Tutorial : Mise en place d'un dashboard de comparaison avec Bootstrap et PureCSS

Auteur : DITO Maxime

1) Description du dashboard

Notre projet est un dashboard comparant les technologies PureCSS et Bootstrap. Nous avons implémenté le dashboard à l'aide de PureCSS et Bootstrap. Chaque élément du dashboard contient un exemple de composant existant dans chacune des deux technologies, ainsi que le code permettant d'intégrer le composant à votre application. De plus, vous pouvez grâce à un bouton, copier rapidement l'intégralité du code que vous observez.

L'exemple fourni permet de visualiser les différences entre 3 composants de PureCSS et Bootstrap (image, navbar et tableau) cependant, nous n'aborderons pas leur inclusion ici car il vous suffit de copier le code des librairies dans les modules défini ci-dessous.

Cependant, ce document vous permettra de mettre en place ce dashboard sans se soucier du contenu qu'il comportera.

Ce dashboard est "responsive", cela signifie qu'il dispose d'une vue mobile et d'une vue tablette/PC.

Les tests de compatibilité ont uniquement été réalisés sur émulateur pour le mobile et la tablette en simulant les appareils suivants :

- Iphone 4
- Nexus 7.

Enfin, seul le navigateur Chrome a été utilisé.

Ce tutorial s'adresse principalement à des personnes ayant un minimum de connaissance dans les langages HTML et CSS.

2) Outils de développement

Pour utiliser les deux technologies, il suffit simplement de les télécharger soit :

- Sur leur site respectif (PureCSS, Bootstrap)
- Via un package manager (Bower, NPM ...)

Vous allez aussi avoir besoin de JQuery pour faire fonctionner les éléments Javascript de Bootstrap. Vous pouvez la télécharger :

- Sur le site (JQuery)
- Via un package manager (Bower, NPM ...)

Une fois les fichiers récupérés, il vous suffira de les inclure dans votre page/application web.
Voici la ligne de code à incorporer pour PureCSS dans entre les balises “header” de votre page. La première ligne inclus la librairie tandis que la deuxième ajoute le système de grille responsive :

```
<link rel="stylesheet" href="bower_components/pure/pure-min.css">
<link rel="stylesheet" href="bower_components/pure/grids-responsive-min.css">
```

Pour Bootstrap, vous devez inclure entre les balises “header” de votre page :

```
<link rel="stylesheet"
href="bower_components/bootstrap/dist/css/bootstrap.min.css">
```

Enfin, il faudra aussi ajouter entre les balises “body”, juste avant la balise de fermeture l’import de script suivant permettant d’utiliser JQuery ainsi que les composants Javascript de Bootstrap :

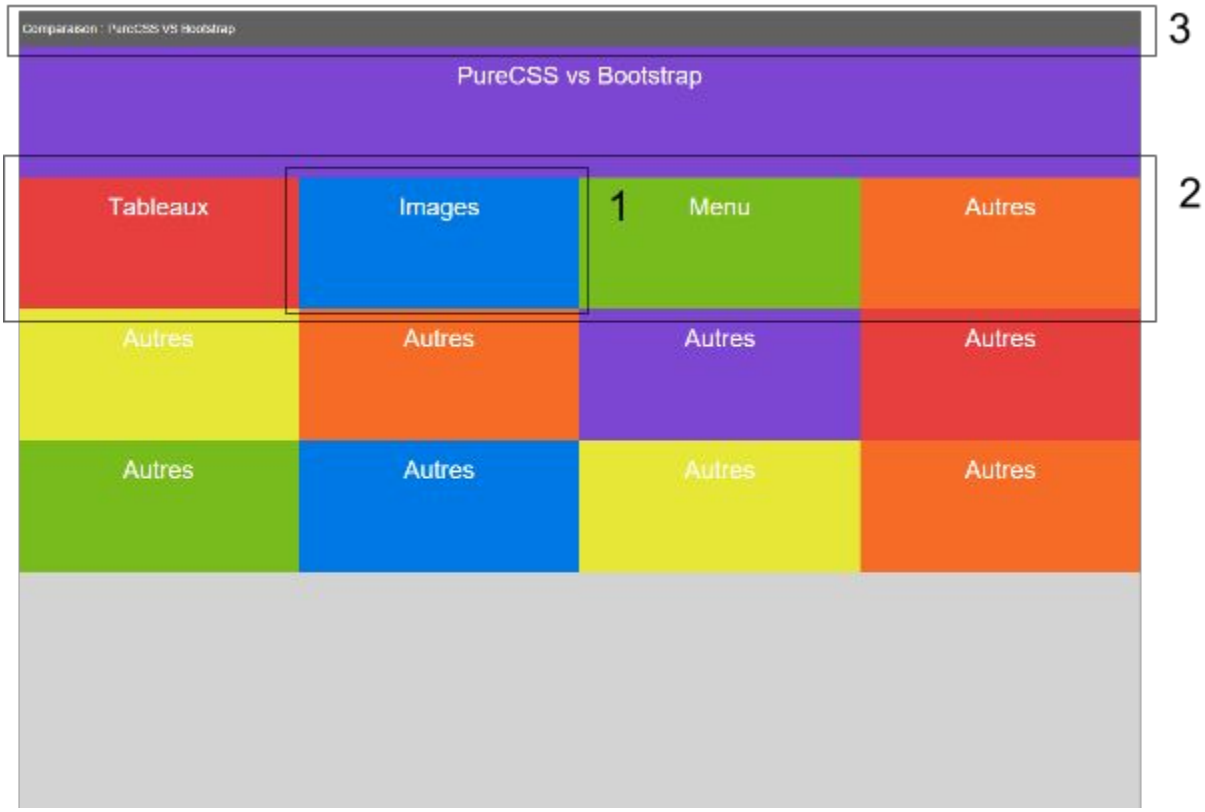
```
<script language="javascript" type="text/javascript"
src="bower_components/jquery/dist/jquery.min.js"></script>
<script language="javascript" type="text/javascript"
src="bower_components/bootstrap/dist/js/bootstrap.min.js"></script>
```

Nos librairies sont maintenant utilisables, nous pouvons donc commencer à développer l’application.

3) Première étape : la vue générale !

1) Architecture générale

Dans un premier temps, nous allons étudier l'architecture de la vue générale de notre dashboard afin de mieux comprendre sa complexité et sa mise en place.
Nous observons dans un premier temps la version PC.
Voici donc notre dashboard décomposé en plusieurs parties :

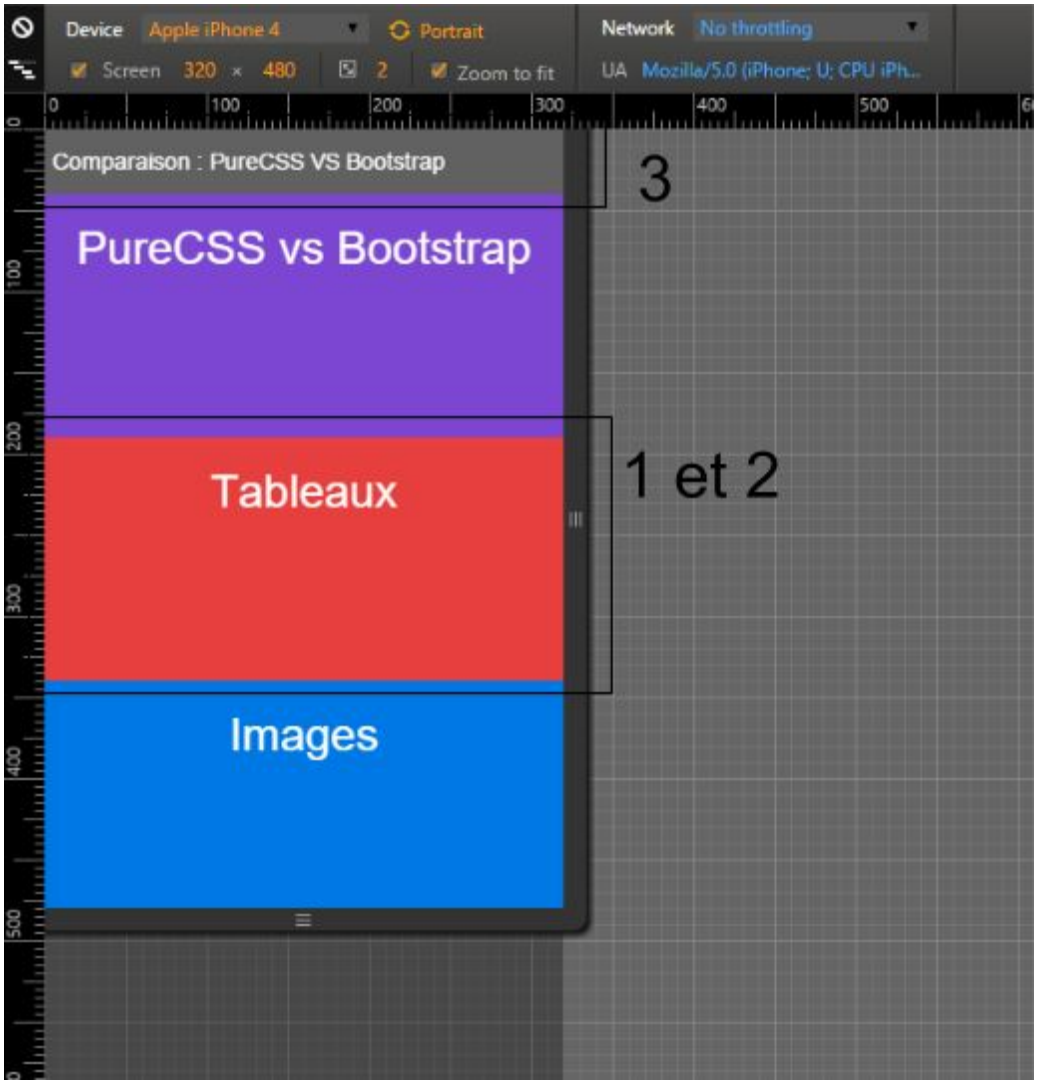


- La 1er partie (accordion) correspond à l'élément principale de notre dashboard. Il servira à afficher les composants PureCSS et Bootstrap que l'on souhaite comparer.
- La 2e partie (row) permet de gérer une ligne d'élément d'accordion.
- La 3e partie (menu-header) correspond à un menu flottant qui pourra implémenter des interactions avec l'utilisateur

Cependant, comme nous utiliserons la grille "responsive" de PureCSS, nous allons commencer par mettre en place ces éléments pour la partie mobile.

Pourquoi ce choix ? Tous simplement parce que PureCSS est pensé "mobile first", de ce fait, il préférable de commencer par mettre en place notre interface pour mobile et par la suite de s'occuper de la version PC.

Voici donc la décomposition de la vue générale mobile :



- La partie 1 et 2 sont fusionnées avec la vue mobile. En effet, pour chaque ligne, on affichera qu'un composant accordion.
- Enfin, la partie 3 reste identique à celle sur PC.

On constate donc que les deux vues sont très proches au niveau de leur conception et de leur découpage. C'est la cas car nous mettons en place une architecture "responsive" disposant d'une vue spécifique pour mobile.

2) Mise en place de la vue mobile

a. Mise en place de la partie 1 - Création du composant Accordion

Commençons par créer un élément de la partie 1, l'accordion. Pour cela, nous utiliserons le système de grille de PureCSS. Ces éléments ne sont rien de plus que des blocs contenant un titre, qui doivent, lors d'un clic ou du survol de la souris, afficher les éléments contenus en son sein.

Voici le code HTML permettant de définir la structure de notre élément :

```
<div class="pure-u-1 accordion dashboard-part orange">
   <h1>Mon Titre</h1>
   <div class="content pure-u-4-5">
   </div>
</div>
```

On distingue deux sections dans cet élément :

- Une première div définissant la structure interne de notre élément.
- Une deuxième div qui contiendra le contenu textuel à afficher lorsque l'élément sera touché ("touch") sur mobile ou au survol de la souris sur PC.

Détaillons maintenant les classes de chaque balises "div" :

- Les classes "pure-u-*" correspond au système de grille de PureCSS, il permet de définir la largeur que prendra notre élément dans notre vue. Le chiffre derrière correspond à la taille qu'il doit avoir, "1" correspond à 100%, "4-5" correspond à ⅘ de la largeur soit 80% etc…
- La classe "accordion" nous permettra de définir l'animation d'affichage de notre contenu.
- La classe "dashboard-part" définie les propriétés génériques de cet élément, comme la hauteur.
- La classe "content" sera utilisée pour savoir quel est l'élément à afficher lorsque l'on que l'utilisateur interagit avec le composant.
- Enfin la classe "orange" correspond à une couleur de fond.

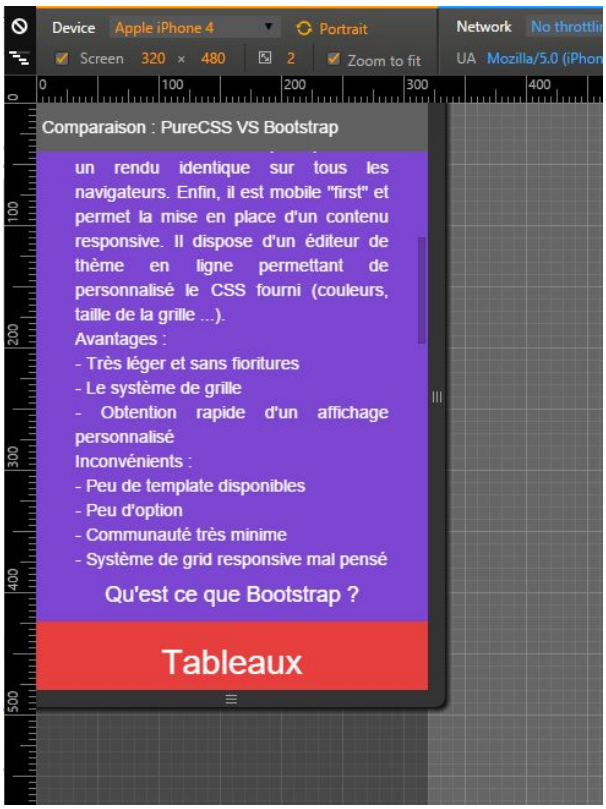
Voici donc le code CSS à incorporer soit dans une balise "style" de votre page soit dans un fichier CSS que vous importez :

```
.dashboard-part {
   /*Permet de fixer la taille de toute les div de manière identique*/
   -webkit-box-sizing: border-box;
   -moz-box-sizing: border-box;
   box-sizing: border-box;
   color: #fff;
   text-align: center;
   position: relative;
   padding-bottom: 40px;
```

```
   min-height: 100px;
   height:150px;
}

/* Définition des couleurs de fond de notre élement principal*/
.red {background-color: #E73F3F;}
.blue {background-color: #0078e7;}
.orange {background-color: #F76C27;}
.yellow {background-color: #E7E737;}
.green {background-color:#79BB1E;}
.purple {background-color:#7E45D3;}
/* ----- END -----*/
/* Définition de l'attitude général de notre composant (en commun pour mobile et
PC) */
.accordion {
   /*On cache le contenu en trop*/
   overflow: hidden;
}

.accordion > .content {
   /*On cache le contenu de la deuxième div ayant la class content*/
   visibility: hidden;
   margin: 5px 0;
}

.accordion:hover > .content, .accordion:focus > .content, .accordion:active >
.content{
   /*Lorsque la première div est touch pour mobile ou survolée (active) pour un PC
on affiche le contenu de la deuxième div*/
   visibility: visible;
}

.accordion:hover, .accordion:focus, .accordion:active {
   /*Taille minimal de la première div lorsqu'elle est active*/
   min-height:150px;
}
/* ----- END -----*/

/* Définition de l'attitude de notre composant pour mobile */
@media screen and (max-width: 768px) {
   .accordion:hover, .accordion:focus, .accordion:active {
   /* On effectue une transition sur les propriétés de notre élements pour faire
une animation */
   -webkit-transition: all .3s ease-in-out;
   -moz-transition: all .3s ease-in-out;
   -o-transition: all .3s ease-in-out;
   -ms-transition: all .3s ease-in-out;
   transition: all .3s ease-in-out;
   /* On lui donne une taille fixe, ici 80% de la hauteur de utilisable de
l'appareil et on permet de pouvoir scroll pour voir tous le contenu */
   height: 80vh;
   overflow-y:scroll;
   }
   .accordion {
```

```
   /* On effectue une autre transition pour quand le composant n'est plus actif
afin qu'il effectue aussi une animation de fermeture */
   -webkit-transition: all .3s ease-in-out;
   -moz-transition: all .3s ease-in-out;
   -o-transition: all .3s ease-in-out;
   -ms-transition: all .3s ease-in-out;
   transition: all .3s ease-in-out;
   }
}
/* ----- END ----- */
```

Voici le résultat de l'affichage lorsque l'on active un accordion (ici nous avons défini un contenu qui est du texte) :



Nous avons donc maintenant un composant accordion permettant d'afficher un bloc disposant d'informations cachées qui s'affichent lorsque l'on active le composant par un toucher ("touch") sur mobile pour un survol avec la souris.

b. Mise en place de la partie 2 - Création du composant Row

Maintenant que notre élément principal est défini, nous allons nous occuper de la partie 2 (row) et mettre en place une ligne d'accordion.

Pourcela il faudra ajouter dans votre code HTML :

```
<section class="dashboard pure-g">
        ...
</section>
```

Cette balise doit encapsuler 1 ou plusieurs composants accordion que l'on a précédemment créés.
Voici donc le code que vous devriez actuellement obtenir :

```
<section class="pure-g">
       <div class="pure-u-1 accordion dashboard-part orange">
          <h1>Mon Titre</h1>
          <div class="content" class="content pure-u-4-5">
          </div>
       </div>

       <!-- On peut ajouter d'autre composant accordion dans la section -->
</section>
```

Détaillons un peu ce composant, il dispose d'une classe propre au système de grille de PureCSS "pure-g". Cela défini que dans cette section, nous définirons des éléments utilisant le système de grille de PureCSS. Nous nous servirons donc de cette section pour définir des lignes (row) dans notre dashboard.

Nous avons donc défini un composant row pour notre dashboard. Il ne reste plus qu'à définir la dernière partie, le menu-header.

c. Mise en place de la partie 3 - Création du composant Menu-header

Il ne nous reste plus qu'à ajouter le menu-header qui flottera sur notre application. Ce menu affichera le titre du dashboard et pourra dans l'éventualité servir pour implémenter des interactions avec l'utilisateur.

Pour cela, ajouter dans votre balise "body", en dessous de la balise "body" ouvrante et au dessus des éléments précédemment créés, le code suivant :

```
<header>
   <nav class="pure-menu pure-menu-horizontal">
       <p>Mon Titre</p>
   </nav>
</header>
```

On remarque que la balise "nav" possède deux classes définies par PureCSS :

- la classe "pure-menu" indique que l'on utilise un menu défini par PureCSS
- la classe "pure-menu-horizontal" va définir la mise en page de notre menu. Contrairement à Bootstrap et son "menu hamburger", il ne sera pas vertical mais permettra de choisir un intitulé en les déplaçant de la gauche à la droite.

Comme nous souhaitons que notre menu soit fixe dans notre vue et afin de lui donner une apparence, nous devons ajouter le CSS suivant :

```
nav{
```

```
    /*On applique un style au menu et on fixe sa position*/
    background-color: rgba(85,85,85,0.9);
    padding-top: 10px;
    font-weight: lighter;
    position:fixed;
    z-index:200;
    top:0;
}

nav > p{
    /*On applique un style au titre du menu*/
    color: white;
    margin-left:5px;
}

section:nth-of-type(1) {
    /*On applique une marge à la première ligne pour pas qu'elle se trouve sous le
menu*/
    margin-top:40px;
}
```

Nous avons donc terminé de mettre en place notre vue générale pour mobile, nous allons ajouter des éléments à cette vue afin de l'adapter pour tablette et PC.

3) Mise en place de la vue PC et tablette

a. Mise en place de la partie 1 - Création du composant Accordion

Modifions le code précédemment écrit afin d'ajouter un affichage et des interactions différentes pour PC et tablette. Pour cela, nous allons adapter notre composant accordion en lui ajoutant la classe suivante "pure-u-md-1-4". Cette classe est rattachée au système de grille "responsive" de PureCSS est indique que notre élement prendra ¼ de la largeur de la ligne (row) si le dispositif à une résolution supérieur à 768px.

Pourquoi ne pas avoir défini la même chose pour le mobile ?
Tous simplement parce que la convention pour PureCSS indique que si on utilise la grille "responsive", il est préférable de définir les éléments pour mobile en utilisant le système de grille simple soit sans indiquer l'appareil que l'on cible (sinon on aurait pu utiliser la classe suivante "pure-u-sm-*" pour le mobile).

Voici donc ce que dois devenir notre composant accordion :

```
<div class="pure-u-1 pure-u-md-1-4 accordion dashboard-part orange">
    <h1>Mon Titre</h1>
    <div class="content pure-u-4-5">
    </div>
```

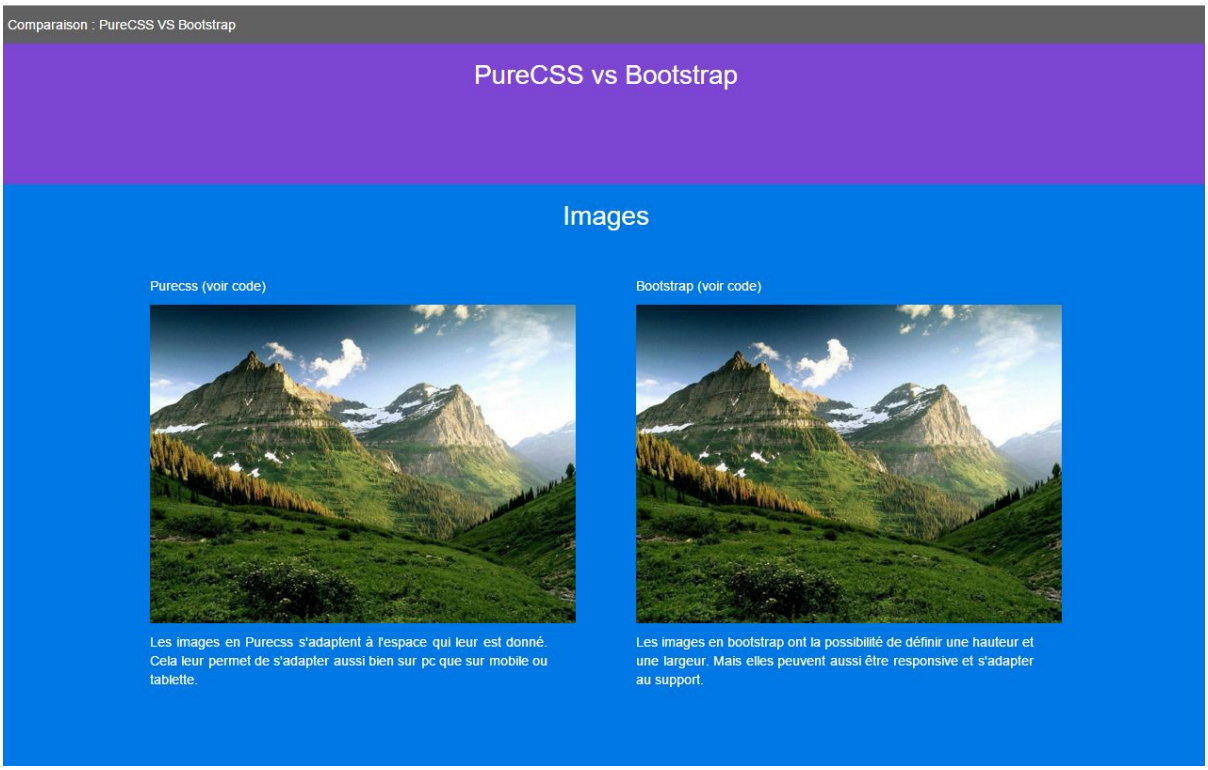
```
</div>
```

Par cet ajout, nous imposons implicitement le fait que l'on aura au maximum 4 composants accordion sur une même ligne. Il ne nous reste plus qu'à ajouter une animation lorsque l'utilisateur va survoler l'élément sur PC ou le "touch" sur tablette :

```
/* Définition de l'attitude de notre composant pour PC et tablette */
@media screen and (min-width: 768px) {
   .accordion:hover , .accordion:focus, .accordion:active  {
       /*On lui fait prendre toute la largeur*/
       width: 100%;
       /*On définit que notre élément n'est plus dans la grille et qu'il est au
dessus des autres éléments*/
       z-index: 100;
       position: absolute;
       /*On rend visible le surplus d'information*/
       overflow: visible;
       /* On effectue une transition sur les propriétés de notre élements pour
faire une animation défini pour le premier élément */
       -webkit-transition: width .3s ease-in-out, height .3s ease-in-out;
       -moz-transition: width .3s ease-in-out, height .3s ease-in-out;
       -o-transition: width .3s ease-in-out, height .3s ease-in-out;
       -ms-transition: width .3s ease-in-out, height .3s ease-in-out;
       transition: width .3s ease-in-out, height .3s ease-in-out;
   }
   /*Comme l'élément selectionné n'est plus dans la grille, on ajoute au suivant
une marge équivalent à son ancienne largeur pour ne pas avoir un effet de
décallage*/
   .accordion:hover+.accordion {
       margin-left:25%;
   }
   /*On effectue pour chaque élément en fonction de sa position (2, 3 ou 4) sa
transformation afin d'avoir un affichage similaire pour tous les accordion*/
   .accordion:hover:nth-child(2) {
       margin-left:-25%;
       /* On effectue une transition sur les propriétés de notre élements pour
faire une animation défini pour le deuxième élément */
       transition: width .3s ease-in-out, height .3s ease-in-out, margin .3s
ease-in-out;
   }
   .accordion:hover:nth-child(3) {
       margin-left:-50%;
       /* On effectue une transition sur les propriétés de notre élements pour
faire une animation défini pour le troisième élément */
       transition: width .3s ease-in-out, height .3s ease-in-out, margin .3s
ease-in-out;
   }
   .accordion:hover:nth-child(4) {
       margin-left:-75%;
       /* On effectue une transition sur les propriétés de notre élements pour
faire une animation défini pour le quatrième élément */
       transition: width .3s ease-in-out, height .3s ease-in-out, margin .3s
ease-in-out;
   }
```

```
    /*On fixe une taille minimal au cas ou le contenu est vide, sinon on utilise la
taille nécessaire*/
    .accordion:hover, .accordion:focus, .accordion:active {
        height: auto;
        min-height:150px;
    }
}
/* ----- END ----- */
```

Voici le résultat de l'affichage lorsque l'on active un accordion (ici nous avons défini un contenu qui sont des images) :



Notre élément est donc maintenant spécifique pour tablette et PC et s'affiche correctement comme sur mobile.

b. Mise en place de la partie 2 - Création du composant Row

Aucune modification dans le code n'est à prévoir hormis le fait que notre composant row ne peut maintenant recevoir que 4 éléments accordion.

c. Mise en place de la partie 3 - Création du composant Menu-header

De même pour notre menu, il n'y a aucun changement à mettre en place.

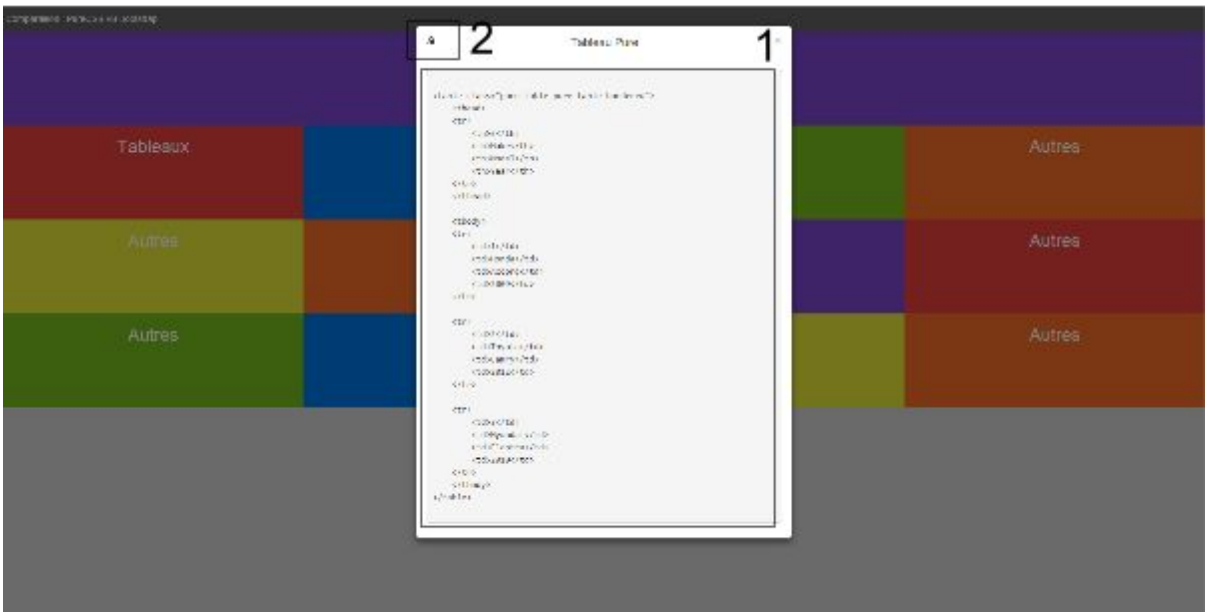
La vue principale de notre application est terminée, et nous avons créé un dashboard pouvant contenir des informations qui seront affichées de manière différentes sur PC et tablette ou mobile.

Il ne reste qu'à mettre en place notre dernière vue, celle permettant de visualiser en détail le code source de l'élément souhaitait.

4) Deuxième étape : la vue spécifique de détails !

1) Architecture générale

Voici à quoi va ressembler la vue que nous allons développer :



Dans la première partie de la vue, on remarque une section permettant d'afficher des informations détailées, ici le code source de l'élément contenu dans l'accordion. Et enfin, une seconde partie qui permet de copier rapidement les informations par un simple clic.

Pour mettre en place cette vue, nous utiliserons le composant "modal" de Bootstrap. De ce fait, la vue sera identique aussi bien pour mobile que pour PC et tablette.

2) Mise en place de la vue

Nous allons donc ajouter dans notre page, entre les balises “body”, juste après notre menu, le code suivant :

```
<div class="modal fade" id="myModal" tabindex="-1" role="dialog"
aria-labelledby="myModalLabel">
   <div class="modal-dialog" role="document">
       <div class="modal-content">
           <div class="modal-header">
               <button type="button" class="close" data-dismiss="modal"
aria-label="Close"><span aria-hidden="true">&times;</span></button>
             <!-- Zone dans laquelle sera injecté le titre par le script JS -->
               <h4 class="modal-title" id="myModalLabel"></h4>
           </div>
           <div class="modal-body" >
             <!-- Zone dans laquelle sera injecté le contenu par le script JS -->
               <pre id="modalBody"></pre>
           </div>
       </div>
   </div>
</div>
```

Ce code est fournit par Bootstrap est permet de déclarer un composant de type “modal”. Une modal est une fenêtre qui va s’ouvrir par dessus la vue actuelle en grisant le fond ne permettant d'interagir qu’avec elle.
Elle se compose de la sorte :

- Un contenu header “modal-header” qui affiche un bouton permettant de la fermer et un titre
- Un contenu body “modal-body” permettant d’afficher des informations (pour nous le code source par exemple)

Comme vous pouvez le constater, la modal est vide, elle ne contient ni titre,ni contenu. Cependant, ne voulant pas créer une modal pour chaque exemple, nous allons utiliser du javascript afin de la remplir avec les informations qui nous intéressent.

Pour cela, nous utiliserons le code javascript suivant qui permet de récupérer le code source de notre exemple, de supprimer l’indentation en trop, et de l’injecter dans le body de notre modal tout en injectant un titre dans le header de la modal :

```
//Permet de set le contenu et le titre de la modal
function setModal(id) {
     //Suppresion des tabulation inutile dans l'indentation du code et affectation
au contenu de la modal
   $("#modalBody").text($("#"+id.replace(" ", "")).html().replace(/[\t ]{1,20}\</g,
"<"));
   $("#myModalLabel").html(id) ;
```

Il vous suffit d'ajouter ce code entre une balise "script" entre les balises "body" de votre page.

Maintenant, nous disposons d'un affichage utilisable sur mobile, PC et tablette sachant récupérer du contenu. Nous allons ajouter la possibilité de pouvoir copier ces informations par un simple clic.

Il nous faut donc ajouter un bouton dans notre vue, pour cela, ajouter dans le header de la modal le code suivant au dessus de la balise "h4" :

```
<span class="glyphicon glyphicon-copy copyBtn" onclick="copyclipboard()"
title="Copier le code"></span>
```

On remarque qu'on utilise une balise "span" avec deux classes glyphycon fournies par Bootstrap permettant d'afficher une petit icône de copie, elle dispose aussi d'un attribut "onclick" qui, lors d'un clic sur l'icône, va déclencher une action : la méthode "copyclipboard" que nous allons implémenter.

Voici la méthode javascript que vous devez ajouter :

```
// Fonction permettant la copie du code dans le clipboard
function copyclipboard() {
   var $temp = $("<input>")
   $("#myModal").append($temp);
   $temp.val($("#modalBody").text()).select();
   document.execCommand("copy");
   $temp.remove();
}
```

Cette fonction permet de copier le contenu de la modal directement dans le "presse-papier" de l'utilisateur. Ainsi, il peut rapidement copier le code pour l'utiliser (ce qui est plutôt pratique sur mobile).

Enfin, nous allons ajouter un peu de style dans le header de la modal afin que le titre et ce bouton soit alignés :

```
#myModalLabel {
   display:inline-block;

}
.modal-header {
    text-align:center;
}

.copyBtn {
   float:left;
}
```

La vue est donc terminé complète, il ne reste plus qu'à relier entre elles ces deux vues.

3) Relier les deux vues

Pour cela, rien de très compliqué, il suffit simplement de permettre lors de l'appuie d'un bouton par exemple dans la première vue, d'afficher la deuxième. Pour cela, nous ajoutons le code suivant dans la balise "div" ayant la classe "content" d'un de nos accordion :

```
<p><span class="codeDemo" data-toggle="modal" data-target="#myModal"
onclick="setModal('Le Nom de mon contenu');">L'intitulé de mon bouton</span></p>
```

Ce code permet d'appeler l'ouverture de la seconde vue lors d'un clic sur cet intitulé.

Enfin, pour que notre script Javascript sache quel est le contenu qu'il doit afficher dans la modal. Ajoutons le code suivant dans dans la balise "div" ayant la classe "content" de notre accordion :

```
<div class="codeToLoad" id="LeNomDeMonContenu">
```

Ainsi, la mise en place du dashboard est terminé. Il ne vous reste plus qu'à injecter les données que vous souhaitez comme par ceux fourni en exemple de ce tutorial. Pour cela, ajouter le contenu entre la balise "`<div class="codeToLoad" id="LeNomDeMonContenu">`" définie ci-dessus.

Pourquoi ne pas aborder cette mise en place ? Tous simplement car elle ne présente aucune difficulté et que les éléments sont eux même présenté dans la documentation de leurs API respectives. De plus, il suffit simplement de copier le code fourni comme expliqué ci-dessus. A vous de faire votre choix d'intégration !

5) Conclusion

Vous pouvez maintenant visualiser votre résultat en ouvrant simplement la page HTML avec votre navigateur préféré. De plus, si celui-ci dispose d'outil de développement simulant l'affichage sur mobile ou tablette, vous serez en mesure d'observer l'adaptation de votre vue.