Étude Technologique sur le Responsive Web Design : Comparaison entre les frameworks Bootstrap et Fountation 5

Ying Jiang - Falou Seck

22/10/2015

Table des matières

Introduction

Dans le cadre du cours d'**Adaptation des interfaces à l'environnement**, il nous a été demandé de produire une étude comparative de technologies permettant aux IHMs d'être utilisables avec différents supports. Nous avions le choix entre le *Responsive Web Design* (RWD) et la programmation mobile *Cross-Platform*. Nous avons opté pour le *Responsive Web Design* en utilisant deux frameworks très populaires : Bootstrap et Foundation.

Pour mener à bien notre étude, nous avons développé une application web dont le but est de présenter un artiste connu : Paul McCartney. Cette application web a été développée en deux versions : une avec Bootstrap et une autre avec Foundation.

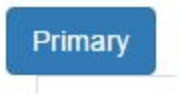
Présentation des technologies

Bootstrap

Bootstrap est un framework, dédié au web, créé par Twitter en 2010. C'est un outil, assez complet, facilitant le développement d'une application. En effet, il peut assurer la gestion complète des styles dans un site web. Il propose des feuilles de style en Less, en Sass et en CSS. Celles-ci se chargent de mettre en forme les composants HTML les plus utilisés parmi lesquels : les boutons, les tableaux, les formulaires et les menus de navigation.

Une des principales forces de ce framework est sa structure en grille. C'est cette division qui permet entre autres d'adapter le contenu d'une application web aux trois principaux supports : PC, tablette et smartphone. Pour utiliser Bootstrap avec un composant HTML, il suffit juste d'ajouter l'attribut `class` à la balise que l'on souhaite mettre en forme. Par exemple, pour un bouton "primary", c'est-à-dire large, ayant un fond bleu et une couleur de police blanche, il suffit de faire soit :

- `<span class="btn btn-primary">Primary</span>` ou
- `<button class="btn btn-primary">Primary</button>` ou encore
- `<input type="button" class="btn btn-primary" value="Primary">`, le résultat sera strictement identique et ressemblera à :



Fountation

Foundation est un framework web créé par Zurb. Il se définit comme "le framework front-end 'responsive' le plus avancé du monde". Comme Bootstrap, il fournit des feuilles de style en

CSS et gère les composants HTML les plus utilisés. Aussi, il divise l'écran en douze colonnes, par défaut. La seule différence est que le nombre de colonnes n'est pas fixé à douze comme en Bootstrap. Il est paramétrable en Foundation. Le nombre de colonnes est modifiable à souhait par le développeur.
Foundation fournit de plus des fonctionnalités plus complexes, comme le Joyride qui permet de stocker tous les points d'arrêts d'un site. Tout ceci simplifie le travail pour construire une page web.

Présentation de l'application

Nous avons développé un site web présentant Paul Mc Cartney et une partie de son oeuvre. C'est une *Single Page Application* (SPA) qui permet de voir ses derniers concerts et ses clips les plus mythiques, de découvrir des photos de l'artiste, de pouvoir écouter ou réécouter une partie de sa discographie et enfin de réserver des places pour ses prochaines prestations. Nous montrerons comment une partie de l'application a été développée en utilisant les deux frameworks. L'intégralité du code source est jointe à ce fichier. Mais avant d'utiliser ces frameworks forts utiles, parlons du maquettage de l'application.

Maquettage

Le maquettage est une bonne technique pour avoir idée claire de la composition d'une interface. C'est d'autant plus utile lorsque l'on décide de développer une application web "responsive". En effet cela permet d'avoir dès le départ la structure de l'IHM sur les supports cibles : PC, tablette et smartphone.

Le principe des SPA consiste à retrouver tout le contenu d'un site web sur une seule et même page. Ainsi les liens présents dans le menu de navigation en haut de la page ne redirigent plus vers d'autres pages mais vers des sections ; qui occupent toute la page lorsqu'on y accède. Un des principaux avantages de l'utilisation des SPA est une navigation plus fluide. Dans notre application, nous aurons donc les sections suivantes : **vidéos, photos, discographie** et **biographie**.

Nous avons réalisé deux maquettes à l'aide d'un outil de maquettage basse fidélité : Evolus Pencil. Nous avons ainsi établi une maquette pour écran desktop et une autre pour smartphone. Le but de ces deux maquettes est d'exploiter la puissance de la structure en grilles des deux frameworks. Comme elles le montrent ci-dessous :

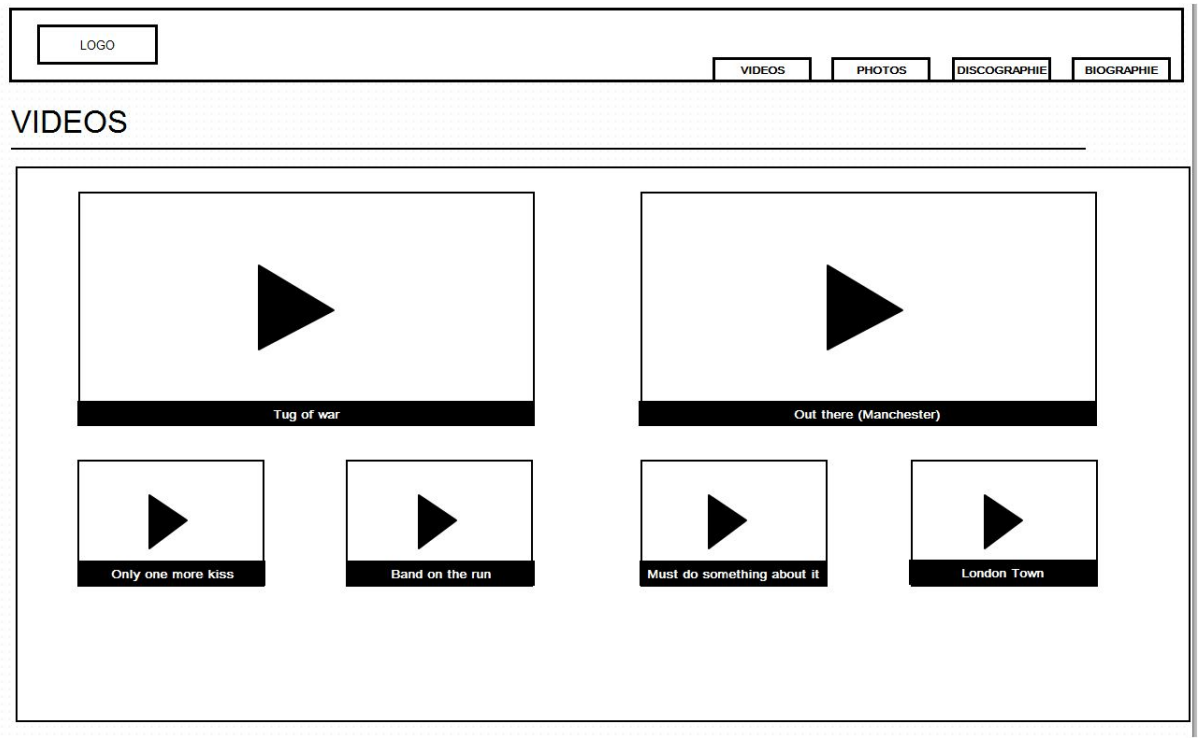


Figure 1 : Maquette de l'application version desktop

Nous pouvons voir que pour la maquette desktop, une section occupe en réalité une ligne et l'intégralité des douze colonnes de l'écran. Dans cette même ligne, les vidéos occupent deux lignes. Sur la première ligne, les vidéos occupent seulement 2 colonnes alors qu'elles en occupent quatre sur la seconde.

Dans la version smartphone (Taille de l'écran : 360 x 640), l'agencement, est tout autre. En effet, les vidéos occupent chacune une ligne, la section vidéo étant divisé en 3 lignes. On remarque de plus que la barre de navigation d'en haut disparaît pour laisser à un bouton communément appelé "hamburger", qui lorsqu'on clique dessus affiche les liens du menu de navigation.

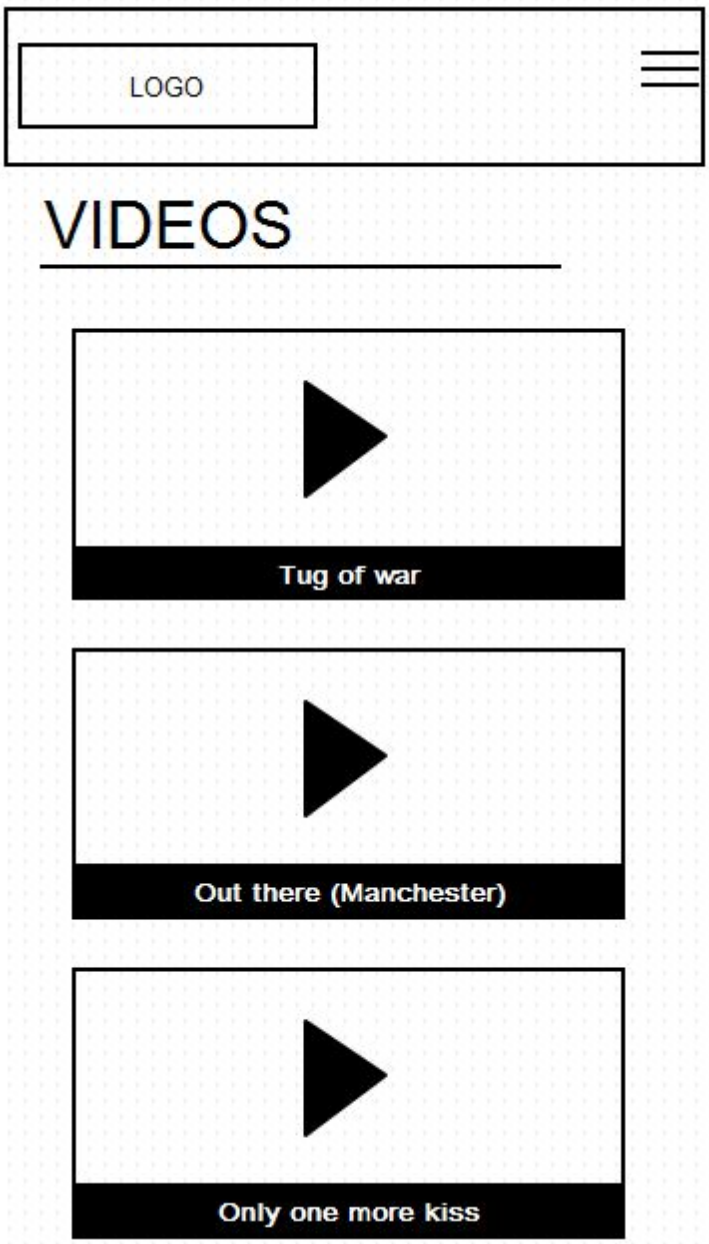


Figure 2 : Maquette de l'application version smartphone

Le maquettage nous a donc permis de prévoir comment notre IHM s'adaptera à des écrans de taille différente. Nous pouvons à présent passer à l'implémentation.

Installation des frameworks

Après avoir créé un dossier pour votre projet, vous pouvez télécharger Bootstrap et Foundation de deux manières :

- Soit en récupérant les fichiers archivés (en .zip) à partir des sites officiels de Bootstrap et de Foundation.

- Soit en utilisant le gestionnaire de packages front-end bower. Placez-vous à la racine du dossier de votre projet et tapez les commandes :
  - `bower install --save`[1] `boostrap` ou
  - `bower install --save foundation`

Nous avons, pour notre part, choisi la seconde option. L'avantage de l'utilisation de bower est qu'il se charge de télécharger toutes les dépendances d'un package. Par exemple pour Bootstrap, bower installera aussi jquery. Pour Foundation, le gestionnaire de packages installera en plus fastclick, jquery, jquery-cookie, jquery-placeholder et modernizr. Tous ces packages téléchargés seront mis dans le dossier bower_components.

Maintenant que nous avons tous les packages nécessaires, nous pouvons commencer l'implémentation de notre site web. Vu que nous avons choisi de développer une SPA, nous n'aurons besoin que d'une seule page html que nous renommerons index.html. Le fichier HTML devra avoir une structure minimale similaire à celle - ci :

```
<html>
      <head>
            <title>PMC | {Bootstrap | Foundation} version</title>
      </head>
      <body>
      </body>
</html>
```

Outils de développement

Pour cette application nous avons utilisé l'environnement de développement Webstorm et l'éditeur de texte Sublime Text 3. Webstorm a l'avantage d'être très complet et est adapté aux grands projets. Sublime Text quant à lui est un éditeur de texte léger, très utilisé et qui gère une multitude de langages (Java, C, C#, Bash, JavaScript, HTML, etc). Pour cette petite application web, Sublime Text est donc largement suffisant.

Utilisation des frameworks

Nous allons inclure le fichier css de Bootstrap ou de Foundation dans la balise `<head>` du fichier `index.html`.

```
      <head>
            <link
      rel="stylesheet"href="../bower_components/bootstrap/dist/css/b
      ootstrap.css">
            <title>PMC | {Bootstrap | Foundation} version</title>
      </head>
```

ou

[1] L'utilisation de l'option --save permet d'inclure le package que l'on souhaite télécharger comme dépendance dans le fichier bower.json. Cela implique donc la présence d'un fichier de configuration bower.json. Il peut être créé via la commande bower init.

```
<head>
      <link                                          rel="stylesheet"
href="../bower_components/foundation/css/foundation.min.css">
      <title>PMC | {Bootstrap | Foundation} version</title>
</head>
```

C'est le seul fichier dont nous aurons besoin pour l'instant.

Implémentation du menu de navigation

En Bootstrap

L'implémentation d'un menu de navigation se fait très rapidement et suit quasiment la même démarche pour beaucoup d'applications web. Juste après la balise ouvrante `<body>`, on utilise le tag `<nav>` pour définir notre menu de navigation. Bootstrap étant un framework intuitif, la classe CSS pour construire une barre de navigation s'appellent `navbar`. Il faut aussi savoir que souvent en Bootstrap, pour mettre en forme un composant, on indique sa classe générale avant d'indiquer sa classe particulière. C'est ce que nous avons fait plus haut avec le bouton `primary`. Nous avons utilisé la classe générale `btn`, pour indiquer que c'est un bouton et la classe `btn-primary` pour dire que c'est un bouton de type "primary".

Donc pour la barre de navigation, la balise `<nav>` sera de la forme `<nav class="navbar navbar-default"></nav>`. Nous allons donc utiliser la barre de navigation par défaut de Bootstrap. Ce sera un bandeau avec un fond grisé occupant toute la largeur de l'écran. Mais nous allons fixer la barre de navigation en haut de la page. Ainsi, l'utilisateur pourra facilement changer de section où qu'il se trouve dans la page. Pour ce faire, nous allons rajouter la classe **`navbar-fixed-top`** à notre balise `<nav>`. Nous allons aussi rajouter l'attribut `role="navigation"`. C'est un attribut de HTML 5 dont une de ses raisons est de faciliter l'accessibilité d'une application. Ainsi, les lecteurs d'écran pourront savoir que la balise <nav> et tous ses fils servent à la navigation. Nous allons passer à l'entête de notre menu de navigation.

Dans cette entête, nous aurons la configuration de l'"hamburger" (qui apparaîtra uniquement sur des écrans de taille réduite) et au logo du site web. Il est également simple à implémenter et à la forme suivante :

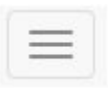
```
<nav       class="navbar       navbar-default       navbar-fixed-top"
role="navigation">
      <div class="navbar-header">
                <button  class="navbar-toggle"  data-toggle="collapse"
data-target="#menu-principal">
            <span class="sr-only">Toggle navigation</span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
          </button>
```

```
        <a class="navbar-brand text-uppercase" href="#page-top"><img
id="logo-page"

src="photos/pmc_signature.png"/></a>
      </div>
</nav>
```

Nous avons donc vu que l'entête a pour classe `navbar-header`. En dessous de cette `div`, nous avons une balise `<button>` avec trois attributs :

- un attribut `class="navbar-toggle"` pour basculer en mode navigation.
- un attribut `data-toggle="collapse"` pour avoir un effet plier / déplier
- un attribut `data-target="#menu-principal"` qui fonctionne comme l'attribut `href` dans les balises `<a></a>`. Cet attribut permet d'indiquer les données à afficher lorsqu'on clique sur ce bouton. Nous allons donc accéder à la `div` ayant pour id menu-principal lorsqu'on clique sur ce bouton "hamburger".

A l'intérieur de la balise `<button></button>`, nous avons mis une balise ayant pour classe sr-only. C'est une classe uniquement destinée aux lecteurs d'écran d'où le sr (screen reader). Nous avons ensuite trois autres balises `<span>` qui ont tous la même classe `icon-bar`. Cela permet d'avoir les tirets composant notre hamburger qui aura cette forme : 

Et enfin la dernière balise de notre entête de menu de navigation contient le logo du site web d'où l'utilisation de la classe `navbar-brand`. Attention cependant, Bootstrap ne gère pas encore très bien l'ajustement automatique du logo sur la barre de navigation. Il faut donc gérer soi même la forme (principalement la largeur et la hauteur) du logo. C'est ce que nous avons fait en créant un autre fichier `style.css` (qu'il faudra bien sûr inclure juste après le fichier `bootstrap.css`). La balise `<img/>` contenant le logo de notre page a pour id `logo-page`. Dans le fichier css, nous l'avons mis en forme comme suit :

```
#logo-page {
   height: 200%;
   width: 20%;
}
```

Notre entête est terminée. Nous pouvons passer aux liens du menu de navigation. Les liens seront mis dans une balise `<div>` qui aura donc pour id `menu-principal` et qui contiendra une liste (balise `<ul><li>...</li></ul>`) des sections de notre application. Notre menu de navigation ressemblera donc à cela :

```
<div class="collapse navbar-collapse" id="menu-principal">
  <ul class="nav navbar-nav navbar-right">
     <li>
                          <a   class="page-scroll   text-uppercase"
href="#videos">Vidéos</a>
     </li>
  </ul>
```

```
</div>
```

Cette div a donc pour id menu-principal et a pour classe `collapse` (pour indiquer que c'est un composant qui pourra être plié et déplié) et `navbar-collapse` pour dire que c'est une div de navigation pliable / dépliable. Dans cette div, nous avons une liste non ordonnée (`<ul>`) qui contiendra les éléments de notre menu de navigation. C'est pourquoi la balise `<ul>` a pour classe `nav navbar-nav navbar-right` (par défaut les éléments sont positionnés à gauche). Il suffira juste de copier la balise `<li></li>` de "vidéos" et de la coller à la suite de celle-ci en prenant soin de changer l'id et l'intitulé du lien :)).

Notre menu de navigation est presque terminé. Nous pouvons à présent le tester. Vous pouvez ouvrir le fichier index.html avec tous les navigateurs (nous utiliserons pour notre part Mozilla Firefox) à partir de l'explorateur de fichiers. C'est une application web statique qui ne nécessite pas de serveur. Il n'y a donc pas de commande particulière à taper.

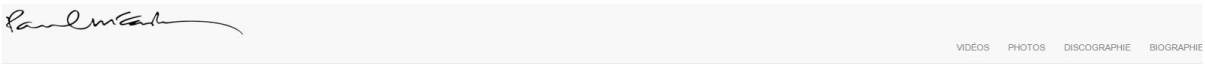


Figure 3 : 1er rendu de la barre de navigation

Sur un navigateur web, vous aurez le rendu ci - dessus. C'est très simpliste mais nous avons un menu de navigation qui répond à nos besoins. L'objectif de Bootstrap étant de nous aider à créer des applications web "responsive", nous allons tester notre application sur des écrans de taille réduite. Pour ce faire, nous allons utiliser l'émulateur mobile de Firefox (F12 > ). Vous pourrez ainsi choisir toutes les tailles d'écran pour bien tester l'adaptation de l'application web. Vous verrez que si nous choisissons un petit écran (360 x 640), l'hamburger apparaît et le menu de navigation horizontal disparaît et nous avons le rendu suivant :

Figure 4 : 1er rendu du menu de navigation en Bootstrap avec l'émulateur de Firefox

Vous avez remarqué que la barre de navigation occupait toute la largeur de l'écran et qu'il n'y avait de marge ni pour la photo ni pour les liens de navigation. Nous allons corriger cela en mettant nos liens de navigation (juste après `<nav>`) dans une balise `<div>` ayant pour classe `container`. C'est une classe Bootstrap qui permet de rendre le composant "responsive" avec une taille fixe. Mais pour ce cas ci, il nous permet d'avoir des marges pour le logo et les liens de navigation et d'avoir ainsi une application plus ergonomique. Nous aurons donc le rendu ci-dessous :

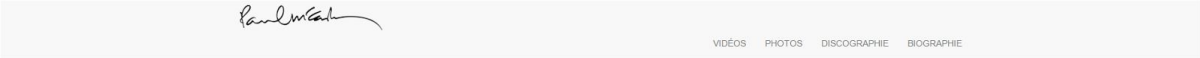


Figure 6 : 2nd rendu du menu de navigation (version desktop) en Bootstrap

En Foundation

Dans la version Foundation, nous avons aussi réalisé un menu de navigation. Comme Bootstrap, Foundation fournit plusieurs types de barres de navigation. Pour les utiliser, vous allez juste ajouter un attribut `class`, et mettre le nom de la barre de navigation comme valeur. Ce qui va la rendre responsive.
Dans cet exemple, pour avoir un effet personnalisé, nous avons fait un menu de navigation sans utiliser les barres navigation prédéfinies. La structure du menu est une combinaison d'un texte et un groupe de boutons, comme c'est affiché dans l'image suivante :

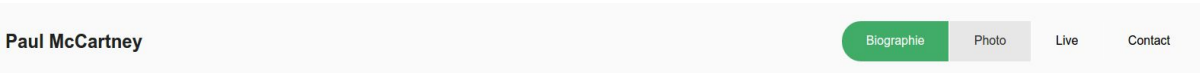


Figure 7 : 1er rendu du menu de navigation (version desktop) en Foundation

Pour la version en Foundation, nous n'aurons pas d'image comme logo. C'est le nom de l'artiste qui fera office de logo. Ce qui donne comme code :

```
<div class="small-12 medium-4 large-6 columns namelogo">
  <h1>Paul McCartney</h1>
</div>
```

Nous expliquerons plus tard les valeurs de l'attribut `class` dans la balise `<div>`.

À droite du menu, il y a un groupe de boutons, qui forment nos liens de navigation et qui vont nous rediriger vers chaque section de la page: biographie, photo, live et contact. Pour réaliser ce groupe de boutons, nous avons utilisé la classe `button-group`, et bien sûr, nous pouvons encore ajouter des effets au-dessus. Dans cet exemple, nous utilisons en plus la classe `round` comme nous avons vu dans l'image précédente. La structure du groupe de boutons est la suivante:

```
<div class="small-12 medium-8 large-6 columns">
    <ul class="button-group  round">
    <li><a href="#biographie" class="button success">Biographie</a></li>
    <li><a href="#photo" class="button secondary">Photo</a></li>
    <li><a href="#live" class="button">Live</a></li>
    <li><a href="#contact" class="button ">Contact</a></li>
    </ul>
</div>
```

Les boutons de Foundation sont très proches de ceux de Bootstrap. En effet Nous pouvons des classes comme `success` (`btn-success` en Bootstrap), `secondary` pour changer les apparences des boutons. Ou nous modifions directement dans le fichier .css, comme suivant:

```
.button-group a{
    color:rgba(0,0,0,1);
    background-color:transparent;
}
```

Maintenant, vu que nous n'avons pas utilisé le menu prédéfini dans Foundation, nous devons le rendre adaptative avec les système de grilles de Foundation. C'est ce que nous avons vu dans les codes précédents :

```
<div class="small-12 medium-4 large-6 columns namelogo">
    <div class="small-12 medium-8 large-6 columns">
```

La grille de Foundation ressemble beaucoup à celle de Bootstrap. Elles ont toutes 12 colonnes par défaut, mais dans Foundation, nous n'avons que trois types d'écran : `small`, `medium` et `large`, qui correspondent respectivement aux écrans de petite, moyenne et grande taille.
Dans cet exemple, quand l'écran est de grande taille, le logo et le groupe de boutons partagent une ligne, chacun prend un moitié de place, soit 6 colonnes;

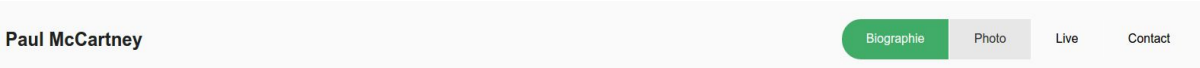


Figure 8 : Rendu du menu de navigation sur un écran large en Foundation

Lorsque l'écran est de taille moyenne, le logo prend 4 colonnes et le groupe de boutons prend 8 colonnes, et ils sont toujours sur la même ligne;

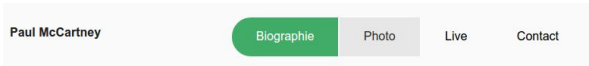


Figure 9 : Rendu du menu de navigation sur un écran de taille en Foundation

Par contre, lorsque l'écran est petit, le logo et le groupe de boutons prennent chacun 12 colonnes, c'est-à-dire que chaque composant occupe une ligne.

Figure 10 : Rendu du menu de navigation sur un écran de petite taille en Foundation

Implémentation du corps de l'application

Nous allons rentrer dans le vif du sujet. Pour cette partie, nous allons reprendre la maquette , mais annotée cette fois, afin d'illustrer et d'exploiter les principaux atouts du système de grilles des deux frameworks.

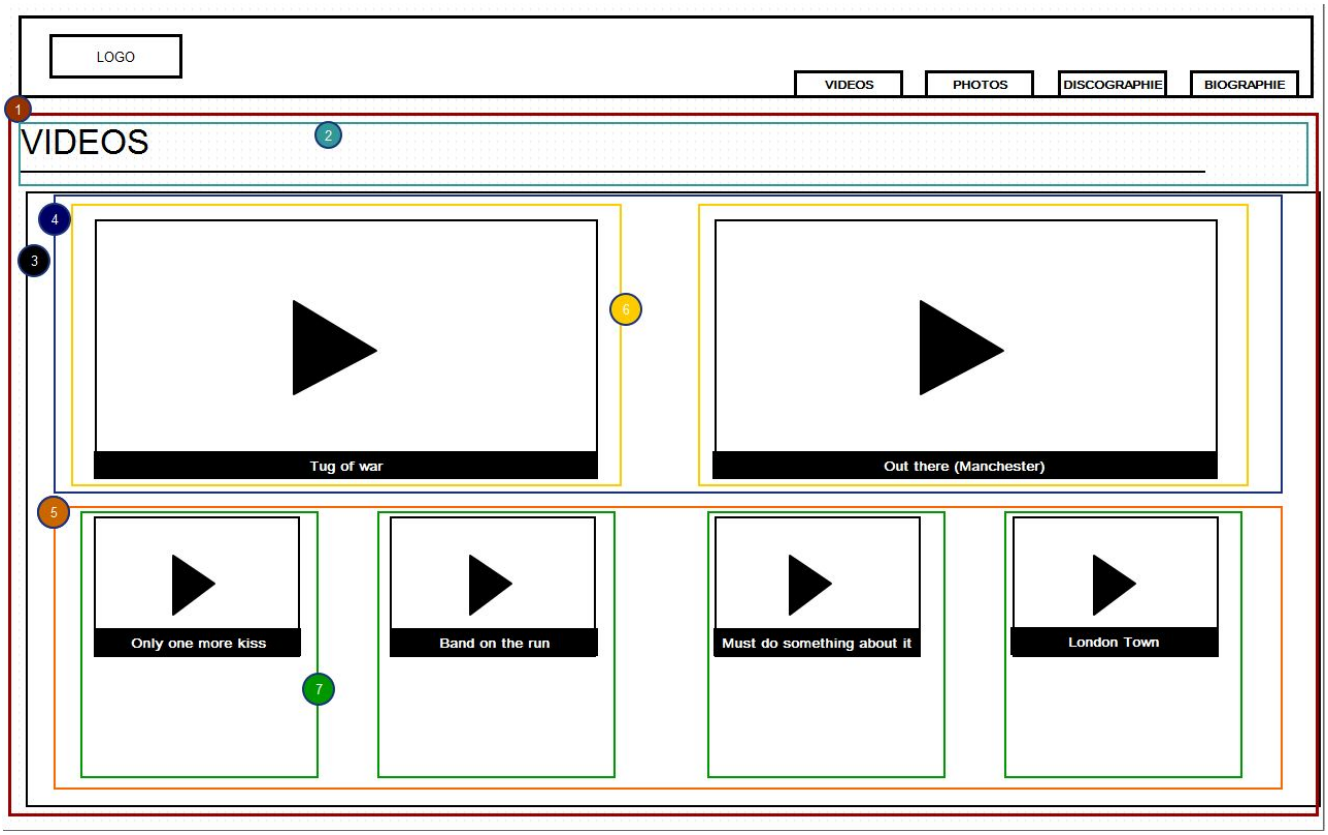


Figure 11 : Maquette annotée version desktop

A la suite de la balise `</nav>`, nous allons rajouter une balise `<div>` avec comme classe `container`. C'est dans cette div que nous mettrons tout le contenu de l'application. Nous allons donc créer notre section "vidéos".

Section "vidéos"

En Bootstrap

Pour créer notre section, nous avons besoin d'un tag... `<section>` avec comme id `videos`. Juste après la balise `<section>`, nous allons mettre une `<div class="container">`. Ce qui donnera le résultat suivant :

```
<div class="container">
    <!-- Content : sections -->
    <section id="videos">
    </section>
</div>
```

Il ne nous faut pas plus pour créer une section. Nous allons à présent construire la grille en nous basant sur les badges que nous avons mis sur la nouvelle maquette :

Nous allons commencer par la ligne principale :  . C'est une ligne qui aura donc la classe `row`. Ce qui donne :

```
<div class="row"></div>
```

C'est dans cette ligne que nous mettrons le titre (balise <h1>) de la section qui occupera toute la largeur de l'écran quelle que soit la taille de l'écran. Nous utiliserons donc toutes les douze colonnes de l'écran. Ce qui donne le résultat suivant :

```
<div class="row">
  <div class="col-lg-12">
     <h1 class="page-header text-uppercase">vidéos</h1>
  </div>
</div>
```

Il est important de remarquer qu'en Bootstrap (en tous cas jusqu'à la version 3.3.5 qui a été utilisée pour cette application), on peut utiliser les colonnes d'un écran selon quatre tailles :

- xs (extra-small), les très petits : les smartphones
- sm (small), les petits : les tablettes
- md (medium), les moyens : écrans d'ordinateur
- lg (large), le maximum pour les grands écrans d'ordinateur.

Pour la balise contenant le titre de la page, nous avons utilisé les douze colonnes des grands écrans. Cela aura comme conséquence l'occupation de toutes les colonnes des écrans de taille inférieure si aucune règle spécifique n'a été définie pour ceux-ci.

Vous remarquerez que le titre de la page n'apparaît pas. C'est normal. Il est caché par la barre de navigation. C'est un des inconvénients des barres de navigation ancrées en haut de l'écran. Pour résoudre le problème, il faut ajuster le CSS et ajouter de la marge intérieure pour le tag <body> (donc pour toute la page). Ce qui donnera comme résultat :

```
body {
   padding-top: 70px;
}
```

Pour construire la ligne  , vous aurez donc remarqué qu'il nous faut une balise . Dans cette div, nous allons rajouter une autre

pour construire la ligne  . C'est dans première cette ligne que nous mettrons nos premières vidéos. Sur des écrans de grande et de petite taille, chaque vidéo occupera six colonnes, par contre une vidéo occupera toutes les douze colonnes de cette ligne sur un smartphone. Ce qui nous donne :

```
<div class="row video-row">
  <div class="col-lg-6 col-md-6 col-xs-12 video-row">
     <div class="embed-responsive embed-responsive-16by9">
                                <iframe   class="embed-responsive-item"
src="https://www.youtube.com/embed/zIfPIfuTFXA"
```

```
                frameborder="0" allowfullscreen></iframe>
        </div>
    </div>
    <div class="col-lg-6 col-md-6 col-xs-12 video-row">
        <div class="embed-responsive embed-responsive-16by9">
                                <iframe    class="embed-responsive-item"
src="https://www.youtube.com/embed/_wlhdWiq77o"
                frameborder="0" allowfullscreen></iframe>
        </div>
    </div>
</div>
```

Les classes `embed-responsive embed-responsive-16by9` sont utiles pour rendre les vidéos embarquées (tirées de Youtube) “responsive”. En utilisant l’émulateur de Firefox vous verrez bien que les vidéos s’adaptent très bien quelle que soit la taille de l’écran.

Vous vous rendez maintenant compte que le système de grilles de Bootstrap et très simple à utiliser et est très efficace. Ainsi, pour construire la ligne 

, nous allons nous inspirer de 

. Sur cette deuxième ligne, les vidéos occuperont 3 colonnes sur un écran de grande taille, 6 sur un écran de taille moyenne et douze sur un écran de smartphone. Ce qui donnera comme code :

```
<div class="row video-row">
    <div class="col-lg-3 col-xs-12 col-md-6 video-row">
        <div class="embed-responsive embed-responsive-16by9">
                                <iframe    class="embed-responsive-item"
src="https://www.youtube.com/embed/azSY7W3U4nM"
                frameborder="0" allowfullscreen></iframe>
        </div>
    </div>
    <div class="col-lg-3 col-xs-12 col-md-6 video-row">
        <div class="embed-responsive embed-responsive-16by9">
                                <iframe    class="embed-responsive-item"
src="https://www.youtube.com/embed/yDzhrO5K02c"
                frameborder="0" allowfullscreen></iframe>
        </div>
    </div>
    <div class="col-lg-3 col-xs-12 col-md-6 video-row">
        <div class="embed-responsive embed-responsive-16by9">
                                <iframe    class="embed-responsive-item"
src="https://www.youtube.com/embed/Pzm7oSWEh6s"
                frameborder="0" allowfullscreen></iframe>
        </div>
```

```
  </div>
  <div class="col-lg-3 col-xs-12 col-md-6 video-row">
     <div class="embed-responsive embed-responsive-16by9">
                                <iframe    class="embed-responsive-item"
src="https://www.youtube.com/embed/5B5HNOI7IbE"
                  frameborder="0" allowfullscreen></iframe>
     </div>
  </div>
</div>
```

Vous avez sans doute remarqué l'utilisation de la classe `video-row` dans les lignes et colonnes de notre section. En fait, c'est une classe ad hoc que nous avons faite et qui permet de créer des marges entre les lignes et colonnes. En effet, en Bootstrap quand deux lignes se suivent, la division se voit à l'écran mais l'espacement est trop ténu. Nous avons aussi utilisé cette classe avec des colonnes pour la simple raison que sur certains écrans, certains composants tiennent sur quelques colonnes alors qu'ils occupent toute la largeur de l'écran (les douze colonnes) sur des écrans de plus petite taille et deviennent par conséquent… des lignes. C'est pourquoi dans notre feuille CSS, nous avons :

```
.video-row{
   margin-bottom : 20px;
}
```

Vous verrez que le "grid layout" de Foundation est très proche de celui de Bootstrap.

En Foundation

Comme présenté dans la maquette, le bloc de vidéo 

inclut trois parties: une grille de séparation 

qui contient le titre, une première ligne 

composée de 2 vidéos, et une seconde ligne composée de 4 vidéos 

.

La ligne de titre est simple à réaliser. Tout d'abord, nous séparons le bloc de vidéo des autres parties dans la page en utilisant une balise `<hr>` que nous avons mise en forme pour qu'elle ait cet effet :

Pour la première ligne 

, comme pour Boostrap, il suffit juste de créer une balise `<div>` ayant pour classe `row`. C'est elle qui contiendra nos deux grilles principales : 

et 

.

Pour construire la ligne 

, le principe est le même qu'en Bootstrap. La seule différence est qu'ici on cible trois types d'écran et que l'inclusion de vidéos embarquées est beaucoup plus simple. Ainsi, pour des vidéos positionnées sur la première ligne, nous aurons le code suivant :

```
<div class="row small-uncollapse medium-uncollapse">
    <div class="small-12 medium-6 large-6 columns">
```

Nous incluons les classes 'small-uncollapse' et 'medium-uncollapse' pour ajouter des marges intérieures entres les vidéos quand l'écran est petit ou moyen.

Pour rendre les vidéos *responsive*, il suffit d'inclure la classe `flex-video` et Foundation se charge du reste:

```
<div class="flex-video widescreen vimeo">
   <iframe width="1280" height="720" src="https://www.youtube.
   com/embed/tRnFHfI7WAQ" frameborder="0" allowfullscreen></iframe>
</div>
```

Pour construire la seconde ligne 

, nous suivons la même démarche que pour la grille précédente et le résultat reste fidèle à la maquette. Ce qui donne comme code au début:

```
<div class="row small-uncollapse medium-uncollapse">
    <div class="small-12 medium-6 large-3 columns">
```

Dans la version Foundation, après avoir suivi la maquette, nous avons aussi essayé d'afficher la vidéo dans une fenêtre pop-up. Tout d'abord, nous définissons un bouton qui déclenche l'ouverture de la fenêtre popup :

```
<a data-reveal-id="videoModal" class="radius button">Hey Jude Live at
Hyde Park</a>
```

ici, le `data-reveal-id` représente l'objet que nous voulons appeler. Ensuite, nous définissons l'objet relié à ce bouton:

```
<div id="videoModal" class="reveal-modal large" data-reveal aria-
labelledby="videoModalTitle" aria-hidden="true" role="dialog">
    <h2 id="videoModalTitle">Hey Jude Live at Hyde Park</h2>
    <div class="flex-video widescreen vimeo">
       <iframe width="1280" height="720" src="https://www.youtube.
       com/embed/tRnFHfI7WAQ" frameborder="0" allowfullscreen></iframe>
    </div>
    <a class="close-reveal-modal" aria-label="Close">&#215;</a>
</div>
```

le code au-dessus représente le pop-up block. Dedans, nous mettons le titre de la vidéo, et la vidéo elle-même, ainsi qu'un label qui nous permettra de fermer cette fenêtre pop-up.

Il faut noter que la classe `flex-video` existe non seulement dans Foundation mais aussi dans Bootstrap. Elle permet d'ajouter des vidéos de façon *responsive*.

Conclusion

Boostrap et Foundation sont deux frameworks qui ont le vent en poupe depuis quelques années. Ils sont très simples à utiliser, intuitifs, faciles à prendre en main et efficaces. Les deux technologies partagent la même philosophie. C'est-à-dire, une division de l'écran en grilles (grid layout). Elles ont une syntaxe proche (Par exemple `col-md-x` pour Bootstrap contre `medium-x columns` pour Foundation). Elles présentent cependant quelques limites. Par exemple, elles ne gèrent pas encore efficacement les tableaux. Si un tableau a plusieurs colonnes, le contenu du tableau sera difficilement adaptable aux écrans de petite taille.

Références

http://foundation.zurb.com/docs/components/topbar.html#clickable
http://foundation.zurb.com/docs/index.html
http://foundation.zurb.com/docs/components/kitchen_sink.html
http://foundation.zurb.com/docs/components/grid.html
http://responsive.vermilion.com/compare.php
http://getbootstrap.com/components/