epsi

SN2 - DEVE304 - Atelier - Responsive Web Design & développement plateforme mobile

1. Rappels théoriques

1.1 Responsive Web Design et conception d'une IHM

1.1.1 Définition

Le terme de responsive design a été inventé par Ethan Marcotte en 2010, et décrit la combinaison de trois techniques.

- La première était l'idée des grilles fluides.
    - La deuxième technique était l'idée d'images fluides. En utilisant une technique très simple de réglage de la propriété max-width à 100%, les images deviennent plus petites si leur colonne de contenu devient plus étroite que la taille intrinsèque de l'image, mais ne deviennent jamais plus grandes. Cela permet à une image de se réduire pour s'intégrer dans une colonne de taille flexible, plutôt que de la déborder, mais de ne pas s'agrandir et de devenir pixélisée si la colonne devient plus large que l'image.
    - Le troisième élément clé était la media query. Les Media Queries permettent de changer le type de mise en page en utilisant uniquement CSS. Au lieu d'avoir une seule mise en page pour toutes les tailles d'écran, la mise en page pouvait être modifiée. Les barres latérales pouvaient être repositionnées pour l'écran plus petit, ou une autre navigation pouvait être affichée.

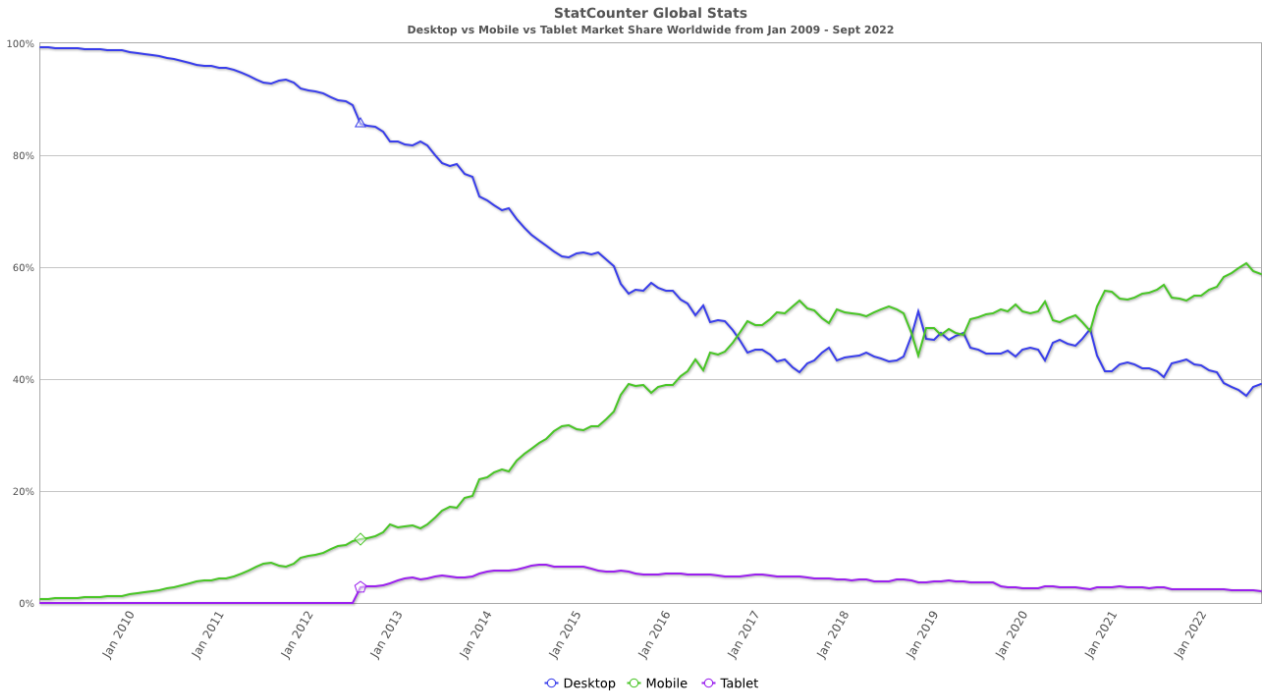
Il est important de comprendre que le responsive web design n'est pas une technologie à part. C'est un terme utilisé pour décrire une approche de la conception web, ou un ensemble de bonnes pratiques, utilisées pour créer une mise en page qui peut correspondre à l'appareil utilisé pour visualiser le contenu. Dans la recherche initiale de Marcotte, cela impliquait des grilles flexibles (en utilisant des flotteurs) et des média queries, mais depuis la rédaction de cet article, il y a presque 10 ans, le concept de responsive design est devenu la norme. Les méthodes modernes de mise en page CSS sont par nature responsives, et nous avons intégré de nouvelles choses à la plateforme Web pour faciliter la conception de sites responsives.

1.1.2 Les différents types de terminaux

Il existe 3 principaux types de terminaux à toujours prendre en compte:

- Format Desktop (ou laptop)
- Format Tablet
- Format Mobile

Aujourd'hui, la part d'utilisation du format Mobile a dépassé celui du Desktop (environ 58% vs 39%).



1.1.3 L'approche Mobile First

En plus de cette forte utilisation du format Mobile, celui-ci est plus contraignant à concevoir (limite de la taille de l'écran, beaucoup de breakpoints, bande passante en moyenne moins performante, ..).

L'approche mobile-first est un principe d'amélioration progressive. C'est l'idée que le design mobile, qui est plus dur, devrait être fait en premier. Une fois les questions de design mobile réglées, la conception sur d'autres appareils sera plus facile. Le plus petit des designs aura seulement les caractéristiques essentielles, donc en premier lieu vous devez concevoir le cœur de votre UX.

Cette approche pousse la conception jusqu'au choix du contenu à afficher (Content-First) ou encore la simplification des éléments d'interface.

1.2 Techniques de mise en page

1.2.1 Media Queries

Le responsive design n'a pu voir le jour uniquement grâce aux média queries. La spécification de niveau 3 des média queries est devenue une Recommandation Candidate en 2009, ce qui signifie qu'elle a été jugée prête à être mise en œuvre dans les navigateurs.

Les Media Queries nous permettent d'effectuer une série de tests (par exemple, si l'écran de l'utilisateur dépasse une certaine largeur, ou une certaine résolution) et d'appliquer le CSS de manière sélective pour donner à la page le style approprié aux besoins de l'utilisateur.

Par exemple, la média query suivante teste si la page Web actuelle est affichée comme média d'écran (donc pas un document à imprimer) et si la fenêtre de visualisation a au moins 800 pixels de large. Le CSS du sélecteur .container ne sera appliqué que si ces deux éléments sont vrais.

```
@media screen and (min-width: 800px) {
  .container {
    margin: 1em 2em;
  }
}
```

Vous pouvez ajouter plusieurs medias queries dans une feuille de style, afin de modifier votre mise en page ou certaines de ses parties pour les adapter aux différentes tailles d'écran. Les endroits où une média query est introduite et où la mise en page est modifiée sont appelés "breakpoints" (points d'arrêt).

Une approche courante lors de l'utilisation de Media Queries consiste à créer une disposition à colonne unique simple pour les appareils à écran étroit (par exemple les téléphones portables), puis à vérifier si les écrans sont plus grands et à mettre en œuvre une disposition à colonnes multiples lorsque vous savez que vous avez suffisamment de largeur d'écran pour la prendre en charge. Ceci est souvent décrit dans la conception mobile en priorité (Mobile First).

https://www.w3schools.com/css/css_rwd_mediaqueries.asp

1.2.2 Grilles flexibles

Les sites responsives ne se contentent pas de changer leur mise en page entre les points de rupture, ils sont construits sur des grilles flexibles. Une grille flexible signifie que vous n'avez pas besoin de cibler toutes les tailles d'appareils possibles et de construire une mise en page parfaite au pixel près. Cette approche serait impossible étant donné le grand nombre de dispositifs de tailles différentes qui existent, et le fait que sur les ordinateurs de bureau au moins, les gens n'ont pas toujours la fenêtre de leur navigateur maximisée.

En utilisant une grille flexible, vous n'avez qu'à ajouter un point d'arrêt et à modifier le design au moment où le contenu commence à avoir une piètre apparence. Par exemple, si la longueur des lignes devient illisible à mesure que la taille de l'écran augmente, ou si une boîte se retrouve écrasée avec deux mots sur chaque ligne lorsqu'elle se rétrécit.

Aux débuts du responsive design, notre seule option pour réaliser la mise en page était d'utiliser des "floats" (flotteurs). Des mises en page flottantes flexibles ont été réalisées en donnant à chaque élément un pourcentage de largeur et en s'assurant que les totaux ne dépassent pas 100 % dans toute la mise en page.

Cette approche se retrouve aujourd'hui à de nombreux endroits sur le Web. Il est probable que vous rencontrerez des sites web utilisant cette approche dans votre travail, donc il est utile de la comprendre, même si vous ne construiriez pas un site moderne en utilisant une grille flexible basée sur le flottement.

Exemple:

https://mdn.github.io/css-examples/learn/rwd/float-based-rwd.html

https://github.com/mdn/css-examples/blob/main/learn/rwd/float-based-rwd.html

1.2.3 Images responsives

L'approche la plus simple pour les images responsive est celle décrite dans les premiers articles de Marcotte sur le design responsive. En gros, vous preniez une image qui était à la plus grande taille possible et vous la réduisiez. C'est encore une approche utilisée aujourd'hui, et dans la plupart des feuilles de style, vous trouverez le CSS suivant quelque part :

```
img {
  max-width: 100%:
}
```

Cette approche présente des inconvénients évidents. L'image peut être affichée à une taille beaucoup plus petite que sa taille réelle, ce qui est un gaspillage de bande passante - un utilisateur mobile peut télécharger une image dont la taille est beaucoup plus grande que ce qu'il voit réellement dans la fenêtre du navigateur. De plus, il se peut que vous ne vouliez pas le même rapport hauteur/largeur de l'image sur le mobile que sur le bureau. Par exemple, il peut être agréable d'avoir une image carrée pour le mobile, mais de montrer la même image en format paysage sur le bureau. Ou, en tenant compte de la taille plus petite d'une image sur le mobile, vous pouvez vouloir montrer une image différente, qui est plus facile à comprendre sur un écran de petite taille. Ces choses ne peuvent pas être réalisées par une simple réduction de la taille d'une image.

Les images responsives, en utilisant l'élément `<picture>` et les attributs `<img>` `srcset` et `sizes` permettent de résoudre ces deux problèmes. Vous pouvez fournir plusieurs tailles ainsi que des " indices " (méta-données qui décrivent la taille de l'écran et la résolution pour lesquelles l'image est la mieux adaptée), et le navigateur choisira l'image la plus appropriée pour chaque dispositif, en s'assurant qu'un utilisateur téléchargera une taille d'image appropriée pour le dispositif qu'il utilise.

Vous pouvez également créer des images directes utilisées à différentes tailles, ce qui permet d'obtenir un recadrage différent ou une image complètement différente pour différentes tailles d'écran.

1.3 Méthodes de mise en page modernes

Les méthodes de mise en page modernes telles que Multiple-column layout, Flexbox, et Grid sont responsives par défaut. Elles partent toutes du principe que vous essayez de créer une grille flexible et vous donnent des moyens plus faciles de le faire.

1.3.1 Multiple-column layout

La plus ancienne de ces méthodes de mise en page est multicol. Lorsque vous spécifiez un `column-count`, cela indique en combien de colonnes vous voulez que votre contenu soit divisé. Le navigateur calcule alors la taille de celles-ci, une taille qui changera en fonction de la taille de l'écran.

```
.container {
  column-count: 3;
}
```

Si vous spécifiez plutôt une largeur de colonne, vous spécifiez une largeur minimale. Le navigateur créera autant de colonnes de cette largeur qu'il en faudra pour que le conteneur soit parfaitement adapté, puis répartira l'espace restant entre toutes les colonnes. Par conséquent, le nombre de colonnes changera en fonction de l'espace disponible.

```
.container {
  column-width: 10em;
}
```

1.3.2 Flexbox

Dans Flexbox, les articles flexibles se rétréciront et répartiront l'espace entre les articles en fonction de l'espace dans leur conteneur, en fonction de leur comportement initial. En modifiant les valeurs de `flex-grow` et `flex-shrink` vous pouvez indiquer comment vous souhaitez que les articles se comportent lorsqu'ils rencontrent plus ou moins d'espace autour d'eux.

Dans l'exemple ci-dessous, les éléments flex prendront chacun autant d'espace dans le conteneur flex, en utilisant la notation `flex: 1`.

```
.container {
  display: flex;
}

.item {
  flex: 1;
}
```

Exemple:

https://mdn.github.io/css-examples/learn/rwd/flex-based-rwd.html

https://github.com/mdn/css-examples/blob/main/learn/rwd/flex-based-rwd.html

1.3.3 Grid

Dans la mise en page en grille CSS, l'unité `fr` permet la répartition de l'espace disponible sur les différentes sections de la grille. L'exemple suivant crée un conteneur de grille avec trois rangées dont la taille est de `1fr`. Cela créera trois colonnes, chacune prenant une partie de l'espace disponible dans le conteneur.

```
.container {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
}
```

1.5 Outillage et frameworks utiles

1.5.1 Modernizr

Modernizr est une bibliothèque JavaScript open-source qui permet de détecter si un navigateur supporte une fonctionnalité HTML5 / CSS3 ou pas.

Dans le cas de figure où la fonctionnalité n'est pas supporté, alors Modernizr proposera automatiquement une classe CSS avec le préfix "no-" pour que vous n'ayez plus qu'à la définir dans vos CSS.

https://modernizr.com

1.5.2 css3pie

Css3pie est une librairie à inclure dans vos projets pour s'assurer des compatibiltés avec les anciennes versions d'Internet Explorer.

http://css3pie.com/

1.5.3 Framework Bootstrap

Ce framework a été créé en 2011 en interne chez Twitter pour harmoniser le développement de la mise en page entre les développeurs. Il a alors été déposé sur GitHub en open source pour bénéficier à tous.

Le but de Bootstrap est de permettre, par exemple, de rendre facilement un site responsive design sans avoir besoin de coder toute la partie CSS.

Le framework propose des modèles en HTML, CSS et JavaScript, pour mettre en page des composants de navigations, des boutons, des images, des blocs. Il définit un système de grille qui lui est propre.

https://getbootstrap.com/

1.5.4 Alternatives à Bootstrap

- Skeleton.
- Bulma.
- Pure.css.
- Groundwork.
- Cardinal.
- PowertoCSS.
- Mueller.
- Bootflat.

2. Projet

2.1 Mise en contexte

Vous êtes une équipe d'intégrateurs au sein d'une entreprise de service informatique. Vous êtes affectés à un projet de développement d'une application web responsive pour une agence immobilière.

Il s'agit d'une application destinée uniquement aux salariés de l'agence, qui permettra de centraliser et faciliter la gestion de toutes les informations internes notamment sur les ventes, les clients, les informations liés aux visites/états des lieux, etc…

Les salariés cibles de cette application sont:

- Les vendeurs / commerciaux qui utiliseront beaucoup l'application via un support mobile ou tablette, notamment lors des visites et états des lieux sur place.
- Le personnel en charge de la gestion de l'agence utiliseront eux l'application plutôt au format Desktop avec leur ordinateurs fixes à l'agence.

Un des besoins principaux de votre client est donc l'adaptabilité de l'application aux différents supports.

2.2 Travail attendu

En tant qu'intégrateurs, votre intervention sur ce projet consistera en la production de maquettes HTML/CSS destinées aux équipes de développement.

Ces maquettes seront réalisés à partir du cahier des charges détaillant quels elements sont présents sur chaque interface. Vous devrez donc utiliser vos connaissances pour concevoir une UX responsive à partir de simplement une liste d'elements à intégrer.

2.3 Cahier des charges

Votre client vous impose d'utiliser un thème Bootstrap Material: https://github.com/coreui/coreui-free-bootstrap-admin-template

Voici la liste des interfaces à réaliser avec les élements contenus:

- Une barre de menu latérale, commune à toutes les pages, donnant accès à:
  - Actualités
  - Base Clients
  - Base Annonces
  - Chat et appels
  - Calendrier
  - Mes ventes
  - Centre de contact
  - Analytique
- Un bandeau haut-de-page commun à toutes les pages contenant:
  - Fil d'arianne
  - Icone image de profil avec menu déroulant vers une partie "Mon Compte" (Notifications, Chat, Calendrier, Mes ventes) et une partie "Paramètres" (Profil, Se Déconnecter)
- Une page Login contenant:
  - Un formulaire Login / MDP
  - Un lien vers une page récupération de mot de passe
- Une page Actualités contenant:
  - Une liste d'articles type blog avec photo / texte
- Une page Base Clients contenant:
  - Un formulaire de recherche avec des filtres
  - Un tableau de clients découlant de la recherche et affichant par défaut toutes les annonces
- Une page Base Annonces contenant:
  - Un formulaire de recherche avec des filtres

    - Un tableau d'annonce découlant de la recherche et affichant par défaut toutes les annonces
- Une page Chat et appels contenant:
    - Un formulaire de recherche avec de contacts
    - Une liste de toutes les discussions liées à l'utilisateur.
- Une page Calendrier contenant:
    - Un calendrier avec des évènements généraux à l'agence et des évènements personnels
- Une page Mes ventes contenant:
    - Un tableau contenant toutes les ventes et dossiers en cours du salarié authentifié.
    - Un tableau contenant l'historique des ventes et dossiers liés au salarié authentifié.
- Une page Centre de contact contenant:
    - des encarts avec les informations de contact des différents référents (référent commercial, référent administratif et financier, directeur d'agence)
- Une page Analytique contenant:
    - divers graphiques et métriques sur les ventes et dossiers de l'agence.

2.4 Méthodologie suggérée

Afin de vous aider à vous organiser dans la réalisation de ce projet, voici une suggesion de méthodologie à suivre.

Travail collaboratif

Il s'agit d'un travail en groupe, il faudra donc articuler vos travaux autour d'outils facilitant la collaboration et la parallèlisation des tâches.

Pour cela, vous pourrez utiliserer au moins un outil de versionning (GIT ?) et un outil de suivi de tâches (Kanban), à mettre en place au début du projet avec des règles et conventions comprises et maîtrisées par tous.

Mobile First

Comme vu précédemment, l'approche Mobile First consiste à concevoir le Responsive Design d'abord sur le support mobile car celui-ci présente le plus de contraintes dans son développement.

Vous pourrez donc commencer à maquetter pour le support mobile et adapter votre Responsive Design aux plus grands supports.

Qualité de code et d'arborescence

Afin de fournir un code de qualité aux équipes de développement, il s'agira donc de respecter certaines normes qui amélioreront l'intégration, la maintenabilité ou encore l'évolutivité de vos maquettes.