

CSS Phase 8 : Utiliser un framework CSS : Bootstrap

Objectifs

- Maîtriser un framework CSS implémentant les contraintes du web responsive
- Utiliser un système de grille

Cheminement

Le framework CSS [Bootstrap](#) vous permet de mettre en oeuvre facilement des sites dits "responsive", c'est-à-dire avec une mise en page qui s'adapte automatiquement en fonction des écrans (PC, smartphones, tablettes).

Qu'est-ce que Bootstrap ?

Ce document a été rédigé pour la version 5.3 de Bootstrap. En fonction de l'évolution des versions des points peuvent avoir changé.

Afin d'aider les « front designers » (appelés il y a peu encore des intégrateurs) dans la conception « front-end » (le web design), des frameworks (« librairies ») CSS ont été créés, le plus populaire en termes d'utilisation étant Bootstrap.

Bootstrap a été développé en 2010 par deux ingénieurs de Twitter qui souhaitent répondre à la problématique d'une librairie servant de base commune à tous les projets de leur société. Bootstrap est actuellement (fin 2018) disponible dans sa version 4.1

Bootstrap permet principalement de structurer une page HTML en la rendant adaptable aux différentes tailles d'écran (*web responsive*) grâce à un découpage en grille, le *grid system*.

Outre le web responsive, Bootstrap embarque une multitude d'aides pour la conception front-end entre autres :

- menus (accordéons etc.)
- gestion des polices, couleurs etc.
- messages (alertes, infobulles)
- gestion des marges et paddings
- habillage des formulaires et des tableaux
- gestion des images
- fenêtres modales (iframes)
- carrousels (diaporamas) etc.

Intégrer Bootstrap dans une page web

Dans la partie <head> de la page HTML :

1. Ajouter la balise meta *viewport* :

```
<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
```

2. Puis le code pour intégrer la feuille de style CSS de Bootstrap, toujours dans <head> :

```
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC" crossorigin="anonymous">
```

Ici, on importe Bootstrap via une URL pointant sur un CDN (un serveur externe hébergeant des fichiers) mais rien n'empêche de télécharger Bootstrap directement dans l'arborescence de votre site.

Les attributs **integrity** et **crossorigin** sont devenus indispensables lors du chargement de ressources, pour permettre au navigateur de vérifier leur provenance ;

Vous pourrez ensuite ajouter vos propres fichiers CSS qui devront être chargés **APRES** celui de Bootstrap. Par exemple :

```
<!-- [...] -->
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC" crossorigin="anonymous">
<link rel="stylesheet" href="css/messtyles.css">
</head>
```

A la fin de la page

1. Enfin, intégrez les fichiers Javascript nécessaires à Bootstrap; placez ce code **avant** la fermeture de la balise body (</body>), l'ordre des fichiers est à respecter (Popper puis Bootstrap) :

```
<script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.8/dist/umd/popper.min.js"
integrity="sha384-I7E8VVD/ismYTF4hNIPjVp/Zjvgyol6VFvRkX/vR+Vc4jQkC+hVqc2pM80Dewa9r"
crossorigin="anonymous"></script>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.min.js"
integrity="sha384-BBt1+eGJRgqQAUMxJ7pMwbEyER411g+O15P+16Ep7Q9Q+zcX6gSbd85u4mG4QzX+"
crossorigin="anonymous"></script>
</body>
```

Popper est optionnel.

Vous pourrez ensuite ajouter vos propres fichiers JS qui devront être chargés **APRES** ceux de Bootstrap. Par exemple :

```
<!-- [...] -->
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.min.js"
integrity="sha384-
BBt1+eGJRgqQAUMxJ7pMwbEyER411g+O15P+16Ep7Q9Q+zqX6gSbd85u4mG4QzX+"
crossorigin="anonymous"></script>
<script src="js/mesanimations.js"></script>
</body>
```

2. Barre de navigation (exemple), à placer dans <body> :

```
<nav class="navbar navbar-expand-sm bg-dark navbar-dark">

  <!-- Toggler/collapsible Button -->
  <button class="navbar-toggler" type="button" data-toggle="collapse"
data-target="#collapsibleNavbar">
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse" id="collapsibleNavbar">
    <ul class="navbar-nav">
      <li class="nav-item">
        <a class="nav-link" href="index.php">Accueil</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="tableau.php">Tableau</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="formulaire.php">Formulaire
wazaa</a>
      </li>
    </ul>
  </div>
</nav>
```

Sur le fonctionnement des *navbar* :






- ligne 1 : on ouvre la balise `<nav>` qui contient plusieurs classes obligatoires (mais variables) ; ici :
 - `navbar` permet à Bootstrap d'identifier le début de formatage de la nav, et d'aligner automatiquement les marges
 - `navbar-expand-sm` indique à partir de quelle dimension (ici `sm`, donc à plus de 576px ; cf. le système de grilles de Bootstrap) la barre de navigation sera visible entièrement, au lieu d'afficher le bouton *hamburger*
 - `bg-dark` donne un rendu sur fond sombre
 - `navbar-dark` ajuste la couleur du texte et des liens pour les adapter à un rendu foncé
- lignes 4-6 : on trouve ici une balise `<button>` : il s'agit du code du bouton *hamburger*. L'attribut `data-target` est obligatoire ; sa valeur doit être la même que celle de l'attribut `id` du `<div>` suivant (ici `collapsibleNavbar`) et préfixé du signe `#`.
- lignes 7-19 : un `<div>` qui contient une liste de liens classiques.

Les barres de navigation peuvent supporter :

- un nom de site/marque, éventuellement une image (logo) : `brand`
- un formulaire (zone de saisie pour un champ de recherche) : `form`
- des sous-menus, appelés `dropdowns`

Le système de grille

Avec Bootstrap, il n'est plus nécessaire d'écrire des requêtes CSS media queries : Bootstrap l'a déjà fait pour vous ! Il ne "suffit plus" qu'à utiliser des classes CSS correspondant aux points de rupture selon le tableau suivant :

xs	Extra small <576px	 portrait mobile
sm	Small ≥576px	 landscape mobile
md	Medium ≥768px	 portrait tablets <i>navbar collapse</i>
lg	Large ≥992px	 landscape tablets
xl	Extra large ≥1200px	 laptops, desktops, TVs

	Extra small <576px	Small ≥576px	Medium ≥768px	Large ≥992px	Extra large ≥1200px
Max container width	None (auto)	540px	720px	960px	1140px
Class prefix	.col-	.col-sm-	.col-md-	.col-lg-	.col-xl-
# of columns	12				
Gutter width	30px (15px on each side of a column)				

Le principe de grille consiste à découper votre page en cellules, comme dans un tableau, donc à créer des lignes (classes `.row`) et des colonnes (classes `.col-X`). Pour affiner la mise en forme, on utilise souvent des conteneurs (classes `.container` ou `.container-fluid`) dans lesquels nos grilles seront formatées.

Bootstrap découpe la largeur d'écran en **12 blocs de largeurs égales** (égales mais variables selon la largeur du dispositif sur lequel on affiche). Les classes `col-` vont spécifier combien de blocs on veut utiliser.

Conteneurs

La classe CSS `container` permet d'encadrer un bloc de contenu, avec une largeur par défaut de 980px max sur PC (ce qui veut donc dire que sur un écran plus grand que 980px, on obtient un effet de centrage, avec des marges latérales de plus en plus grandes).

Dans ces conteneurs, on peut ensuite placer le reste des outils de grille, lignes (= rangées) et colonnes. Par exemple :

```
<div class="container">
  <!-- [...] -->
</div>
```

NB: il existe une seconde classe de type conteneur : `container-fluid`, permettant d'utiliser toute la largeur de l'écran (100%)

Pour harmoniser une page HTML, on utilise **au moins** un conteneur ; dans ce cas, celui-ci est placé immédiatement sous la balise `<body>` et se termine à la fin de la page (juste avant la fermeture de `</body>`).

Il est néanmoins possible d'utiliser plusieurs conteneurs dans une page, y compris en mixant les classes `container` et `container-fluid`, et en imbriquant les conteneurs les uns dans les autres !

Les lignes (= rangées)

On crée une rangée (ligne) en appliquant la classe `.row` à un élément. Depuis l'exemple ci-dessus, on peut créer une rangée en affectant cette classe à une balise `<div>`, dans notre conteneur :

```
<div class="container">
  <div class="row">
    <!-- [...] -->
  </div>
</div>
```

NB: Chaque élément contenant la classe `.row` entrainera un retour à la ligne au sein de l'élément parent !

Les colonnes

Bootstrap découpe l'écran en **12 blocs de largeur égale, et proportionnelle** à la largeur du dispositif.

Les classes **col-x** vont ensuite nous permettre spécifier combien de blocs on veut occuper.

Toujours dans notre exemple, on ordonne nos éléments en colonnes en appliquant la classe **col** à une ou plusieurs balises **<div>** :

```
<div class="container">
  <div class="row">
    <div class="col-12">Une (seule) colonne occupant toute la largeur, soit 12 blocs</div>
  </div>
</div>
```

Testez l'exemple !

Si l'on veut maintenant aligner plusieurs éléments :

```
<div class="container">
  <div class="row">
    <div class="col-8">Une première colonne qui utilise 8 blocs</div>
    <div class="col-4">Une seconde colonne qui utilise les 4 blocs restants</div>
  </div>
</div>
```

Dans cet exemple, il n'y a que 2 colonnes, une de 8 blocs et une de 4 blocs, le total étant bien de 12 ; les deux éléments sont alignés sur notre page !

Il est aussi possible d'aller plus loin :

```
<div class="container">
  <div class="row">
    <div class="col-8">Une première colonne de 8 blocs</div>
    <div class="col-4">Une seconde colonne de 4 blocs</div>
    <div class="col-6">Une troisième colonne de 6 blocs, sur une 2e ligne puisque les 12 colonnes précédentes ont été occupées</div>
    <div class="col">Une quatrième colonne, occupant la largeur disponible sur la ligne en cours, soit 6 blocs</div>
  </div>
</div>
```

Testez l'exemple !

Ici, nous avons 24 blocs (2 x 12) :

- la première ligne est occupée par les deux premières **<div>** : $8 + 4 = 12$ blocs ; les deux éléments sont alignés
- la **<div class="col-6">** est renvoyée sur une seconde ligne pour occuper l'espace dont elle a besoin
- la classe **.col** en fin de bloc permet à sa **<div>** d'occuper automatiquement le nombre de blocs restant sur la ligne, soit ici, 6 blocs encore

Il existe de nombreuses possibilités de découpe, le total de blocs utilisés devant toujours être égal à 12. Par exemple :

- 3 colonnes de 4 blocs
- 6 colonnes de 2 blocs
- 2 colonnes de 6 blocs
- 1 colonne de 7 blocs et 1 colonne de 5 blocs etc.

Allons encore plus loin :

- Chaque grille peut être **redécoupée à volonté**, en créant par exemple dans une colonne une rangée, elle-même redécoupée en colonnes !
- Les colonnes peuvent dimensionnées automatiquement,
 - en indiquant uniquement la classe `.col` (cf. ci-dessus),
 - ou encore en utilisant `.col-auto` ; dans ce cas la colonne est ajustée à la largeur de son contenu
- Des colonnes peuvent être "sautées" via la classe `.offset-X` avant une classe `.col`, dans le même élément

Ces 3 points permettent d'effectuer des mises en page assez complexes !
[documentation sur la grille](#)

Affichage responsive

Toute la "magie" de Bootstrap est ici ! Nous allons mettre en œuvre les **points de rupture, ou breakpoints**.

Reprenons l'exemple précédent avec 2 colonnes :

```
<div class="container">
  <div class="row">
    <div class="col-8">Une première colonne qui utilise 8 blocs</div>
    <div class="col-4">Une seconde colonne qui utilise les 4 blocs
restants</div>
  </div>
</div>
```

Ce code s'affiche de la même façon sur tous les dispositifs. Mais on pourrait souhaiter obtenir un affichage différent selon la largeur d'écran !

Par exemple :

- **Règle 1** - Une page présentant deux éléments : une partie centrale, d'une largeur de 8 blocs et une seconde colonne de largeur 4 sur les grands écrans ;
- **Règle 2** - Pour les écrans supérieurs à 992px, la colonne de droite doit être un peu plus large ; on aura donc une largeur de 9 pour la partie centrale et une largeur de 3 pour l'autre colonne ;
- **Règle 3** - Sur tablettes, en mode portrait, la colonne principale aura plutôt une largeur de 7 blocs et la seconde occupera les 5 blocs restant ;
- **Règle 4** - Nos 2 colonnes auront une largeur égale (6 blocs) pour les dispositifs à partir de 576px et jusqu'au format tablette / portrait (768px) ;
- **Règle 5** - sur les mobiles enfin, l'affichage se fait sur une seule colonne, les deux éléments se retrouvant l'un en-dessous de l'autre

Pour cela, on va ajouter un préfixe de **point de rupture** entre le **col-** et le nombre de blocs.

Par exemple, la classe **.col-md-7** va indiquer que pour les dimensions d'écran à partir de **md**, c'est-à-dire de 768 pixels, la colonne occupera 7 blocs.

Ce qui nous donne, pour l'affichage voulu :

```
<div class="container">
  <div class="row">
    <div class="col-12 col-sm-6 col-md-7 col-lg-8 col-xl-9">Colonne 1</div>
    <div class="col-12 col-sm-6 col-md-5 col-lg-4 col-xl-3">Colonne 2</div>
  </div>
</div>
```

Testez l'exemple.

Rappel : Bootstrap étant *Mobile first*, on part d'abord de la dimension la plus petite, celle pour mobile (< 576px), pour aller vers la plus grande (xl).

Ici on a toujours 2 colonnes, sauf sur mobile puisqu'on a indiqué que les colonnes y occupaient les 12 blocs disponible (la seconde colonne se placera alors sous la 1^{ère}).

Remarque : on a indiqué ici des souhaits pour tous les points de rupture, mais ce n'est pas obligatoire !

En effet, un *breakpoint* (point de rupture) s'applique à une taille d'écran minimale et **pour toutes les tailles d'écran supérieures**.

Cela signifie que si, dans notre exemple, nous n'avions pas précisé de répartition pour les tailles d'écran **-lg**, c'est la règle inférieure qui se serait appliquée (= **md**), autrement-dit : 7 + 5.

Masquer une zone ?

Sur mobile, il peut être utile de ne pas afficher une zone, pour améliorer le temps de chargement ou parce que le contenu n'est pas pertinent (publicité par exemple).

Bootstrap propose un ensemble de classes pour afficher/masquer des éléments HTML selon les *breakpoints*, basées sur les classes **display** (= *affichage*) du CSS.

Par exemple :

```
<div class="d-none d-sm-block">
    Ce bloc sera masqué par défaut, pour tout (d-none).
    L'affichage sera rétabli à partir de -sm, c'est-à-dire pour les écrans plus
    grands que 576px.
    Cela signifie plus généralement que la zone sera masquée pour les mobiles,
    mais visible pour tous les écrans plus larges.
</div>

<div class="d-lg-none d-xl-block">
    Ce bloc sera masqué à partir de la plage de breakpoints -lg (>= 992px), puis
    rétabli à partir des écrans -xl, soit à partir de 1199px.
    En-dessous de la plage -lg (jusqu'à 991px), la zone sera donc visible aussi !
</div>
```

Explications :

- le préfixe **d** signifie *display*, c'est-à-dire *affichage* ;
- **none** signifie *aucun* ; autrement dit, **d-none** est utilisé pour *masquer* (équivalent de **display: none** en CSS) ;
- **block** signifie *en bloc* ; autrement dit, **d-block** est utilisé pour *afficher* (équivalent de **display: block** en CSS) ;
- les préfixes de breakpoints, **-xs**, **-sm** etc. indiquent à partir de quelle(s) plage(s) d'écran afficher / masquer les éléments ;
- s'il n'y a pas de préfixe de classes (par exemple **d-none**), la règle s'applique par défaut à toutes les tailles d'écran, sauf mention contraire annulant cette règle (exemple du bloc 1 : masqué par défaut mais visible à partir des écrans de taille 576px et supérieur).

Documentation

CSS 08

Réalisation : Guillaume DELACROIX Formateur AFPA

06 janvier 2023

Afpa

Marges (spacing)

Bootstrap gère les marges externes (appelées **margins**) et internes (appelées **padding**s) de chaque élément avec des classes représentées par un système de lettres :

- **m** pour margin
- **p** pour padding

Ensuite, d'autres lettres peuvent venir préciser sur quel(s) côté(s) s'applique(nt) la marge :

- **t** pour *top* (*margin-top* ou *padding-top*) : en **haut**.
- **b** pour *bottom* (*margin-bottom* ou *padding-bottom*) : en **bas**.
- **l** pour *left* (*margin-left* ou *padding-left*) : à **gauche**.
- **r** pour *right* (*margin-right* or *padding-right*) : à **droite**.
- **x** pour l'**axe horizontal**, c'est-à-dire à la fois à gauche et à droite.
- **y** pour l'**axe vertical**, c'est-à-dire à la fois en haut et en bas.

Enfin, il faut indiquer la taille de la marge, sur une échelle de 0 à 5.

2 valeurs particulières :

- **0** : pas de marge ou de padding
- **auto** : marges automatiques appliquées par le CSS à l'élément.

Exemples

```
<p class="mt-5 bg-info">Un paragraphe avec une marge supérieure de 5.</p>
<p class="pl-5 bg-info">Un paragraphe avec une marge interne gauche de 5.</p>
<p class="mx-3 bg-info">Un paragraphe avec une marge externe gauche et droite de 3.</p>
<p class="py-3 bg-info">Un paragraphe avec une marge externe en haut et en bas de 3.</p>
<p class="mt-0 bg-info">Un paragraphe SANS marge externe supérieure.</p>
<p class="mx-auto bg-info">Un paragraphe avec une marge externe automatique à gauche et à droite, qui sera donc centré</p>
```

Pour appliquer une marge ou un padding sur les 4 côtés, aucune indication n'est à spécifier avant la taille de marge :

```
<p class="m-5 bg-info">Un paragraphe avec une marge de 5 sur les 4 côtés.</p>
<p class="p-3 bg-danger">Un paragraphe avec un padding de 3 sur les 4 côtés.</p>
```

Il est bien sûr possible de spécifier à la fois une marge externe et une marge interne :

```
<p class="m-5 p-3 bg-info">Un paragraphe avec une marge externe de 5 sur les 4 côtés et un padding de 3 sur les 4 côtés.</p>
```

Bootstrap permet aussi d'adapter les marges en fonction des *breakpoints* ; pour cela il faut ajouter le préfixe du breakpoint voulu :

```
<p class="px-sm-2 px-md-3 bg-info">Un paragraphe avec une marge interne à gauche et à droite de 2 pour le breakpoint sm (>= 576 px) puis de 3 à partir du breakpoint md (>= 768 px).</p>
```

Testez les différents exemples.

Tableaux

Bootstrap fournit un ensemble de classes pour habiller les tableaux HTML.

Pour commencer, il convient d'ajouter la classe `table` à la balise HTML `<table>` :

```
<table class="table">
  [ ... ]
</table>
```

Différentes classes peuvent ensuite y être ajoutées :

Classe Rôle

- `.table-bordered` : affiche les bordures
- `.table-striped` : ajoute un fond de couleur à une ligne sur deux pour faciliter la lisibilité
- `.table-hover` : surligne la ligne en cours de survol de la souris.

Les tableaux HTML doivent eux aussi être responsive : pour cela il est nécessaire d'englober tout le tableau dans un `<div>` supplémentaire avec en attribut classe `table-responsive`.

Exemple :

```
<div class="table-responsive">
  <table class="table table-striped table-hover table-bordered">
    <thead>
      <tr>
        <th>Nom</th>
        <th>Prénom</th>
        <th>Adresse</th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td>LOPER</td>
        <td>Dave</td>
        <td>30 rue de Poulainville 80000 Amiens</td>
      </tr>
      <tr>
        <td>CHARPENTIER</td>
        <td>Joseph</td>
        <td>12 place de la grotte 99000 Bethléem</td>
      </tr>
    </tbody>
  </table>
</div>
```

Testez l'exemple

Bootstrap propose bien d'autres options pour les tableaux : [documentation](#).

Images

Bootstrap fournit des classes pour placer, habiller (bordures, arrondis) ou redimensionner les **images**.

Par exemple, afin de les rendre responsive, il suffit d'appliquer la classe `img-fluid` sur la balise `` tout en veillant à ne pas spécifier de dimensions fixes (= suppression des attributs `width` et `height`) ; ce qui donne :

```

```

La classe `img-fluid` applique les règles CSS suivantes aux balises `` :

- `max-width: 100%;` : fixe la largeur de l'image à celle de son parent (pour résumer).
- `height: auto;` : adapte la hauteur de l'image en fonction de sa largeur, afin de ne pas obtenir de distorsion de l'image.

NB: En dehors de cette classe et de Bootstrap, [d'autres solutions](#) existent pour gérer la problématique responsive des images : plusieurs dimensions d'une même image (attributs `srcset` et `sizes`, balise `<picture>`), propriétés css (`cover` etc.).

En outre, Bootstrap fournit d'autres classes applicables aux images pour gérer les miniatures, les arrondis ou encore leur placement.

[Documentation](#)

Boutons

Bootstrap permet d'habiller des boutons !

La classe principale est la classe `.btn`, qui ajoute des marges et une bordure arrondi au bouton classique :

Bouton "classique"

Bouton avec classe `.btn`

De même, il est également possible d'utiliser des classes couleur et de taille.

Exemples :

Un bouton avec fond bleu et texte blanc :

```
<button type="button" class="btn btn-primary">bouton bleu</button>
```

Résultat : Bouton bleu

Un bouton vert, avec ajout de la classe `btn-lg` pour obtenir un bouton plus grand :

```
<button type="button" class="btn btn-success btn-lg">Grand bouton vert</button>
```

Résultat : Grand bouton vert

Ces classes de boutons peuvent être utilisées sur les liens (balises `<a>`) :

```
<a class="btn btn-warning btn-sm">Lien en forme de petit bouton orange</a>
```

Résultat : Lien en forme de petit bouton orange

[Documentation](#)

Formulaires

Un champ de formulaire standard (`input type="text"`) est défini par le code suivant dans Bootstrap :

```
<form>
  <div class="form-group">
    <label for="courriel">Adresse mail</label>
    <input type="email" class="form-control" name="mail_address" id="courriel"
placeholder="Saisir votre adresse mail">
  </div>
</form>
```

- Les champs `input` et le `label` associé sont groupés dans un `<div>` qui prend en attribut la classe `form-group`
- Les champs `input` prennent une classe `form-control`.

Ensuite, il existe différentes variantes, dont des classes permettant

- d'afficher des champs désactivés (`disabled`), ou en lecture seule (`readonly`)
- d'aligner des champs en ligne (classes `-inline`), au lieu de les mettre les uns sous les autres
- de personnaliser les champs (avec des logos, des labels alignés)
- d'ajouter des *feedbacks* pour valider les contrôles de saisie
- de gérer les couleurs, les tailles
- d'adapter les affichages pour que le formulaire soit *responsive*
- etc.

[Documentation](#)

Messages (alertes)

Bootstrap propose un système de messages (boîtes d'alerte, ou *alert boxes*) pour communiquer avec l'utilisateur :

```
<div class="alert alert-warning alert-dismissible fade show" role="alert">
  <h4 class="alert-heading">La classe <b>.alert</b> permet ce format d'affichage
!</h4>
  <small><i><b>.alert-heading</b> dans la balise < h4 > indique qu'il s'agit de
l'entête.</i></small>
  <hr>
  <p class="mb-0"><b>.alert-warning</b> gère la couleur de fond du message.</p>
  <p class="mb-0"><b>.alert-dismissible</b> adapte l'emplacement du bouton de
fermeture de l'alerte.</p>
  <p class="mb-0"><b>.fade</b> rend un effet dégradé lors de la disparition du
message.</p>
  <p class="mb-0"><b>.show</b> permet de rendre l'alerte visible.</p>
  <br>
  <p class="mb-0"><a href="#" class="alert-link">La classe alert-link utilisée
ici permet d'insérer un lien hypertexte.</a></p>
  <br>
  <p>Enfin, dans le bouton, la classe <b>.close</b> applique le rôle de
fermeture pour l'élément ayant le <b>role="alert"</b> (renseigné dans <i>data-
dismiss</i>),
    <button type="button" class="close" data-dismiss="alert">
      <span>&times;</span>
    </button>
</div>
```

Résultat :

La classe .alert permet ce format d'affichage !

.alert-heading dans la balise < h4 > indique qu'il s'agit de l'entête.

.alert-warning gère la couleur de fond du message.

.alert-dismissible adapte l'emplacement du bouton de fermeture de l'alerte.

.fade rend un effet dégradé lors de la disparition du message.

.show permet de rendre l'alerte visible.

La classe alert-link utilisée ici permet d'insérer un lien hypertexte.

Enfin, dans le bouton, la classe **.close** applique le rôle de fermeture pour l'élément ayant le **role="alert"** (renseigné dans *data-dismiss*), ×

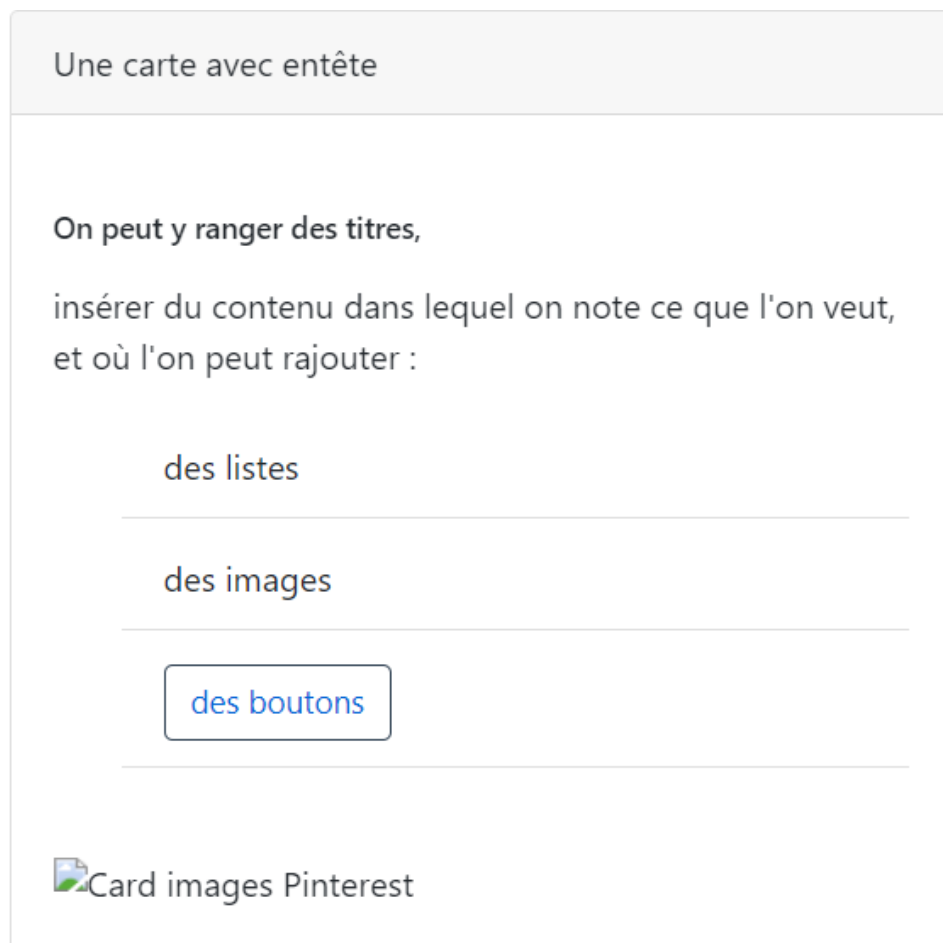
[Documentation](#)

Cards

Bootstrap, à partir de la version 4, apporte un nouveau composant : les **cartes**, qui permettent l'habillage de blocs de contenus:

```
<div class="row mt-5">
  <div class="offset-1 col-5">
    <div class="card">
      <div class="card-header">
        Une carte avec entête
      </div>
      <div class="card-body">
        <h5 class="card-title">On peut y ranger des titres,</h5>
        <p class="card-text">insérer du contenu dans lequel on note ce que
l'on veut, et où l'on peut rajouter :</p>
        <ul class="list-group list-group-flush mb-3">
          <li class="list-group-item">des listes</li>
          <li class="list-group-item">des images</li>
          <li class="list-group-item">
            <a href="#" class="btn btn-primary">des boutons</a>
          </li>
        </ul>
        <br>
        
      </div>
    </div>
  </div>
</div>
```

Résultat :



Vous trouverez [ici, des exemples de cartes personnalisées](#).
A vous de trouver l'habillage qui correspond à vos besoins !

[Documentation](#)

edit: depuis la version 5 de Bootstrap également, les **médias** sont une alternative aux éléments *cards*.

[Documentation](#)

Typographie

Bootstrap fournit des classes pour gérer la typographie : taille et alignement de polices, stylisation (gras, italique, barré, etc.), mises en exergue (`<mark>`, ``), citations (`.blockquotes`), habillage des listes (``, ``), etc.

Voici quelques exemples des outils proposés :

Polices par défaut

La police de caractère affichée par défaut varie en fonction du [système d'exploitation](#), par exemple :

- *Sego UI* sous Windows,
- *Roboto* sous Android

La police par défaut peut bien entendu être changée.

Taille de police

Bootstrap définit par défaut la taille de police à **16 pixels**, donc une unité de taille fixe. En effet, Bootstrap ne gère pas le redimensionnement des textes pour qu'ils deviennent « responsive » : il faudra ajouter des media queries spécifiques.

Néanmoins, Bootstrap propose

- 6 classes *headings*, notées `.hX` et offrant le même rendu que les balises du même nom
- 4 classes *display*, notées `.display-X` et permettant d'agrandir un texte

```
<p class="h1">h1 heading</p>
<p class="h2">h2 heading</p>
<p class="h3">h3 heading</p>
<p class="h4">h4 heading</p>
<p class="h5">h5 heading</p>
<p class="h6">h6 heading</p>
<p class="display-1">Display 1</p>
<p class="display-1">Display 1</p>
<p class="display-2">Display 2</p>
<p class="display-3">Display 3</p>
<p class="display-4">Display 4</p>
```

Testez l'exemple

Alignement

Il existe des classes pour l'alignement du texte, équivalents de `text-align` en CSS :

- `text-left` : aligne le texte à gauche de son conteneur (par défaut)
- `text-right` : aligne le texte à droite de son conteneur
- `text-center` : centre le texte dans son conteneur
- `text-justify` : justifie le texte

Exemple :

```
<div class="row">
  <div class="col-12">
    <p class="text-left">Texte aligné à gauche (normal)</p>
    <p class="text-right">Texte aligné à droite</p>
    <p class="text-center">Texte centré</p>
    <p class="text-justify">Texte justifié. Lorem ipsum dolor sit amet,
consectetur adipisicing elit. Donec eget mi lobortis, bibendum nibh eu, semper quam.
In eget felisurna. Maecenas luctus lacus quis semper semper. Donec pellentesque
ligula tortor, utpulvinar massa consequat ac.</p>
  </div>
</div>
```

Ces classes peuvent être spécifiées pour un dispositif en particulier grâce aux préfixes de breakpoints : par exemple `text-sm-center` centrera le texte uniquement à la dimension correspondant à `sm`, c'est-à-dire à partir de 576 pixels.

- [Documentation](#)

Mise en pratique

C'est parti pour le projet du groupe AC/DC !!!!

Ressources

- [Bootstrap](#)
- [w3schools](#)
- **Hackerthemes** : [Bootstrap4 Cheat Sheet](#) et [Bootstrap 4 Buffet](#) "Bootstrap 4 Buffet"), permet de sélectionner des éléments Bootstrap plus facilement.