Web mobile : introduction et glossaire

Le Web mobile a depuis longtemps envahi notre quotidien de concepteur de sites web : aujourd'hui, smartphones et tablettes font partie intégrante du parc de périphériques sur lesquels nous jouissons de notre dose quotidienne d'Internet, sur lesquels nous consultons nos sites web préférés, et pestons - à juste titre - lorsque celui-ci ne s'affiche pas correctement.

Afin de mieux dégrossir ce vaste sujet, voici une introduction à la conception web mobile sous forme d'un glossaire des principaux termes et d'une double méthodologie pratique.

Glossaire

Application native
: Correspond aux logiciels à télécharger que l'on peut trouver sur les boutiques "stores" (AppStore pour Apple, GooglePlay pour Android,…) développés dans un langage spécifique à chaque plateforme : Objective-C pour Apple, Java pour Android, …

Application web (Web App)
: Une application web est le nom que l'on donne à un site web "classique" que l'on a adapté pour les mobiles.
La structure existante reste en place (HTML, scripts, bases de données, médias) et est complétée par une couche graphique dynamique (en CSS) adaptée aux écrans de taille réduite dans un principe général de design adaptatif (*Responsive web design* et avec des outils CSS3 tels que les Media Queries » http://www.alsacreations.com/article/lire/930-css3-media-queries.html . Des retouches ou sur-couches JavaScript sont parfois également à l'ordre du jour.

Site web dédié
: Il s'agit d'une nouvelle version du site web complètement adaptée aux terminaux nomades en terme de technologies (HTML, vidéo, audio, images) et de performances (optimisations, préchargements, mises en cache).
Habituellement, une détection du périphérique utilisateur (via User Agent, JavaScript, autre) permet de le rediriger vers l'URL de la version mobile dédiée du site (m.monsite.com, monsite.mobi).

Navigateurs mobiles
: Le marché des navigateurs pour mobiles se distingue de celui des ordinateurs de bureau par sa diversité et sa vivacité concurrentielle et technologique. Parmi les plus employés dans le monde : Opera Mobile, Opera Mini, Safari Mobile (sur iPhone, iPad), navigateurs Android, Internet Explorer mobile, Blackberry, Firefox mobile, Chrome mobile, UCweb, etc.

Densité de pixels
: Correspond à la résolution d'un écran numérique exprimée en DPI (*Dots Per Inch* : points par pouce) ou PPI (pixels par pouce). Une densité élevée correspond à un écran dit "HD" ou "Haute Définition". Quelques exemples de densités : 326 PPI pour iPhone 4 et 4S, 264 PPI pour iPad 3, 220 PPI pour MacBook Pro 3.

Pixel ratio
: Correspond au ratio entre les pixels réels (`screen width`) et la valeur de `device-width` du navigateur. Par exemple, un iPhone 4 a un ratio de 2 (640/320) tandis qu'un iPhone 3 a un ratio de 1 (320/320). Il est possible de détecter le pixel-ratio via Media Queries ou en JavaScript via `window.devicePixelRatio`.

Retina
: "Retina display" est une marque déposée par Apple pour désigner des écrans dont la densité de pixels est telle que l'oeil humain n'est plus censé pouvoir distinguer de défauts dans les détails. Il ne s'agit ni plus ni moins d'écrans de haute définition

(densité de pixels élevés). Cette technologie équipe des terminaux iPod, iPhone, iPad et MacBook Pro. Du côté de la concurrence, par exemple Samsung, on retrouve le terme (Super-)AMOLED.

Orientation
: L'orientation correspond, pour un écran numérique, à son sens d'affichage : si le périphérique est tenu dans le sens de la largeur, on parle d'orientation paysage (landscape en anglais); s'il est tenu dans le sens de la hauteur, on parle d'orientation portrait.
Les récents navigateurs de bureau reconnaissent également l'orientation : si la fenêtre est plus large que haute, l'orientation sera définie en paysage par exemple. Il est possible de détecter l'orientation via CSS Media Queries.

screen width (ou screen height)
: Correspond à la largeur (ou hauteur) d'un écran numérique. Que cela soit sur un écran de bureau, une tablette ou un téléphone mobile, cela se traduit par la (mal nommée) résolution : par exemple 1024x768, 320x640, etc.

device-width (ou device-height)
: Correspond aussi à la largeur (ou hauteur) d'un écran numérique. Ou plutôt celle qui est déclarée par le périphérique.
Un certain nombre de terminaux mentent sur la valeur de `device-width` et `device-height`. En effet, ceux-ci ne correspondent *pas* à la largeur réelle de l'écran (`screen-width`) mais à une valeur exprimée en "Device Independant Pixels (dip)" ou "Pixels Indépendants du Terminal". Ainsi, un iPhone 4 dont la largeur physique réelle est de 640px retournera une largeur `device-width` de 320px, ceci pour des raisons de compatibilité ascendante avec son aîné iPhone 3.

viewport
: Désigne la zone de la fenêtre du navigateur.
Les navigateurs mobiles trichent en reconnaissant par défaut une surface de viewport plus élevée que la surface physique réelle en pixels.
Par exemple la largeur de viewport par défaut sur Safari mobile est de 980px, elle est de 850px sur Opera, 800px sur un navigateur Android, 1024px sur IE mobile.
Pour s'affranchir des tailles de sites minuscules (puisque le mobile affiche une largeur de 980px dans une surface de 320px), il est possible de modifier cette valeur par défaut et d'imposer ses valeurs à l'aide de l'élément HTML - propriétaire - `<meta name="viewport"…>`, ou selon les spécifications CSS, à l'aide d'une règle-at `@viewport`.

Media queries
: Module des spécifications CSS3 permettant la sélection fine de périphériques selon divers critères (largeur, hauteur, résolution, orientation,…), puis d'appliquer spécifiquement un ensemble de styles CSS dédiés à cette sélection.
Cet outil est devenu une étape incontournable de l'adaptation de designs aux écrans de différentes tailles. Pour tous les détails, voir l'article Media Queries » http://www.alsacreations.com/article/lire/930-css3-media-queries.html .

Responsive web design
: Ensemble de techniques et principes permettant d'adapter un design web à différentes tailles d'écran : de l'écran de bureau géant au smartphone, en passant par les tablettes, les netbooks et autres formats de TV. Ceci afin d'offrir aux visiteurs la meilleure expérience possible quel que soit le support de consultation et quitte à réorganiser l'affichage et la disposition des éléments (par exemple passer d'un site à 4 colonnes sur grand écran à un site mono-colonne sur mobile). Ce terme a été introduit par Ethan Marcotte dans son livre éponyme » http://www.alsacreations.com/livres/lire/1320-responsive-web-design.html et se fonde sur un trio de techniques : CSS Media Queries, gabarits fluides et contenus fluides.

Point de rupture
: Les points de rupture sont les différents palliers, qui une fois franchis, "cassent" votre design, les largeurs qui nécessiteront des adaptations voire des réorganisations en CSS. Testez votre site web sous différentes résolutions (par exemple avec les excellents outils Screenqueri.es » http://screenqueri.es/ ou Responsive.is » http://responsive.is), vous identifiez rapidement quelques palliers problématiques : de sont vos points de rupture.

Mobile first
: Philosophie de conception à contre-courant de nos principes habituels consistant à commencer prioritairement par la version épurée (mobile) puis à enrichir progressivement pour les écrans de plus en plus larges.

User agent
: L'agent utilisateur est une chaîne de texte envoyée par le navigateur au serveur pour s'identifier. Elle contient des informations comme par exemple : le nom de l'application, la version, le système d'exploitation, la langue, etc.
Il est possible, via un langage serveur, de récupérer et exploiter ces informations, par exemple en redirigeant le visiteur sur une version mobile dédiée en reconnaissant un agent utilisateur mobile. La technique n'est malheureusement pas toujours fiable car il existe des milliers de chaînes d'identification différentes et peu de sites les maintiennent à jour, ou les gèrent avec exhaustivité.

Par quoi commencer ?

Avant de débuter un projet de site web mobile, vous aurez bien entendu un bon million de questions à vous poser, autant techniques qu'ergonomiques, sans oublier les aspects de performances navigateurs, de compatibilités, d'accessibilité à tous, etc.

En outre, selon l'existant du projet (la version "bureau" existe-t-elle déjà, ou tout est-il à construire à partir de zéro ?), vos considérations ne seront pas les mêmes. De même que la procédure suivie. Il ne s'agira pas ici d'entrer dans tous les détails, mais d'aborder les principales étapes à parcourir.

Nous avons distingué deux méthodologies courantes, adaptées à deux cas de figure : l'adaptation d'un design existant (idéale pour "aller vite" mais pas forcément très optimisée), et la démarche "mobile first" (parfaite pour débuter de zéro, mais nécessitant une réflexion initiale plus poussée).

Méthodologie 1 : adapter un design existant

Votre site web "classique" est déjà en ligne mais inadapté aux smartphones et tablettes ?

En suivant une procédure simple en 10 points, il est relativement aisé et rapide d'adapter un design existant à différentes tailles d'écran. En contrepartie, sachez que gérer les écueils d'ergonomie et de performance web ne sera pas toujours une partie de plaisir.

1. Fixez le viewport du terminal » /article/lire/1490-comprendre-le-viewport-dans-le-web-mobile.html (`<meta name="viewport"…>`)
2. Identifiez les points de rupture
3. Faites un choix de navigation mobile (en haut, en bas, select, déroulant, masqué, etc.)
4. Appliquez des styles mobiles via media queries, soit intégrés dans les styles globaux soit dans une feuille CSS séparée
5. Débutez vos styles mobiles par une feuille de styles mobile de base » http://www.alsacreations.com/astuce/lire/1177-une-feuille-de-styles-de-base-pour-le-web-mobile.html ("reset")
6. Traitez un par un tous les éléments possédant des largeurs problématiques (`width`, `min-width`, `height`, `min-height`, `margin`, `padding`) en annulant ces largeurs (`width: auto`, `min-width: 0`, par exemple)
7. Réintégrez les éléments dans le flux pour obtenir des blocs verticaux (écrasez les valeurs de `float` et `position: absolute` par `float: none` et `position: static`)
8. Adaptez les contenus et les médias pour éviter des débordements » http://www.alsacreations.com/tuto/lire/1038-gerer-debordement-contenu-et-cesures-css.html
9. Optez pour des gabarits de largeur fluide
0. Et pour finir en beauté, prenez en compte les écrans HD (rétina) » http://www.inpixelitrust.fr/blog/remplacement-dimage-css-adapte-au-retina-display-des-appareils-mobiles/ et les performances web mobile » http://wdfriday.com/blog/2012/04/introduction-a-la-performance-pour-le-web-mobile/ .

Méthodologie 2 : la démarche "mobile-first"

Cette fois, changeons radicalement de stratégie : en pensant "à l'envers", nous évitons tous problèmes de ressources inadaptées, de temps de chargements inutiles, d'éléments superflus et non ergonomiques.

Etape longue et nécessaire : pensez dès le début du projet à l'ergonomie de votre site web dans différents types de terminaux (généralement smartphones, tablettes, écrans de portables, écrans de bureau). En conséquence, prévoyez un temps de conception graphique multiplié par 3 ou 4, mais c'est pour la bonne cause.

Les étapes sont parfois sensiblement identiques à la méthodologie précédente (à savoir : fixer le viewport, emploi des media queries, prise en compte des terminaux HD et de la performance web), avec quelques particularités à prendre en compte :

- Le wireframing » http://www.alsacreations.com/article/lire/1183-mockup-rough-maquette-zoning.html doit être initialement orienté pour les petits écrans (des outils tels que Codiqa » http://www.codiqa.com/ sont dédiés aux mobiles)
- Le design et l'ergonomie principalement adaptés aux petits écrans
- Les styles CSS de base sont dédiés aux smartphones (puis améliorés via respond.js » https://github.com/scottjehl/Respond/ ou des classes conditionnelles pour assurer la compatibilité aux anciens navigateurs de bureau)
- Les ressources et medias sont très optimisés (puis remplacés si écrans plus larges)
- idem pour JS et scripts de "confort"

Pour en savoir plus sur cette méthode, je vous invite à consulter le très bon article publié sur WebdesignFriday : "CSS et Mobile First : procéder par amélioration progressive" » http://wdfriday.com/blog/2012/03/css-et-mobile-first-proceder-par-amelioration-progressive/ .

Ressources

- http://mobilewebbestpractices.com/ » http://mobilewebbestpractices.com/
- http://validator.w3.org/mobile/ » http://validator.w3.org/mobile/
- http://html5boilerplate.com/mobile » http://html5boilerplate.com/mobile
- http://mobitest.me/ » http://mobitest.me/
- http://lab.goetter.fr/tagged/mobile » http://lab.goetter.fr/tagged/mobile