Initiation RWD : Aide, outils, ressources

Concepts généraux sur le développement de sites responsives

Qu'est ce que le **responsive web design** ? C'est une technique qui permet de concevoir des sites qui s'adaptent automatiquement à l'espace disponible sur l'écran. Ainsi, d'un support à l'autre, et simplement via des règles CSS, certains éléments du site peuvent être masqués, ajoutés et changés de place pour optimiser l'expérience utilisateur. Voici un exemple qui met en évidence la différence entre : statique, liquide, adaptatif et responsive.

Les approches de développement

Dégradation gracieuse

Cette technique est relativement répandue, surtout lors de l'adaptation en responsive d'un site déjà existant. Elle consiste à reprendre le site en l'état actuel et jouer sur la taille des différents blocs pour les rendre adaptatifs tout en modifiant l'affichage et le positionnement de certains blocs suivant la taille de l'écran.

Cette approche favorise l'expérience utilisateur sur ordinateur de bureau tout proposant une version dégradée sur mobile.

Amélioration progressive / mobile first

A l'inverse de la première technique, l'amélioration progressive met le mobile au premier plan : l'application est pensée en tout premier lieu pour les mobiles. Cette approche nécessite une réflexion sur les fonctionnalités proposées par l'application et sur la manière de les rendre facilement utilisables sur mobile. Par la suite, il est possible d'enrichir l'application en proposant d'avantages de fonctionnalités si les capacités du navigateur ou du terminal utilisé le permet.

Cette technique est plus lourde à mettre en place lors de développement sur des produits existants mais devrait être privilégiée dans un projet ayant une portée mobile.

Comment cela se passe-t-il en HTML/CSS

Le Viewport

C'est une composante très importante à prendre en compte lors de la réalisation d'un site web responsive. C'est une balise HTML qui sert à donner les instructions du comportement du viewport du client.

Qu'est ce que le viewport ? On pourrait résumer cela à la *fenêtre d'affichage/une surface d'affichage* du site. L'article "Comprendre le viewport dans le web mobile" vous aidera à comprendre les concepts liés à cette balise HTML.

Il arrive régulièrement que cela soit source de problème lors de la réalisation de sites responsives et le site mydevice.io permet d'avoir un récapitulatif complet des caractéristiques du navigateur dans lequel est affiché cette page.

Voici un autre article de la documentation Apple résumant le fonctionnement du viewport sur ses terminaux.

Les unités relatives

Avant toute chose, nous vous conseillons la lecture d'un petit article nommé "the Lenghts of CSS" qui vous rappellera les possibilités offertes par CSS afin de spécifier une taille.

Auparavant, lors du développement d'un site de 1200 px de large, il arrivait fréquemment de rencontrer des notations du genre :

Imaginons un site avec une structure comme ci-dessous

Structure HTML

```
<div id="header" class="wrapper"></div>

 <div class="wrapper">

    <div class="content"></div>
    <div class="menu"></div>
</div>

<div class="footer" class="wrapper"></div>
```

CSS

```
.wrapper {
   width: 1200px;
}
.content {
   width: 900px;
}
.menu {
   width: 300px;
}
```

Dans le cas d'un développement de site responsive, il n'est pas possible d'utiliser des unités figées tel que le pixel : il faut donc se tourner vers d'autres unités dites relatives, comme `%`, `vh`, `em`, ...

CSS (version responsive)

```
.wrapper {
    max-width: 1200px; /* le site peut prendre n'importe quelle taille entre 0 et 1200px */
}

.content {
    width : 75%; /* 900/1200 = 75% */
}

.menu {
    width: 25%; /* 300/1200 = 25% */
}
```

Grâce aux quelques modifications apportées dans le code CSS ci-dessus, nous avons maintenant un site dit "fluide" quand on le redimensionne en largeur.

Les media-queries

Il s'agit d'une spécification CSS3 éditée par le W3C qui permet d'exécuter des règles CSS conditionnelles en fonction de la largeur de la fenêtre du terminal client, de son orientation et/ou de ses capacités.

La syntaxe est très simple :

Exemple de media-query

```
@media screen and (max-width : 1000px) {
    p {
        color: red;
    }
}

p {
    color: blue;
}
```

Dans ce cas, tous les paragraphes seront bleus, sauf pour les écrans qui ont une largeur inférieure à 1000 pixels.

La communauté Alsacréations a publié un article très complet sur le sujet.

Reprenons notre exemple précédent. Si, lorsque la page fait moins de 600px, le menu de droite n'est pas assez large, nous allons lui appliquer une règle CSS conditionnelle afin de lui faire prendre toute la largeur de la page.

```
@media screen and (max-width: 600px) {
    .menu,
    .content {
        width: auto;
    }
}
```

Dans ce cas, la valeur **`auto`** permet de dire que la `div` doit reprendre son comportement initial par rapport à la propriété **`width`**, et comme cette `div` un élément HTML de type "bloc", elle prendra alors 100% de la largeur de la page. Pour plus d'informations sur le "pourquoi/comment" cet élément HTML prend toute la largeur de la page, consultez cet article portant sur le contexte de formatage block.

Performances

L'un des enjeux du responsive web design est sans nul doute la vitesse d'affichage du site et la fluidité lors de son utilisation. Le poids moyen d'une page web est d'environ 1,2Mo. Les contraintes qu'apportent les mobiles nous obligent à repenser la façon de développer et forcent les développeurs à faire attention aux différentes librairies utilisées.

Pour cela, il devient nécessaire d'analyser la pertinence des scripts utilisés... Peut-on par exemple se passer de jQuery ? Tout dépend l'utilisation que l'on en fait dans le contexte, mais il toujours intéressant de se poser la question.

Depuis quelques temps on découvre de nouveaux outils basés sur NodeJS qui permettent d'apporter un niveau d'automatisation très intéressant pour le développement front-end. Cela permet de gérer beaucoup de tâches fastidieuses et/ou répétitives de façon automatique et de s'assurer par la même occasion d'avoir un haut niveau de qualité. On pense notamment à GruntJS ou Gulp, qui permettent par exemple de concaténer tous les fichiers CSS insérés dans une page et "minifier" cette nouvelle feuille de style, etc...

Des outils d'analyses de performances de site existent et permettent d'identifier les axes d'amélioration : Chrome developper tools , Google page insight, WebPageTest, ...

Prise en charge des écrans haute-densité

Depuis juin 2010, avec l'arrivée de l'iPhone 4 d'Apple, on retrouve de plus en plus de terminaux à écran haute-densité sur le marché.

Cette haute densité ne permet pas à l'oeil humain de distinguer les pixels les uns des autres. Or il a été constaté que dans ces cas là, les images ont tendance à être *floues* sur le terminal. Pour pallier à ce soucis, il existe plusieurs techniques permettant de prendre en compte ces nouveaux écrans. Cela passe par un travail au niveau des visuels du site.

Un article assez exhaustif présente 5 techniques pour adapter ses images aux écrans haute densité.

Une autre approche relativement intéressante pour régler ces problèmes est la prise en charge côté serveur de la partie gestion des ressources à transmettre au client. Cette technique est appelée Responsive Server-Side ou plus communément RESS : voir ce slideshare qui présente ce qu'il est possible de faire.

Les solutions techniques

Twitter Bootstrap

Twitter Bootstrap est un framework CSS mobile-first créé par deux développeurs de Twitter : c'est une librairie comprenant une grille et un ensemble de composants CSS réutilisables. L'avantage de l'utilisation de ce système est qu'il est testé sur un grand nombre de navigateurs, évitant ainsi les problèmes de compatibilité.

L'inconvénient de Bootstrap est son poids car il nécessite de charger la librairie jQuery. Il n'est donc pas forcément pertinent dans des projets de petite envergure.

Jasig ayant choisi Bootstrap pour développer le thème Respondr de uPortal 4.1, il peut être pertinent de l'utiliser au sein de portlets du portail. Toutefois, l'utilisation de ce framework au niveau des portlets connaît quelques limitations qui ont été comblées -en partie- par la création d'une version customizée de Bootstrap. Pour plus d'information, consultez la partie Adaptation de Twitter Bootstrap pour fonctionner dans un contexte portlet

Fonctionnement de la grille

Bootstrap définit quatre catégories d'écrans à savoir :

1. Extra-Small (=> `xs`) : Largeur inférieure à 768px
2. Small (=> `sm`) : Largeur comprise entre 768px et 1200px
3. Medium (=> `md`) : Largeur comprise entre 990px et 1200px
4. Large (=> `lg`) : Largeur supérieure à 1200px

Dans chacune de ces catégories, les composants et la grille fournis par Bootstrap auront un comportement différent.

Voir l'explication du fonctionnement de la grille

Considérons un site découpé en 12 colonnes sur sa largeur. Si l'on veut afficher 4 blocs côte-à-côte, on devra leur donner une largeur de 3 colonnes chacun.

Exemple Bootstrap #1 (colonnes)

```
<div class="row">
    <div class="col-xs-12 col-md-3"></div>
    <div class="col-xs-12 col-md-3"></div>
    <div class="col-xs-12 col-md-3"></div>
    <div class="col-xs-12 col-md-3"></div>
</div>
```

Dans l'exemple ci-dessus, lorsque la fenêtre du navigateur sera inférieure à 768px (`xs`), chaque `div` prendra toute la largeur de la page (12/12 => 100%). Et lorsqu'elle sera comprise entre 990px et 1200px, chaque `div` occupera 3/12 soit 25% de la largeur de la page.

Styles de base

Il existe beaucoup de classes CSS qui aident au développement de l'interface : on retrouve toutes ces `class` dans l'onglet CSS de la documentation. La réutilisation de ces classes permet de rapidement mettre en forme des *boutons, tableaux, formulaires*.

Exemple Bootstrap #2 (classes CSS)

```
<div class="table-responsive">
    <table class="table">
        ...
    </table>
</div>
```

```
<button type="button" class="btn btn-warning">Warning</button>
```

Les composants

Une des grandes forces de Twitter Bootstrap réside dans la richesse des composants qu'il propose. Un composant peut être considéré comme un *module* dans une page, comme par exemple un menu déroulant ou un carrousel.

Dans ce cas, Bootstrap propose un ensemble de composants prêts à l'emploi en se basant sur ses styles de base. L'avantage de les utiliser réside dans leur nature à être -de base- responsives.

Adaptation de Twitter Bootstrap pour fonctionner dans un contexte portlet

Dans le cadre de nos tests de développement avec Bootstrap, nous avons identifié certaines limitations du framework lors d'un usage au sein d'une portlet.

Effectivement, les media-queries détectent la largeur de la fenêtre dans son intégralité, tandis qu'une portlet ne représente qu'un fragment de la page. Dans ce cadre là, nous avons mis en place une version modifiée du framework pour s'adapter à ces contraintes. Les sources sont disponibles sur Github.

Cette solution se veut iso-fonctionnelle par rapport à la version non modifiée. Cependant, il reste aujourd'hui quelques conflits de classes lors d'une utilisation "simultanée" de la version modifiée (importée par les portlets) avec celle d'origine (utilisée par le thème Respondr d'uPortal 4.1).

Des échanges sur cette problématique sont en cours avec la communauté Jasig => Historique de la mailing list

Nécessité d'implémentation de modification au sein du framework

Comme évoqué précédemment les media-queries ne se basent que sur la largeur de la fenêtre, or les portlets peuvent prendre différentes tailles au sein d'une page (25%, 33%, 40%, 50%, 60% ou même 100% de la largeur). Il était donc nécessaire de trouver une solution "similaire" (règles CSS "conditionnelles") mais ne se basant pas sur les media-queries (inexploitables dans ce contexte).

Pour ce faire, nous sommes donc partis du constat suivant :

- Objectif : permettre d'appliquer différentes règles CSS en fonction de la largeur d'une portlet
- Contraintes :
    - un page peut contenir plusieurs portlets
    - une même portlet peut se trouver à plusieurs endroits différents sur la même page...
    - .... et ne pas avoir la même largeur à chaque fois
    - la largeur d'une portlet peut varier lors d'un drag-n-drop (déplacement de portlet d'une colonne A à une colonne B), d'un changement de mode (normal => maximisé, maximisé => normal), d'un redimensionnement de la fenêtre du navigateur

=> Il nous fallait donc trouver une solution permettant de recalculer dynamiquement la largeur de chaque portlet présente sur la page.

Notre solution se base sur :

- un `div` (dit "conteneur") englobant l'intégralité du contenu de la portlet
- un script JS permettant de :
    - calculer à la volée les largeurs de chacune des portlets...
    - ... et, en fonction de la valeur obtenue pour chacune d'elles, attribuer la classe CSS correspondant au mode d'affichage (`xs` , `sm`, `md`, `lg`) à appliquer.

Fonctionnement de la version modifiée

Il faut insérer un petit snippet JavaScript dans la page

Snippet JS

```
(function($) {
        var $portletContainers;
        $(document).ready(function() {
            // all portlets must have the CSS class .pc
            $portletContainers = $(".pc");
            // Resize event isn't fired on DOM Content Loaded, we launch the function manually
                   onWindowResize();
                 $(window).resize(onWindowResize);
        });
        function onWindowResize() {
            $portletContainers.each(function(index) {
        var $that = $(this);
        var portletWidth = $that.width()
        $that.removeClass("xs sm md lg");
        if(portletWidth < 768)
            $that.addClass("xs");
        if(portletWidth >= 768 && portletWidth < 992)
            $that.addClass("sm");
        if(portletWidth >= 992 && portletWidth < 1200)
            $that.addClass("md");
        if(portletWidth >= 1200)
            $that.addClass("lg");
    });
}
})(jQuery);
```

A chaque redimensionnement de la fenêtre, ce script va parcourir tous les éléments de la page pour recalculer la taille des conteneurs de porlet et leur appliquer la classe CSS en fonction de leur nouvelle largeur.

La limitation actuelle de ce système est qu'il ne gère pas le déplacement de portlet en drag-n-drop => Cela fait partie des évolutions futures du développement.

Exemples de projets tirant parti du framework

1. Portlet Jasig modifiée : email-preview
2. Portlet Jasig modifiée : news-reader-portlet
3. Esup-Dematec (Bootstrap en version 2.3)

Intégration d'une portlet en pure CSS

Dans quels cas envisager cette solution ?

L'utilisation de framework CSS n'est pas toujours utile. Dans certains cas, lorsque l'existant est trop lourd ou que l'interface se révèle très sommaire notamment, il peut s'avérer fastidieux et/ou inutile de se baser sur des librairies aussi conséquentes.

>> Dans le cadre de l'utilisation d'un framework CSS dans un projet existant d'ampleur non négligeable, le développeur sera amené à revoir :

- la structure HTML générale de son site pour se conformer à la philosophie du framework => Il lui faudra reprendre chaque bloc, un par un, pour y appliquer les classes CSS nécessaires à l'affichage des styles correspondants.
- la feuille de style CSS afin d'en retirer les éléments entrants en conflit avec les classes fournies par le framework => Il faudra en effet vérifier que les règles CSS de la portlet ne surchargent pas certaines règles du framework au risque sinon de perdre l'aspect responsive de certains éléments
- le choix des librairies JS (ou de leurs versions) utilisées => il faut s'assurer que les plugins JS (ou jQuery) utilisés dans la portlet soient eux aussi responsives... ou, dans le cas contraire, trouver une librairie iso-fonctionnelle pour remplacer l'actuel (avec tous les problèmes que cela peut occasionner...)

=> On comprend donc que cela peut vite devenir lourd et compliqué à mettre en place pour obtenir le résultat désiré. Des adaptations de la feuille de style CSS pourraient alors être suffisantes...

Dans le cadre d'un développement proposant une vue minimaliste, le développeur devra évaluer -en fonction du résultat attendu- l'intérêt de charger (ou non) toute une librairie... Peut-être que quelques lignes de CSS et JS peuvent suffire pour répondre à ses besoins ! => Dans ces cas là, il paraît plus pertinent de partir sur une solution "homemade"...

Les classes CSS conditionnelles

Comme expliqué précédemment, il est malheureusement impossible avec des media-queries de cibler la largeur d'une portlet => nous sommes obligés de nous baser sur la largeur de la fenêtre (pour mieux comprendre cette problématique, un article explique bien les problèmes d'implémentation de cette fonctionnalité nommée Element Query).

Il faudra donc s'abstraire des media-queries et passer par une technique alternative. Nous proposons donc de détecter en JavaScript le redimensionnement de la fenêtre et d'appliquer une classe CSS à un élément HTML en fonction de sa taille (calculée à la volée).

Code HTML (Container)

```
<div class="container">
        <!-- prendra la classe "small" si la taille de ce bloc est inférieur à 600px -->
        <!-- prendra la classe "medium" si la taille de ce bloc est supérieur à 600px et inférieur à 1100px -->
        <!-- prendra la classe "large" si la taille de ce bloc est supérieur à 1100px -->
</div>
```

Avec une CSS classique tirant parti des media-queries nous aurions écrit ceci :

CSS (avec media-queries)

```
@media screen and (max-width: 600px) {
  /* règles pour les écrans "small" */
}

@media screen and (min-width: 601px) and (max-width: 1100px) {
  /* règles pour les écrans "medium" */
}

[...]
```

Dans la mesure où nous ne pouvons pas nous appuyer sur les media-queries, il nous faut écrire quelque chose comme cela :

CSS (sans media-queries)

```
.container.small .mon-selecteur-css {
   /* mes propriétés ... */
}

.container.medium .mon-selecteur-css {
         /* ... */
}

.container.large .mon-selecteur-css {
   /* ... */
}
```

Snippet Javascript

Pour mettre en oeuvre ce qui est présenté dans la partie précédente, nous n'avons d'autres choix que de nous reposer sur JavaScript. Il existe actuellement deux solutions :

Selector Queries

C'est une toute petite librairie JavaScript qui permettra d'écrire les "simili media-queries" dans le code HTML afin d'appliquer une classe CSS si les conditions sont validées. Voir les sources sur Github

Selector Queries (exemple)

```
<div data-squery="min-width:1100px=large max-width:600px=small">
    <!-- ... -->
</div>
```

Cette solution s'appuie sur les attributs `data-*` (compatibles à partir de IE8+) : ces attributs `data-*` ne sont pas prévus pour cela à la base, mais aucune solution CSS n'existe actuellement pour le réaliser. L'avantage de cette librairie est qu'elle est relativement légère et qu'elle permet de mettre en place rapidement des points de bascules personnalisés.

Solution "homemade"

Il est également possible de reprendre le snippet JavaScript proposé pour la version Bootstrap, en l'adaptant à vos besoins :

```
(function($) {
        var $portletContainers;
        $(document).ready(function() {
            // all portlets must have the CSS class .container
            $portletContainers = $(".container");
            // Resize event isn't fired on DOM Content Loaded, we launch the function manually
                   onWindowResize();
                  $(window).resize(onWindowResize);
        });
        function onWindowResize() {
            $portletContainers.each(function(index) {
                var $that = $(this);
                var portletWidth = $that.width()
                $that.removeClass("small medium large");
                if(portletWidth < 600)
                    $that.addClass("small");
                if(portletWidth >= 600 && portletWidth < 1100)
                    $that.addClass("medium");
                if(portletWidth >= 1100)
                    $that.addClass("large");
            });
        }
})(jQuery);
```

Utilisation d'une grille

L'élément le plus important dans la réalisation d'une portlet ou d'un site responsive est la grille qu'il utilise. Les frameworks sont toujours fournis avec une grille, on le retrouve chez Bootstrap, Foundation, Unsemantic, Knacss, ...

Une grille fonctionne toujours de la même manière, c'est tout d'abord un nombre de colonnes défini avec un espacement entre chaque (que l'on appellera "gouttière") qui va former une ligne. Le système de grille le plus répandu se base sur 12 colonnes mais il est autant possible d'en avoir 16 ou 8.

Il est hautement recommandé de se baser sur une grille existante, comme celle de Bootstrap ou de Foundation par exemple. Ces deux frameworks offrent la possibilité de customizer ces librairies en sélectionnant les parties/modules du framework que l'on souhaite utiliser => Il est alors possible de ne "charger" que le système de grille...

Intégration de portlet au sein d'une iframe

Comme évoqué précédemment, le concept d'Element Query (appliquer des CSS conditionnelles par rapport à un élément de la page et non par rapport à la fenêtre globale) n'existe pas en CSS. Cependant l'élément HTML iframe applique les media-queries de la page incluse par rapport à la taille de l'iframe.

Cela permet donc d'utiliser les frameworks CSS sans les modifications présentées dans la partie précédente. Toutefois, les i*frames* apportent plusieurs problématiques : leur largeur et leur hauteur sont fixes par exemple.

Pour contourner le problème de hauteur fixe, il existe une solution reposant sur "postMessage" en HTML5 afin de communiquer entre la page HTML et la page incluse dans l'iframe. Le fonctionnement basique de cette solution est simple :

- la page incluse va transmettre à la page parente la hauteur de son contenu...
- ... ce qui va permettre de positionner la bonne valeur à l'attribut `height` de l'élément `iframe`.

Il est possible de retrouver une présentation de ce travail réalisé par Pascal R. sur Github. Une documentation est aussi disponible sur Esup-Portail.

A noter que d''après le site CanIUse, la fonction postMessage est compatible avec Internet Explorer 8 et supérieure. Cette fonctionnalité est donc prise en charge par plus de 90% des navigateurs du marché.

Autres frameworks CSS

1. Foundation : concurrent direct de Twitter Bootstrap, on retrouve beaucoup de similitude avec ce dernier
2. Knacss : micro framework, très léger et ne contient aucun style graphique, juste une grille et des fonctionnalités de placement d'éléments, idéal pour des demandes graphiques très spécifiques
3. ...

Ressources utiles

Ci-dessous un agrégat d'articles, sites et outils utiles qui n'ont pas trouvés leur place au sein de cette documentation

Sites

- Can I use : liste de la prise en charge des différentes propriétés CSS par les différentes versions des navigateurs
- Browserstack : Service permettant de lancer des VM en spécifiant l'OS, le navigateur et la version du navigateur (y compris pour des mobiles)
- Discover DevTools : Cours sur les outils de développeurs Google Chrome

Articles

- Writing Efficient CSS : Article du Mozilla Developer Network
- Getting to 60fps using Chrome devtools : utilisation des outils de développeurs chrome pour améliorer la fluidité général d'un site
- Code Guide : Guide de bonne pratique de développement front par @mdo un des développeurs de Twitter Bootstrap

Outils

- Modernizr : Librairie JavaScript de détection des capacités du navigateur
- HTML5Boilerplate : Bâse de projet regroupant toutes les bonnes pratiques du développement front-end