*Formation Bootstrap*

*Notes de cours*



*Johnny Piette*

Table des matières

[I. A propos de ce cours 4](#_Toc66949546)

[II. Prérequis 4](#_Toc66949547)

[III. Présentation de Bootstrap 4](#_Toc66949548)

[1. Brainstorming : Qu’est-ce que Bootstrap pour vous ? 4](#_Toc66949549)

[2. Brève présentation de Bootstrap 4](#_Toc66949550)

[3. Pourquoi utiliser Bootstrap ? 5](#_Toc66949551)

[4. Bootstrap 4.x et Bootstrap 5 5](#_Toc66949552)

[5. Comment Utiliser Bootstrap ? 5](#_Toc66949553)

[IV. Intégration des Fichiers Bootstrap 6](#_Toc66949554)

[V. Fonctionnement de Bootstrap 6](#_Toc66949555)

[VI. Manipulation du texte 9](#_Toc66949556)

[1. Alignement du texte 9](#_Toc66949557)

[2. Alignement du texte fonction de l’écran 9](#_Toc66949558)

[3. Les classes de mise en forme 9](#_Toc66949559)

[4. Transformation du texte 10](#_Toc66949560)

[VII. Les images 10](#_Toc66949561)

[1. Responsive 10](#_Toc66949562)

[2. Thumbnail 10](#_Toc66949563)

[3. Alignement 10](#_Toc66949564)

[VIII. Le composant Jumbotron 11](#_Toc66949565)

[IX. Lecteur d’écran 12](#_Toc66949566)

[X. Le Composant Carousel 12](#_Toc66949567)

[1. Construction du carousel Bootstrap 12](#_Toc66949568)

[2. Ajouter des contrôles Précédent et Suivant 13](#_Toc66949569)

[Bouton précédent 13](#_Toc66949570)

[Bouton suivant 13](#_Toc66949571)

[3. Contrôles accessibles 14](#_Toc66949572)

[Exemple complet de contrôles accessibles 14](#_Toc66949573)

[XI. La Grille de Bootstrap 14](#_Toc66949574)

[1. Classes pour un nombre fixe de colonnes : 14](#_Toc66949575)

[2. Classes pour un nombre de colonnes variables en fonction de la résolution 15](#_Toc66949576)

[3. Mixe entre classes à nombre de colonnes fixes et variables 16](#_Toc66949577)

[XII. Les couleurs 17](#_Toc66949578)

[XIII. Les boutons 18](#_Toc66949579)

[1. Boutons avec une couleur de fond 18](#_Toc66949580)

[2. Boutons avec bordures 18](#_Toc66949581)

[3. Boutons sur la balise <a> 19](#_Toc66949582)

[4. La taille des boutons 19](#_Toc66949583)

[5. Activer/Désactiver un bouton 19](#_Toc66949584)

[XIV. Les tableaux 20](#_Toc66949585)

[1. Simple tableau Bootstrap 20](#_Toc66949586)

[2. Description du tableau 21](#_Toc66949587)

[3. Entêtes clairs ou sombres 22](#_Toc66949588)

[4. Lignes en rayures de zèbre 22](#_Toc66949589)

[5. Tableau à bordures ou sans bordure 23](#_Toc66949590)

[6. Effet lors du passage de la souris sur une ligne 23](#_Toc66949591)

[7. Couleurs des lignes 23](#_Toc66949592)

[8. Tableaux adaptables 23](#_Toc66949593)

[XV. Les formulaires 24](#_Toc66949594)

[1. Classes form-group et form-control 24](#_Toc66949595)

[2. Classe .form-control-file 25](#_Toc66949596)

[3. Attribut readonly 25](#_Toc66949597)

[4. Attribut readonly et la classe .form-control-plaintext 25](#_Toc66949598)

[XVI. Les displays 26](#_Toc66949599)

[XVII. Les navs 26](#_Toc66949600)

[XVIII. Les navbars 26](#_Toc66949601)

[XIX. Classes d’espacement 26](#_Toc66949602)

[1. Les marges 26](#_Toc66949603)

[a. Soit une classe du type .m{côtés}-{taille}. 26](#_Toc66949604)

[b. Soit une classe du type .m{côtés}-{media}-{taille} 27](#_Toc66949605)

[2. Les paddings 27](#_Toc66949606)

[XX. Les bordures 27](#_Toc66949607)

[XXI. Les barres de progression 27](#_Toc66949608)

[XXI. Les fenêtres modales 27](#_Toc66949609)

# A propos de ce cours

Ce document contient les notes de cours sur Bootstrap. En complément, vous avez un autre document qui contient les instructions pour réaliser les différents exercices/manipulations que nous ferons en classe. A ce document d’exercices viendra se greffer un ensemble de répertoires reprenant les différents exercices/manipulations. Si vous vous en sortez, n’allez pas voir le résultat final de ces exercices.

De plus, il est complémentaire au site <http://eqla.ddns.net> que je vous invite à visiter.

# Prérequis

Pour aborder Bootstrap les prérequis suivants sont nécessaires :

* HTML
* CSS
* Un peu de JS pour certains composants.

En effet, Bootstrap étant un Framework basé sur la présentation, les notions de class et id sur des balises sont nécessaires.

Petit rappel : Quelle est la différence entre les attributsid et class ?

Tous les deux se trouvent sur des balises HTML.

* id : cet attribut doit être unique. Il peut être utilisé pour appliquer un style très précis à une balise.
* class : n’a pas la prétention d’être unique mais peut l’être. Il est utilisé pour appliquer un même style à différentes balises.

L’utilisation des class dans Bootstrap est omniprésente donc il est important de bien comprendre ce qu’est un attribut.

De plus, il est très courant de savoir qu’un élément puisse avoir plusieurs classes. Et dans Bootstrap c’est monnaie courante d’ajouter plusieurs classes à un élément.

# Présentation de Bootstrap

## Brainstorming : Qu’est-ce que Bootstrap pour vous ?

On commence par demander aux stagiaires ce qu’est Bootstrap ? Quels sont ses intérêts ?

## Brève présentation de Bootstrap

Bootstrap est le Framework CSS le plus célèbre au monde. Il fournit une liste d’outils qui permet de simplifier le design de sites internet. L’adresse officielle est <https://getbootstrap.com/>

C’est un ensemble qui contient des boutons, des formulaires, des outils de navigations, une typographie et divers composants.

Il est dit « Responsive » ou « Responsive Web Design » : permet la consultation confortable sur des écrans de différentes tailles. Ça peut allez du desktop, smartphone, tablette, télévision, etc.

En avril 2020, il est le 7ème projet le plus populaire sur GitHub.

## Pourquoi utiliser Bootstrap ?

Il n’est pas obligatoire de l’utiliser mais il peut vous faire gagner pas mal de temps dans certaines situations. Il permet d’avoir une certaine cohérence graphique pour les sites. Il est possible de trouver des thèmes gratuits et payants.

Les avantages d’utiliser Bootstrap sont les suivants :

* Gain de temps en développement.
* Cohérence dans le design de votre site.
* Pensé Mobile First : Android, IOS, Windows 10 mobile.
* Compatible à la majorité des navigateurs : tous les navigateurs. Et pour le cas d’IE à partir de la version 10.
* Responsive.
* Open Source.

## Bootstrap 4.x et Bootstrap 5

En ce moment, février 2021, nous sommes au moment où Bootstrap 5 va bientôt pointer le bout de son nez. Il est toujours en phase béta. Comme la version 4.x est la plus déployée, je pense qu’il est plus intéressant pour vous de voir la version 4.x qui est la plus installée et utilisée.

La version 4.x dépend de la librairie JavaScript jQuery qui doit impérativement être chargée avant le fichier JavaScript de Bootstrap.

Mais le passage à Bootstrap 5 ne devrait pas être trop difficile normalement. De plus Bootstrap 5 va se libérer de sa dépendance à jQuery et fera du pur JavaScript (appelé parfois Vanilla JS).

## Comment Utiliser Bootstrap ?

Bootstrap est un Framework qui est composé d’un ensemble de fichiers. Pour utiliser Bootstrap, il nous faut donc utiliser des fichiers que vous pourrez trouver à cette adresse : <https://getbootstrap.com/docs/4.6/getting-started/download/>

Il y a deux manières d’utiliser ces fichiers :

* Soit vous les téléchargez (1 fichier CSS et 1 ou 2 fichiers JS) sur le site Bootstrap. Et vous ajoutez le lien dans votre HTML. Vous voyez que j’ai mis 1 ou 2 fichiers JS :
  + 2 fichiers JavaScript :
    - L’un pour Popper.js qui permet d’avoir des Tooltips (info-bulles) sur des éléments de votre page. Donnant un bel effet.
    - L’autre pour le JavaScript de Bootstrap.
  + 1 fichier JavaScript : c’est un bundle (un paquet) qui contient Popper et Bootstrap.
* Soit vous utilisez des adresses qui pointent sur ce qu’on appelle des CDN(Content Delivery Network). L’avantage des CDN c’est qu’ils sont super rapides mais si vous voulez les utiliser, vous devez ajouter l’attribut integrity pour vérifier que c’est le code javascript que vous voulez et qu’il n’a pas été remplacé par un hacker. Les navigateurs modernes vérifieront grâce à la valeur mise pour l’attribut integrity qu’il s’agit bien du fichier que vous voulez.

Cependant, avec Bootstrap 4.x et versions antérieures, il est impératif d’ajouter la librairie JavaScript jQuery. Elle doit être chargée avant les fichiers JavaScript Popperet Bootstrap. C’est-à-dire que dans votre page HTML vous mettrez la balise <javascript> de jQuery avant celles de Popper et de Bootstrap. Le cas échéant, Bootstrap ne fonctionnera pas. C’est pourquoi Bootstrap 5 n’utilise plus jQuery. Le JavaScript moderne permet de s’en affranchir.

Vous aurez des exemples d’intégration à la page suivante : <https://getbootstrap.com/docs/4.6/getting-started/introduction/>

Pour vous simplifier la tâche, j’ai créé dans le répertoire **Exercices/Templates** un fichier html modèle pour commencer vos exercices. Il se nomme **Template.html** et les ressources nécessaires pointent sur des CDN. Maintenant, libre à vous de l’utiliser ou non.

Dans ce fichier, j’ai fait pointer vers la dernière version 4.6 de Bootstrap. De plus, j’ai ajouté un fichier CSS supplémentaire (Icones Bootstrap) que nous discuterons plus tard mais comme ça nous avons notre page web modèle Bootstrap déjà prête pour cela.

# Intégration des Fichiers Bootstrap

Sur le site Bootstrap : <https://getbootstrap.com/docs/4.6/getting-started/introduction/>

Allez dans le répertoire ***Note-de-Cours/IV.Integration-de-Bootstrap***

Vous verrez dans le fichier exemple1.html comment intégrer les CDN pour utiliser Bootstrap.

Ensuite dans le fichier exemple2.html, vous verrez comment héberger vos fichiers au lieu d’utiliser les CDN. C’est assez trivial mais comme ça vous aurez un exemple de chaque.

Vous pouvez tester et constater que le résultat est le même avec et sans CDN. Sauf que dans le cas des CDN, vous aurez besoin d’une connexion Internet et pas pour des fichiers locaux.

# Fonctionnement de Bootstrap

Sur le site Bootstrap : <https://getbootstrap.com/docs/4.6/layout/overview/>

Bootstrap fonctionne principalement avec l’utilisation de classes. Il faut savoir que le fichier CSS de Bootstrap quand il n’est pas minifié (ramené sur une ligne pour qu’il prenne moins de place) fait 10600 lignes… Il n’est pas nécessaire de connaître par cœur toutes les classes. Personnellement j’utilise le site principal et Google.

La première classe que l’on va utiliser est la classe .**container** que l’on applique à un div. Elle permettra d’adapter la largeur du div en fonction de la résolution de l’écran du périphérique utilisé.

Exemple :

|  |
| --- |
| <body>  <div class= "**container**">  <h1>Hello, World !</h1>  <p>  <span class="font-weight-bold">Le Lorem Ipsum</span> est simplement du faux texte employé dans la composition et la mise en page avant impression. Le Lorem Ipsum est le faux texte standard de l'imprimerie depuis les années 1500, quand un imprimeur anonyme assembla ensemble des morceaux de texte pour réaliser un livre spécimen de polices de texte. Il n'a pas fait que survivre cinq siècles, mais s'est aussi adapté à la bureautique informatique, sans que son contenu n'en soit modifié. Il a été popularisé dans les années 1960 grâce à la vente de feuilles Letraset contenant des passages du Lorem Ipsum, et, plus récemment, par son inclusion dans des applications de mise en page de texte, comme AldusPageMaker.  </p>  </div>  </body> |

Si l’on regarde le CSS de Bootstrap, on voit que dès que l’on utilise la classe .container, que tout va s’adapter.

Regardons rapidement ce CSS pour la classe .container On voit que tout changera automatiquement en fonction du média utilisé. Vous pouvez en même temps voir qu’il y a d’autres classes **.container**, .**container-fluid**, etc

|  |
| --- |
| .container,  .container-fluid,  .container-lg,  .container-md,  .container-sm,  .container-xl {  width: 100%;  padding-right: 15px;  padding-left: 15px;  margin-right: auto;  margin-left: auto  }  @media (min-width:576px) {  .container,  .container-sm {  max-width: 540px  }  }  @media (min-width:768px) {  .container,  .container-md,  .container-sm {  max-width: 720px  }  }  @media (min-width:992px) {  .container,  .container-lg,  .container-md,  .container-sm {  max-width: 960px  }  }  @media (min-width:1200px) {  .container,  .container-lg,  .container-md,  .container-sm,  .container-xl {  max-width: 1140px  }  } |

**.container** agit différemment en fonction des résolutions des périphériques (d’après l’analyse du morceau de CSS mis plus haut) :

Cas n°1 : résolution inférieure à 576 pixels => 100% de l’écran

Cas n°2 : résolution supérieure ou égale à 576 pixels et inférieure à 768 pixels, l’élément s’affichera au centre de l’écran et sa largeur sera de 540 pixels.

Cas n°3 : résolution supérieure ou égale à 768 pixels et inférieure à 992 pixels, l’élément s’affichera au centre de l’écran et sa largeur sera de 720 pixels.

Cas n°4 : résolution supérieure ou égale à 992 pixels et inférieure à 1200 pixels, l’élément s’affichera au centre de l’écran et sa largeur sera de 960 pixels.

Cas n°5 : résolution ou égale à 1200 pixels, l’élément s’affichera au centre de l’écran et sa largeur sera de 1140 pixels.

.**container-fluid** permet d’utiliser 100% de la taille de votre écran et n’est pas fixée comme l’est .**container**.

Cependant, avec la classe .**container**, comme nous l’avons vu plus haut au cas n°1, si l’écran est inférieur à 576 pixels alors 100% sera utilisé comme **.container-fluid**.

Reprenons l’exemple précédent avec .**container-fluid**

|  |
| --- |
| <body>  <div class= "**container-fluid**">  <h1>Hello, World !</h1>  <p>  <span class="font-weight-bold">Le Lorem Ipsum</span> est simplement du faux texte employé dans la composition et la mise en page avant impression. Le Lorem Ipsum est le faux texte standard de l'imprimerie depuis les années 1500, quand un imprimeur anonyme assembla ensemble des morceaux de texte pour réaliser un livre spécimen de polices de texte. Il n'a pas fait que survivre cinq siècles, mais s'est aussi adapté à la bureautique informatique, sans que son contenu n'en soit modifié. Il a été popularisé dans les années 1960 grâce à la vente de feuilles Letraset contenant des passages du Lorem Ipsum, et, plus récemment, par son inclusion dans des applications de mise en page de texte, comme AldusPageMaker.  </p>  </div>  </body> |

Si vous en avez la possibilité, comparez le résultat de la page sur un smartphone et un ordinateur de bureau. Ou encore, jouez sur la largeur du navigateur sur votre desktop, la largeur s’auto adapte.

# Manipulation du texte

Sur le site Bootstrap : <https://getbootstrap.com/docs/4.6/utilities/text/>

## Alignement du texte

**a. La classe .text-justify**

Elle permet de justifier le texte.

**b. La classe .text-center**

Elle permet de centrer le texte.

**c. La classe .text-left**

Elle permet d’aligner à gauche votre texte.

**d. La classe .text-right**

Elle permet d’aligner à droite votre texte.

## Alignement du texte fonction de l’écran

Des abréviations peuvent s’ajouter à certaines classes pour conditionner l’action en fonction de l’écran.

* sm (small) :résolution supérieure ou égale à 576 pixels et inférieure à 768 pixels.
* md (medium): résolution supérieure ou égale à 768 pixels et inférieure à 992 pixels.
* lg (large): résolution supérieure ou égale à 992 pixels et inférieure à 1200 pixels
* xl (extra large): résolution ou égale à 1200 pixels.

Dans la littérature Bootstrap on voit souvent l’utilisation du caractère \* pour certaines classes. Ça veut dire qu’il le faut remplacer par une valeur numérique ou du texte.

Exemples :

.text-\*-right : .text-xl-right

.text-\*-center : .text-md-center

.text-\*-left : .text-sm-left

D’après la feuille de style de Bootstrap, je n’ai pas trouvé ces abréviations pour la classe text-justify.

## Les classes de mise en forme

La classe .font-weight-bold: met en gras.

La classe .font-weight-bolder: met en plus gras.

La classe .font-weight-normal: met le texte normal.

La classe .font-weight-light: met dans une font claire.

La classe .font-weight-lighter: met dans une font encore plus claire.

La classe .font-italic: met en le texte en italic.

## Transformation du texte

La classe **.text-lowercase**: texte converti en minuscules.

La classe **.text-uppercase**: TEXTE CONVERTI EN MAJUSCULES.

La classe **.text-capitalize**: Première Lettre De Chaque Mot Est En Majuscule.

Exemples :

|  |
| --- |
| <p>.text-lowercase: <span class="text-lowercase">TEXTE CONVERTI EN MINUSCULES </span>.</p>  <p>.text-uppercase: <span class="text-uppercase">texte converti en majuscules</span>.</p>  <p>.text-capitalize: <span class="text-capitalize">première lettre de chaque mot est en majuscule.</span></p> |

# Les images

Sur le site Bootstrap : <https://getbootstrap.com/docs/4.6/content/images/>

Nous allons voir que Bootstrap permet de facilement rendre une image responsive, en faire une jolie vignette, aligner celle-ci.

Evidemment toutes ces classes peuvent être combinées entre elles comme dans le précédent exemple.

## Responsive

Bootstrap permet de rapidement permettre à une image d’être responsive. Ajouter la classe .img-fluid et votre image va s’auto-adapter en fonction de l’écran.

|  |
| --- |
| <img src="Images/logo\_eqla.png" class="**img-fluid**" alt="logo d'Eqla"width="10000px"  /> |

Ici on a ajouté l’attribut width="10000px". L’image ne fera bien sûr jamais 10000 pixels. Bootstrap veille au grain. 😊

## Thumbnail

La classe .img-thumbnail ajoute à l’image un bord blanc arrondi.

|  |
| --- |
| <img src="/Images/paris.jpg" class="**img-thumbnail**" alt="Image de paris" /> |

## Alignement

Permet de mettre des images à gauche **.float-left** ou à droite **.float-right**quel que soit la taille de l’écran.

|  |
| --- |
| <img src="/Images/logo\_eqla.png" class="**img-fluid float-left**" alt="logo d'Eqla" />  <img src="/Images/paris.jpg" class="**img-fluid img-thumbnail float-right**" alt="Image de paris" /> |

On peut aussi définir ces alignements en fonction du périphérique :**.float-\*-left** ou **.float-\*-right** où **\*** peut avoir différentes valeurs.

* sm : small
* md : medium
* lg : large
* xl : extra-large

# Le composant Jumbotron

Sur le site Bootstrap : <https://getbootstrap.com/docs/4.6/components/jumbotron/>

Le Jumbotron est une boite rectangulaire qui va permettre de mettre en avant un certain message. Il est très grand et donne directement un impact visuel. On ne peut pas passer à côté quand on arrive sur une page.

Pour définir un jumbotron, on va ajouter une classe .jumbotron à un élément conteneur de type div. On va ensuite pouvoir placer plus ou moins n’importe quel contenu HTML à l’intérieur.

Exemple :

|  |
| --- |
| <div class="**jumbotron**">  <h1> Hello, world!</h1>  <p class="lead">Ceci est un exemple du composant jumbotron pour capter l’attention pour son contenu, une information importante.</p>  </div> |

Maintenant, vous n’êtes pas obligés de mettre le jumbotron dans un div avec une class .container mais vous pouvez avoir un div container après le jumbotron. Donnant un effet que le jumbotron prend toute la largeur de la page et que le div ayant la classe .container soit lui centré.Ça dépend de ce que l’on veut mais ce n’est pas mal non plus.

Et pour terminer vous pourriez aussi avoir un div ayant la classe container à l’intérieur de votre jumbotron. Ça n’est pas interdit.

|  |
| --- |
| <div class="**jumbotron** text-white" style="background-color: #563d7c">  <div class="**container**">  <h1> VII. Le composant Jumbotron</h1>  <p class="lead">Ceci est un exemple du composant jumbotron pour capter l’attention par son contenu, une information importante.</p>  </div>  </div> |

Et pour finir, vous avez la class .jumbotron-fluid qui supprime les coins arrondis et les rends carrés. A vous de décider entre « arrondis » ou « carrés ».

Exemple : <div class="jumbotron jumbotron-fluid">

# Lecteur d’écran

Sur le site Bootstrap : <https://getbootstrap.com/docs/4.6/utilities/screen-readers/>

J’ai découvert cette classe par hasard en me « promenant » sur le site Bootstrap. Il m’a semblé évident de vous la présenter.

La classe .sr-only, elle n’est accessible que par un lecteur d’écran.

Exemple :

|  |
| --- |
| <div class = "container">  <p> Votre formateur s’appelle Johnny </p>  <a class = "sr-only" href="#">Votre formateur doit absolument aller chez le coiffeur mais ça restera entre nous.</a>  <p>Votre formateur est très bien coiffé.</p>  </div> |

La classe .sr-only combinée avec .sr-only-focusable n’est visible que si on utilise le clavier (TAB) pour passer d’un lien à un autre.

Exemple :

|  |
| --- |
| <div class = "container">  <p> Votre formateur s’appelle Johnny </p>  <a class = "sr-only" href="#">Votre formateur doit absolument aller chez le coiffeur mais ça restera entre nous.</a>  <a class = "sr-only sr-only-focusable" href ="#">Votre formateur est très bien coiffé.</a>  </div> |

# Le Composant Carousel

Sur le site Bootstrap : <https://getbootstrap.com/docs/4.0/components/carousel/>

Bootstrap fournit nativement un composant visuel de diaporama très sympathique que l’on appelle le carrousel. Il est fréquent de le voir sur des pages webs. Il présente par exemple des promos qui défilent via des images, du texte avec un changement cyclique définissable.

On peut y mettre des images, du texte, etc.

## Construction du carousel Bootstrap

Il suit le schéma suivant basé sur l’imbrication de plusieurs div.

1. Un div contenant notre carrousel ayant les classes **.carousel** et **.slide**

Ce div doit avoir l’attribut suivant data-ride="carousel"

1. Un div enfant inclus dans le premier avec la classe **.carousel-inner**
2. Ensuite nous ajoutons des div pour le contenu (slide) à faire défiler. Chaque contenu est inclus dans un div ayant la classe **.carousel-item**. Notons qu’un des divs de slide doit impérativement avoir la classe **.active**. Sinon le carousel ne sera pas visible.

Exemple :

|  |
| --- |
| <div id="monCarousel" class="**carousel slide**" data-ride="carousel">  <div class="**carousel-inner**">  <div class="**carousel-item active**">  <img src="Images/la.jpg" alt="Los Angeles, arrivée d'un groupe sur scène."> Los Angeles, pour <a href="">plus de photos</a>.  </div>  <div class="**carousel-item**">  <img src="Images/chicago.jpg" alt="chicago, image d'un chanteur sur scène avec une guitare.">  </div>  <div class="**carousel-item**">  <img src="Images/ny.jpg" alt="New-York, image dans la foule en direction de la scène.">  </div>  </div>  </div> |

Si vous avoir un fondu pour passer à un slide suivant. Il faut ajouter la classe carousel-fade.

On peut jouer sur la vitesse, en millisecondes, de défilement en ajoutant un attribut data-interval.

Ce qui veut dire que notre div principal contenant le carousel devrait ressembler à ceci

|  |
| --- |
| <div id="monCarousel" class="carousel slide carousel-fade" data-ride="carousel" data-interval="2000"> |

Par défaut il n’a pas de boutons de défilement mais on peut les ajouter.

## Ajouter des contrôles Précédent et Suivant

Lors de l’utilisation du carousel Bootstrap, il est possible de définir les boutons « suivant » et « précédent ». Mais pour cela, il faut impérativement donner un id à notre carousel, sinon il ne sera pas possible d’ajouter ces boutons de contrôle.

### Bouton précédent

1. Ajoutez un id avec comme valeur monCarousel au div ayant la classe "carousel".
2. Positionnez-vous après la fermeture du div ayant la classe "carousel-inner".
3. Ouvrez une balise a et définissez les attributs suivants :
   1. class=".carousel-control-prev"
   2. href="#monCarousel" (C’est l’id du carousel définit au point 1).
   3. role="button"
   4. data-slide="prev"
4. Ajoutez <span class="carousel-control-prev-icon" aria-hidden="true"></span>
5. Fermez la balise a

### Bouton suivant

1. Ouvrez une balise a et définissez les attributs suivants :
   1. class=".carousel-control-next"
   2. href="#monCarousel" (C’est l’id du carousel définit au point 1).
   3. role="button"
   4. data-slide="prev"
2. Ajoutez <span class="carousel-control-next-icon" aria-hidden="true"></span>
3. Fermez la balise a

## Contrôles accessibles

Alors ce carousel est très sympa. Mais il manque d’accessibilité. En effet, si un lecteur d’écran passe sur les boutons Précédent ou Suivant, nous entendrons le lecteur d’écran dire : Bouton. Ce qui est peu clair.

On peut l’améliorer en ajoutant la classe sr-only vue précédemment.

Il suffit d’ajouter dans les deux balises a des contrôles un span ayant la classe sr-only.

<span class="sr-only">Précédent</span>

<span class="sr-only">Suivant</span>

### Exemple complet de contrôles accessibles

|  |
| --- |
| <a class="carousel-control-prev" href="#monCarousel" role="button" data-slide="prev">  <span class="carousel-control-prev-icon" aria-hidden="true"></span>  <span class="sr-only">Précédent</span>  </a>  <a class="carousel-control-next" href="#monCarousel" role="button" data-slide="next">  <span class="carousel-control-next-icon" aria-hidden="true"></span>  <span class="sr-only">Suivant</span>  </a> |

# La Grille de Bootstrap

Sur le site Bootstrap : <https://getbootstrap.com/docs/4.0/layout/grid/>

Bootstrap propose un système de grille très pratique et responsive. Elle permet d’agencer des contenus sous formes de colonnes.

Cette grille est composée d’une ligne de 12 colonnes. On pourrait penser que c’est un tableau. Si on veut mais alors d’une seule ligne. Ce système de grille permet d’ajouter des éléments sous formes de colonnes.

Et comme vous le savez maintenant, utilisez Bootstrap, c’est utiliser ses classes.

On utilise toujours un div conteneur ayant la classe .row.

Les divs enfants seront les colonnes et auront une ou plusieurs classes.

## Classes pour un nombre fixe de colonnes :

1. Les classes .col-\* où \* est un nombre de 1 à 12. Quand on veut absolument un nombre défini de colonnes quel que soit la résolution. Dans ce cas, vous aurez toujours le nombre de colonnes désirés.

Exemple où le nombre de colonnes est fixe par ligne (un .col-4 et .col-8 = 4 colonnes + 8 colonnes = 12 colonnes) :

|  |
| --- |
| <div class="row">  <div class="col-4">    Première colonne avec un .col-4  </div>  <div class="col-8">  Deuxième colonne avec un .col-8  </div>  </div> |

Ensuite, les combinaisons sont comme vous le voulez : Par exemple 3 .col-4 et vous aurez 3 colonnes. En effet, 3 x 4 colonnes = 12 colonnes , 2 .col-6 = 2 x 6 colonnes =12 colonnes,1 .col-2 et 1 .col-4+ et 1 .col-6 = 12 etc.

1. La classe .col peut tout simplement être utilisée si vos colonnes ont exactement la même taille. Donc on pourrait avoir deux colonnes avec .col au lieu de .col-6 Ce qui veut dire que nous aurons deux colonnes quel que soit la résolution.

L’exemple :

|  |
| --- |
| <div class="row">  <div class="col">    Première colonne avec un .col  </div>  <div class="col">    Deuxième colonne avec un .col  </div>  </div> |

Est identique à

|  |
| --- |
| <div class="row">  <div class="col-6">    Première colonne avec un .col-6  </div>  <div class="col-6">    Deuxième colonne avec un .col-6  </div>  </div> |

On peut un peu complexifier en faisait un .col avec un .col-3. Ici notre .col sera équivalent à un .col-9

|  |
| --- |
| <div class="row">  <div class="col">    Ceci est un texte assez répétitif. Ceci est un texte assez répétitif.  </div>  <div class="col-3">    Ceci est un texte assez répétitif. Ceci est un texte assez répétitif  </div>  </div> |

## Classes pour un nombre de colonnes variables en fonction de la résolution

Nous venons de voir les classes pour un nombre fixe de colonnes. C’est top quand on veut absolument avoir ce nombre de colonnes.

Mais sachez que plus la résolution est petite et plus vos colonnes seront étroites. En effet, la lecture dans des colonnes si étroites n’est pas aisée.

Pour pallier à ce problème, Bootstrap propose de conditionner nos colonnes en fonction de la résolution.

Si cette résolution n’est pas atteinte alors au lieu de mettre les colonnes les une à côté des autres, ils les mettra les une au-dessus des autres.

Les classes à utiliser seront :

* .col-sm-\* où \* est un nombre compris entre 1 et 12 (sm>= 576 pixels)
* .col-md-\* où \* est un nombre compris entre 1 et 12 (md >= 768 pixels)
* .col-lg-\* où \* est un nombre compris entre 1 et 12 (lg >= 992 pixels)
* .col-xl-\* où \* est un nombre compris entre 1 et 12 (xl >= 1200 pixels)

Par exemple si on a deux colonnes ayant chacune comme classe un .col-sm-6

|  |
| --- |
| <div class="row">  <div class="col-sm-4">  Première colonne avec un .col-sm-4  </div>  <div class="col-sm-6">  Ceci est la deuxième colonne avec un .col-sm-6  </div>  </div> |

Ça signifie qu’il faut afficher deux colonnes si on a une résolution d’au moins (576 pixels). Dans le cas contraire, Bootstrap affichera la première colonne et la seconde ira à la ligne. Et la lecture sera aisée sur un petit écran.

## Mixe entre classes à nombre de colonnes fixes et variables

Prenons un cas concret, vous souhaitez afficher 3 colonnes si la résolution le permet (md). Dans le cas contraire, vous afficher la première colonne sur une ligne et les deux autres sur la seconde.

Pour cela, vous aurez le html suivant

|  |
| --- |
| <h1>.col et .col-3</h1>  <div class="row text-white">  <div class="col-12 col-md-4 text-justify bg-success">  </div>  <div class="col-6 col-md-4 text-justify bg-warning"></div>  <div class="col-6 col-md-4 text-justify bg-danger"></div>  </div> |

Expliquons-le petit à petit :

* .col-12 col-md-4 : la colonne fera le 1/3 de l’écran si la résolution est supérieure ou égale à md. Dans le cas contraire, la colonne prendra toute la ligne (.col-12).
* .col-6 col-md-4 : la seconde colonne fera le 1/3 (même raison). Dans le cas contraire, la colonne prendra la moitié de l’écran (.col-6).
* .col-6 col-md-4 : idem que précédemment mais sera à côté de la colonne précédente si résolution inférieure à md.

Donc si la résolution est > = md, on aura 3 colonnes identiques (.col-md-4). Dans le cas contraire, on aura une colonne qui prendra toute une ligne (.col-12). Et sur une autre ligne, nous aurons deux colonnes identiques (.col-6) qui prendront toute la ligne.

# Les couleurs

Sur le site Bootstrap : <https://getbootstrap.com/docs/4.0/utilities/colors/>

Bootstrap dispose de plusieurs classes pour les couleurs. Aux classes .bg-\* (pour le background d’un élément) et aux classes .text-\* on ajoute un des suffixes suivants :

|  |
| --- |
| - primary : bleu  - secondary : gris  - success : vert  - danger : rouge  - warning : jaune  - info : cyan  - light : gris clair  - dark : gris très foncé  - body : noir  - muted : gris  - white : blanc |

Par exemple un div avec un fond gris clair :

|  |
| --- |
| <div class= "container bg-light">  <p>Le fond de ce conteneur est gris clair</p>  </div> |

Autre exemple, un div avec un fond rouge avec du texte blanc :

|  |
| --- |
| <div class= "container bg-danger text-white">  <p>Le fond de ce conteneur est rouge et le texte est blanc</p>  </div> |

Pour le texte les classes suivantes peuvent aussi être utilisées :

- black-50 : texte noir semi transparent

- white-50 : texte blanc semi transparent

Comme on peut le constater, Bootstrap fournit une cohérence dans l’utilisation des couleurs standards. Vous verrez que cette logique est suivie pour la couleur des boutons que nous allons justement voir au chapitre suivant.

# Les boutons

Sur le site Bootstrap : <https://getbootstrap.com/docs/4.0/components/buttons/>

Les classes Bootstrap pour les boutons sont faciles à utiliser. On utilise la balise <bouton>. Avec l’attribut type="button". Si on ne donne pas d’attribut type, il agira par défaut comme un type="submit" et essaiera d’envoyer un formulaire.

## Boutons avec une couleur de fond

Vous devez en premier mettre la classe .btn suivit de la couleur du bouton via la classe .btn-\* : .btn-primary, .btn-success, .btn-warning, .btn-danger, etc.

Il a un léger effet lorsqu’on survole le bouton : il change légèrement de couleur (un peu plus foncé) et reprend sa couleur quand on bouge sa souris.

Exemple :

|  |
| --- |
| div class="container">  <h1>Boutons avec couleur de fond</h1>  <button type="button" class="btn btn-primary">Primary</button>  <button type="button" class="btn btn-secondary">Secondary</button>  <button type="button" class="btn btn-success">Success</button>  <button type="button" class="btn btn-danger">Danger</button>  <button type="button" class="btn btn-warning">Warning</button>  <button type="button" class="btn btn-info">Info</button>  <button type="button" class="btn btn-light">Light</button>  <button type="button" class="btn btn-dark">Dark</button>  <button type="button" class="btn btn-link">Lien</button>  </div> |

## Boutons avec bordures

On utilise la classe .btn-outline-\* à la place de .btn-\* : .btn-outline-primary, .btn-outline-success, .btn-outline-warning, .btn-outline-danger, etc.

Exemple :

|  |
| --- |
| <div class="container">  <h1>Boutons à bordures</h1>  <button type="button" class="btn btn-outline-primary">Primary</button>  <button type="button" class="btn btn-outline-secondary">Secondary</button>  <button type="button" class="btn btn-outline-success">Success</button>  <button type="button" class="btn btn-outline-danger">Danger</button>  <button type="button" class="btn btn-outline-warning">Warning</button>  <button type="button" class="btn btn-outline-info">Info</button>  <button type="button" class="btn btn-outline-light">Light</button>  <button type="button" class="btn btn-outline-dark">Dark</button>  <button type="button" class="btn btn-outline-link">Lien</button>  </div> |

## Boutons sur la balise <a>

Rien ne nous empêche de mettre les classes des boutons sur la balise <a> pour avoir un bouton à la place d’un lien. Exemple : le bouton « Nouvelle Inscription ».

Exemple :

|  |
| --- |
| <a class="btn btn-primary" href="http://www.google.be">Nouvelle Inscription</a>  <a class="btn btn-warning" href="http://www.google.be">Désinscription</a> |

## La taille des boutons

On peut modifier la taille standard des boutons en utilisant les .btn-sm (petit bouton) et .btn-lg (gros bouton). Il existe aussi la classe .btn-block qui va prendre tout l’espace de l’élément conteneur.

Exemple :

|  |
| --- |
| <div class="container">  <h1>Taille des Boutons</h1>  <button type="button" class="btn btn-primary btn-lg mb-2">Gros bouton</button>  <button type="button" class="btn btn-outline-primary btn-lg mb-2">Gros bouton avec bordures</button>  <button type="button" class="btn btn-success btn-sm mb-2">Petit bouton</button>  <button type="button" class="btn btn-outline-success btn-sm mb-2">Petit bouton avec bordures</button>  <br><br>  <button type="button" class="btn btn-warning btn-lg btn-block">Bouton de type block</button>  <button type="button" class="btn btn-outline-warning btn-lg btn-block">Bouton avec bordures de type block</button>  </div> |

## Activer/Désactiver un bouton

On le fait en utilisant la classe .active pour l’activer et .disabled pour le désactiver. Attention de ne pas penser que c’est sécurisé de mettre un bouton avec la classe .disabled. En effet, si vous pensez par exemple désactiver la vente d’un produit jusqu’à 20h00. Sur le frontend, mettre cette classe à .disabled est facilement modifiable avec des navigateurs comme Firefox/Chrome. Il faut en plus côté backend, empêcher la vente du produit tant qu’il n’est pas 20h00.

Exemple :

|  |
| --- |
| <button type="button" class="btn btn-primary active">Active Primary</button>  <button type="button" class="btn btn-primary disabled">Disabled Primary</button> |

# Les tableaux

Sur le site Bootstrap : <https://getbootstrap.com/docs/4.0/content/tables/>

Bootstrap permet de styliser nos tableaux.

## Simple tableau Bootstrap

La classe .table permet de donner du style à nos tableaux on l’ajoute à la balise <table>.

Le tableau est plus grand et respire plus que le tableau par défaut qui est plus compact. Les lignes par défaut non pas de bords à gauche et à droite. L’entête du tableau définit dans <thead> ou par un <th> est en gras.

Exemple :

|  |
| --- |
| <div class="container"> <table class="table"> <thead> <tr> <th scope="col">Id</th> <th scope="col">Nom</th> <th scope="col">Prénom</th> <th scope="col">Age</th> </tr> </thead> <tbody> <tr>  <th scope="row">1</th> <td>Piette</td> <td>Johnny</td> <td>46</td> </tr> <tr>  <th scope="row">2</th> <td>Dupont</td> <td>Patrick</td> <td>35</td> </tr> <tr> <th scope="row">3</th> <td>Jacques</td> <td>Gabriel</td> <td>6</td> </tr> </tbody> </table>  </div> |

## Description du tableau

Pour informer l’utilisateur du contenu du tableau, soit on met un titre clair ou une description à l’aide d’un titre et un paragraphe.

On peut aussi utiliser la balise caption juste après la balise table. Nous aurons en bas à gauche du tableau la description du tableau.

Exemple :

|  |
| --- |
| <div class="container"> <table class="table"> <caption>Liste des utilisateurs</caption>  <thead> <tr> <th scope="col">Id</th> <th scope="col">Nom</th> <th scope="col">Prénom</th> <th scope="col">Age</th> </tr> </thead> <tbody> <tr>  <th scope="row">1</th> <td>Piette</td> <td>Johnny</td> <td>46</td> </tr> <tr>  <th scope="row">2</th> <td>Dupont</td> <td>Patrick</td> <td>35</td> </tr> <tr> <th scope="row">3</th> <td>Jacques</td> <td>Gabriel</td> <td>6</td> </tr> </tbody> </table>  </div> |

## Entêtes clairs ou sombres

Pour avoir l’entête du tableau en noir ou en gris clair on utilise les classes .thead-dark ou .thead-light.

Exemples : <thead class="thead-dark" > ou <thead class="thead-light" >

## Lignes en rayures de zèbre

Pour une lecture efficace, on utilise souvent cette technique d’alterner deux couleurs, une ligne sur deux. Ça permet de plus facilement lire un tableau. Sinon on est vite « noyé » dans un tableau s’il n’applique pas cette technique.

Pour alterner une ligne sur deux, en plus de la classe .table, on utilise la classe .table-striped sur la balise <table>.

Exemple :

|  |
| --- |
| <div class="container"> <table class="table table-striped"> <thead> <tr> <th scope="col">Id</th> <th scope="col">Nom</th> <th scope="col">Prénom</th> <th scope="col">Age</th> </tr> </thead> <tbody> <tr>  <th scope="row">1</th> <td>Piette</td> <td>Johnny</td> <td>46</td> </tr> <tr>  <th scope="row">2</th> <td>Dupont</td> <td>Patrick</td> <td>35</td> </tr> <tr> <th scope="row">3</th> <td>Jacques</td> <td>Gabriel</td> <td>6</td> </tr> </tbody> </table>  </div> |

## Tableau à bordures ou sans bordure

On utilise la classe .table-bordered pour ajouter une bordure à toutes les cellules du tableau et la classe .table-borderless pour les supprimer partout.

Exemple avec bordures : <table class="table table-bordered">

Exemple sans bordure : <table class="table table-borderless">

## Effet lors du passage de la souris sur une ligne

Un effet de survol lorsqu’on passe la souris sur une ligne est assez sympathique. Il se caractérise par un changement de couleur de la cellule lorsqu’on la survole.

Pour ajouter cet effet, on utilise la classe .table-hover sur la balise <table>.

Exemple : Exemple avec bordures : <table class="table table-hover table-bordered">

## Couleurs des lignes

Nous allons pouvoir utiliser les couleurs contextuelles de Bootstrap pour changer la couleur de chaque ligne ou de chaque cellule d’un tableau. On va pouvoir utiliser les classes suivantes :

* .table-active ;
* .table-primary ;
* .table-secondary ;
* .table-success ;
* .table-danger ;
* .table-warning ;
* .table-info ;
* .table-light ;
* .table-dark.

## Tableaux adaptables

Si dans un tableau vous avez vraiment besoin d’afficher tous les colonnes quel que soit le média, il existe une classe .table-responsive qui va ajouter une barre une défilement horizontale. Ce n’est pas super génial en termes de présentation mais au moins le tableau ne sort pas de l’écran…

# Les formulaires

Sur le site Bootstrap : <https://getbootstrap.com/docs/4.0/components/forms/>

Bootstrap fournit un ensemble de classes qui permettent de rendre un formulaire beaucoup plus joli visuellement. De base, les formulaires html sont assez austères.

Pour utiliser les classes que nous allons voir, il faut bien entendu utiliser la balise « form » pour nos formulaires.

## Classes form-group et form-control

La classe .form-group est utilisée pour grouper le contrôle du formulaire avec sa description.

Les éléments <input>, <select> et <textarea> sont stylisés via la classe .form-control. Ce sont des contrôles dont les bords sont arrondis, grands et prennent toute la largeur du conteneur.

Avant ces contrôles, on ajoute sa description via la balise <label> avec l’attribut for= "#ControleADecrire".

L'élément HTML <label> représente une légende pour un objet d'une interface utilisateur. Il peut être associé à un contrôle en utilisant l'attribut for ou en plaçant l'élément du contrôle à l'intérieur de l'élément <label>. Un tel contrôle est appelé contrôle étiqueté par l'élément <label>.

Exemple :

|  |
| --- |
| <form>  < !-- Premier form-group -->  <div class="form-group">  <label for="exempleInputEmail">Adresse Email</label>  <input type="email" class="form-control" id="exempleInputEmail" aria-describedby="emailHelp" placeholder="Entrer un email">  <small id="emailHelp" class="form-text text-muted">Nous ne partagerons jamais votre email avec qui que ce soit.</small>  </div>  < !-- Deuxième form-group -->  <div class="form-group">  <label for="exempleInputPassword">Password</label>  <input type="password" class="form-control" id=" exempleInputPassword " placeholder="Password">  </div>  < !-- Troisième form-group -->  <div class="form-check">  <input type="checkbox" class="form-check-input" id="exempleCheck">  <label class="form-check-label" for="exempleCheck">Check me out</label>  </div>  <button type="submit" class="btn btn-primary">Soumettre</button>  </form> |

Rappelons que l’attribut placeholder permet d’afficher un texte dans un champ du formulaire. Exemple : « Entrer un email ». Ce message sera affiché tant que rien n’est indiqué dans le champ du formulaire.

Ensuite, il est important de bien spécifier le type. En effet, ce type permettra d’être intercepté en cas de mauvais type. Exemple le type email qui testera un peu la validité d’un email.

L’attribut required indique au formulaire que ce champ est nécessaire et ne peut valider le formulaire tant que ce champ est vide. Attention qu’il ne faut pas croire aveuglément un formulaire reçu côté back-end. Normalement les mêmes règles de validations doivent être de rigueur côté serveur. Une règle : « Ne jamais croire ce que nous recevons du front-end. On reteste les données côté back-end. »

## Classe .form-control-file

Pour le contrôle d’envoi de fichier on ajoute simplement cette classe au contrôle.

Exemple :

|  |
| --- |
| <div class="form-group">  <label for="exempleFormControlFile">Example file input</label>  <input type="file" class="form-control-file" id="exempleFormControlFile">  </div> |

## Attribut readonly

Quand un élément comprend l’attribut readonly, le contrôle du formulaire sera désactivé et on ne saura pas le cliquer ou y écrire quelque chose. Le contrôle est juste en lecture seule.

Le contrôle sera en gris foncé avec son texte prédéfini. Exemple, dans un formulaire si on connait déjà le nom d’utilisateur et qu’il ne changera pas, on peut le préremplir et mettre l’attribut readonly.

Exemple :

|  |
| --- |
| <input id="username"  class="form-control" type="text" placeholder="mcfly" readonly> |

## Attribut readonly et la classe .form-control-plaintext

Ici le comportement sera identique : on ne saura pas interagir. Cependant la classe .form-control-plaintext supprime le bord du contrôle input. Donnant l’impression que c’est un simple texte.

Exemple :

|  |
| --- |
| <input type="text" readonly class="form-control-plaintext" id="staticEmail" value="johnny.piette@gmail.com"> |

# Les displays

Sur le site Bootstrap : <https://getbootstrap.com/docs/4.6/utilities/display/>

# Les navs

Sur le site Bootstrap : <https://getbootstrap.com/docs/4.6/components/navs/>

# Les navbars

Sur le site Bootstrap : <https://getbootstrap.com/docs/4.6/components/navbar/>

# Classes d’espacement

Sur le site Bootstrap : <https://getbootstrap.com/docs/4.0/utilities/spacing/>

Dans Bootstrap, nous avons des classes qui permettent d’ajouter des paddings ou des marges

## Les marges

Ces classes ont deux formats :

### Soit une classe du type .m{côtés}-{taille}.

La taille varie de 0 à 5 (0 = 0rem, 1=0,25 rem, 2 = 0,5 rem, 3=1rem, 4 = 1,5 rem, 5 = 3 rem)

0,25 rem = 0,25 \* la variable SASS $spacer qui est définie par défaut à 1rem).

« côtés » peut prendre les valeurs suivantes : rien, t (top), b(bottom), l (lef), r (right), x (gauche et droite), y (haut et bas)

m-\* : ajoutera une marge aux 4 côtés.

mt-\* : ajoutera une marge en haut.

mb-\* : ajoutera une marge en bas.

ml-\* : ajoutera une marge à gauche.

mr-\* : ajoutera une marge à droite.

mx-\* : ajoutera une marge à gauche et à droite.

my-\* : ajoutera une marge en haut et en bas.

### Soit une classe du type .m{côtés}-{media}-{taille}

« media » peut prendre une des valeurs suivantes : sm, md, lg, xl. C’est donc conditionné en fonction de la taille de l’écran du périphérique.

Exemples : m-sm-7, mx-lg-5, etc.

## Les paddings

Même principe que pour les marges.

Donc on pourra avoir deux formes : .p{côtés}-{taille} et p{côtés}-{media}-{taille}.

# Les bordures

# XXI. Les barres de progression

# Les fenêtres modales