Bootstrap

Pour utiliser efficacement Bootstrap, vous devez déjà être convaincu de son utilité, vous devez aussi savoir l'installer. Ce chapitre est destiné à vous faire découvrir en quoi Bootstrap peut vous aider et comment vous devez le mettre en œuvre pour l'utiliser efficacement. Nous aborderons également les Medias Queries que Bootstrap utilise pour que les pages s'adaptent automatiquement au support utilisé pour les visualiser.

Un framework ?

Avant toute chose, il faut définir ce qu'est un framework. Il s'agit d'un ensemble de composants structurés qui sert à créer les bases et à organiser le code informatique pour faciliter le travail des programmeurs, que ce soit en terme de productivité ou de simplification de la maintenance. Il en existe beaucoup pour les applications web qui ciblent de nombreux langages : Java, Python, Ruby, PHP…

Il existe des frameworks côté serveur (désignés *backend* en anglais), et d'autres côté client (désignés *frontend* en anglais). Bootstrap fait partie de cette deuxième catégorie. Les frameworks CSS sont spécialisés, comme leur nom l'indique, dans les CSS ! C'est-à-dire qu'ils nous aident à mettre en forme les pages web : organisation, aspect, animation… Ils sont devenus à la mode et il en existe un certain nombre :

- Knacss
- Blueprint
- unsemantic
- YUI
- BlueTrip
- ez-css
- Pure
- Gumby
- Materialize
- etc.

Bootstrap est un framework CSS, mais pas seulement, puisqu'il embarque également des composants HTML et JavaScript. Il comporte un système de grille simple et efficace pour mettre en ordre l'aspect visuel d'une page web. Il apporte du style pour les boutons, les formulaires, la navigation… Il permet ainsi de concevoir un site web rapidement et avec peu de lignes de code ajoutées. Le framework le plus proche de Bootstrap est sans doute Foundation qui est présenté comme « *The most advanced responsive front-end framework in the world* ». Cette absence de modestie est-elle de mise ? Je pense que c'est surtout une affaire de goût et de préférence personnelle. En tout cas en terme de popularité c'est Bootstrap qui l'emporte haut la main.

Les intérêts

- Les navigateurs sont pleins de fantaisies et ont des comportements très différents malgré leur lente convergence vers les standards. Les frameworks sont *cross-browser*, c'est à dire que la présentation est similaire quel que soit le navigateur utilisé et d'une parfaite compatibilité.
- Les frameworks CSS font gagner du temps de développement parce qu'ils nous proposent les fondations de la présentation.
- Les frameworks CSS normalisent la présentation en proposant un ensemble homogène de styles.
- Les frameworks CSS proposent en général une grille pour faciliter le positionnement des éléments.

- Les frameworks CSS offrent souvent des éléments complémentaires : boutons esthétiques, barres de navigation, etc...
- La grande diffusion de nouveaux moyens de visualisation du web (smartphones, tablettes…) impose désormais la prise en compte de tailles d'écran très variées ; les frameworks CSS prennent généralement en compte cette contrainte.

Les inconvénients

- Pour utiliser efficacement un framework il faut bien le connaître, ce qui implique un temps d'apprentissage.
- La normalisation de la présentation peut devenir lassante en lissant les effets visuels.

Par rapport aux deux inconvénients énoncés, Bootstrap est d'une prise en main rapide même pour un débutant et est totalement configurable parce qu'il est construit avec LESS(que nous verrons dans le cours).

Découverte de Bootstrap

Origine de Bootstrap

Vous connaissez forcément Twitter, un des principaux réseaux sociaux qui inondent la planète de liens virtuels entre les humains devenus des noyaux cybernétiques. Le projet Bootstrap a été créé au départ par Mark Otto et Jacob Thornton pour répondre à des besoins internes de développement de cette entreprise au niveau de l'uniformisation de l'aspect des pages web. Il s'agissait juste de stylisation CSS, mais le framework s'est ensuite enrichi de composants Javascript.

Il a ensuite été publié en 2011 en devenant rapidement populaire parce qu'il est venu se positionner dans un espace vacant du développement. Son système de grille de 12 colonnes est devenu une référence. D'autre part sa mise en œuvre est aisée et se limite à référencer quelques librairies, comme nous allons bientôt le voir.

Il a été mis à disposition du public sous licence Apache. Le framework en est actuellement à la version 3. Celle-ci a pris un virage particulier en intégrant l'aspect « responsive » par défaut, alors qu'auparavant cette fonctionnalité faisait l'objet d'un fichier séparé. Cette version est même déclarée comme « mobile-first ». Avec l'utilisation croissante d'appareils mobiles, le framework s'est adapté pour offrir une solution censée couvrir tous les besoins.

Contenu du kit

Bootstrap propose :

- Une mise en page basée sur une grille de 12 colonnes bien pratique. Bien sûr, si vous avez besoin de plus de 12 colonnes, ou de moins, il est toujours possible de changer la configuration ;
- L'utilisation de Normalize.css ;
- Du code fondé sur HTML5 et CSS3 ;
- Une bibliothèque totalement open source sous license MIT;
- Du code qui tient compte du format d'affichage des principaux outils de navigation (*responsive design*) : smartphones, tablettes… ;
- Des plugins jQuery de qualité ;
- Un résultat *cross-browser* (la prise en charge de IE7 a été abandonnée avec la version 3), donc une garantie de compatibilité maximale ;
- Une bonne documentation ;
- La garantie d'une évolution permanente ;

- Une mine de ressources variées sur le web ;
- Une architecture basée sur LESS, un outil bien pratique qui étend les possibilités de CSS (un portage sur SASS existe également).

Vous pouvez trouver toutes les informations sur Bootstrap directement sur le site dédié.

C'est quoi un`normalize.css`?

Les navigateurs n'adoptent pas tous les mêmes valeurs par défaut pour les styles des éléments HTML. Cela peut générer quelques surprises au rendu des pages web selon le navigateur utilisé. D'autre part certains navigateurs présentent des défauts de prise en compte de certains éléments.`Normalize`est un petit fichier CSS qui établit des règles pour avoir un rendu identique quel que soit le navigateur utilisé. Au lieu d'agir brutalement comme les reset CSS qui remettent toutes les valeurs à zéro,`normalize`agit intelligemment en conservant ce qui est utile et en jouant finement sur les éléments. Vous pouvez trouver le détail des règles appliquées avec leur explication ici.

Évolution de Bootstrap

Bootstrap est un framework très populaire qui évolue très rapidement avec l'arrivée fréquente de nouvelles versions. C'est à la fois un avantage (il s'améliore de plus en plus) et un inconvénient (le code qu'on a écrit pour une mise en page devient rapidement obsolète pour les nouvelles versions). Le passage à la version 3 a été une petite révolution avec de très nombreux changements, en particulier une refonte complète de la grille. Un site écrit avec la version 2 doit être totalement réécrit pour cette nouvelle version, mais ce n'est évidemment pas une obligation. L'évolution du framework s'est faite essentiellement en direction des appareils nomades qui constituent peu à peu le parc le plus important d'appareils pour surfer sur Internet.

La principale source d'information pour connaître les changements des nouvelles versions est le blog officiel. Il est aussi intéressant de s'informer en amont sur la page GitHub du projet pour connaître les demandes des utilisateurs (Pull Requests) et les problèmes rencontrés (Issues).

La prochaine version majeure sera la version 4 avec de nombreuses améliorations en vue.

Ce cours a été mis à jour pour la version 3.3 et tous les renseignements et exemples donnés ont été testés sur cette version.

Installation

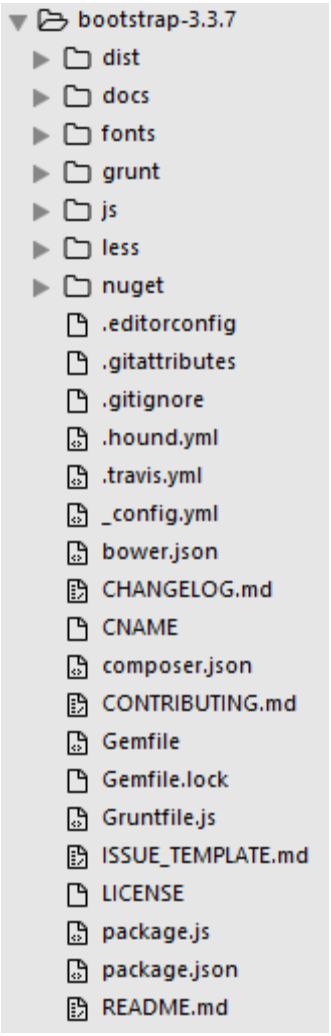
Installation de Bootstrap

L'installation de Bootstrap est simple : cliquez sur le bouton de téléchargement sur le site du framework. Vous avez à disposition trois boutons :

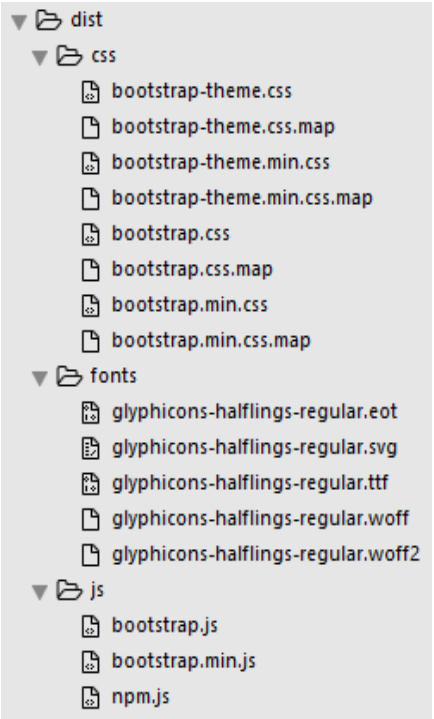
- "Download Bootstrap" : permet de récupérer juste les fichiers nécessaires au fonctionnement de Bootstrap.
- "Download source" : permet de récupérer en plus tous les fichiers sources.
- "Download Sass" : c'est un portage de Bootstrap en Sass pour les utilisateurs de projets qui utilisent Sass (Rails, Compass...).

Vous pouvez aussi aller consulter directement le code source sur github, ou même l'installer avec Bower si vous utilisez`node`.

Quand vous téléchargez la librairie avec le bouton "Download source", vous obtenez un fichier zippé contenant un répertoire bootstrap-3.3.7 qui contient lui-même un certain nombre de fichiers et de dossiers, comme le montre la figure suivante.



Les fichiers utiles pour simplement utiliser Bootstrap se situent dans le dossier`dist`(« distribution »), ce sont les seuls fichiers que vous obtenez si vous utilisez le bouton "Download Bootstrap".



Le dossier « dist » contient les fichiers utiles à l'utilisation de Bootstrap
Voyons un peu les principaux fichiers :

- `bootstrap.css`comporte les classes de base de Bootstrap ;
- `bootstrap.min.css`comporte les mêmes classes de base que`bootstrap.css`mais est minifié ;
- `bootstrap-theme.css`contient des règles de styles particulières pour créer un thème spécifique pour Bootstrap ;
- `bootstrap-theme.min.css`est identique à`bootstrap-theme.css`mais est minifié ;
- `glyphicons-halflings-regular.svg`comporte la collection d'icônes au format svg ;
- `glyphicons-halflings-regular.ttf`comporte la collection d'icônes au format ttf ;
- `glyphicons-halflings-regular.woff`comporte la collection d'icônes au format woff ;
- `glyphicons-halflings-regular.eot`comporte la collection d'icônes au format eot ;
- `bootstrap.js`contient le code JavaScript des composants de Bootstrap ;
- `bootstrap.min.js`contient le même code JavaScript mais est minifié ;

Les fichiers`min`(`bootstrap.min.css`,`bootstrap.min.js`et`bootstrap-theme.min.css`) contiennent le même code que leurs équivalents (`bootstrap.css`,`bootstrap.js`et`bootstrap-theme.css`) mais ont été épurés des commentaires et compressés pour les alléger et accélérer ainsi leur chargement.

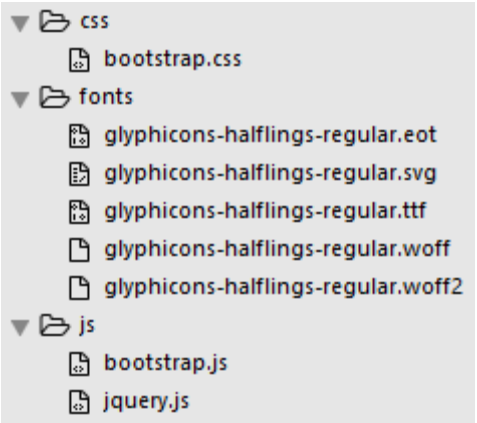
Les fichiers **bootstrap-theme.css.map** et **bootstrap.css.map** permettent de retrouver l'emplacement original d'une ligne de code à partir du code minifié. Cette fonctionnalité est utilisable avec les dernières versions de Chrome et Firefox. Ces fichiers ne sont pas indispensables au fonctionnement de Bootstrap.

D'accord, mais que mettre sur un site pour que Bootstrap fonctionne ?

Il faut distinguer deux situations :

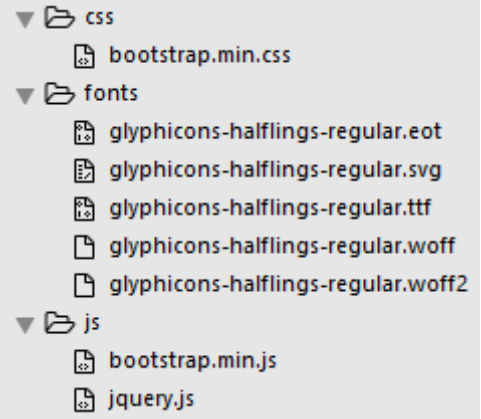
1. Pendant la phase de développement, il est intéressant de pouvoir naviguer dans le code de Bootstrap, il faut donc mettre les fichiers non compressés.
2. Lorsque votre site est en ligne, vous n'avez plus à vous inquiéter du code et seule compte la vitesse de chargement, il faut donc cette fois utiliser les versions compressées (`min`).

La figure suivante montre les fichiers utiles pendant la phase de développement (les fichiers`bootstrap.js`et`jquery.js`ne sont à prévoir que si vous utilisez des plugins jQuery et le dossier`fonts`uniquement si vous utilisez les icônes).



Les fichiers utiles durant le développement

La figure suivante montre les fichiers utiles lorsque le site est en ligne (les fichiers`bootstrap.min.js` et`jquery.js` ne sont à prévoir que si vous utilisez des plugins jQuery et le dossier`fonts` uniquement si vous utilisez les icônes).



Les fichiers utiles en production

Dans les exemples de ce cours, je pars du principe qu'un répertoire`bootstrap` a été créé à la racine du site avec les trois répertoires`css`,`fonts` et`js` comportant les fichiers cités précédemment.

Pour que Bootstrap fonctionne il faut le déclarer dans les pages HTML, qui doivent être impérativement au format HTML 5, il faut donc prévoir le bon DOCTYPE :

```
<!DOCTYPE html>

    …
```

Il faut ensuite déclarer au minimum le fichier`bootstrap.min.css` (ou`bootstrap.css`) dans l'en-tête de la page web :

```
<head>

        …

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

</head>
```

À partir de là toutes les classes sont accessibles… Évidemment vous devez adapter le lien selon la localisation de vos fichiers.

Si vous utilisez des composants JavaScript, vous devez également référencer la librairie de Bootstrap ainsi que`jQuery` (la librairie jQuery ne fait pas partie des fichiers téléchargés avec Bootstrap et doit être récupérée indépendamment sur http://jquery.com/) :

```
<script src="bootstrap/js/jquery.js"></script>

<script src="bootstrap/js/bootstrap.min.js"></script>
```

Et le fichier de thème ?

Ce fichier est un exemple d'amélioration de l'esthétique de Bootstrap, il n'est pas utile pour son fonctionnement. Si vous observez l'apparence des composants de Bootstrap, vous allez constater qu'ils sont « plats », sans aucun relief. Dans la version 2, ce n'était pas le cas. Pour des raisons de

performances, il a été décidé d'offrir cette amélioration visuelle dans un fichier séparé. Si vous avez la curiosité de regarder ce fichier, vous verrez qu'il contient des définitions de dégradés et d'ombrages. Vous trouvez le résultat de l'application de ce thème sur cette page d'exemple. Nous reparlerons de ce fichier dans le chapitre concernant la configuration de Bootstrap.

Les CDN

CDN est l'acronyme de « *Content delivery network* » ; c'est un réseau de serveurs qui met à disposition des librairies. Il devient ainsi inutile de prévoir ces librairies sur son propre serveur, il suffit de « pointer » vers eux. Il y a des avantages à utiliser un CDN :

- Libération de ressources et de la bande passante sur son propre serveur ;
- Parallélisation des téléchargements (un CDN est sur plusieurs serveurs) ;
- Accélération du chargement ;
- Diminution de la latence ;
- Actualisation automatique des librairies ;
- Amélioration du référencement…

Y-a-t-il des inconvénients ?

D'après certains, l'utilisation d'un CDN, qui impose une requête DNS supplémentaire, ne serait judicieuse que si l'on a beaucoup de librairies à charger. D'autre part si vous modifiez le fichier CSS de Bootstrap pour l'adapter à vos besoins, vous ne pourrez plus bénéficier des avantages d'un CDN. Mais si vous voulez profiter de cette possibilité pour Bootstrap, utilisez bootstrapcdn. Vous pouvez évidemment utiliser un CDN également pour jQuery, par exemple chez Google.

Si vous utilisez des CDN, l'installation en est d'autant simplifiée. Vous n'avez qu'à adapter les appels des librairies :

```
<link   <link   href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">


…


<script src="https://code.jquery.com/jquery-1.12.4.min.js"></script>


<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
```

Une trame pour démarrer

Le template de base

Vous trouvez aussi sur le site de Bootstrap un template de base et des exemples comportant l'essentiel des éléments pour vous aider à initier un projet. Pour le moment vous n'allez pas encore comprendre l'utilité de tous ces éléments, mais vous y reviendrez par la suite lorsque vous aurez fait le tour de ce cours.

Dans le template de base, vous trouvez en premier cette ligne :

```
<meta http-equiv="X-UA-Compatible" content="IE=edge">
```

Cette déclaration ne concerne que le navigateur Internet Explorer. Elle permet de s'assurer qu'il utilise la dernière version du moteur de rendu. Notez que cette ligne ne passe pas la validation W3C.

On trouve ensuite cette ligne :

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```
Cette ligne concerne uniquement les mobiles. On demande que l'affichage occupe tout l'espace disponible avec une taille de 1, autrement dit sans zoom. Vous pouvez aller encore plus loin en interdisant le zoom ou en le limitant à certaines valeurs.

On trouve ensuite la déclaration du fichier CSS dans sa version minifiée :

```
<link href="css/bootstrap.min.css" rel="stylesheet">
```
On trouve ensuite ces déclarations :

```
<!-- HTML5 Shim and Respond.js IE8 support of HTML5 elements and media queries -->

<!-- WARNING: Respond.js doesn't work if you view the page via file:// -->

<!--[if lt IE 9]>

  <script src="https://oss.maxcdn.com/html5shiv/3.7.3/html5shiv.min.js"></script>

  <script src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></script>

<![endif]-->
```
Le but de ces appels est de permettre aux navigateurs ne prenant pas en charge HTML5 et les Media Queries CSS3 de le faire ; ils visent essentiellement IE8. Pour obtenir plus de renseignements sur`respond.js` vous pouvez aller sur la page GitHub. Méfiez-vous en particulier si vous utilisez un CDN pour charger vos styles CSS, parce que`respond.js` utilise AJAX et vous pouvez buter sur le problème d'accès à plusieurs domaines.

Le but du fichier`html5shiv` est de créer les éléments (avec createElement) de type bloc du HTML 5 (header, section, aside…) qui sont ignorés par IE8.

En fin de page (pour ne pas ralentir le chargement) se trouvent les appels JavaScript (utiles si vous utilisez des plugins jQuery, comme nous allons le voir dans ce cours) :

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>

<script src="js/bootstrap.min.js"></script>
```
En résumé on obtient ce code (en simplifiant les références et en supposant que tous les fichiers sont situés dans un dossier`bootstrap`et des sous-dossiers fonctionnels) :

```
<!DOCTYPE html>

<html lang="fr">

  <head>

    <title>Bootstrap template</title>

    <meta charset="utf-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1">

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

    <!-- HTML5 Shim and Respond.js IE8 support of HTML5 elements and media queries -->

    <!-- WARNING: Respond.js doesn't work if you view the page via file:// -->

    <!--[if lt IE 9]>

      <script src="https://oss.maxcdn.com/html5shiv/3.7.3/html5shiv.min.js"></script>

      <script src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></script>

    <![endif]-->

  </head>

  <body>

    <h1>Hello, world!</h1>

    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>

    <script src="bootstrap/js/bootstrap.min.js"></script>

  </body>

</html>
```

Un template de démarrage

Vous pouvez trouver également sur le site 18 templates d'exemple. Les éléments nécessaires à la compréhension de ces exemples seront exposés tout au long de ce cours. Voyons toutefois le premier exemple, qui est le plus simple :

```
<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="utf-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1">
```

```
    <!-- The above 3 meta tags *must* come first in the head; any other head content must
come *after* these tags -->

    <meta name="description" content="">

    <meta name="author" content="">

    <link rel="icon" href="../../favicon.ico">



    <title>Starter Template for Bootstrap</title>



    <!-- Bootstrap core CSS -->

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



    <!-- IE10 viewport hack for Surface/desktop Windows 8 bug -->

    <link href="../../assets/css/ie10-viewport-bug-workaround.css" rel="stylesheet">



    <!-- Custom styles for this template -->

    <link href="starter-template.css" rel="stylesheet">



    <!-- Just for debugging purposes. Don't actually copy these 2 lines! -->

    <!--[if      lt      IE      9]><script      src="../../assets/js/ie8-responsive-file-
warning.js"></script><![endif]-->

    <script src="../../assets/js/ie-emulation-modes-warning.js"></script>



    <!-- HTML5 shim and Respond.js for IE8 support of HTML5 elements and media queries -->

    <!--[if lt IE 9]>

      <script src="https://oss.maxcdn.com/html5shiv/3.7.3/html5shiv.min.js"></script>
```

```
      <script src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></script>

    <![endif]-->

  </head>



  <body>



    <nav class="navbar navbar-inverse navbar-fixed-top">

      <div class="container">

        <div class="navbar-header">

          <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-
target="#navbar" aria-expanded="false" aria-controls="navbar">

            <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" href="#">Project name</a>

        </div>

        <div id="navbar" class="collapse navbar-collapse">

          <ul class="nav navbar-nav">

            <li class="active"><a href="#">Home</a></li>

            <li><a href="#about">About</a></li>

            <li><a href="#contact">Contact</a></li>

          </ul>

        </div><!--/.nav-collapse -->
```

```
      </div>

    </nav>



    <div class="container">



      <div class="starter-template">

        <h1>Bootstrap starter template</h1>

        <p class="lead">Use this document as a way to quickly start any new project.<br> All
you get is this text and a mostly barebones HTML document.</p>

      </div>



    </div><!-- /.container -->







    <!-- Bootstrap core JavaScript

    ================================================== -->

    <!-- Placed at the end of the document so the pages load faster -->

    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>

    <script>window.jQuery                    ||                    document.write('<script
src="../../assets/js/vendor/jquery.min.js"><\/script>')</script>

    <script src="../../dist/js/bootstrap.min.js"></script>

    <!-- IE10 viewport hack for Surface/desktop Windows 8 bug -->

    <script src="../../assets/js/ie10-viewport-bug-workaround.js"></script>

  </body>

</html>
```

On retrouve ce qu'on a vu précédemment avec la ligne pour les mobiles et l'appel au fichier CSS de Bootstrap. On trouve la référence d'un autre fichier CSS :`starter-template.css`. Voyons ce que contient ce fichier :

```
body {

  padding-top: 50px;

}

.starter-template {

  padding: 40px 15px;

  text-align: center;

}
```

Nous verrons, lorsque nous parlerons du composant "Barre de navigation" de Bootstrap, qu'il faut parfois réserver un espace en haut de la page (avec`padding-top: 50px`) pour éviter que le texte du début ne se retrouve sous la barre. On trouve aussi ici la classe`starter-template` qui se contente de fixer un espace interne et un alignement centré du texte. Cette classe ne sert que pour le texte de présentation du template et peut être supprimée ou modifiée selon votre convenance.

On trouve aussi dans la page une trame de la barre de navigation (je vous explique tout ça en détail dans ce cours) :

```
<nav class="navbar navbar-inverse navbar-fixed-top">

  <div class="container">

    <div class="navbar-header">

      <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-
target="#navbar" aria-expanded="false" aria-controls="navbar">

        <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" href="#">Project name</a>

    </div>
```

```
    <div id="navbar" class="collapse navbar-collapse">

      <ul class="nav navbar-nav">

        <li class="active"><a href="#">Home</a></li>

        <li><a href="#about">About</a></li>

        <li><a href="#contact">Contact</a></li>

      </ul>

    </div><!--/.nav-collapse -->

  </div>

</nav>
```

Le contenu de la page est ensuite inséré dans une DIV comportant la classe`container`:

```
<div class="container">

  <div class="starter-template">

    <h1>Bootstrap starter template</h1>

    <p class="lead">Use this document as a way to quickly start any new project.<br> All you
get is this text and a mostly barebones HTML document.</p>

  </div>

</div><!-- /.container -->
```

Nous verrons également l'intérêt de cette classe et son utilisation. Considérez les exemples de templates fournis sur le site de Bootstrap à la fois comme des guides de démarrage et des aides à la compréhension du framework. Évitez de les prendre tels quels sans en comprendre tous les éléments. À l'issue de ce cours vous aurez tout en main pour le faire…

Les Media Queries

Les Media Queries sont destinées à simplifier la création de pages web pour les rendre consultables sur des supports variés (tablettes, smartphones…). Cette section n'est qu'une introduction rapide à ce domaine qui mériterait un cours à lui tout seul.

Avec les Media Queries on peut cibler :

- La résolution ;
- Le type de media ;
- La taille de la fenêtre (width et hight) ;
- La taille de l'écran (device-width et device-height);
- Le nombre de couleurs ;
- Le ratio de la fenêtre (par exemple le 16/9) ;

- etc.

Les plus courageux peuvent consulter directement les spécifications du W3C.
La liste est longue, malheureusement peu de navigateurs actuels sont capables de digérer tout ça. On se limite en général à la taille de l'affichage, l'orientation et parfois le ratio.

Je ne comprends pas la différence entre taille de l'écran et taille de la fenêtre !

Ah ! Bonne réflexion. Cette distinction n'a aucun sens pour les écrans d'ordinateurs mais trouve tout son intérêt pour les smartphones. La taille de l'écran est la surface physique réelle alors que la taille de la fenêtre est celle dont le smartphone "pense" disposer. Par exemple pour un iPhone 4 la surface réelle est de 640x960 px et la surface de la fenêtre est de 320x480 px. Autrement dit un iPhone est pessimiste sur ses possibilités d'affichage !

Mais ce n'est pas le seul souci ! Le navigateur embarqué sur un smartphone a lui aussi une certaine idée de la surface d'affichage dont il dispose (le viewport). Par exemple, pour continuer avec l'exemple de l'iPhone 4, qui utilise Safari, ce dernier pense disposer d'une largeur de 980 px ! Sur ces appareils il est aussi possible d'utiliser le zoom et notre mise en page risque de s'en ressentir. On peut se demander comment s'en sortir dans tout ce bazar... Heureusement on dispose de la balise`META viewport` pour fixer certaines valeurs, voici ce qui est préconisé dans le template de Bootstrap :

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

- On ouvre la fenêtre à la largeur de l'écran :`width=device-width`
- On règle le zoom :`initial-scale=1`

Vous pouvez aussi fixer d'autres valeurs. Par exemple, empêcher l'utilisateur de zoomer. Les principaux navigateurs permettent de tester le rendu sur mobile. Par exemple avec Firefox il faut utiliser la commande "Vue adaptative" dans le menu "Développement web".

Syntaxe des Media Queries

Une media query est une expression dont la valeur est soit vraie, soit fausse. Voici un exemple dans du code CSS :

```
@media (max-width: 767px) {

  …

}
```

`@media` est une règle apparue avec le CSS2. Elle permet à la base de cibler un media : écran, imprimante, projecteur… Cette règle est reprise et étendue avec le CSS3. L'expression située entre parenthèses est évaluée, et si elle est « vraie », les règles situées dans le bloc sont prises en compte.

Ici l'expression est `(max-width: 767px)`, elle se comprend facilement. Elle est « vraie » si le support de visualisation de la page a une largeur d'affichage inférieure ou égale à 767px.

Oui, mais on n'a pas spécifié de média là !

Oui parce qu'on veut prendre en compte tous les médias existants ! Voici une syntaxe équivalente :

```
@media all and (max-width: 767px) { … }
```

On en profite pour voir qu'on peut combiner des éléments avec des opérateurs logiques.

Voici une autre expression avec le même opérateur logique :

```
@media (min-width: 768px) and (max-width: 979px) { … }
```

Ici, on veut prendre en compte les règles du bloc si l'affichage se situe entre 768px et 979px.

Si on veut appliquer des règles lorsque l'orientation est « portrait », voici la syntaxe à utiliser :

```
@media (orientation: portrait) { … }
```

Comme vous pouvez le voir, la syntaxe est simple, ce qui l'est moins ce sont les règles à définir. Mais quand vous utilisez Bootstrap, vous n'avez pas à vous soucier de tout ça… Si vous explorez le code de Bootstrap, vous trouverez beaucoup d'utilisation des Media Queries, comme par exemple ici, où est définie une règle de dimension maximale pour la classe `container` lorsque l'affichage est d'au moins 992 pixels :

```
@media (min-width: 992px) {

  .container {

    max-width: 970px;

  }

   …

}
```

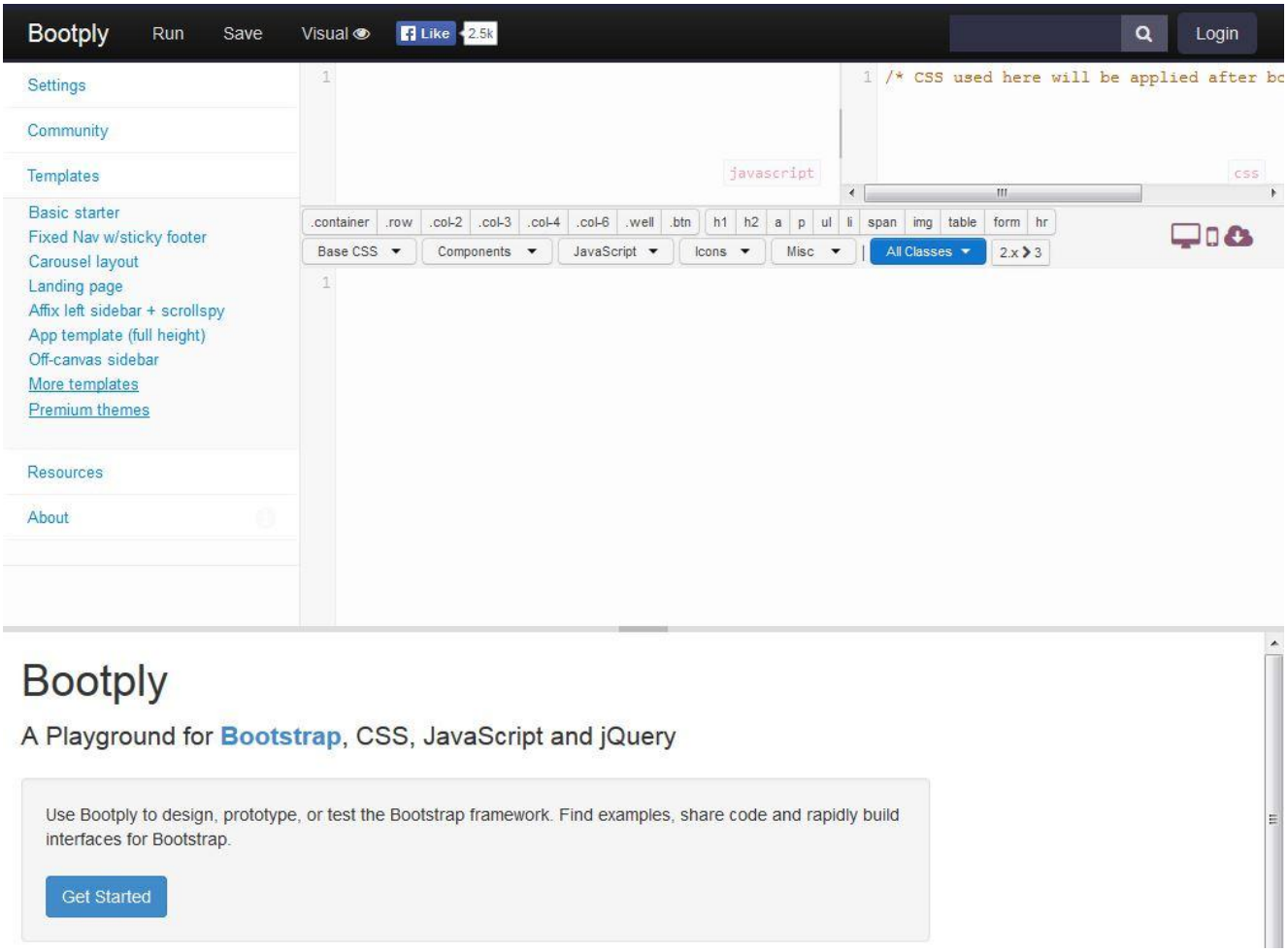
Nous verrons plus loin dans ce cours que Bootstrap définit quatre types de medias selon la dimension de l'affichage. Voici la syntaxe utilisée :

```
/* Extra small devices (phones, up to 480px) */

/* No media query since this is the default in Bootstrap */



/* Small devices (tablets, 768px and up) */

@media (min-width: @screen-tablet) { ... }



/* Medium devices (tablets, 992px and up) */

@media (min-width: @screen-desktop) { ... }



/* Large devices (large desktops, 1200px and up) */

@media (min-width: @screen-large-desktop) { ... }
```

Les ressources

Bootstrap est un framework très populaire et les ressources correspondantes sont très nombreuses. Dans ce cours je vais présenter très peu de ces ressources pour me limiter à celles qui me semblent les plus importantes et en évitant de parler des extensions existantes qui mériteraient un chapitre entier.

Parmi tout ce qui est disponible il y a le site Bootply qui se démarque par son utilité. Je vous en parle dès cette introduction parce que c'est autant un outil d'apprentissage que de développement. La page de création se présente ainsi :



La page de création de Bootply
Il y a différentes zones de saisie : Javascript, css... Des assitants permettent de générer automatiquement du code : un simple élement ou un thème complet.

Vous pouvez l'utiliser comme boîte à outil pour tester les exemples de ce cours.

Pour toutes les autres ressources je vous renvoie à cet excellent site.

En résumé

- Bootstrap est un framework CSS qui comporte en plus des plugins jQuery pour composer des pages web.
- Bootstrap est un framework récent qui a connu un développement et une popularité très rapide.

- Bootstrap s'installe facilement en référençant quelques fichiers sur son serveur ou même en passant directement par des CDN.
- Bootstrap propose des templates de démarrage pour éviter de partir avec une page blanche.
- Bootstrap intègre des medias queries pour adapter les pages web à tous les supports de visualisation.

Une grille ?

L'organisation spatiale des pages web est l'une des premières préoccupations lorsque l'on crée un site web. Prévoit-on une bannière ? Faut-il un espace pour un menu à gauche ou en haut ? Y aura-t-il des blocs sur un des côtés pour recevoir certaines fonctionnalités comme la connexion ou des infos ? Faut-il prévoir un bas de page ?

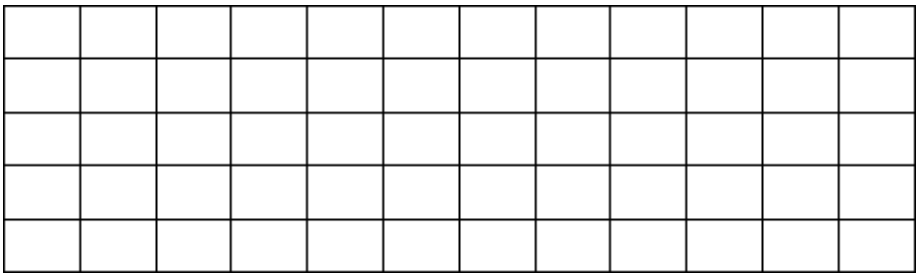
Bootstrap ne répond évidemment pas à ces questions, mais il peut grandement vous faciliter la tâche, avec son système de grille, pour obtenir le résultat que vous souhaitez.

Petite visite guidée…

Présentation

Le principe d'une grille

Une grille est tout simplement un découpage en cellules :



Une grille

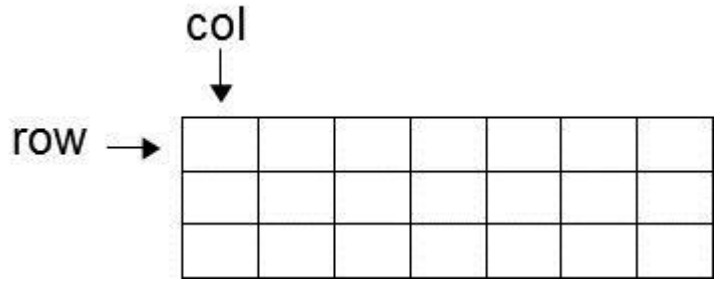
On peut alors décider d'organiser du contenu en utilisant pour chaque élément une ou plusieurs cellules :

On peut placer des éléments sur la grille

La grille de Bootstrap comporte 12 colonnes comme dans l'illustration précédente. Vous commencez sans doute à comprendre l'utilité de cette organisation.

Terminologie

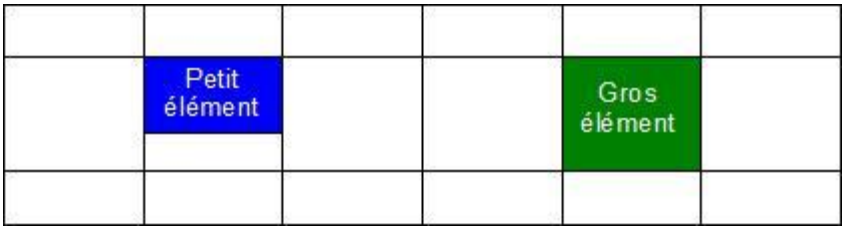
Une grille est découpée en rangées (appelées *row*, parce que tout est en anglais) et colonnes (*col*) :



Une grille est composée de rangées et de colonnes

La grille de Bootstrap

La grille de Bootstrap n'est pas aussi idéale que celle présentée précédemment. Le découpage en colonnes est tout simplement une division en pourcentage de la largeur de la fenêtre de visualisation et correspond donc à ce qu'on vient de voir. En revanche, il n'en est pas vraiment de même pour les rangées. Ces dernières ont la hauteur de leur contenu :



Les rangées ont la hauteur de leur contenu

Moralité, une rangée prend la hauteur du plus gros élément qu'elle contient. Puisque la largeur des colonnes est contrainte, le flux des données s'écoule verticalement, ce qui est un comportement HTML classique. Il faudra en tenir compte lors de la mise en page.

Organisation de la grille

Bootstrap est essentiellement un fichier CSS. Il comporte de nombreuses classes que l'on peut utiliser directement dans les balises HTML.

La première classe à connaître est `row`, qui représente une rangée. Cette classe établit des marges négatives à droite et à gauche :

```
.row {

  margin-right: -15px;

  margin-left: -15px;

}
```

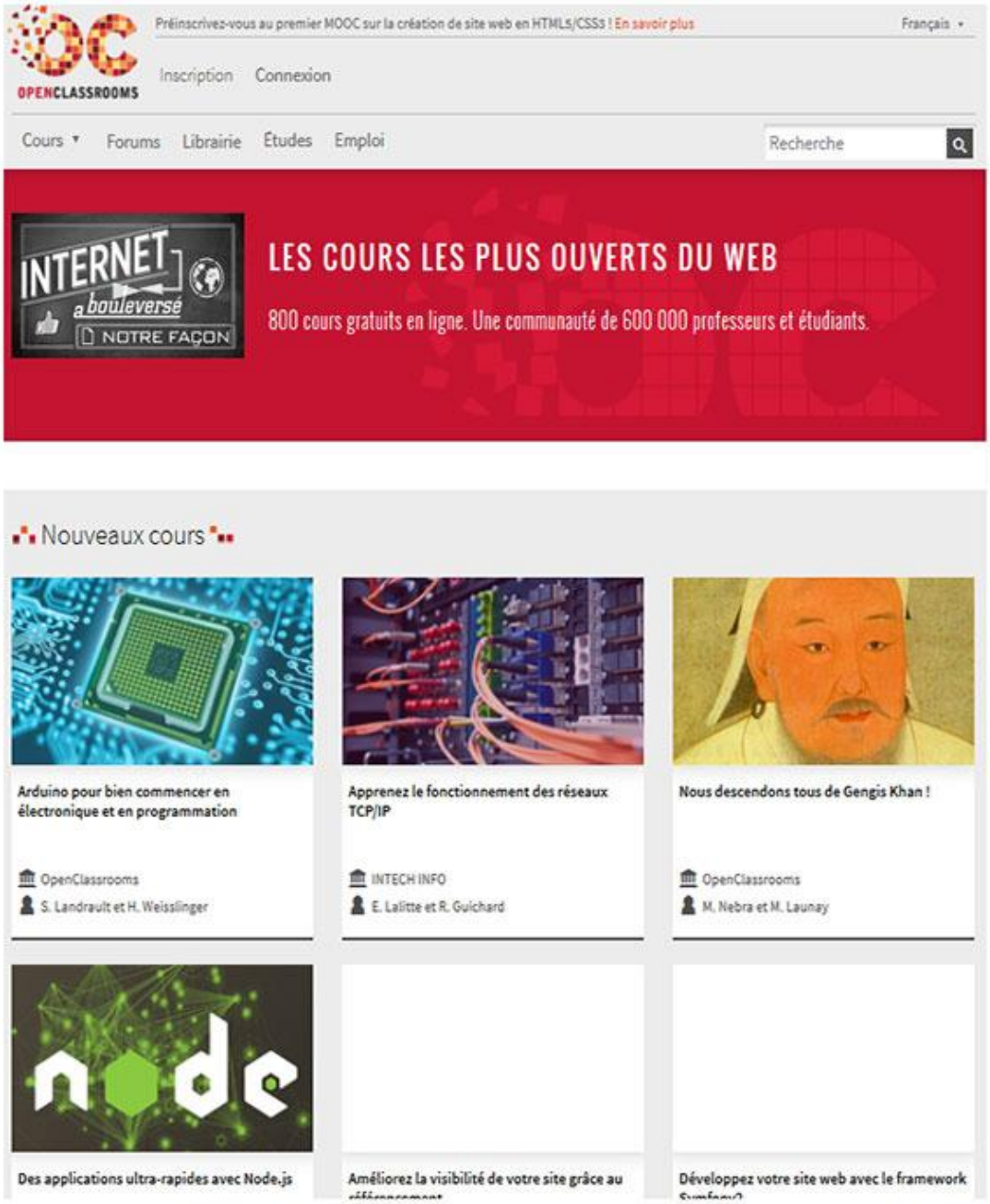
Il faut ensuite définir le nombre de colonnes pour chaque élément en sachant qu'il y en a au maximum 12. Pour définir le nombre de colonnes utilisées pour chaque élément, on dispose de quatre batteries de 12 classes :

- `col-xs-1` ou`col-sm-1` ou`col-md-1` ou`col-lg-1`
- `col-xs-2` ou`col-sm-2` ou`col-md-2` ou`col-lg-2`
- `...`
- `col-xs-12` ou`col-sm-12` ou`col-md-12` ou`col-lg-12`

Pourquoi 4 sortes de classes pour les colonnes ?

Je vous ai déjà dit que Bootstrap est « responsive », ce qui veut dire qu'il s'adapte à la taille de l'écran. Il permet une visualisation aussi bien sur un écran géant que sur un smartphone. Mais que se passe-t-il pour les éléments d'une page web lorsque la fenêtre diminue ou s'élargit ? On peut envisager deux hypothèses : les éléments se redimensionnent en restant positionnés, ou alors ils s'empilent quand la fenêtre devient plus étroite et se positionnent côte à côte quand elle s'élargit. Voici à la figure suivante l'exemple de la page d'accueil du site OpenClassrooms qui illustre ce phénomène avec trois situations (écran large, écran moyen et petit écran) :

Aspect sur grand écran



Aspect sur écran moyen

Aspect sur petit écran

Plutôt que de réduire les éléments au risque de les rendre illisibles, le choix a été fait de les empiler petit à petit quand la fenêtre devient plus étroite. Une autre option pourrait consister à supprimer des éléments pas vraiment utiles, mais nous reviendrons plus tard sur ce point. Pour le moment on va juste se demander comment on peut choisir entre les deux options présentées à la figure suivante.

Empilement ou réduction ?

C'est ici qu'interviennent les 4 sortes de classes vues précédemment pour les colonnes. Bootstrap considère 4 sortes de médias : les petits, genre smartphones (moins de 768 pixels), les moyens, genre tablettes (moins de 992 pixels), les écrans moyens (moins de 1200 pixels)

et enfin les grands écrans (plus de 1200 pixels). Vous trouverez à la figure suivante un tableau pour illustrer les différences de réaction selon la catégorie.

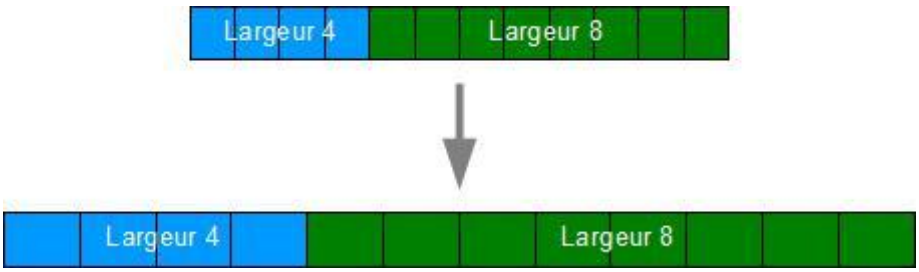
	Petit écran (smartphone)	**Écran réduit (tablette)**	**Écran moyen (desktop)**	**Grand écran (desktop)**
				
Comportement	Redimensionnement	Redimensionnement	Redimensionnement	Empilage puis redimensionnement
Classe	col-xs-*	col-sm-*	col-md-*	col-lg-*
Valeur de référence	< 768 px	>= 768 px	>= 992 px	>= 1200 px

Le nom des classes est intuitif :`xs` pour *x-small*, `sm` pour *small*, `md` pour *medium* et`lg` pour *large*.

Un petit exemple ?
Pour avoir un élément de 4 colonnes de large accouplé avec un élément de 8 colonnes de large sur un smartphone, on a :

```
<div class="row">

  <div class="col-xs-4">Largeur 4</div>

  <div class="col-xs-8">Largeur 8</div>

</div>
```

La figure suivante illustre ce code avec l'effet quand on passe sur un écran plus grand.

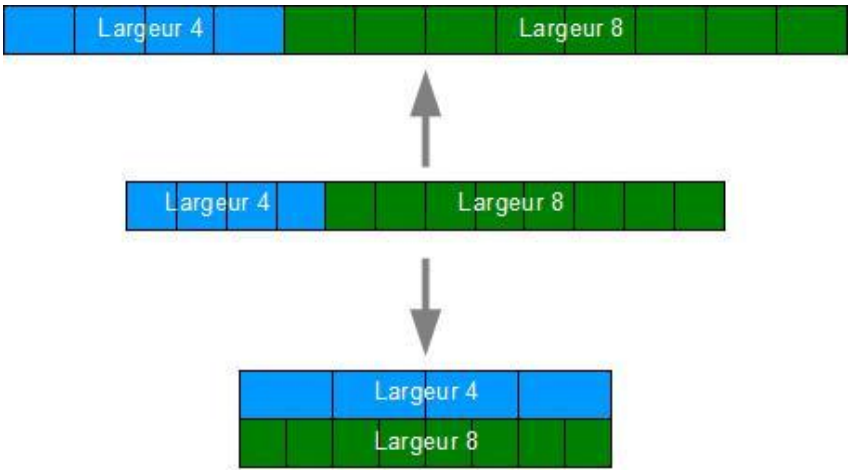


Un élément de 4 colonnes à côté d'un élément de 8 colonnes

Voici la version tablette :

```
<div class="row">

  <div class="col-sm-4">Largeur 4</div>

  <div class="col-sm-8">Largeur 8</div>

</div>
```

La figure suivante illustre ce code avec l'effet quand on passe sur un écran plus petit ou plus grand.

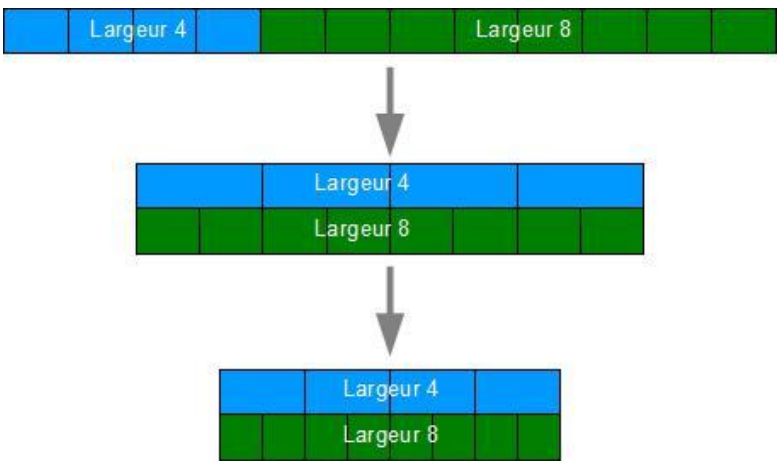


L'affichage change en fonction de la taille de l'écran

Vous remarquez que lors de l'empilage, les colonnes prennent toute la place disponible.
Et enfin, voici la version grand écran :

```
<div class="row">

  <div class="col-lg-4">Largeur 4</div>

  <div class="col-lg-8">Largeur 8</div>

</div>
```

La figure suivante illustre ce code avec l'effet quand on passe sur un écran de plus en plus petit.



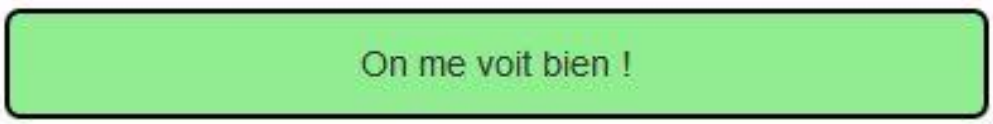
La version grand écran

La largeur des éléments de la grille est calculée en pourcentage selon la fenêtre de visualisation. Rien n'interdit évidemment de mélanger des classes des 3 catégories pour créer des effets particuliers avec certains éléments qui doivent s'empiler et d'autre pas (c'est même la stratégie utilisée pour la mise en page "responsive" comme nous allons bientôt le voir)…

Après cette petite mise en jambe pour vous présenter la notion de grille dans Bootstrap, on va passer en revue les possibilités dans le détail. Mais pour visualiser tous les exemples qui vont suivre, je vous propose d'ajouter un peu de style, histoire d'afficher les éléments de façon explicite. Pour éviter de polluer le code HTML on va mettre ce style spécifique dans un fichier CSS particulier (ce fichier sera nommé`tuto.css` dans le code) :

```
body {

  padding-top: 10px;

}

[class*="col-"], footer {

  background-color: lightgreen;

  border: 2px solid black;

  border-radius: 6px;

  line-height: 40px;

  text-align: center;

}
```

Le seul but de ces règles de style est de faire apparaître nettement les éléments à l'écran. Une petite marge interne en haut pour le corps, pour éviter que tout soit collé en haut de l'écran. Des bordures (avec des coins arrondis pour l'esthétique) et un fond coloré pour distinguer les éléments. Une hauteur fixée à 40 pixels, parce qu'étant donné que la hauteur des rangées dépend du contenu, on aurait un rendu hétéroclite. La figure suivant présente l'aspect obtenu.



Le résultat final

La grille en pratique

Le conteneur

La grille de Bootstrap doit être placée dans un conteneur. Bootstrap propose les classes`container` et`container-fluid`. Leur utilisation était auparavant optionnelle. Il est désormais clairement indiqué dans la documentation qu'il faut la mettre en œuvre systématiquement si on veut obtenir des alignements et des espacements corrects. La classe`container` contient et centre la grille sur une largeur fixe, qui s'adapte en fonction de la largeur de l'écran. La classe`container-fluid` permet à la grille d'occuper toute la largeur. Dans les exemples de ce chapitre, je vais utiliser systématiquement`container` pour avoir une visualisation plus facile des éléments. Ce conteneur a une largeur maximale selon le média concerné, comme indiqué au tableau suivant.

	Petit écran (smartphone)	**Ecran réduit (tablette)**	**Ecran moyen (desktop)**	**Grand écran (desktop)**
				
Conteneur maximum	Automatique	750 px	970 px	1170 px
Valeur de référence	< 768 px	>= 768 px	>= 992 px	>= 1200 px

Le centrage du conteneur est fait de façon classique avec de petites marges internes de 15 pixels et les marges droite et gauche automatiques :

```
.container {

  padding-right: 15px;

  padding-left: 15px;

  margin-right: auto;

  margin-left: auto;

}
```

Ce sont les mêmes règles pour la classe`container-fluid`. Ce qui est ajouté pour la classe`container` est la limite de largeur spécifiée par média :

```
@media (min-width: 768px) {

  .container {
```

```
    width: 750px;
  }
}
@media (min-width: 992px) {
  .container {
    width: 970px;
  }
}
@media (min-width: 1200px) {
  .container {
    width: 1170px;
  }
}
```

Remarquez l'ordre des medias, du plus étroit vers le plus large. De cette manière on a une surcharge des règles cohérente.

Une seule rangée

Dans ce premier exemple, on va déclarer une seule rangée avec deux éléments qui occupent tout l'espace :

```
<!DOCTYPE html>
<html>
  <head>
    <link href="assets/css/bootstrap.css" rel="stylesheet">
    <link href="assets/css/tuto.css" rel="stylesheet">
  </head>
  <body>
    <div class="container">
```

```
      <div class="row">

        <div class="col-lg-4">4 colonnes</div>

        <div class="col-lg-8">8 colonnes</div>

      </div>

    </div>

  </body>

</html>
```

Ce qui nous donne la figure suivante.

Une seule rangée avec deux éléments qui occupent tout l'espace

Mise en page rapide et simple non ?

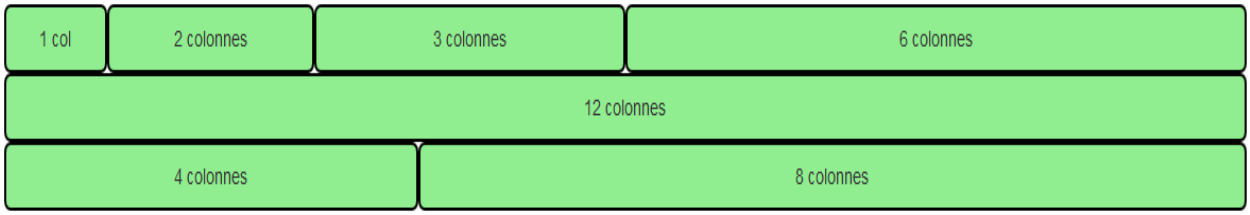
Plusieurs rangées

Pour obtenir plusieurs rangées, il suffit d'utiliser plusieurs fois la classe`row`. Un petit exemple avec 3 rangées pour voir l'effet :

```
<!DOCTYPE html>

<html>

  <head>

    <link href="assets/css/bootstrap.css" rel="stylesheet">

    <link href="assets/css/tuto.css" rel="stylesheet">

  </head>

  <body>

    <div class="container">

      <div class="row">

        <div class="col-lg-1">1 col</div>

        <div class="col-lg-2">2 colonnes</div>

        <div class="col-lg-3">3 colonnes</div>
```

```
        <div class="col-lg-6">6 colonnes</div>

      </div>

      <div class="row">

        <div class="col-lg-12">12 colonnes</div>

      </div>

      <div class="row">

        <div class="col-lg-4">4 colonnes</div>

        <div class="col-lg-8">8 colonnes</div>

      </div>

    </div>

  </body>

</html>
```

Ce qui nous donne la figure suivante.



Trois rangées

Sauter des colonnes

Bootstrap permet aussi de sauter des colonnes. Il y a quelques classes d'offset pour ça :

- col-*-offset-1
- col-*-offset-2
- ...
- col-*-offset-11

Ces classes se contentent d'ajouter une marge gauche. Par exemple :

```
.col-lg-offset-6 {

  margin-left: 50%;

}
```

Dans ce cas, 6 colonnes représentent la moitié de la fenêtre donc 50%. Voici un exemple :

```
<!DOCTYPE html>

<html>

  <head>

    <link href="assets/css/bootstrap.css" rel="stylesheet">

    <link href="assets/css/tuto.css" rel="stylesheet">

  </head>

  <body>

    <div class="container">

      <div class="row">

        <div class="col-lg-3">3 colonnes</div>

        <div class="col-lg-6">6 colonnes</div>

        <div class="col-lg-3">3 colonnes</div>

      </div>

      <div class="row">

        <div class="col-lg-3">3 colonnes</div>

        <div class="col-lg-offset-6 col-lg-3">3 colonnes</div>

      </div>

    </div>

  </body>

</html>
```

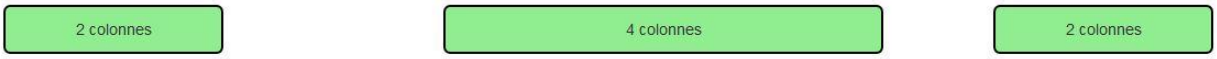
Voici le résultat à la figure suivante.

On a sauté des colonnes

On a sauté 6 colonnes grâce à la classe`col-lg-offset-6`. On peut évidemment multiplier les sauts :

```
<div class="container">

  <div class="row">

    <div class="col-lg-2 col-lg-offset-1">2 colonnes</div>

    <div class="col-lg-4 col-lg-offset-2">4 colonnes</div>

    <div class="col-lg-2 col-lg-offset-1">2 colonnes</div>

  </div>

</div>
```

Voici le résultat à la figure suivante.



On a sauté des colonnes plusieurs fois

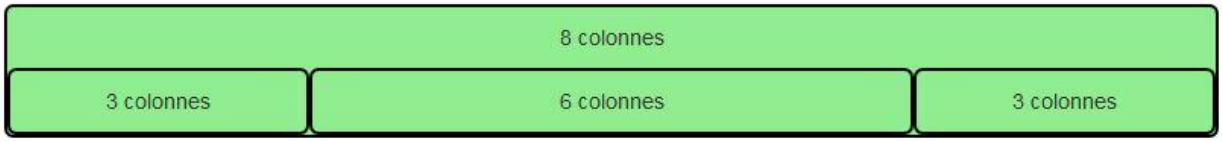
Imbrication d'éléments

On doit souvent imbriquer des éléments dans une page web, est-ce possible avec Bootstrap ? Autrement dit inclure un`row` dans un`col`. Eh bien, on va tester ça tout de suite :

```
<!DOCTYPE html>

<html>

  <head>

    <link href="assets/css/bootstrap.css" rel="stylesheet">

    <link href="assets/css/tuto.css" rel="stylesheet">

  </head>

  <body>

    <div class="container">

      <div class="row">

        <div class="col-lg-8">8 colonnes

          <div class="row">
```

```
            <div class="col-lg-3">3 colonnes</div>

            <div class="col-lg-6">6 colonnes</div>

            <div class="col-lg-3">3 colonnes</div>

          </div>

        </div>

      </div>

    </div>

  </body>

</html>
```

Voici le résultat à la figure suivante.



On a imbriqué des rangées

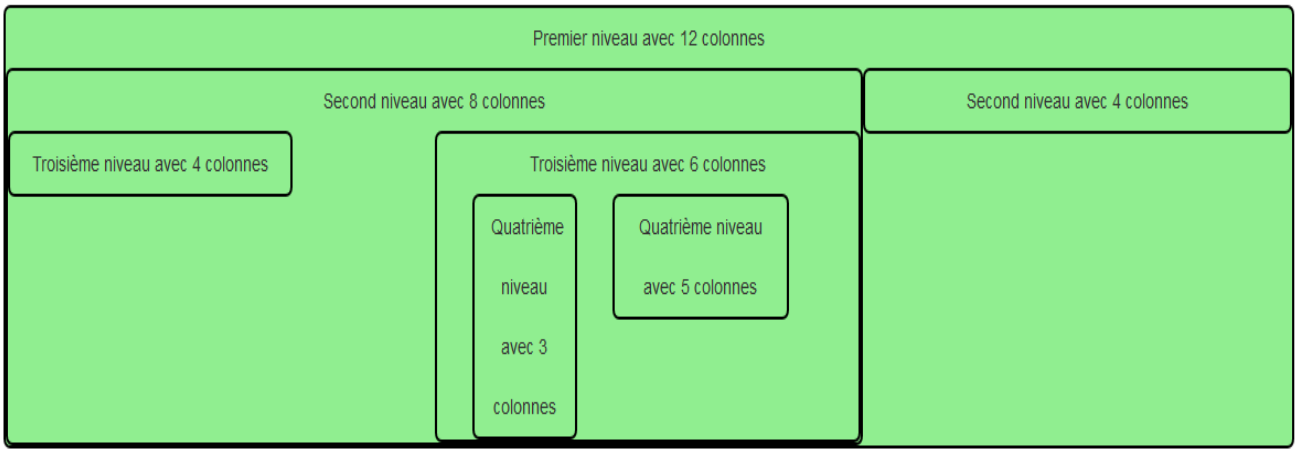
On dirait bien que ça fonctionne !

Les imbrications peuvent s'enchaîner pour répondre à des besoins complexes :

```
<div class="container">

  <div class="row">

    <div class="col-md-12">Premier niveau avec 12 colonnes

    <div class="row">

      <div class="col-md-8">Second niveau avec 8 colonnes

      <div class="row">

        <div class="col-md-4">Troisième niveau avec 4 colonnes</div>

        <div class="col-md-6 col-md-offset-2">Troisième niveau avec 6 colonnes

        <div class="row">

          <div class="col-md-3 col-md-offset-1">Quatrième niveau avec 3 colonnes</div>
```

```
          <div class="col-md-5 col-md-offset-1">Quatrième niveau avec 5 colonnes</div>

        </div>

        </div>

      </div>

      </div>

      <div class="col-md-4">Second niveau avec 4 colonnes</div>

    </div>

  </div>

</div>
```

Voici le résultat à la figure suivante.



Une imbrication complexe avec des sauts de colonnes

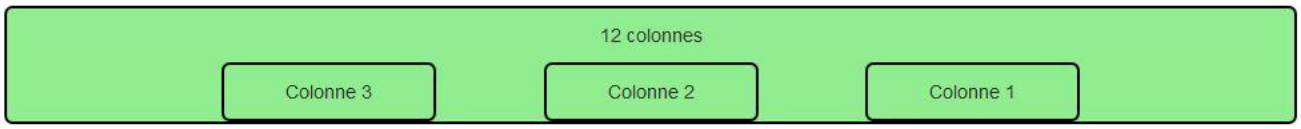
Ordre des colonnes

Très logiquement, les colonnes s'affichent dans l'ordre du flux. Il peut arriver qu'on veuille bouleverser cet ordre. Regardez cet exemple :

```
<!DOCTYPE html>

<html>

  <head>

    <link href="assets/css/bootstrap.css" rel="stylesheet">

    <link href="assets/css/tuto.css" rel="stylesheet">

  </head>
```

```
  <body>

    <div class="container">

      <div class="row">

        <div class="col-lg-12">12 colonnes

          <div class="row">

            <div class="col-lg-2 col-lg-push-8">Colonne 1</div>

            <div class="col-lg-2 col-lg-push-3">Colonne 2</div>

            <div class="col-lg-2 col-lg-pull-2">Colonne 3</div>

          </div>

        </div>

      </div>

    </div>

  </body>

</html>
```

Et voici le résultat à la figure suivante.

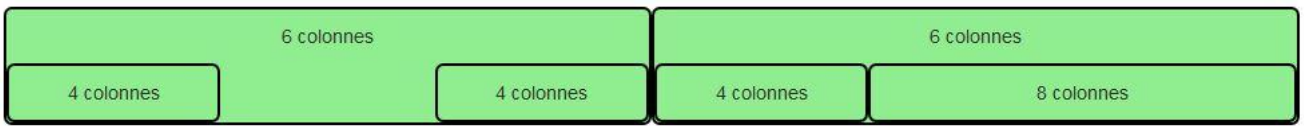


On peut choisir l'ordre des colonnes

La classe`col-lg-push-*` permet de décaler une colonne vers la droite et la classe`col-lg-pull-*` fait l'inverse.

Un petit TP

Pour vous entraîner, je vous propose d'obtenir le résultat visible à la figure suivante.



Essayez d'obtenir ce résultat

Donc deux zones séparées avec un offset de 4 colonnes dans la partie gauche.

Voici la solution :

```
<!DOCTYPE html>

<html>

  <head>

    <link href="assets/css/bootstrap.css" rel="stylesheet">

    <link href="assets/css/tuto.css" rel="stylesheet">

  </head>

  <body>

    <div class="container">

      <div class="row">

        <div class="col-lg-6">6 colonnes

          <div class="row">

            <div class="col-lg-4">4 colonnes</div>

            <div class="col-lg-offset-4 col-lg-4">4 colonnes</div>

          </div>

        </div>

        <div class="col-lg-6">6 colonnes

          <div class="row">

            <div class="col-lg-4">4 colonnes</div>

            <div class="col-lg-8">8 colonnes</div>

          </div>

        </div>

      </div>

    </div>
```

```
  </body>

</html>
```

Mise en page

L'intérêt principal d'une grille est de réaliser une mise en page. Nous allons voir quelques exemples pour structurer correctement une page.

Premier cas

Commençons par quelque chose de simple avec une en-tête, un menu à gauche, une section à droite et un pied de page :

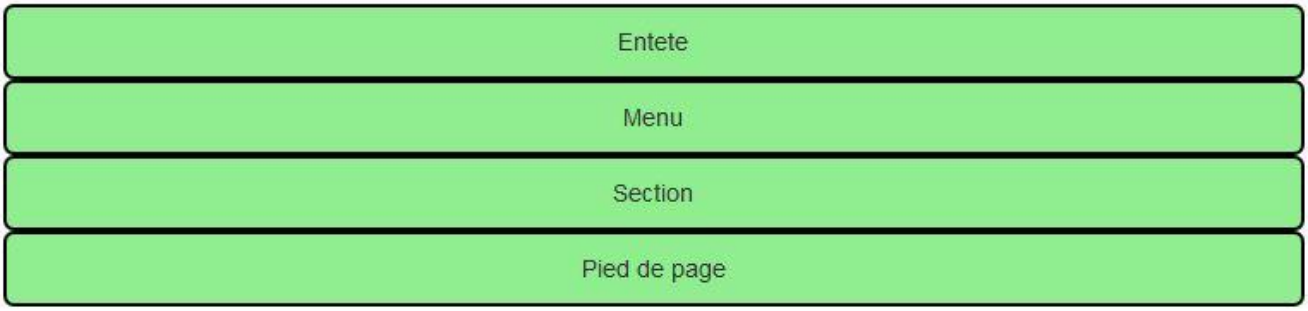
```
<!DOCTYPE html>

<html>

  <head>

    <link href="assets/css/bootstrap.css" rel="stylesheet">

    <link href="assets/css/tuto.css" rel="stylesheet">

  </head>

  <body>

    <div class="container">

      <header class="row">

        <div class="col-lg-12">

          Entete

        </div>

      </header>

      <div class="row">

        <nav class="col-lg-2">

          Menu

        </nav>

        <section class="col-lg-10">
```

```
          Section

        </section>

      </div>

      <footer class="row">

          Pied de page

      </footer>

    </div>

  </body>

</html>
```

Le résultat se trouve à la figure suivante.

Une mise en page simple

J'ai utilisé les classes pour grand écran avec les classe`col-lg-*`, ce qui fait qu'à la réduction, je me retrouve avec un empilage dès que je passe en dessous de 1200 pixels (voir figure suivante).



Empilage à la réduction de l'affichage

Si je veux que ma mise en forme reste stable aussi pour les écrans moyens, comment faire ? Il me suffit d'utiliser les classes`col-md-*` à la place de`col-lg-*`, ce qui a pour effet de déplacer la limite à 992 pixels. De la même façon, l'utilisation des classes`col-sm-*` déplace la limite à 768 pixels…

J'ai le même résultat sans utiliser la classe`col-lg-12`, est-elle vraiment utile ?
On obtient effectivement la même mise en page, mais les classes`col-*` ont un padding droite et gauche de 15 pixels, et si on n'utilise pas la classe`col-lg-12`, on va avoir une incohérence au niveau de l'affichage du contenu.

Second cas

Considérons maintenant un cas un peu plus riche :

```
<!DOCTYPE html>

<html>

  <head>

    <link href="assets/css/bootstrap.css" rel="stylesheet">

    <link href="assets/css/tuto.css" rel="stylesheet">

  </head>

  <body>

    <div class="container">

      <header class="row">

        <div class="col-sm-12">

          Entete

        </div>

      </header>

      <div class="row">

        <nav class="col-sm-2">

          Menu

        </nav>

        <section class="col-sm-10">

          Section

          <div class="row">

            <article class="col-md-10">

              Article

            </article>
```

```
            <aside class="col-md-2">

              Aside

            </aside>

          </div>

        </section>

      </div>

      <footer class="row">

        <div class="col-sm-12">

          Pied de page

        </div>

      </footer>

    </div>

  </body>

</html>
```

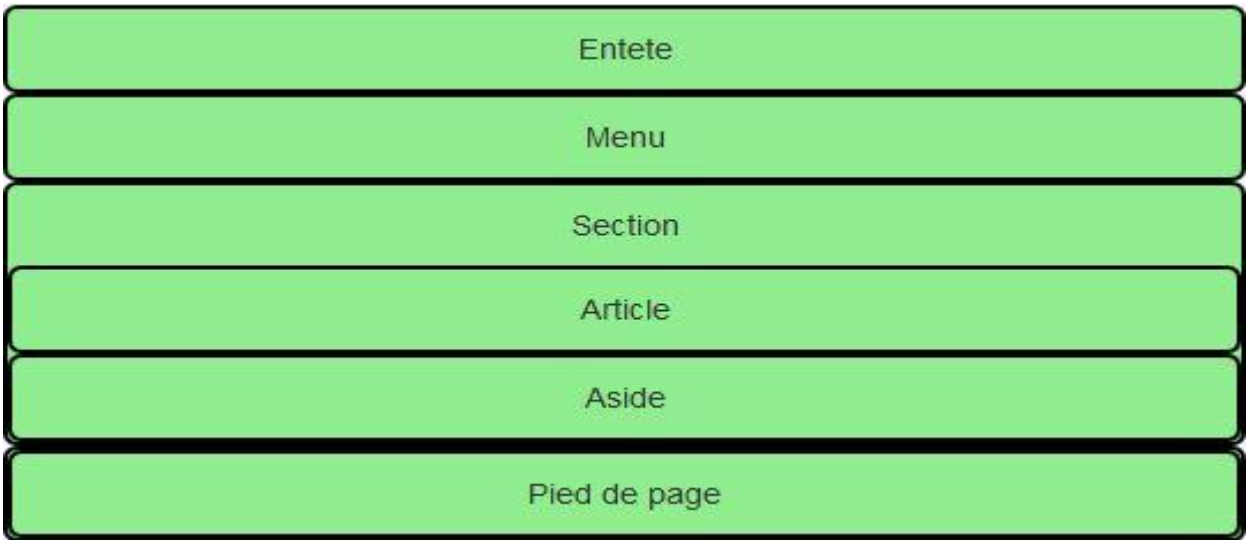
Le résultat se trouve à la figure suivante.

Une mise en page plus riche

On a conservé l'en-tête et le pied de page, ainsi que le menu, mais on a intégré dans le contenu un`article` et un`aside`. L'ensemble est prévu pour fonctionner sur desktop (grand et petit). Pour les tablettes, on se retrouve avec un empilage pour le contenu de la section (utilisation de classes`col-md-*` pour`article` et`aside`), comme à la figure suivante.

Résultat sur une tablette

Et comme on n'a rien prévu de spécial pour les smartphones, tout s'empile (voir figure suivante).



Résultat sur un smartphone

Troisième cas

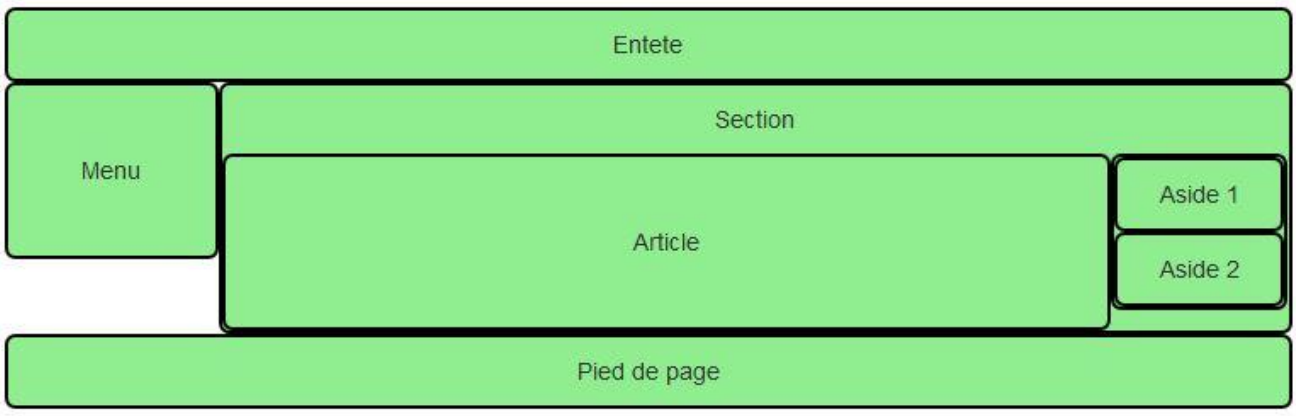
Enrichissons encore la mise en page :

```
<!DOCTYPE html>

<html>

  <head>

    <link href="assets/css/bootstrap.css" rel="stylesheet">

    <link href="assets/css/tuto.css" rel="stylesheet">

    <style type="text/css">

     /* Style pour l'exemple*/
```

```
    article.col-sm-10, nav.col-sm-2 {
      line-height: 100px;
    }
  </style>
</head>
<body>
  <div class="container">
    <header class="row">
      <div class="col-lg-12">
        Entete
      </div>
    </header>
    <div class="row">
      <nav class="col-sm-2">
        Menu
      </nav>
      <section class="col-sm-10">
        Section
        <div class="row">
          <article class="col-sm-10">
            Article
          </article>
          <div class="col-sm-2">
            <div class="row">
```

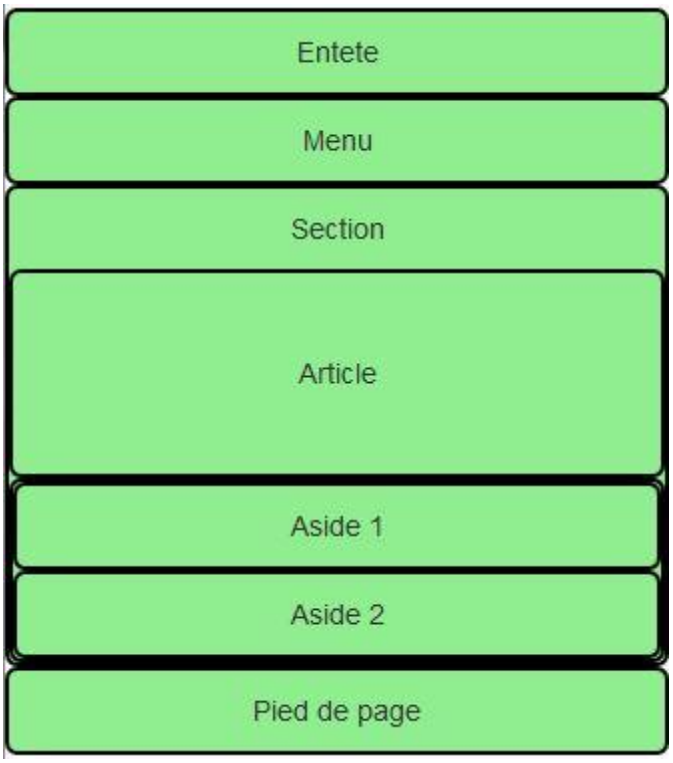
```
                <aside class="col-sm-12">

                  Aside 1

                </aside>

                <aside class="col-sm-12">

                  Aside 2

                </aside>

              </div>

            </div>

          </div>

        </section>

      </div>

      <footer class="row">

        <div class="col-lg-12">

          Pied de page

        </div>

      </footer>

    </div></body></html>
```

Le résultat se trouve à la figure suivante.



Une mise en page plus complexe

Maintenant nous avons 2`asides` à côté de l'article. Pour y arriver, il faut 2 niveaux d'imbrication. On a aussi un empilage complet à la réduction, comme le montre la figure suivante.



Résultat à la réduction

Pourquoi l'en-tête et le pied de page ont maintenant la classe`col-lg-12`, alors que dans l'exemple précédent on avait`col-sm-12`?
Comme la largeur est de 12 colonnes, tout ce qui nous intéresse est le centrage, que ces 2 classes possèdent en commun, sans influence du média utilisé. On peut donc utiliser indifféremment ces 2 classes dans notre cas.

Quatrième cas

Terminons avec un cas plus fourni :

```
<!DOCTYPE html>

<html>

  <head>

    <link href="assets/css/bootstrap.css" rel="stylesheet">

    <link href="assets/css/tuto.css" rel="stylesheet">

    <style type="text/css">

     /* Style pour l'exemple*/
```

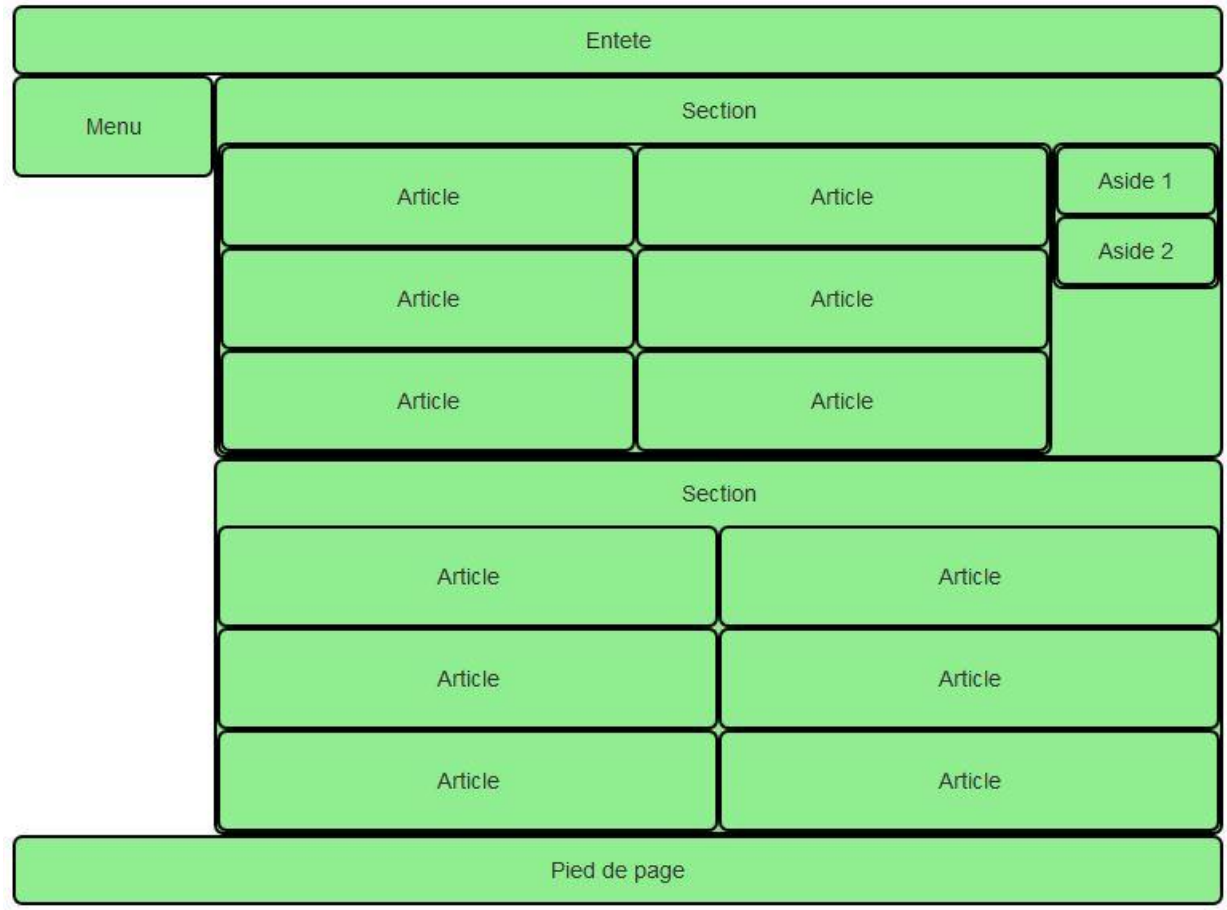
```
    footer {
      border: 0;
    }
    article.col-sm-6, nav.col-sm-2 {
      line-height: 60px;
    }
  </style>
</head>
<body>
  <div class="container">
    <header class="row">
      <div class="col-lg-12">
        Entete
      </div>
    </header>
    <div class="row">
      <nav class="col-sm-2">
        Menu
      </nav>
      <section class="col-sm-10">
        Section
        <div class="row">
          <div class="col-sm-10">
            <div class="row">
```

```
            <article class="col-sm-6">
              Article
            </article>
            <article class="col-sm-6">
              Article
            </article>
            <article class="col-sm-6">
              Article
            </article>
            <article class="col-sm-6">
              Article
            </article>
            <article class="col-sm-6">
              Article
            </article>
            <article class="col-sm-6">
              Article
            </article>
            <article class="col-sm-6">
              Article
            </article>
          </div>
        </div>
        <div class="col-sm-2">
          <div class="row">
            <aside class="col-sm-12">
              Aside 1
```

```
            </aside>
            <aside class="col-sm-12">
              Aside 2
            </aside>
          </div>
        </div>
      </div>
    </section>
    <section class="col-sm-offset-2 col-sm-10">
      Section
      <div class="row">
        <article class="col-sm-6">
          Article
        </article>
        <article class="col-sm-6">
          Article
        </article>
        <article class="col-sm-6">
          Article
        </article>
        <article class="col-sm-6">
          Article
        </article>
        <article class="col-sm-6">
```

```
            Article
          </article>
          <article class="col-sm-6">
            Article
          </article>
        </div>
      </section>
    </div>
    <footer class="row">
      <div class="col-lg-12">
        Pied de page
      </div>
    </footer>
  </div>
 </body>
</html>
```

Le résultat se trouve à la figure suivante.



Une mise en page très chargée

Dans les 2 sections, j'ai mis 6 articles d'une largeur de 6 colonnes qui se positionnent parfaitement. Il n'est pas utile de créer 3 rangées. Dans la deuxième section, je n'ai pas mis d'`aside` pour montrer un mélange de compositions. Le décalage de la deuxième section pour tenir compte de la largeur du menu se fait tout simplement avec une classe d'offset. En ce qui concerne le choix des classes pour les colonnes je me suis arrangé pour que la première section garde sa structure quand on rétrécit.

Vous avez pu voir avec ces quelques exemples qu'on peut obtenir facilement une structure de page aussi complexe que l'on veut. Le point le plus délicat demeure le choix des classes pour les colonnes en fonction du résultat que l'on désire selon les médias utilisés. Dans tous les cas, il faut tester les différents formats parce que le rendu ne donne pas forcément du premier coup ce que l'on avait prévu. Si vous n'y

parvenez pas avec une seule classe, alors la solution est d'en combiner plusieurs, c'est ce que nous allons voir bientôt…

En résumé

- Bootstrap propose une grille pour positionner tous les éléments des pages web.
- La grille est versatile et permet de nombreuses combinaisons comme des inclusions, des sauts de colonnes.
- La grille rend possible également une adaptation selon les dimensions du support de visualisation en réorganisant les éléments ou en masquant certains.

Un peu de pratique

Nous avons vu en détail les possibilités de la grille de Bootstrap. Nous avons vu également comment faire des mises en pages adaptées à différents types de supports. Il est temps maintenant de mettre en application toutes ces connaissances avec des applications pratiques.

Vous trouverez ici les images qui vous permettront de construire les exemples ci-dessous.

Combiner les formats et exemple de page

Nous allons maintenant utiliser ce que nous avons vu dans les chapitres précédents pour construire une page pratique.

Combinaison de classes « col-* »

Nous avons vu des mises en page utilisant sélectivement les classes pour les colonnes prévues pour les différents formats de supports. Nous allons à présent envisager leur combinaison pour gérer certains cas. Dans la page à réaliser, je veux avoir des petites photos côte à côte. Supposons que je parte de cette structure pour cette partie de la page :

```
<!DOCTYPE HTML>

<html>



  <head>

    <meta charset="utf-8">

    <link href="assets/css/bootstrap.css" rel="stylesheet">

    <style type="text/css">

      [class*="col"] { margin-bottom: 20px; }

      img { width: 100%; }

      body { margin-top: 10px; }
```

```
    </style>
  </head>

  <body>

    <div class="container">
      <section class="row">
        <div class="col-lg-2"><img src="images/t1.jpg" alt="Tigre"></div>
        <div class="col-lg-2"><img src="images/t2.jpg" alt="Tigre"></div>
        <div class="col-lg-2"><img src="images/t3.jpg" alt="Tigre"></div>
        <div class="col-lg-2"><img src="images/t4.jpg" alt="Tigre"></div>
        <div class="col-lg-2"><img src="images/t5.jpg" alt="Tigre"></div>
        <div class="col-lg-2"><img src="images/t6.jpg" alt="Tigre"></div>
        <div class="col-lg-2"><img src="images/t7.jpg" alt="Tigre"></div>
        <div class="col-lg-2"><img src="images/t8.jpg" alt="Tigre"></div>
        <div class="col-lg-2"><img src="images/t9.jpg" alt="Tigre"></div>
        <div class="col-lg-2"><img src="images/t10.jpg" alt="Tigre"></div>
        <div class="col-lg-2"><img src="images/t11.jpg" alt="Tigre"></div>
        <div class="col-lg-2"><img src="images/t12.jpg" alt="Tigre"></div>
      </section>
    </div>

  </body>
</html>
```

Le rendu sur grand écran est parfait avec 6 photos sur la largeur (voir figure suivante).

Affichage sur grand écran

Mais ça se gâte quand je rétrécis la fenêtre, puisque je sais qu'en dessous de 1200 pixels les éléments s'empilent. Du coup je me retrouve avec une image sur la largeur, et comme je les ai prévues en basse résolution, elle pixellise, comme à la figure suivante.

Affichage au-dessous de 1200 pixels

L'idéal serait d'avoir :

- Sur moyen et grand écran : 6 images sur la largeur.
- Sur tablette : 4 images sur la largeur.
- Sur smartphone : 3 images sur la largeur.

Comment réaliser cela ? Tout simplement en combinant les classes `col-*` :

```
<div class="container">

  <section class="row">

    <div class="col-xs-4 col-sm-3 col-md-2"><img src="images/t1.jpg" alt="Tigre"></div>

    <div class="col-xs-4 col-sm-3 col-md-2"><img src="images/t2.jpg" alt="Tigre"></div>

    <div class="col-xs-4 col-sm-3 col-md-2"><img src="images/t3.jpg" alt="Tigre"></div>

    <div class="col-xs-4 col-sm-3 col-md-2"><img src="images/t4.jpg" alt="Tigre"></div>

    <div class="col-xs-4 col-sm-3 col-md-2"><img src="images/t5.jpg" alt="Tigre"></div>

    <div class="col-xs-4 col-sm-3 col-md-2"><img src="images/t6.jpg" alt="Tigre"></div>

    <div class="col-xs-4 col-sm-3 col-md-2"><img src="images/t7.jpg" alt="Tigre"></div>

    <div class="col-xs-4 col-sm-3 col-md-2"><img src="images/t8.jpg" alt="Tigre"></div>

    <div class="col-xs-4 col-sm-3 col-md-2"><img src="images/t9.jpg" alt="Tigre"></div>

    <div class="col-xs-4 col-sm-3 col-md-2"><img src="images/t10.jpg" alt="Tigre"></div>

    <div class="col-xs-4 col-sm-3 col-md-2"><img src="images/t11.jpg" alt="Tigre"></div>

    <div class="col-xs-4 col-sm-3 col-md-2"><img src="images/t12.jpg" alt="Tigre"></div>

  </section>

</div>
```

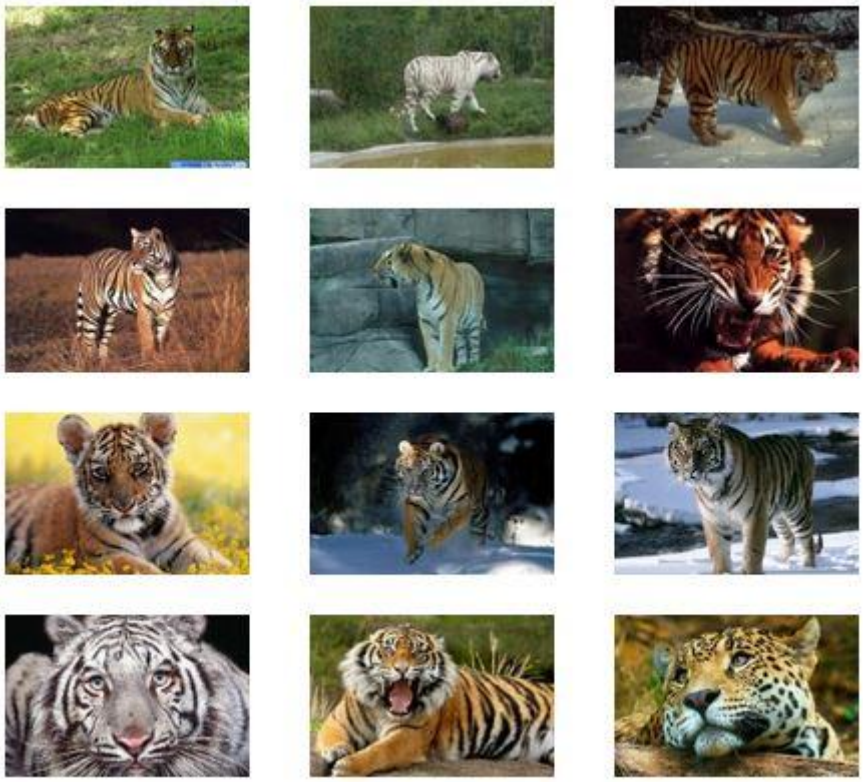
Je vais avoir ainsi :

- Sur un écran de plus de 992 pixels de large : classes `col-md-2` actives.
- Sur un écran entre 768 et 992 pixels de large : classes `col-sm-3` actives.
- Sur un écran de moins de 768 pixels de large : classes `col-xs-4` actives.

Et voici à la figure suivante le résultat sur écran moyen (classes `col-sm-3` actives).

Affichage sur écran moyen

Et à la figure suivante, le résultat sur petit écran (classes`col-xs-4` actives).



Affichage sur petit écran

J'ai donc obtenu facilement une adaptation de la présentation de mes photos en fonction du support utilisé pour les visualiser.

Page d'exemple

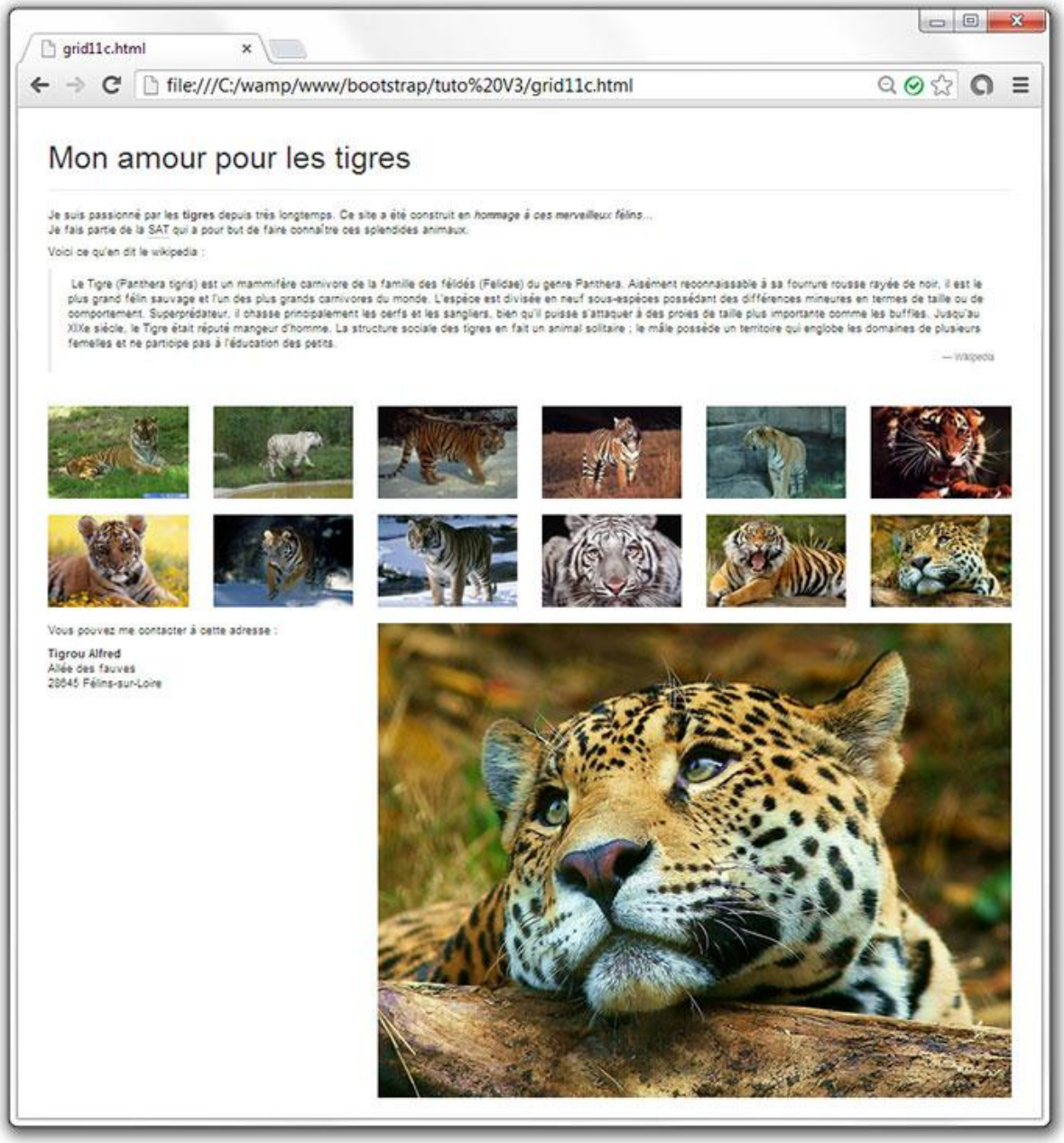
Construisons maintenant une page complète qui intègre la partie que nous venons de traiter :

```
<!DOCTYPE HTML>

<html>



  <head>

    <meta charset="utf-8">

    <link href="assets/css/bootstrap.css" rel="stylesheet">

    <style type="text/css">

      [class*="col"] { margin-bottom: 20px; }

      img { width: 100%; }

    </style>

  </head>



  <body>

    <div class="container">

      <header class="page-header">

        <h1>Mon amour pour les tigres</h1>

      </header>

      <section class="row">

        <div class="col-lg-12">

          <p>

            Je suis passionné par les <strong>tigres</strong> depuis très longtemps. Ce
site a été construit en <em>hommage à ces merveilleux félins...</em><br>

            Je fais partie de la <abbr title="Société des Amoureux des Tigres">SAT</abbr>
qui a pour but de faire connaître ces splendides animaux.

          </p>
```

```
          <p>Voici ce qu'en dit le wikipedia :</p>

          <blockquote>

            Le Tigre (Panthera tigris) est un mammifère carnivore de la famille des félidés
(Felidae) du genre Panthera.

Aisément reconnaissable à sa fourrure rousse rayée de noir, il est le plus grand félin
sauvage et l'un des plus grands carnivores du monde.

L'espèce est divisée en neuf sous-espèces possédant des différences mineures en termes de
taille ou de comportement. Superprédateur,

il chasse principalement les cerfs et les sangliers, bien qu'il puisse s'attaquer à des
proies de taille plus importante comme les buffles.

Jusqu'au XIXe siècle, le Tigre était réputé mangeur d'homme. La structure sociale des
tigres en fait un animal solitaire ; le mâle possède

un territoire qui englobe les domaines de plusieurs femelles et ne participe pas à
l'éducation des petits.<br>

            <small class="pull-right">Wikipedia</small><br>

          </blockquote>

        </div>

      </section>

      <section class="row">

        <div class="col-xs-4 col-sm-3 col-md-2"><img src="images/t1.jpg" alt="Tigre"></div>

        <div class="col-xs-4 col-sm-3 col-md-2"><img src="images/t2.jpg" alt="Tigre"></div>

        <div class="col-xs-4 col-sm-3 col-md-2"><img src="images/t3.jpg" alt="Tigre"></div>

        <div class="col-xs-4 col-sm-3 col-md-2"><img src="images/t4.jpg" alt="Tigre"></div>

        <div class="col-xs-4 col-sm-3 col-md-2"><img src="images/t5.jpg" alt="Tigre"></div>

        <div class="col-xs-4 col-sm-3 col-md-2"><img src="images/t6.jpg" alt="Tigre"></div>

        <div class="col-xs-4 col-sm-3 col-md-2"><img src="images/t7.jpg" alt="Tigre"></div>

        <div class="col-xs-4 col-sm-3 col-md-2"><img src="images/t8.jpg" alt="Tigre"></div>

        <div class="col-xs-4 col-sm-3 col-md-2"><img src="images/t9.jpg" alt="Tigre"></div>
```

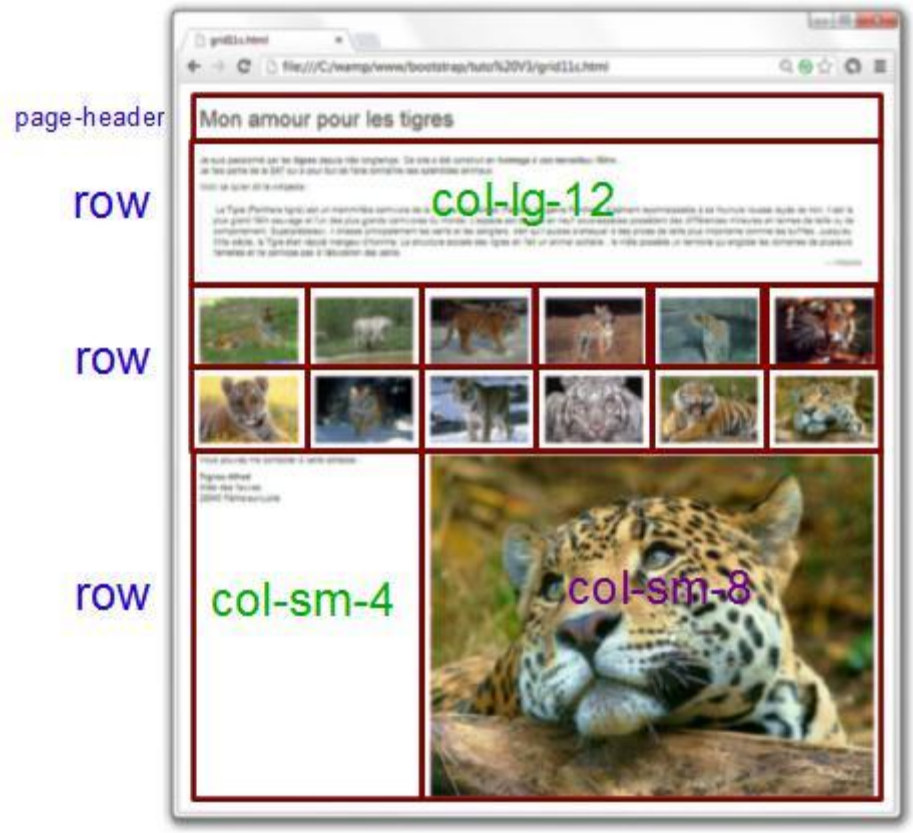
```
        <div class="col-xs-4 col-sm-3 col-md-2"><img src="images/t10.jpg"
alt="Tigre"></div>

        <div class="col-xs-4 col-sm-3 col-md-2"><img src="images/t11.jpg"
alt="Tigre"></div>

        <div class="col-xs-4 col-sm-3 col-md-2"><img src="images/t12.jpg"
alt="Tigre"></div>

      </section>

      <section class="row">

        <aside class="col-sm-4">

          <address>

            <p>Vous pouvez me contacter à cette adresse :</p>

            <strong>Tigrou Alfred</strong><br>

            Allée des fauves<br>

            28645 Félins-sur-Loire<br>

          </address>

        </aside>

        <div class="col-sm-8">

          <img src="images/photo-tigre.jpg" alt="Tigre">

        </div>

      </section>

    </div>

  </body>

</html>
```

Voici à la figure suivante le résultat obtenu.



La page d'exemple

Une simple composition avec un en-tête et 3 rangées. L'en-tête ne comporte que le titre avec une classe un peu particulière que nous allons voir plus loin. La première rangée comporte un seul élément. La deuxième en revanche en comporte 12 et correspond à ce que nous avons vu précédemment. Voici à la figure suivante une visualisation du découpage.



La page d'exemple décryptée

La classe page-header

Cette classe fixe quelques valeurs :

```
.page-header {

  padding-bottom: 9px;

  margin: 40px 0 20px;

  border-bottom: 1px solid #eee;

}
```

Une marge haute de 40 pixels et basse de 20 pixels. Une ligne inférieure de 1 pixel de couleur grise, avec un écart de 9 pixels entre le contenu et cette ligne. Donc une approche sympathique pour un en-tête de page. Il suffit d'y placer un titre comme je l'ai fait ici :

```
<header class="page-header">

   <h1>Mon amour pour les tigres</h1>

</header>
```

Voici le résultat à la figure suivante.

Mon amour pour les tigres

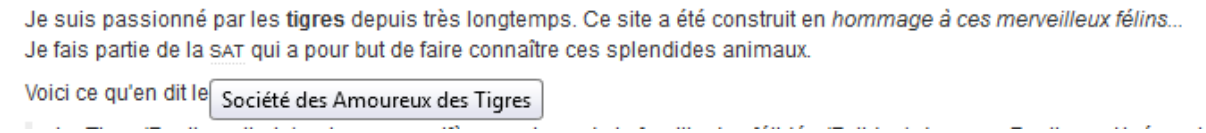
La classe « page-header »

Quelques mises en valeur

Dans le premier paragraphe, j'ai utilisé quelques mises en valeur :

```
<p>Je suis passionné par les <strong>tigres</strong> depuis très longtemps. Ce site a été construit en <em>hommage à ces merveilleux félins...</em><br>


      Je fais partie de la <abbr title="Société des Amoureux des Tigres">SAT</abbr> qui a pour but de faire connaître ces splendides animaux.


</p>
```

Observez à la figure suivante la qualité du popup lorsque l'on passe le curseur au-dessus de l'abréviation SAT.



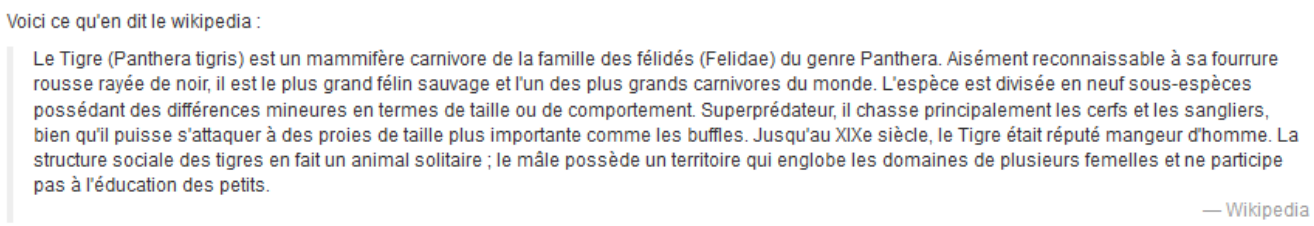
Apparence d'une abréviation

Une citation

Le chapitre suivant comporte une citation avec utilisation de la balise`blockquote`:

```
<blockquote>Le Tigre (Panthera tigris) est un mammifère carnivore...<br>


<small class="pull-right">Wikipedia</small><br>


</blockquote>
```

Le résultat est sobre et élégant (voir figure suivante).



Apparence d'une citation

La référence inférieure « Wikipedia » a été renvoyée à droite avec la classe`pull-right` qui se contente de rendre flottant à droite :

```
.pull-right {


  float: right !important;
```

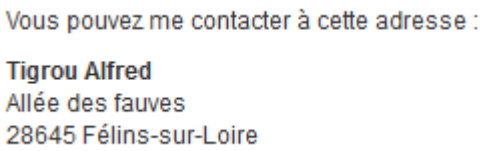
```
}
```

Une adresse

Dans la partie inférieure gauche figure une adresse. J'ai simplement utilisé la balise`<address>` :

```
<address>

   <p>Vous pouvez me contacter à cette adresse :</p>

   <p><strong>Tigrou Alfred</strong><br>

     Allée des fauves<br>

     28645 Félins-sur-Loire<br>

</address>
```

Pour un résultat satisfaisant, visible à la figure suivante.



Apparence d'une adresse

Juste un peu de style ajouté

Pour donner plus d'harmonie à cette page, j'ai dû ajouter une marge basse de 20px à tous les`col` et j'ai obligé les images à occuper tout l'espace disponible :

```
[class*="col"] {

   margin-bottom: 20px;

}

img {

   width: 100%;

}
```

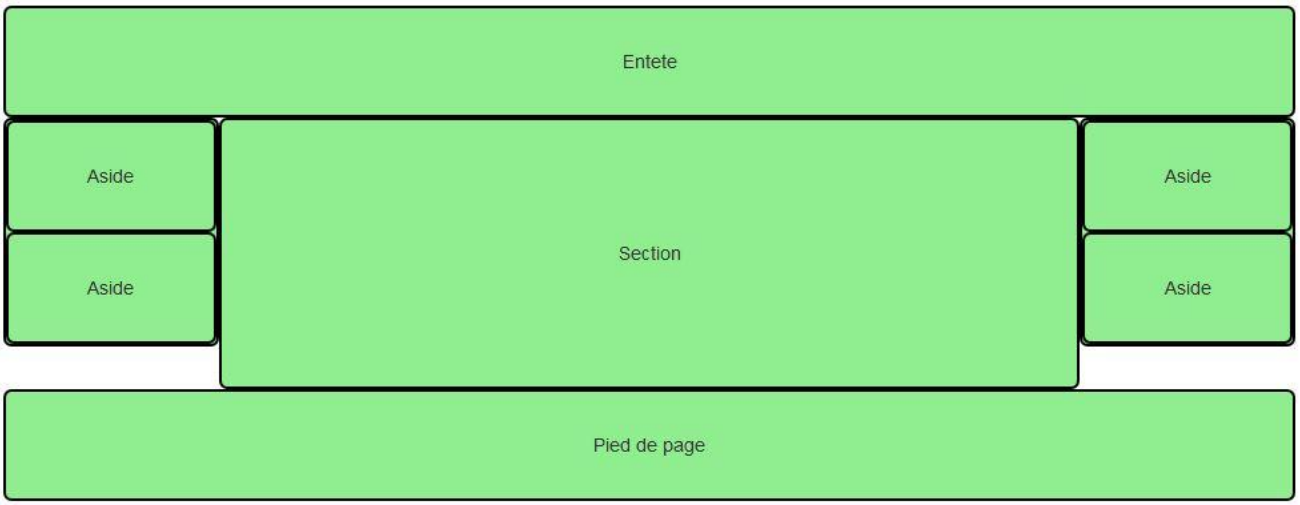
C'est le seul élément de style ajouté à cette page.

Cas pratiques et classes « responsive »

Maintenant, vous avez tous les éléments en main pour construire des pages ! Je vous propose donc quelques exercices pratiques pour vous entraîner.

Exercice 1

Le but est d'obtenir une mise en page pour grand écran identique à celle de la figure suivante.



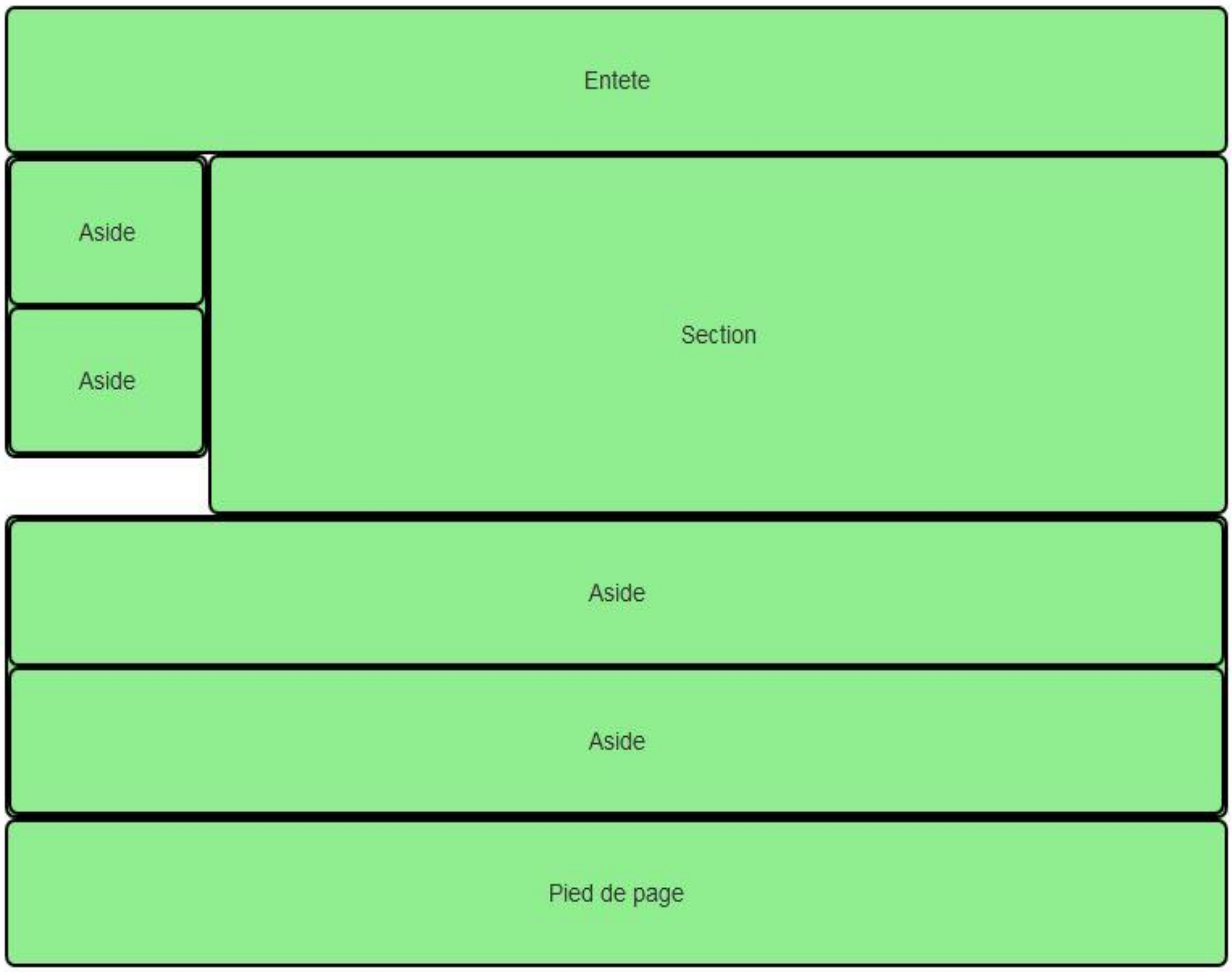
Résultat à obtenir

Réfléchissez un peu avant de regarder la solution.

```
<!DOCTYPE html>
<html>
  <head>
    <link href="assets/css/bootstrap.css" rel="stylesheet">
    <link href="assets/css/tuto.css" rel="stylesheet">
  <!-- Un peu de style pour la visualisation -->
  <style type="text/css">
    .col-lg-8 { line-height: 200px; }
    .col-lg-12 { line-height: 80px; }
  </style>
  </head>
  <body>
    <div class="container">

      <header class="row">
        <div class="col-lg-12">
          Entete
        </div>
      </header>

      <div class="row">

        <div class="col-lg-2">
          <div class="row">
            <aside class="col-lg-12">
              Aside
            </aside>
            <aside class="col-lg-12">
              Aside
            </aside>
          </div>
        </div>

        <section class="col-lg-8">
          Section
        </section>

        <div class="col-lg-2">
          <div class="row">
            <aside class="col-lg-12">
              Aside
            </aside>
            <aside class="col-lg-12">
```

```
            Aside
          </aside>
        </div>
      </div>

    </div>

    <footer class="row">
      <div class="col-lg-12">
        Pied de page
      </div>
    </footer>

  </div>
 </body>
</html>
```

Maintenant la question à se poser est : comment va réagir cette structure quand on va réduire la fenêtre ? Comme j'ai mis des classes`col-lg-*`, tout va s'empiler sous 1200 pixels. Si cet effet me convient, c'est parfait. Mais supposons que, sur tablette je veuille une structure similaire à celle de la figure suivante.



Affichage désiré sur tablette

Comment procéder ? Voici une solution :

```
<!DOCTYPE html>
```

```
<html>
  <head>
    <link href="assets/css/bootstrap.css" rel="stylesheet">
    <link href="assets/css/tuto.css" rel="stylesheet">
  <!-- Un peu de style pour la visualisation -->
  <style type="text/css">
    .col-md-8, .col-sm-10 { line-height: 200px; }
    .col-md-12 { line-height: 80px; }
  </style>
  </head>
  <body>
    <div class="container">

      <header class="row">
        <div class="col-md-12">
          Entete
        </div>
      </header>

      <div class="row">

        <div class="col-sm-2">
          <div class="row">
            <aside class="col-md-12">
              Aside 1
            </aside>
            <aside class="col-md-12">
              Aside 2
            </aside>
          </div>
        </div>

        <section class="col-sm-10 col-md-8">
          Section
        </section>

        <div class="clearfix visible-sm-block"></div>

        <div class="col-md-2">
          <div class="row">
            <aside class="col-md-12">
              Aside 3
            </aside>
            <aside class="col-md-12">
              Aside 4
            </aside>
          </div>
        </div>

      </div>

      <footer class="row">
        <div class="col-md-12">
          Pied de page
        </div>
      </footer>

    </div>
  </body>
</html>
```

J'ai remplacé les classes`col-lg-*` par des classes`col-md-*` pour avoir un comportement de base pour écrans moyens et grands. J'ai ensuite introduit des classes`col-sm-*` pour avoir le comportement désiré sur tablette. En revanche, la ligne 38 mérite quelques commentaires particuliers. Pour le média tablette, vous avez des éléments flottants qui se succèdent avec des

comportements qui ne sont pas forcément ceux que l'on souhaite. La figure suivante vous montre le résultat sans cette ligne de code.

Résultat sans la classe « clearfix »

Que fait cette classe`clearfix` ? Elle réinitialise les éléments flottants qui suivent pour les ramener dans le flux normal, et éviter qu'ils viennent recouvrir les éléments précédents. Nous allons voir maintenant la classe **visible-sm**…

Les classes « responsives »

Pour poursuivre l'exercice précédent, il faut d'abord évoquer quelques classes très utiles qui vont nous permettre non plus de positionner différemment les éléments, mais carrément de les faire disparaître ou apparaître . Vous trouverez un tableau très bien fait dans la documentation, visible à la figure suivante.

	Extra small devices Phones (<768px)	Small devices Tablets (≥768px)	Medium devices Desktops (≥992px)	Large devices Desktops (≥1200px)
`.visible-xs-*`	Visible	Hidden	Hidden	Hidden
`.visible-sm-*`	Hidden	Visible	Hidden	Hidden
`.visible-md-*`	Hidden	Hidden	Visible	Hidden
`.visible-lg-*`	Hidden	Hidden	Hidden	Visible
`.hidden-xs`	Hidden	Visible	Visible	Visible
`.hidden-sm`	Visible	Hidden	Visible	Visible
`.hidden-md`	Visible	Visible	Hidden	Visible
`.hidden-lg`	Visible	Visible	Visible	Hidden

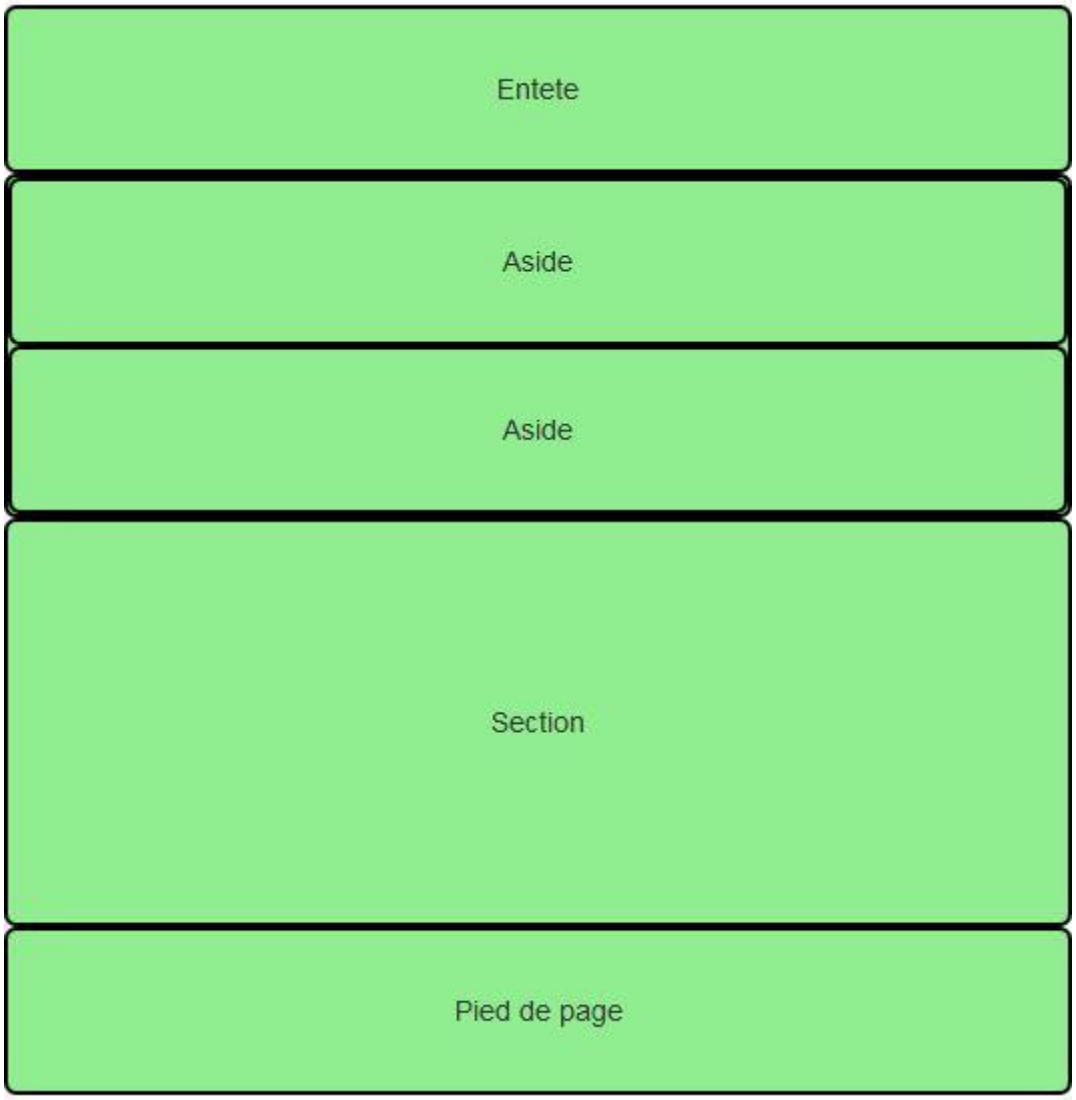
Les classes responsives

Les classes de visibilité se déclinent en trois sortes selon le type de visualisation :

Group of classes	CSS `display`
`.visible-*-block`	`display: block;`
`.visible-*-inline`	`display: inline;`
`.visible-*-inline-block`	`display: inline-block;`

Les classes de visibilité

Nous avons déjà utilisé une de ces classes. Continuons à en utiliser pour poursuivre notre exercice en supprimant 2`asides` pour les smartphones :



Affichage désiré sur smartphone

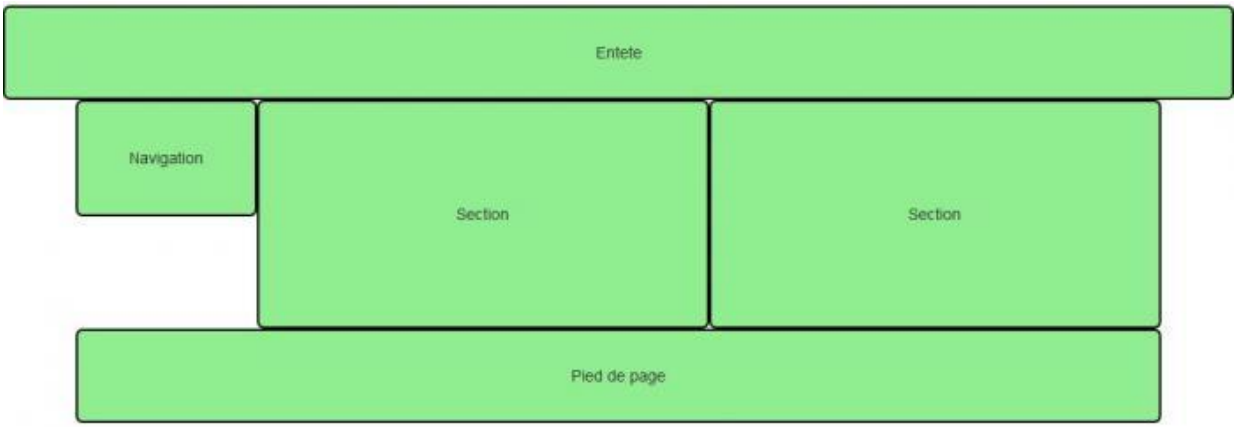
Il suffit de changer une ligne, la ligne 40 :

```
<!DOCTYPE html>
<html>
  <head>
    <link href="assets/css/bootstrap.css" rel="stylesheet">
    <link href="assets/css/tuto.css" rel="stylesheet">
```

```
  <!-- Un peu de style pour la visualisation -->
  <style type="text/css">
    .col-md-8, .col-sm-10 { line-height: 200px; }
    .col-md-12 { line-height: 80px; }
  </style>
  </head>
  <body>
    <div class="container">

      <header class="row">
        <div class="col-md-12">
          Entete
        </div>
      </header>

      <div class="row">

        <div class="col-sm-2">
          <div class="row">
            <aside class="col-md-12">
              Aside
            </aside>
            <aside class="col-md-12">
              Aside
            </aside>
          </div>
        </div>

        <section class="col-sm-10 col-md-8">
          Section
        </section>

        <div class="clearfix visible-sm-block"></div>

        <div class="hidden-xs col-md-2">
          <div class="row">
            <aside class="col-md-12">
              Aside
            </aside>
            <aside class="col-md-12">
              Aside
            </aside>
          </div>
        </div>

      </div>

      <footer class="row">
        <div class="col-md-12">
          Pied de page
        </div>
      </footer>

    </div>
  </body>
</html>
```

Exercice 2

Voyons maintenant un autre cas. Je veux obtenir le résultat visible à la figure suivante.



Résultat à obtenir

Donc une entête qui prend toute la largeur de l'écran, une navigation à gauche, deux sections accolées et un pied de page, le tout pour grand écran. Voici une solution :

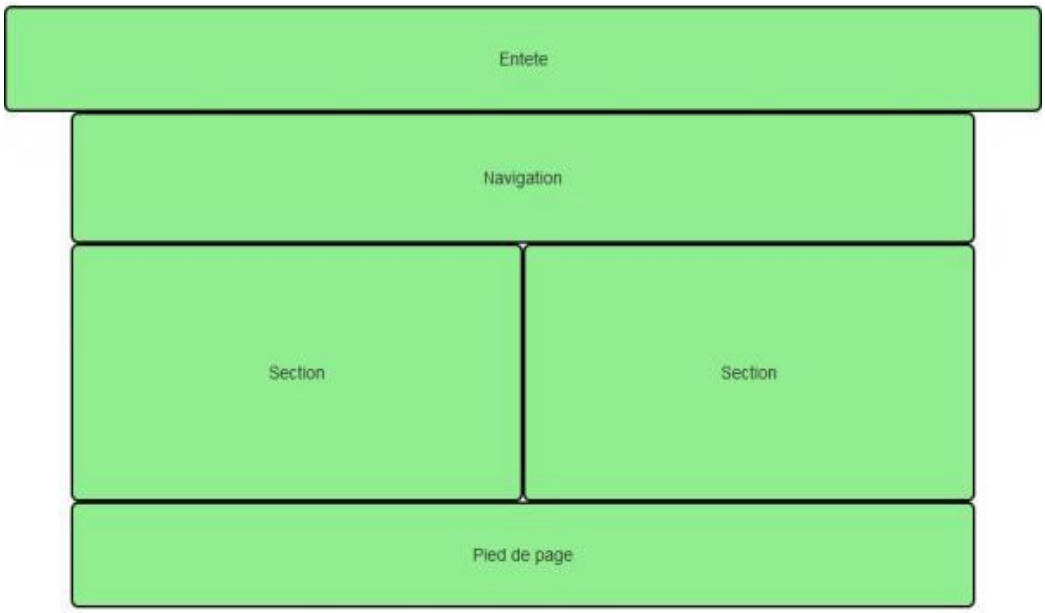
```
<!DOCTYPE html>
<html>
  <head>
    <link href="assets/css/bootstrap.css" rel="stylesheet">
    <link href="assets/css/tuto.css" rel="stylesheet">
  <!-- Un peu de style pour la visualisation -->
  <style type="text/css">
    .col-lg-2 { line-height: 100px; }
    .col-lg-5 { line-height: 200px; }
    .col-lg-12 { line-height: 80px; }
  </style>
  </head>
  <body>

    <header>
      <div class="col-lg-12">
        Entete
      </div>
    </header>

    <div class="container">

      <div class="row">

        <nav class="col-lg-2">
            Navigation
        </nav>

        <section class="col-lg-5">
          Section
        </section>
        <section class="col-lg-5">
          Section
        </section>

      </div>

      <footer class="row">
        <div class="col-lg-12">
          Pied de page
        </div>
      </footer>

    </div>
```

```
  </body>
</html>
```

Maintenant je me dis que mon application serait bien aussi sur écran moyen. Si je ne fais rien, j'obtiens le résultat visible à la figure suivante.

Résultat sur écran moyen

Ça ne me plaît pas trop, j'aimerais que les 2 sections restent accolées, comme à la figure suivante.



Résultat désiré sur écran moyen

Voici une solution :

```
<!DOCTYPE html>
<html>
  <head>
    <link href="assets/css/bootstrap.css" rel="stylesheet">
    <link href="assets/css/tuto.css" rel="stylesheet">
  <!-- Un peu de style pour la visualisation -->
  <style type="text/css">
    .col-lg-2 { line-height: 100px; }
    .col-lg-5 { line-height: 200px; }
    .col-lg-12 { line-height: 80px; }
  </style>
  </head>
  <body>
    <header>
      <div class="col-lg-12">
        Entete
      </div>
    </header>

    <div class="container">

      <div class="row">

        <nav class="col-md-12 col-lg-2">
            Navigation
        </nav>

        <section class="col-md-6 col-lg-5">
          Section
        </section>
        <section class="col-md-6 col-lg-5">
          Section
        </section>

      </div>

      <footer class="row">
        <div class="col-lg-12">
          Pied de page
        </div>
      </footer>

    </div>

  </body>
</html>
```

Après réflexion, je me dis que mon application irait bien aussi sur tablette, l'empilement des éléments me convient, mais j'aimerais avoir une en-tête différente. Après une nouvelle réflexion, j'en veux aussi une différente sur smartphone. Avec les classes « responsives » vues précédemment, c'est facile à réaliser :

```
<!DOCTYPE html>
<html>
  <head>
    <link href="assets/css/bootstrap.css" rel="stylesheet">
    <link href="assets/css/tuto.css" rel="stylesheet">
  <!-- Un peu de style pour la visualisation -->
  <style type="text/css">
    .col-lg-2 { line-height: 100px; }
    .col-lg-5 { line-height: 200px; }
```

```
    .col-lg-12 { line-height: 80px; }
  </style>
  </head>
  <body>


    <header>
      <div class="visible-lg col-lg-12">
        Entete large
      </div>
      <div class="visible-md col-lg-12">
        Entete moyenne
      </div>
      <div class="visible-sm col-lg-12">
        Entete tablette
      </div>
      <div class="visible-xs col-lg-12">
        Entete smartphone
      </div>
    </header>

    <div class="container">

      <div class="row">

        <nav class="col-md-12 col-lg-2">
            Navigation
        </nav>

        <section class="col-md-6 col-lg-5">
          Section
        </section>
        <section class="col-md-6 col-lg-5">
          Section
        </section>

      </div>

      <footer class="row">
        <div class="col-lg-12">
          Pied de page
        </div>
      </footer>

    </div>

  </body>
</html>
```

En résumé

- Il est possible avec la grille de régler très finement le rendu des pages selon les supports de visualisation.
- La plupart des éléments typographiques du HTML trouvent automatiquement une mise en forme esthétique avec Bootstrap.
- On a vu sur des exemples pratiques qu'il est facile de faire des mises en page même complexes sans trop d'effort.