
    - vous pouvez désactiver les données mobiles sur votre téléphone pour éviter que l'ordinateur ne consomme les données du réseau mobile
2. sur votre machine Linux (e.g. votre VM), exécutez la commande "$ python2 -m SimpleHTTPServer"
- devez-vous écrire le symbole dollar "$" pour exécuter la commande ?
3. Si vous travaillez sur la machine physique, continuez avec l'étape numéro 5
4. Si vous êtes sur une machine virtuelle, activez la redirection de port
- Voyez ici comment activer cette redirection
- mais laissez vides les champs "Host IP" et "Guest IP"
- sur "Host Port" et "Guest Port" rentrez "8000" (sans les guillemets)
5. retrouvez l'adresse IP de votre machine physique
- sur Linux, click droit sur le symbole "réseau" de la barre d'état, puis click sur "Connection Information"
- sur Windows, lisez ici comment retrouver votre adresse IP
6. Ouvrez le navigateur de votre smartphone et tapez l'adresse "http://192.168.1.100:8000/", sans les guillemets, supposant que l'adresse IP trouvée dans le point 5 soit par exemple l'adresse "192.168.1.100"
- sur Windows, il est probable qu'un firewall soit configuré et vous empêche d'accéder à la machine virtuelle. Vous pouvez donc désactiver temporairement le firewall en suivant la procédure décrite ici.

Votre progression

Ressources ☐

Rendus TD3 ☐

◄ Applications en HTML/CSS

Le Réseau et le Protocole HTTP/1.1 ►