Bootstrap

1ère édition

Support du Cours & Travaux Pratiques

Noureddine ABABAR

Apprendre à utiliser le framework Bootstrap

Bienvenue dans ce cours complet traitant du framework CSS le plus célèbre au monde : Bootstrap. Ce framework nous fournit une liste d'outils pour simplifier la création de design de sites et d'applications Web.

Prérequis pour suivre ce cours Bootstrap

Le framework Bootstrap est un framework CSS. La connaissance du CSS est donc un prérequis indispensable à l'utilisation de ce framework. De plus, la plupart des composants Bootstrap utilisent également du code JavaScript; une connaissance de base de ce langage est donc également conseillée même si elle n'est pas strictement obligatoire.

Je vais donc considérer pour ce cours que vous disposez d'une bonne maitrise du CSS (et du HTML) et d'une connaissance basique du JavaScript et ne réexpliquerai pas les concepts de base relatifs à ces langages.

Pour tirer pleinement profit de ce cours, je vous recommande donc fortement d'acquérir au préalable une bonne connaissance et une bonne maitrise des langages **HTML**, **CSS** et **JavaScript**.

Table des matières

APPRENDRE A UTILISER LE FRAMEWORK BOOTSTRAP 1
FONCTIONNEMENT DE BASE DU FRAMEWORK BOOTSTRAP 3
PRESENTATION DU SYSTEME DE GRILLES DE BOOTSTRAP 7
GERER L'ORDRE D'AFFICHAGE DES COLONNES DANS UNE GRILLE BOOTSTRAP 18
METTRE EN FORME DES TEXTES AVEC LES CLASSES BOOTSTRAP 19
MODIFIER LA COULEUR, LA COULEUR DE FOND AVEC BOOTSTRAP 22
AJOUTER DES BORDURES AUX ELEMENTS AVEC BOOTSTRAP 25
GERER LES DIMENSIONS DES ELEMENTS AVEC BOOTSTRAP 28
AJOUTER DES MARGES AUX ELEMENTS AVEC BOOTSTRAP 29
L'AFFICHAGE DES ELEMENTS : BOOTSTRAP DISPLAY ET DISPLAY : FLEX 31
GERER LE TYPE DE POSITIONNEMENT DES ELEMENTS AVEC BOOTSTRAP 31
CREER DES TABLEAUX STYLISES AVEC BOOTSTRAP 32
CREER DES BOUTONS STYLISES AVEC BOOTSTRAP 35
CREER DES GROUPES DE LISTES AVEC BOOTSTRAP 37
STRUCTURER ET STYLISER DES FORMULAIRES AVEC BOOTSTRAP 40
CREER UN MENU DE NAVIGATION AVEC BOOTSTRAP 44
CREER DES ELEMENTS ET DES MENUS DEROULANTS AVEC BOOTSTRAP 47
PRESENTATION DES BARRES DE NAVIGATION ET DE LA CLASSE .NAVBAR 49
LES AUTRES COMPOSANTS DE NAVIGATION: BREADCRUMB ET PAGINATION 51
ALERTES, BOITES MODALES ET NOTIFICATIONS TOAST BOOTSTRAP 53
BARRES DE PROGRESSION ET SPINNERS BOOTSTRAP 56
BADGES BOOTSTRAP 59
LES CARTES OU CARDS BOOTSTRAP 61
TRAVAUX PRATIQUES 63
TP N° 1 : INTEGRER UN TEMPLATE BOOTSTRAP 5 63

FONCTIONNEMENT DE BASE DU FRAMEWORK BOOTSTRAP

Qu'est-ce que Bootstrap ?

Bootstrap est un framework CSS. Un framework correspond à un ensemble de librairies regroupées dans un but précis et possédant des règles internes que doivent suivre les utilisateurs.

En d'autres termes, et pour le dire très simplement, Bootstrap est un ensemble de fichiers CSS et JavaScript fonctionnant ensemble et qu'on va pouvoir utiliser pour créer des design complexes de manière relativement simple.

Le framework Bootstrap est donc un ensemble de fichiers CSS et JavaScript qui contiennent des règles prédéfinies et qui définissent des composants. Ces ensembles de règles sont enfermés dans des classes et nous n'aurons donc qu'à utiliser les classes qui nous intéressent afin d'appliquer un ensemble de styles à tel ou tel élément HTML.

Pourquoi utiliser Bootstrap ?

Bootstrap possède trois grands avantages notables par rapport aux autres solutions à notre portée qui se limitent concrètement à écrire tout son code soi-même ou à avoir recours à un autre framework ou librairie CSS.
Ces avantages sont :

1. Un gain de temps de développement qui peut être conséquent ;
2. Une certaine robustesse dans l'architecture globale du code ;
3. Un framework appartenant à une grande société (Twitter).

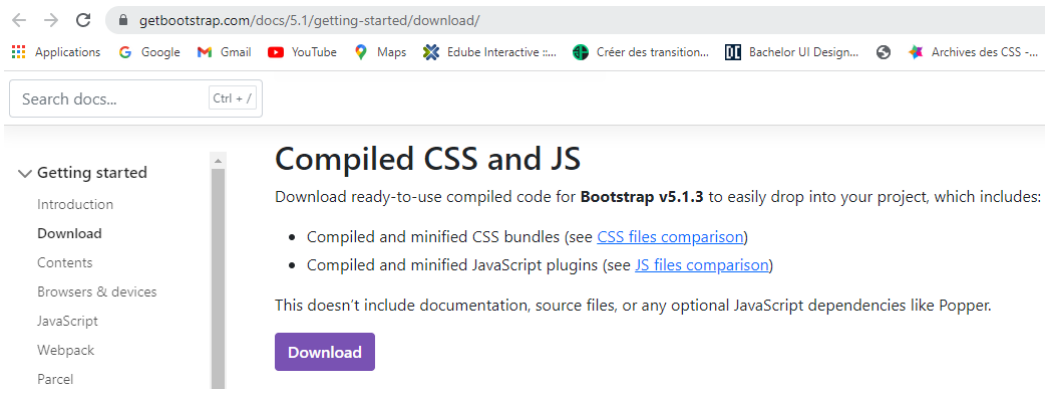
Comment utiliser Bootstrap ?

Bootstrap est un framework, c'est-à-dire un ensemble de fichiers. Pour utiliser Bootstrap, nous allons donc simplement devoir utiliser ces fichiers. On va pouvoir faire cela de deux façons. On peut :

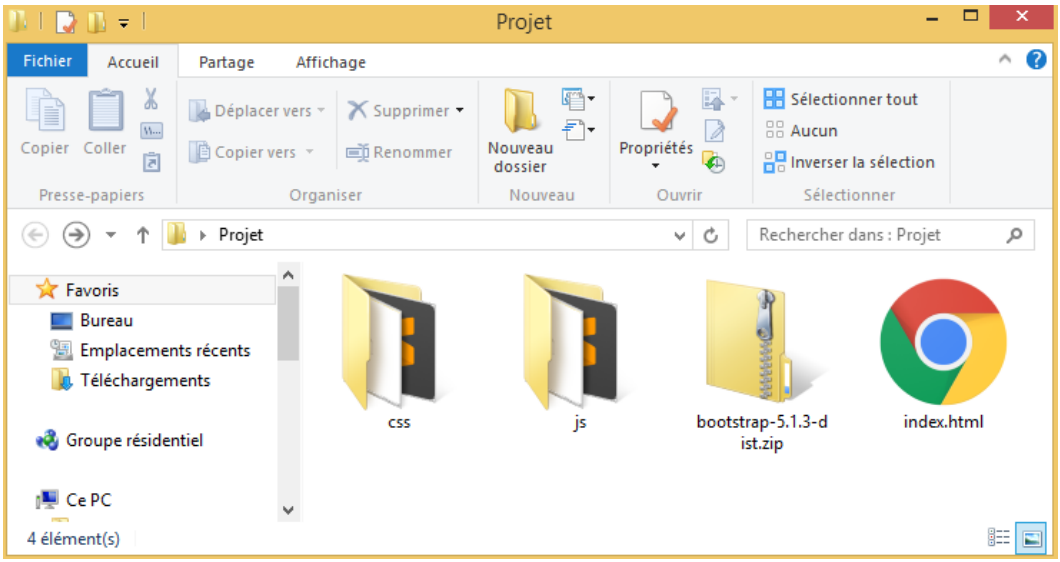
1. Télécharger les fichiers Bootstrap (CSS et JavaScript) sur le site https://getbootstrap.com/ puis les lier à nos fichiers HTML comme n'importe quel autre fichier CSS et JavaScript ;
2. Utiliser un CDN (Content Delivery Network ou réseau de distribution de contenu) et inclure le lien vers les fichiers dans nos fichiers HTML.

Méthode 1 :

1. Télécharger les fichiers Bootstrap(CSS et JavaScript) sur le site https://getbootstrap.com/docs/5.1/getting-started/download/



2. Décompresser le fichier téléchargé dans votre dossier du projet



Lier les fichiers CSS et JavaScript à nos fichiers HTML comme n'importe quel autre fichier CSS et JavaScript :

```
<!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=1.0">
        <title>Projet Bootstrap</title>
        <link rel="stylesheet" href="css/bootstrap.css">
    </head>
    <body>

        <script src="js/bootstrap.js"></script>
    </body>
</html>
```

Méthode 2 :

1. Utiliser un CDN (Content Delivery Network ou réseau de distribution de contenu) et inclure le lien vers les fichiers dans nos fichiers HTML.

```
<!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=1.0">
        <title>Projet Bootstrap</title>
        <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3"
crossorigin="anonymous">
    </head>
```

```
    <body>

        <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min
.js" integrity="sha384-
ka7Sk0Gln4gmtz2MlQnikT1wXgYsOg+OMhuP+IlRH9sENBO0LRn5q+8nbTov4+1p"
crossorigin="anonymous"></script>
    </body>
</html>
```

Présentation du fonctionnement de Bootstrap

Dans cette leçon, nous allons comprendre comment utiliser Bootstrap pour mettre en forme nos fichiers HTML et nous intéresser aux styles CSS directement appliqués à nos éléments.

Nous allons également discuter de la prise en charge du responsive avec Bootstrap ainsi que de l'ordre d'affichage des différents composants.

Le système d'utilisation des classes

La partie CSS pure de Bootstrap dispose de deux fichiers principaux : un fichier utilisant exclusivement des sélecteurs éléments et un fichier n'utilisant que des sélecteurs de classes.

Le principe de fonctionnement de Bootstrap est à la fois simple et efficace : le fichier de sélecteurs éléments va permettre de définir des styles de base qui vont être automatiquement appliqués aux éléments afin de normaliser le comportement de ceux-ci quel que soit le navigateur utilisé par nos visiteurs tandis que le fichier de classes va permettre de grouper des ensembles dans de styles dans des sélecteurs de classes.

Dans tous les cas, nous ne toucherons à aucun de ces deux fichiers directement. Nous allons simplement nous contenter d'ajouter des attributs `class` relatifs aux sélecteurs de classe définis dans le fichier Bootstrap CSS afin de leur appliquer les styles CSS liés.

Le fichier de classes CSS Bootstrap définit par exemple un sélecteur de classe `.container` et lui attribue les styles suivants :

```
.container {
  width: 100%;
  padding-right: var(--bs-gutter-x, 0.75rem);
  padding-left: var(--bs-gutter-x, 0.75rem);
  margin-right: auto;
  margin-left: auto;
}
@media (min-width: 576px) {
  .container { max-width: 540px; }
}
@media (min-width: 768px) {
  .container { max-width: 720px; }
}
```

```
@media (min-width: 992px) {
  .container { max-width: 960px; }
}
@media (min-width: 1200px) {
  .container { max-width: 1140px; }
}
@media (min-width: 1400px) {
  .container { max-width: 1320px; }
}
```

L'un des objectifs principaux de ce cours est de vous présenter la plupart des classes du fichier de classes Bootstrap et de vous apprendre à les utiliser de la meilleure façon.

La mise en place du mobile first et du responsive

Bootstrap a été pensé avec l'idée « mobile first », ce qui signifie que les règles CSS sont créées pour les affichages mobiles. On utilise ensuite un système de « breakpoints » (points d'arrêt) qui reposent sur l'utilisation de règles `@media` pour définir les affichages sur les écrans plus grands.

En reprenant l'exemple de notre classe `.container` ci-dessus, on peut voir que Bootstrap définit déjà des styles généraux liés à ce sélecteur puis utilise ensuite différentes règles `@media` pour modifier les styles de la classe en fonction d'un critère de taille d'écrans.

L'ordre général d'affichage des composants et des éléments

Bootstrap fournit différents composants, c'est-à-dire différents ensembles de code déjà prêt à l'emploi et qui nous permettent d'afficher des fenêtres modales, des barres de navigation, etc. sans avoir à écrire une ligne de code.

Bootstrap utilise la propriété `z-index` pour définir quel composant doit apparaitre au-dessus de quel autre dans le cas où on en insèrerait plusieurs dans une même page. Les valeurs du `z-index` des composants sont les suivants :

```
.dropdown-menu { z-index: 1000;}
.sticky-top { z-index: 1020; }
.fixed-top, .fixed-bottom { z-index: 1030; }
.modal-backdrop { z-index: 1050; }
.modal { z-index: 1055; }
.tooltip { z-index: 1080; }
```

De plus, une valeur de `z-index` est également définie en fonction des états des éléments en cas de chevauchement de ceux-ci :

- 1 par défaut
- 2 pour hover
- 3 pour active

Présentation du système de grilles de Bootstrap

Dans cette leçon, nous allons comprendre les grands principes de disposition et de design sur lesquels Bootstrap repose et voir comment les appliquer à nos pages et éléments HTML.

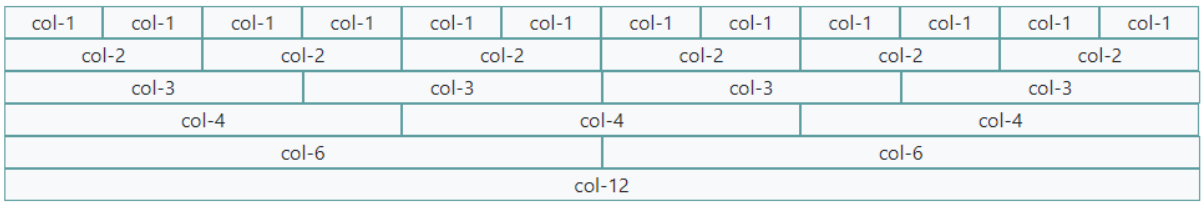
Le système de grille de Bootstrap 5 appelé grid en anglais est l'un des points forts du Framework.

Le positionnement des éléments sur une page peut vite devenir fastidieux notamment lorsqu'on se lance dans la réalisation d'un site complexe et responsive.

C'est là que Bootstrap 5 montre toute l'étendue de son potentiel en créant des grilles qui vont s'adapter automatiquement aux résolutions d'écrans de vos lecteurs.

De plus, il vous sera très facile de changer le design de votre site en fonction de cette même résolution par le biais des breakpoints (points de transition).

La grille de Bootstrap 5 se divise en un maximum de 12 colonnes responsives qui s'étendent sur toute la largeur de la rangée dans laquelle elles sont placées.



Nous allons voir dans cette première partie comment créer une page avec des éléments de différentes largeurs en jouant sur le nombre de colonnes contenues dans chaque rangée.

Les principales classes CSS pour les grilles de Bootstrap 5

Ce sont les classes que vous allez utiliser le plus pour positionner les colonnes sur la grille :

- .container et .container- fluid (conteneur).
- .row (rangée).
- .col (colonne). Sur ces classes vont venir se greffer d'autres classes que nous allons voir plus bas.

La différence entre les classes .container et .container-fluid

Les conteneurs sont fortement conseillés pour utiliser de manière efficace le système de grilles offert par Bootstrap 5. C'est à l'intérieur de ce dernier que vont venir se loger les rangées et les colonnes.

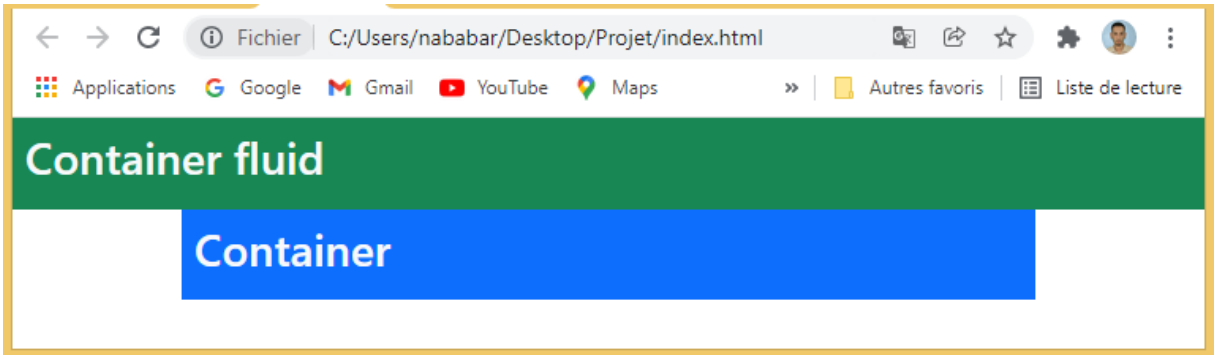
La classe .container va définir une taille fixe et responsive (fluide) pour chacun des breakpoints, c'est-à-dire que pour une même résolution d'écran le conteneur offrira toujours la même taille, mais par contre sa taille évoluera lorsqu'on changera de résolution.

La classe .container-fluid quant à elle va créer une taille fixe et responsive qui occupera 100% de la largeur, peu importe la résolution d'écran des visiteurs.

Exemple :

**Pour plus de clarté, j'ai rajouté une couleur de fond et les marges internes sur le conteneur et le body et mis le texte en blanc.*

```
<div class="container-fluid bg-success p-2">
    <h2 class="text-light">Container fluid</h2>
</div>
<div class="container bg-primary p-2">
    <h2 class="text-light">Container</h2>
</div>
```



Je vous conseille de vous pratiquer les différents conteneurs chez vous pour bien comprendre la différence entre les deux.

La classe .container-fluid est souvent utilisée pour les menus et barres de navigation, Header (entêté) et Footer (pied de page) qui occupe souvent 100% de la largeur et ne nécessite pas de marge.

La classe .container quant à elle et plus utilisée pour ce qui tout ce qui concerne le contenu de la page (situé entre le Header et Footer).

Création d'une grille avec des colonnes de tailles différentes

Bien souvent, nous voudrons que les différents éléments de notre grille occupent des largeurs différentes. On va pouvoir faire cela en indiquant le nombre de colonnes de base que doit occuper chaque colonne qu'on va créer dans la ligne.

Pour indiquer explicitement le nombre de colonnes de base que doivent occuper nos colonnes personnalisées, nous allons plutôt utiliser les classes .col-1, .col-2… .col-12.

En passant une classe .col-1 à un élément, on indique qu'on souhaite créer une colonne d'une taille équivalente à celle d'une colonne de base des grilles Bootstrap. En passant .col-2 on va créer une colonne qui va occuper l'espace de deux colonnes de base et etc.

Modifier la disposition des colonnes en fonction des écrans : le responsive

L'un des intérêts majeurs d'utiliser Bootstrap 5 est que cette version du framework a été développée en pensant à l'affichage sur mobile en premier et donc possède de nombreux outils permettant d'adapter nos designs en fonction de la taille de l'écran (ou plus exactement de la fenêtre) de nos visiteurs.

Avec les grilles, on va notamment pouvoir spécifier qu'une ligne doit posséder tel nombre de colonnes personnalisées qui vont chacune occuper tel nombre de colonnes de base pour une taille de fenêtre donnée puis que ces colonnes personnalisées doivent être réorganisées et occuper plutôt la place de tel autre nombre de colonnes de base pour cette colonne, telle autre nombre pour telle autre, etc. pour une autre taille de fenêtre.

En effet, les grilles Bootstrap vont avoir des tailles différentes en fonction de la taille de la fenêtre de vos visiteurs puisqu'elles dépendent de la taille du conteneur défini avec .container ou .container-fluid qui va dans les deux cas être un conteneur flexible.

Ainsi, les colonnes personnalisées d'une ligne vont pouvoir occuper un nombre différent de colonnes de base selon la taille de la fenêtre de chaque visiteur.

Cette fonctionnalité va être très utile pour proposer un affichage optimisé pour différentes tailles de fenêtre. Par exemple, on pourra vouloir qu'un élément occupe une largeur égale à 3 colonnes de base pour un grand écran puis à 6 colonnes pour une taille d'écran 2 fois plus petite par exemple afin que la taille effective de l'élément ne soit pas modifiée.

Pour faire cela, on va pouvoir utiliser les classes .col-sm, .col-md, .col-lg et .col-xl en plus de .col.

Ces classes sont liées aux **breakpoints** définis par Bootstrap. L'idée ici est très simple : selon la taille de la fenêtre, les styles d'une classe vont être appliqués prioritairement par rapport à ceux des autres.

Bootstrap 5 propose 6 Breakpoints différents pour combler tout le spectre des tailles d'écrans disponibles.

Pour mettre en place ces points de transition, nous allons utiliser les classes suivantes:

	xs <576px	**sm** ≥576px	**md** ≥768px	**lg** ≥992px	**xl** ≥1200px	**xxl** ≥1400px
Container `max-width`	None (auto)	540px	720px	960px	1140px	1320px
Class prefix	`.col-`	`.col-sm-`	`.col-md-`	`.col-lg-`	`.col-xl-`	`.col-xxl-`
# of columns	12					

Nous connaissons déjà la classe simple .col. Comme Bootstrap 5 a été construit sur le principe du « mobile first », c'est la classe par défaut qui va s'appliquer pour toutes les tailles de conteneur si nous ne précisons pas de règle plus précise.

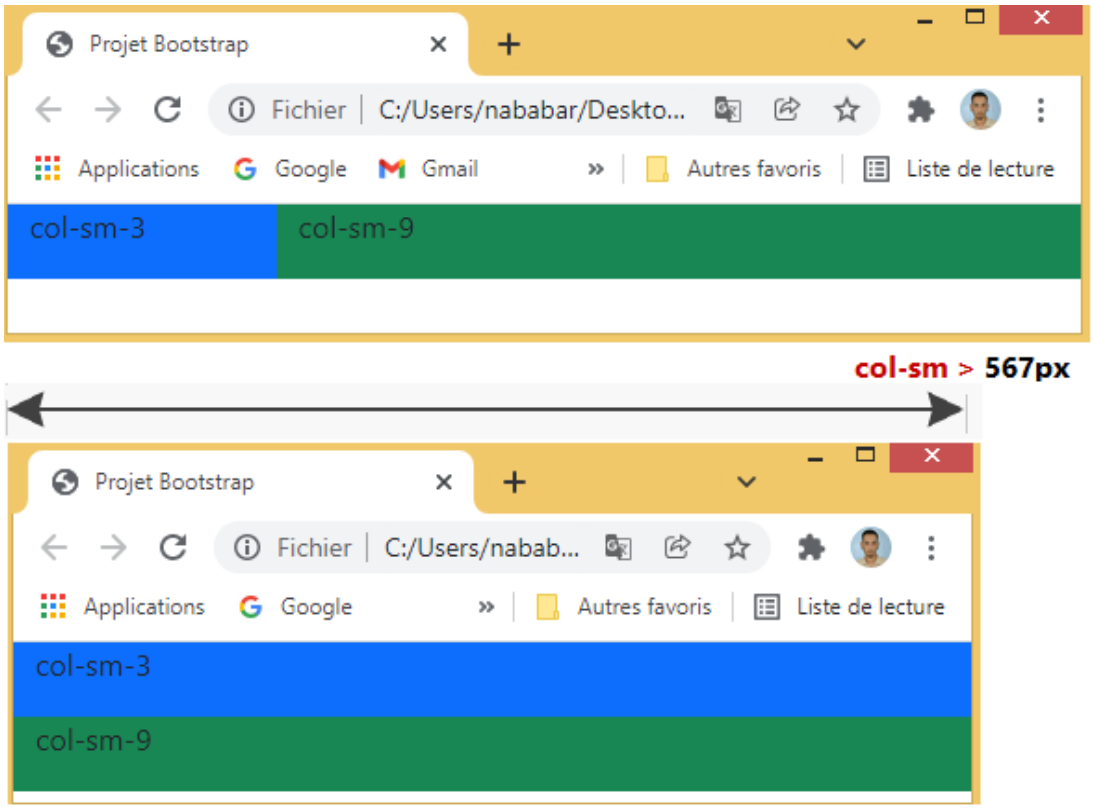
Par ailleurs, vous pouvez noter qu'il n'est pas strictement obligatoire d'utiliser une classe .col avec les éléments de ligne tant qu'au moins une autre classe .col-sm, .col-md, .col-lg ou .col-xl a été définie.

Les exemples sont le meilleur moyen d'expliquer le principe de fonctionnement du système de grille dans le Bootstrap.

col-sm-*

Pour les petits équipements (Small) dont la largeur est supérieure ou égale à **567px.**

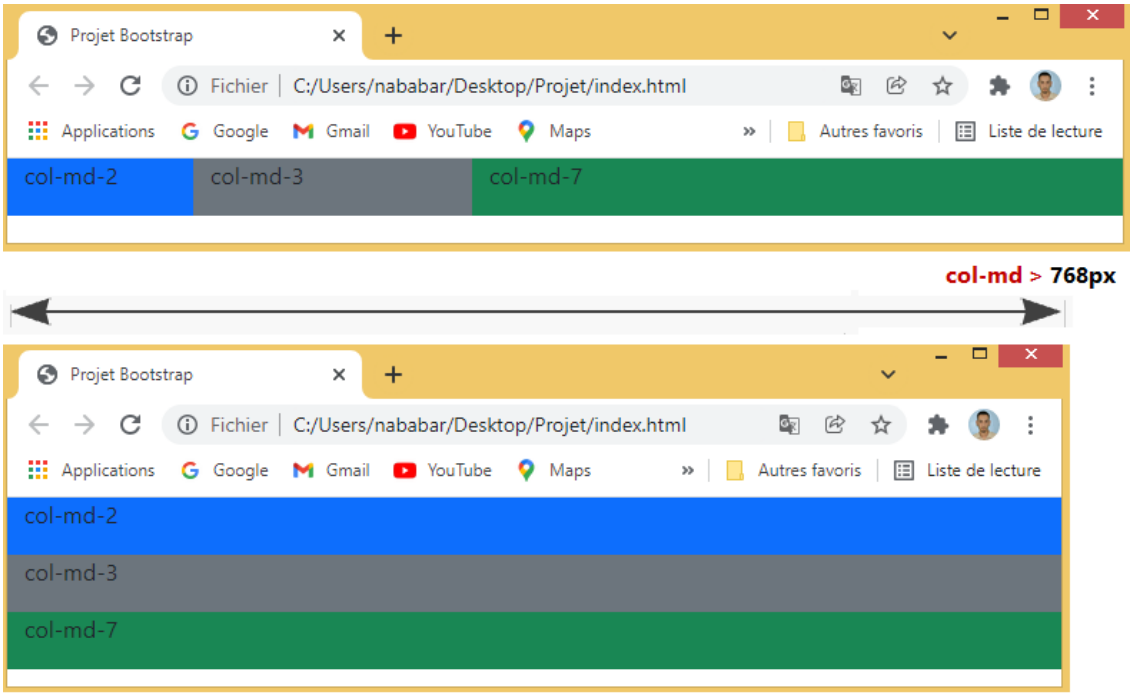
```
<div class= "container-fluid">
    <div class= "row">
        <div class ="col-sm-3 bg-primary">
            <p>col-sm-3</p>
        </div>
        <div class ="col-sm-9 bg-success">
            <p>col-sm-9</p>
        </div>
    </div>
</div>
```



col-md-*

Pour les moyens équipements (Medium) dont la largeur est supérieur ou égale à **768px**

```
<div class= "container-fluid">
    <div class= "row">
        <div class ="col-md-2 bg-primary">
            <p>col-md-2</p>
        </div>
        <div class ="col-md-3 bg-secondary">
            <p>col-md-3</p>
        </div>
        <div class ="col-md-7 bg-success">
            <p>col-md-7</p>
        </div>
    </div>
</div>
```



col-lg-*, col-xl-*

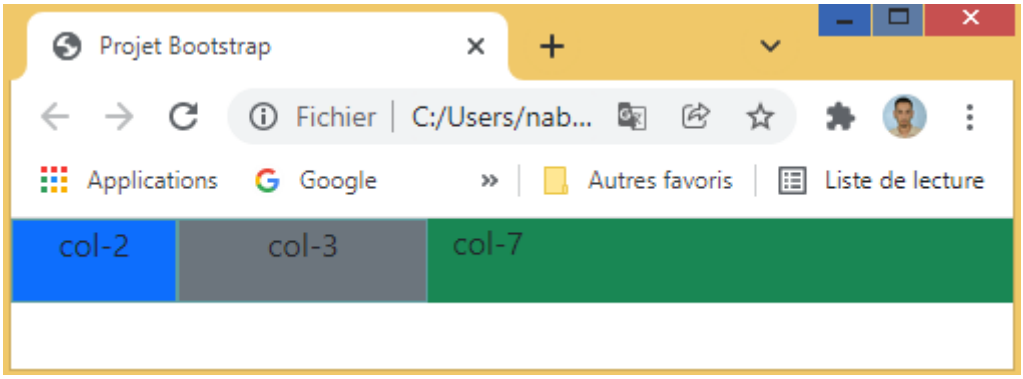
lg : est l'abréviation de "Large", faisant l'allusion aux appareils dont la largeur est supérieure ou égale à **992px**.
xl : est l'abréviation de "Extra Large", faisant l'allusion aux appareils dont la largeur est supérieure ou égale à **1200px**.

Le principe de fonctionnement de ".col-lg-*" & ".col-xl-*" est similaire à celui de ".col-sm-*", ".col-md-*".

col-*

Les appareils dont la largeur de l'écrant est inférieure à **567px**, sont considérés comme ceux très petits (extra small).

```
<div class= "container-fluid">
    <div class= "row">
        <div class ="col-2 bg-primary"><p>col-2</p></div>
        <div class ="col-3 bg-secondary"><p>col-3</p></div>
        <div class ="col-7 bg-success"><p>col-7</p></div>
    </div>
</div>
```

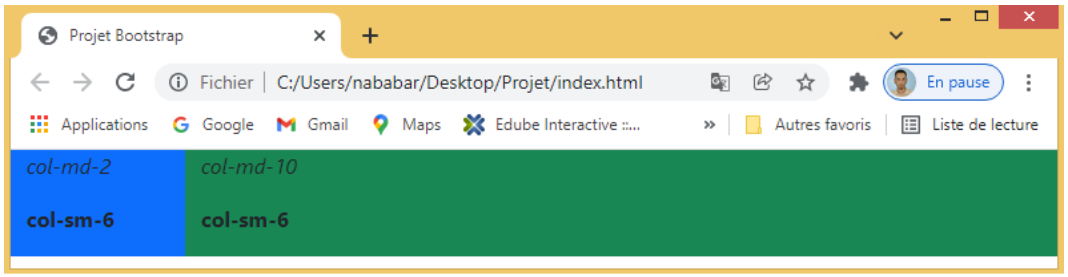


Les éléments ".col-*" sont toujours sur une ligne (row) bien que vous réduisiez la largeur de ".container". Cependant, vous ne pouvez pas rendre la largeur de ".container" inférieure de 320px.

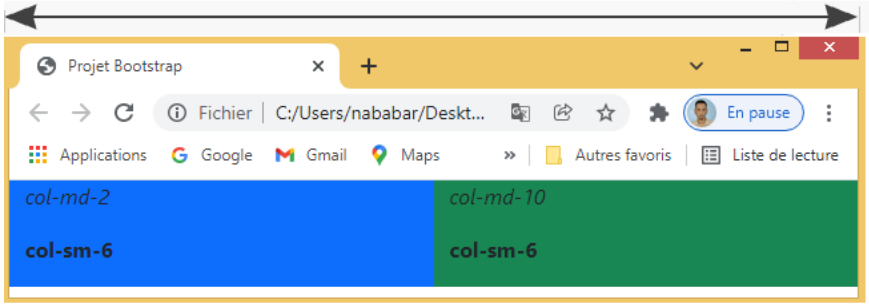
Combinaison : col-sm-*,col-md-*, ..

Un élément (element) peut être combiné de différentes classes col-*, col-sm-*, col-md-*... ensemble. L'exemple ci-dessous vous montrerai le moyen comment Bootstrap s'applique aux telles classes.

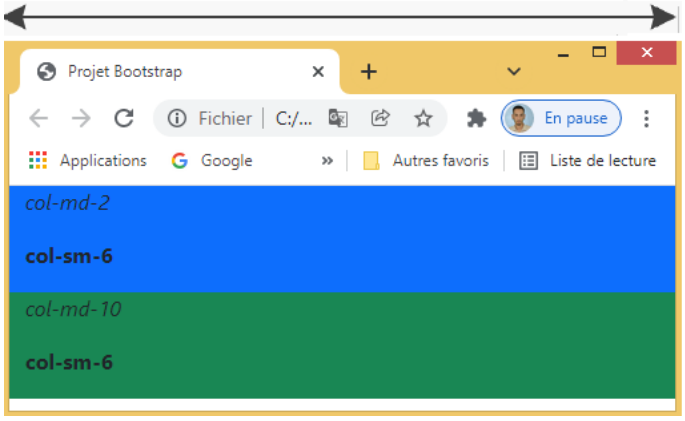
```
<div class= "container-fluid">
    <div class= "row">
        <div class ="col-md-2 col-sm-6 bg-primary">
            <p style="font-style:italic;">col-md-2</p>
            <p style="font-weight:bold;">col-sm-6</p>
        </div>
        <div class ="col-md-10 col-sm-6 bg-success">
            <p style="font-style:italic;">col-md-10</p>
            <p style="font-weight:bold;">col-sm-6</p>
        </div>
    </div>
</div>
```



col-md > 768px



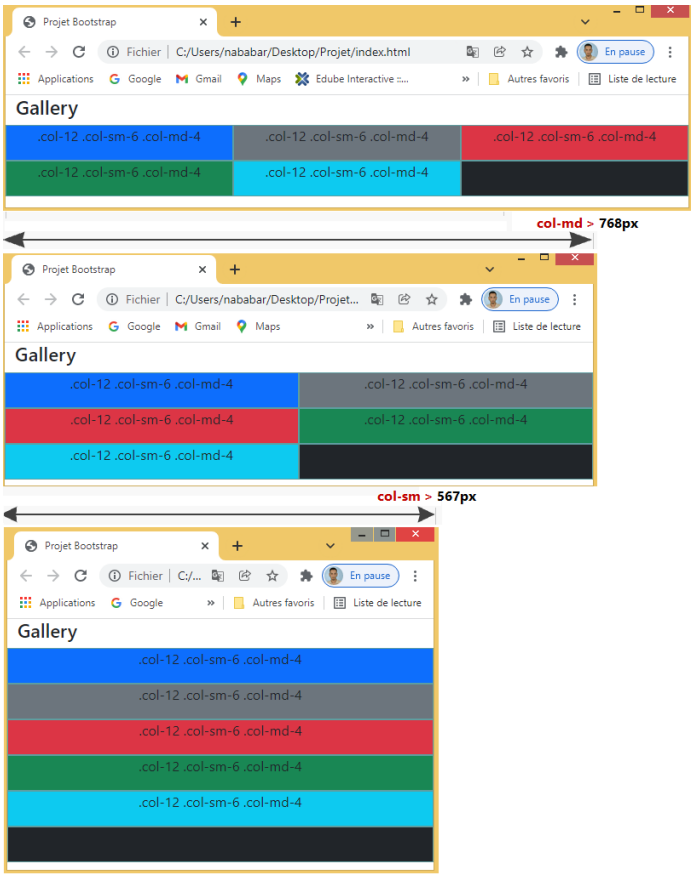
col-sm > 567px



Gallery

L'utilisation des caractéristiques au-dessus de Grid vous permet de créer une Gallery (Galerie) jolie et cette galerie changera sur la base de la taille des appareils.

```
<div class= "container-fluid">
    <h4>Gallery</h4>
    <div class= "row">
        <div class ="col-12 col-sm-6 col-md-4 bg-primary">
            <p>.col-12 .col-sm-6 .col-md-4</p>
        </div>
        <div class ="col-12 col-sm-6 col-md-4 bg-secondary">
            <p>.col-12 .col-sm-6 .col-md-4</p>
        </div>
        <div class ="col-12 col-sm-6 col-md-4 bg-danger">
            <p>.col-12 .col-sm-6 .col-md-4</p>
        </div>

        <div class ="col-12 col-sm-6 col-md-4 bg-success">
            <p>.col-12 .col-sm-6 .col-md-4</p>
        </div>
        <div class ="col-12 col-sm-6 col-md-4 bg-info">
            <p>.col-12 .col-sm-6 .col-md-4</p>
        </div>
        <div class ="col-12 col-sm-6 col-md-4 bg-dark">
            <p>.col-12 .col-sm-6 .col-md-4</p>
        </div>
    </div>
</div>
```



Gérer l'alignement vertical des colonnes dans une ligne

Par défaut, les colonnes vont occuper toute la hauteur d'une ligne. On va cependant pouvoir demander aux colonnes de n'occuper que la place nécessaire à leur contenu et de s'aligner soit au début, soit au milieu, soit en fin de ligne selon l'axe vertical.

Cela va pouvoir être intéressant dans le cas où nos différentes colonnes possèdent des contenus qui utilisent des espaces en hauteur différents ou dans le cas où une hauteur a été explicitement définie pour la ligne.

La hauteur d'une ligne est en effet par défaut déterminée par la colonne dont le contenu prend le plus de place en hauteur. Rien ne nous empêche cependant de déterminer une hauteur pour une ligne.

Bootstrap va nous fournir deux grandes façons d'aligner nos colonnes verticalement dans une ligne : nous allons soit pouvoir aligner toutes les colonnes en même temps, soit pouvoir aligner les colonnes une par une de manière différente.

Pour aligner toutes les colonnes en même temps par rapport à une ligne, nous allons appliquer les classes .align-items-* à nos lignes. Nous pouvons choisir parmi trois classes qui représentent trois positions différentes :

- .align-items-start : les colonnes seront alignées en début (en haut) de la ligne ;
- .align-items-center : les colonnes vont être alignées au centre de la ligne ;
- .align-items-end : les colonnes seront alignées en fin (en bas) de la ligne.

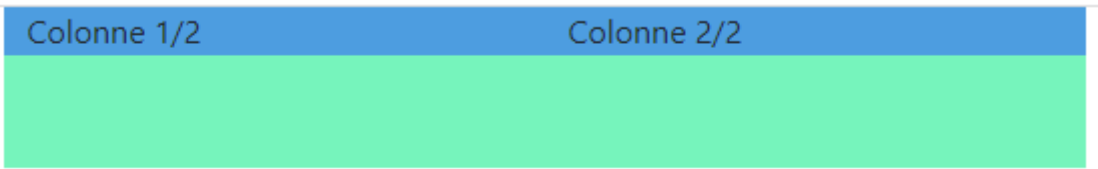
Exemples :

Note : J'ai ici ajouté des styles personnalisés à nos lignes et colonnes.

```
<style>
    .custom-line{
        min-height: 5rem; margin-bottom: 1rem;
        background-color: RGBa(60,240,160,0.7); /*Vert*/
    }
    .custom-line > .col{
        background-color: RGBa(60,120,240,0.7); /*Bleu*/
    }
</style>
```

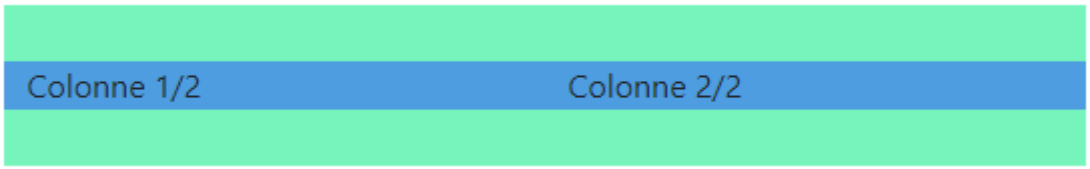
.align-items-start

```
<div class="row align-items-start custom-line">
    <div class="col">Colonne 1/2</div>
    <div class="col">Colonne 2/2</div>
</div>
```



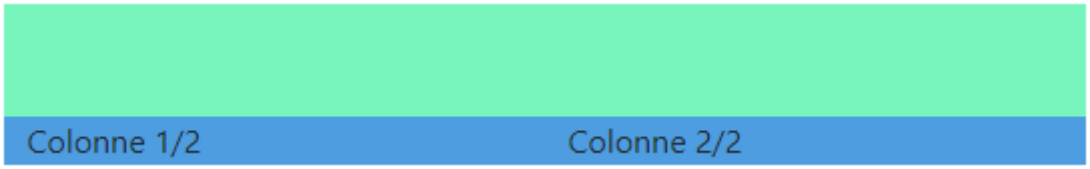
.align-items-center

```
<div class="row align-items-center custom-line">
    <div class="col">Colonne 1/2</div>
    <div class="col">Colonne 2/2</div>
</div>
```



.align-items-end

```
<div class="row align-items-end custom-line">
    <div class="col">Colonne 1/2</div>
    <div class="col">Colonne 2/2</div>
</div>
```

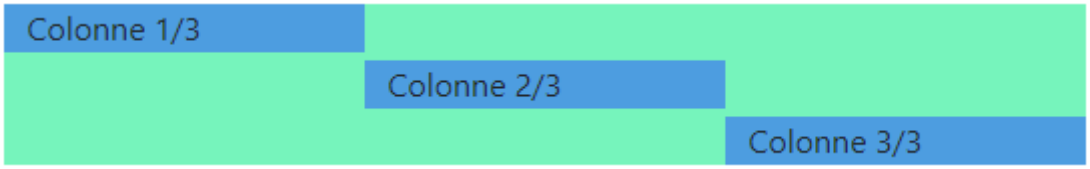


Pour aligner chaque colonne individuellement, nous allons cette fois-ci plutôt utiliser les classes .align-self-* qu'on va utiliser avec chaque élément cette fois-ci. Là encore, nous pouvons choisir parmi trois classes :

- .align-self-start : la colonne sera alignée en début (en haut) de la ligne ;
- .align-self-center : la colonne va être alignée au centre de la ligne ;
- .align-self-end : la colonne sera alignée en fin (en bas) de la ligne.

Exemple :

```
<div class="row custom-line">
    <div class="col align-self-start">Colonne 1/3</div>
    <div class="col align-self-center">Colonne 2/3</div>
    <div class="col align-self-end">Colonne 3/3</div>
</div>
```



Gérer l'alignement horizontal des colonnes dans une ligne

On va principalement vouloir aligner horizontalement des colonnes dans une ligne dans le cas où colonnes créées n'occupent pas tout l'espace de la ligne, c'est-à-dire dans le cas où il reste de l'espace à distribuer entre les colonnes.

Cela va être le cas si on utilise des classes `.col-{nombre}` pour chaque colonne de la ligne et que la somme des nombres fait moins de 12.

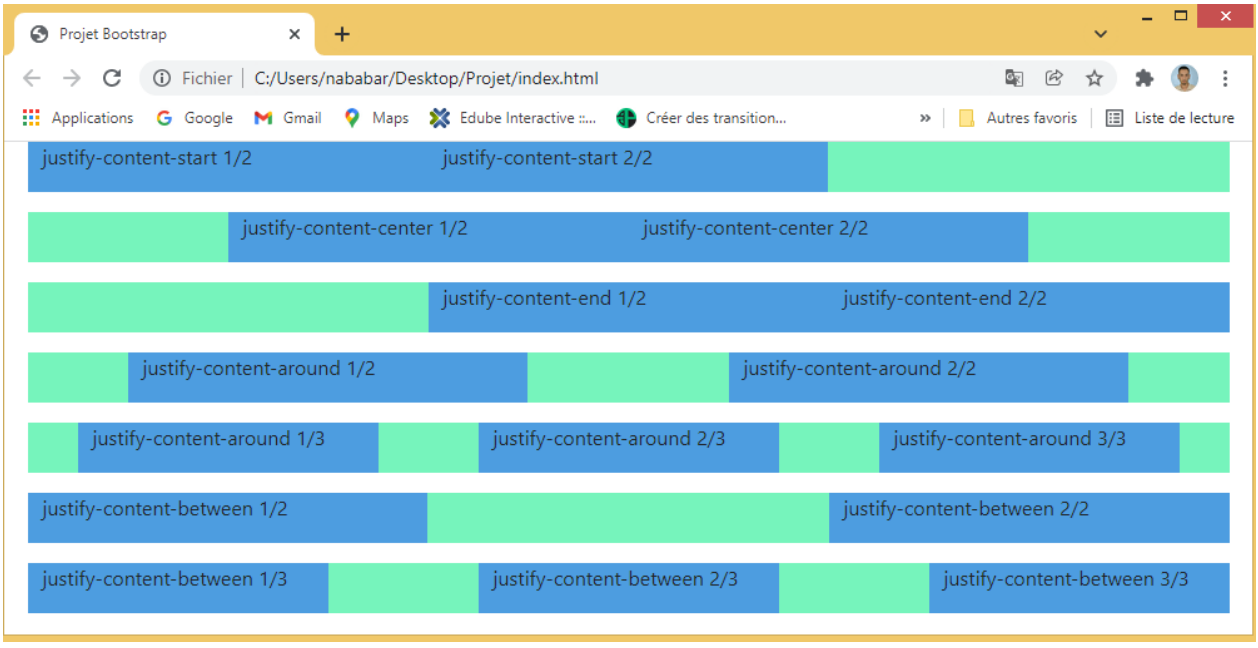
On va pouvoir aligner horizontalement nos colonne dans la ligne grâce aux classes `.justify-content-*` qu'on va devoir appliquer à la ligne en soi. Nous allons pouvoir utiliser les classes suivantes :

- `.justify-content-start` : les colonnes vont se positionner en début de ligne (à gauche par défaut) ;
- `.justify-content-center` : les colonnes vont se positionner au milieu de la ligne ;
- `.justify-content-end` : les colonnes vont se positionner en fin de ligne (à droite par défaut) ;
- `.justify-content-around` : les colonnes vont être réparties équitablement dans la ligne. Chaque colonne va posséder le même écart à droite et à gauche, même celles situées contre les bords de la ligne (l'espacement entre le bord de la ligne et la première / dernière colonnes sera donc deux fois plus petit que l'espacement entre deux colonne) ;
- `.justify-content-between` : les colonnes vont être réparties équitablement dans la ligne. La première colonne va être collée contre le début de la ligne et la dernière va être collée contre la fin de celle-ci.

Exemple :

```
<!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=1.0">
        <title>Projet Bootstrap</title>
        <link rel="stylesheet" href="css/bootstrap.css">
        <style>
            .custom-line{
                margin-bottom: 1rem; background-color: RGBa(60,240,160,0.7);
            }
            .custom-line > .col-3, .custom-line > .col-4{
                background-color: RGBa(60,120,240,0.7); /*Bleu*/
            }
        </style>
    </head>
    <body>
        <div class="container">
            <div class="row justify-content-start custom-line">
                <div class="col-4"><p>justify-content-start 1/2</p></div>
```

```
                <div class="col-4"><p>justify-content-start 2/2</p></div>
            </div>
            <div class="row justify-content-center custom-line">
                <div class="col-4"><p>justify-content-center 1/2</p></div>
                <div class="col-4"><p>justify-content-center 2/2</p></div>
            </div>
            <div class="row justify-content-end custom-line">
                <div class="col-4"><p>justify-content-end 1/2</p></div>
                <div class="col-4"><p>justify-content-end 2/2</p></div>
            </div>
            <div class="row justify-content-around custom-line">
                <div class="col-4"><p>justify-content-around 1/2</p></div>
                <div class="col-4"><p>justify-content-around 2/2</p></div>
            </div>
            <div class="row justify-content-around custom-line">
                <div class="col-3"><p>justify-content-around 1/3</p></div>
                <div class="col-3"><p>justify-content-around 2/3</p></div>
                <div class="col-3"><p>justify-content-around 3/3</p></div>
            </div>
            <div class="row justify-content-between custom-line">
                <div class="col-4"><p>justify-content-between 1/2</p></div>
                <div class="col-4"><p>justify-content-between 2/2</p></div>
            </div>
            <div class="row justify-content-between custom-line">
                <div class="col-3"><p>justify-content-between 1/3</p></div>
                <div class="col-3"><p>justify-content-between 2/3</p></div>
                <div class="col-3"><p>justify-content-between 3/3</p></div>
            </div>
        </div>
         <script src="js/bootstrap.js"></script>
    </body>
</html>
```



Gérer l'ordre d'affichage des colonnes dans une grille Bootstrap

Dans cette nouvelle leçon, nous allons découvrir quelques classes Bootstrap qui vont nous permettre de gérer l'ordre d'affichage (ordre visuel uniquement) des colonnes dans une ligne et allons également voir comment imbriquer des lignes les unes dans les autres pour créer des possibilités de designs avancés.

Gérer l'ordre d'affichage des colonnes

Les classes Bootstrap .order-* vont nous permettre de modifier l'ordre visuel de notre contenu. Nous allons ainsi pouvoir choisir dans quel ordre doit apparaitre notre contenu en passant une classe .order-1, .order-2... .order-12 à chacun de nos éléments représentant nos colonnes et qui va déterminer l'ordre d'affichage visuel de celles-ci dans la ligne.

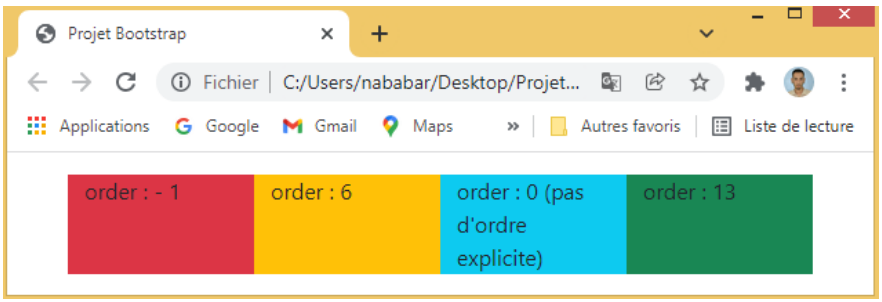
Notez également que les classes .order-* sont responsive et supportent donc les breakpoints. Concrètement, cela signifie que nous allons donc pouvoir choisir des ordres d'affichage différents selon la taille de l'écran d'un visiteur en ajoutant -sm-, -md-, etc. à ces classes.

Finalement, retenez qu'on va également pouvoir utiliser les classes .order-first et .order-last (également responsive) pour afficher un contenu en premier ou en dernier. Pour information, ces deux classes vont respectivement appliquer order: -1 et order : 13 aux colonnes ciblées.

Pour comprendre ces différents exemples, il suffit de savoir que :

- .order-first applique la valeur order: -1 ;
- Les colonnes sans ordre explicite ont la valeur order: 0 ;
- Les colonnes avec un ordre défini par un chiffre vont avoir une valeur entre order: 1 et order: 12 ;
- .order-last applique la valeur order: 13 ;
- On va pouvoir appliquer différents ordres selon les différentes tailles d'écran en utilisant -sm-, -md-, -lg- et -xl.

```
<div class="container">

    <div class="row mt-3">
        <div class="col order-6 bg-warning">order : 6</div>
        <div class="col order-last bg-success">order : 13</div>
        <div class="col bg-info">order : 0 (pas d'ordre explicite)</div>
        <div class="col order-first bg-danger">order : - 1</div>
    </div>
</div>
```

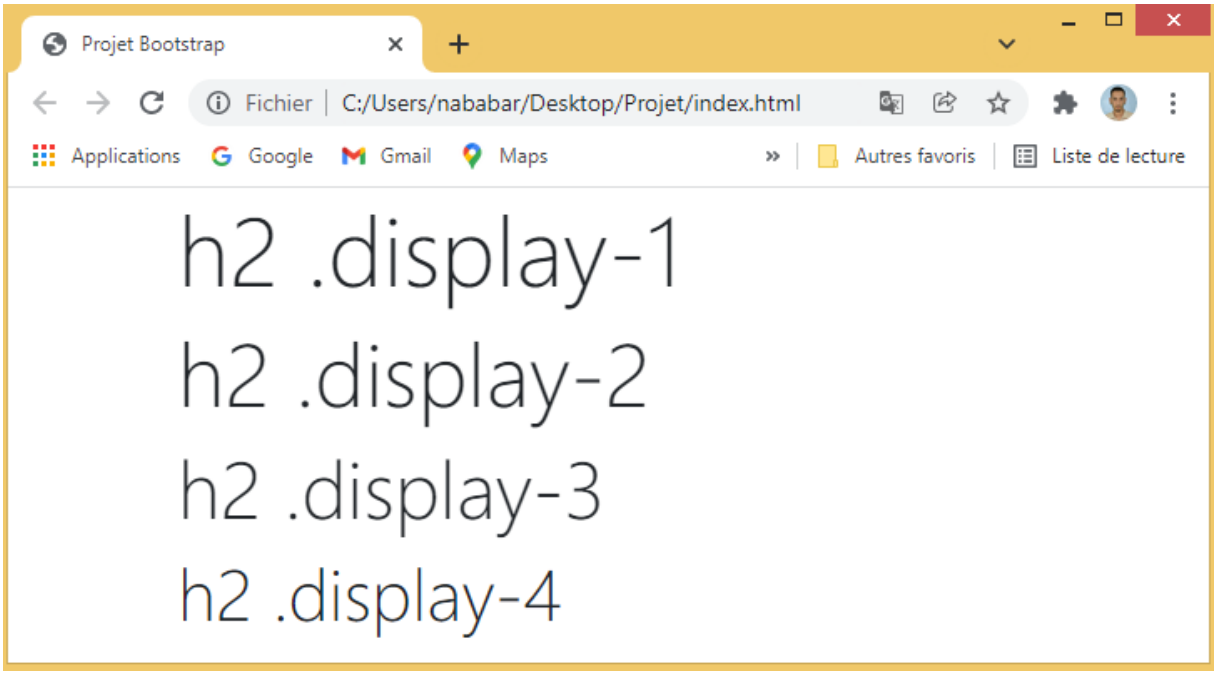


Mettre en forme des textes avec les classes Bootstrap

Faire ressortir visuellement des textes

Les classes .display-1, .display-2, .display-3 et .display-4 permettent de faire ressortir visuellement un titre par rapport au reste du contenu.

```
<div class="container">
    <h2 class="display-1">h2 .display-1</h2>
    <h2 class="display-2">h2 .display-2</h2>
    <h2 class="display-3">h2 .display-3</h2>
    <h2 class="display-4">h2 .display-4</h2>
</div>
```



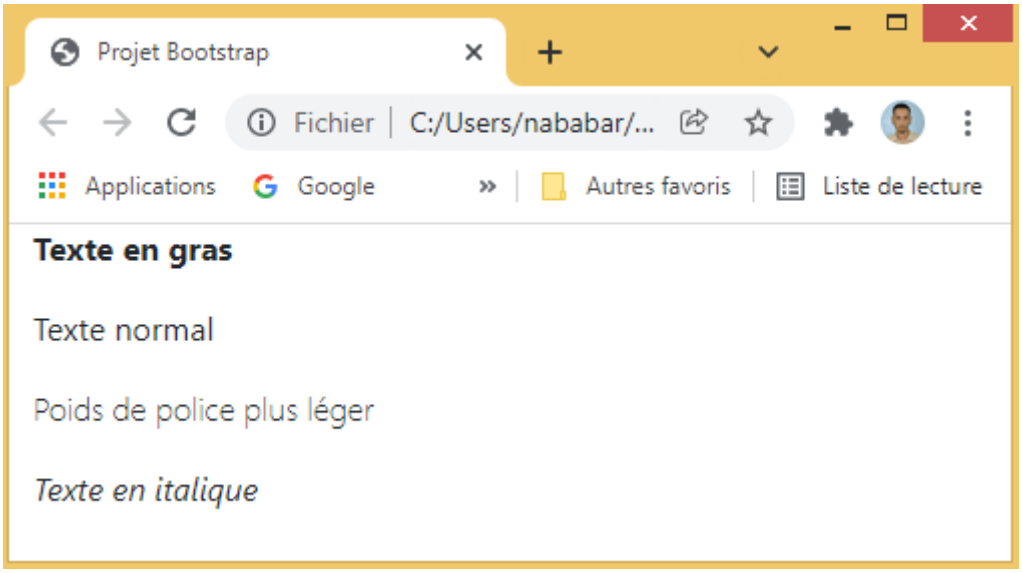
Modifier le poids ou le style d'une police

Pour modifier le poids d'une police, c'est-à-dire son épaisseur, on va pouvoir utiliser l'une des classes suivantes :

- .fw-lighter pour obtenir un texte très fin ;
- .fw-light pour obtenir un texte fin ;
- .fw-normal pour obtenir un texte normal ;
- .fw-bold pour obtenir un texte épais ;
- .fw-bolder pour obtenir un texte très épais.

Pour qu'un texte s'affiche en italique, on peut utiliser la classe .fst-italic.

```
<div class="container-fluid">
    <p class="fw-bold">Texte en gras</p>
    <p class="fw-normal">Texte normal</p>
    <p class="fw-light">Poids de police plus léger</p>
    <p class="fst-italic">Texte en italique</p>
</div>
```

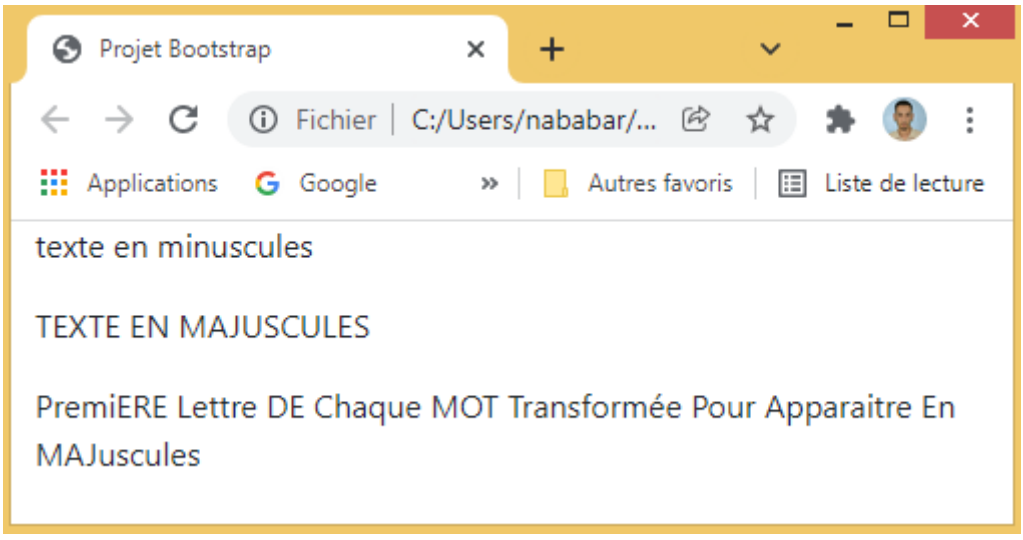


Modifier la casse des textes

Bootstrap nous fournit 3 classes permettant de modifier la casse (le fait que le texte s'affiche en majuscules ou en minuscules) de nos textes :

- La classe .text-lowercase fera qu'un texte s'affiche en minuscules ;
- La classe .text-uppercase fera qu'un texte s'affiche en majuscules ;
- La classe .text-capitalize ne va transformer que la première lettre de chaque mot pour la mettre en majuscules.

```
<div class="container-fluid">
    <p class="text-lowercase">Texte en minuscules</p>
    <p class="text-uppercase">Texte en majuscules</p>
    <p class="text-capitalize">premiERE lettre DE chaque MOT transformée pour
apparaitre en MAJuscules</p>
</div>
```

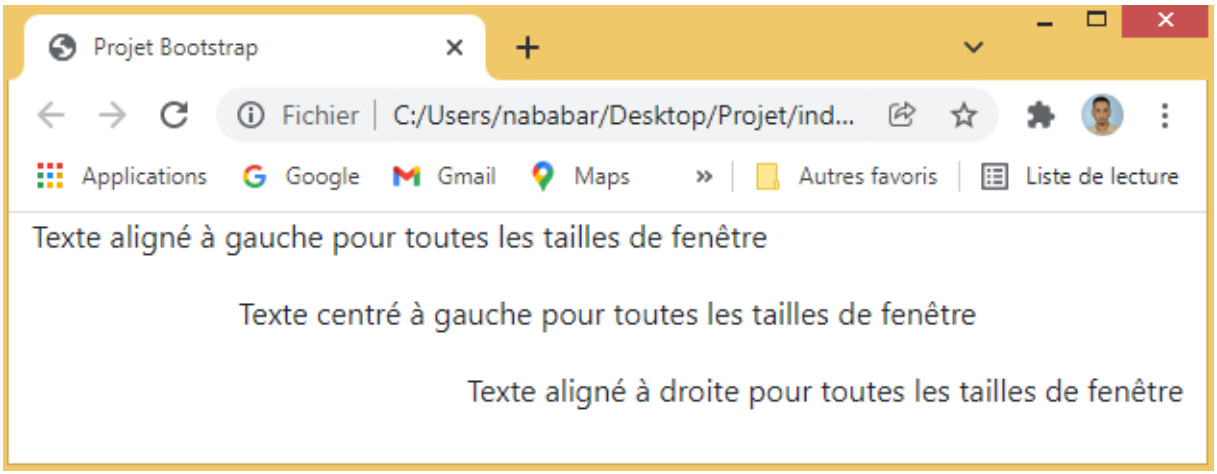


Aligner un texte dans son élément parent

Bootstrap nous fournit différentes classes nous permettant d'aligner un texte à droite, au centre, à gauche ou de le justifier. Les classes d'alignement de base sont les suivantes :

- .text-start va aligner un texte à gauche ;
- .text-center va centrer un texte ;
- .text-end va aligner un texte à droite ;

```
<div class="container-fluid">
    <p class="text-start">Texte aligné à gauche pour toutes les tailles de fenêtre</p>
    <p class="text-center">Texte centré à gauche pour toutes les tailles de fenêtre</p>
    <p class="text-end">Texte aligné à droite pour toutes les tailles de fenêtre</p>
</div>
```



Modifier la couleur, la couleur de fond et l'opacité des éléments avec Bootstrap

Bootstrap possède des classes de couleurs dites « contextuelles ». Le but tel que décrit par Bootstrap est « d'apporter du sens à travers les couleurs ».

Couleur et opacité des textes

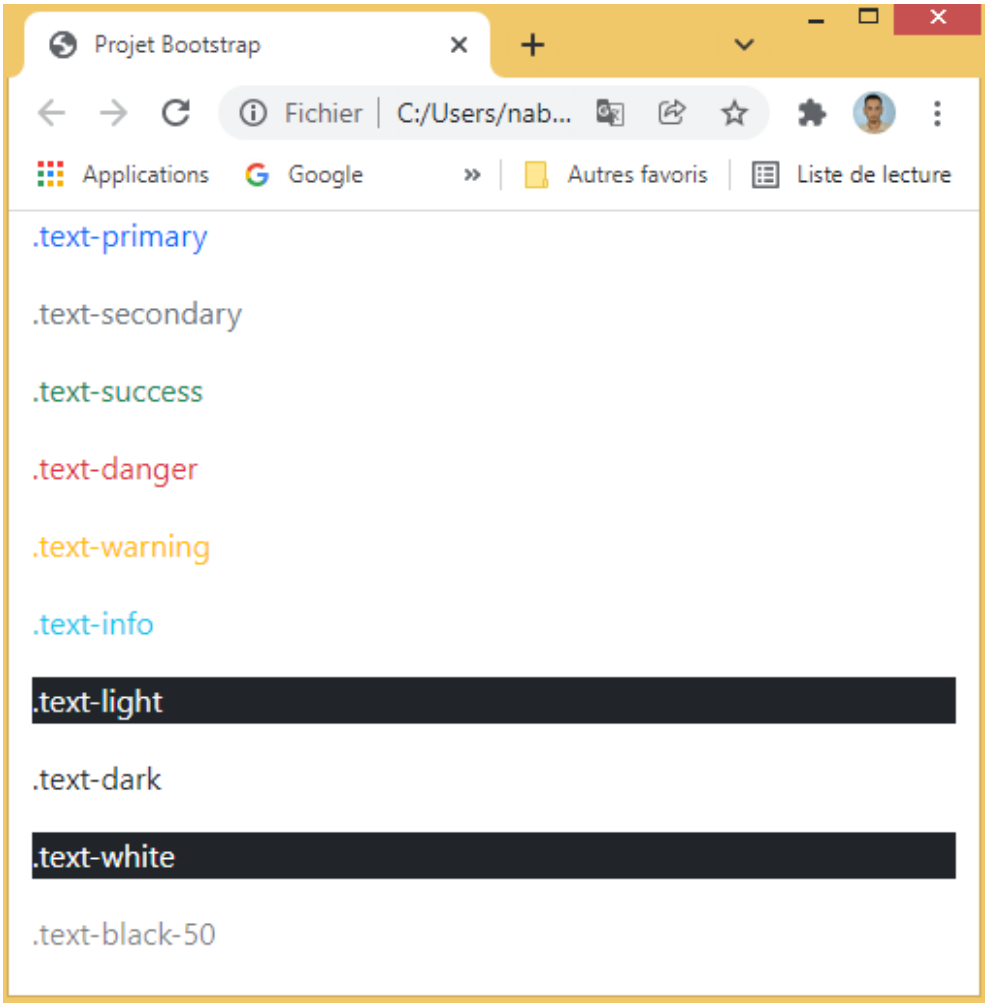
Les classes Bootstrap nous permettant de modifier la couleur des textes sont les suivantes :

- .text-primary : texte bleu ;
- .text-seconday : texte gris-bleu ;
- .text-success : texte vert ;
- .text-danger : texte rouge ;
- .text-warning : texte jaune ;
- .text-info : nuance de bleu ;
- .text-light : texte gris clair ;
- .text-dark : texte gris foncé ;
- .text-body : texte noir ;
- .text-muted : texte gris ;
- .text-white : texte blanc ;
- .text-black-50 : texte noir semi transparent ;
- .text-white-50 : texte blanc semi transparent ;

Notez ici que les couleurs de fond ont été ajoutées sur certains textes pour bien que vous voyiez le texte avec la classe .bg-dark. Cette classe n'a rien à voir avec les classes de couleurs de texte.

Les classes .text-black-50 et .text-white-50 servent à créer des textes noir et blanc semi-transparents.

```
<div class="container">
    <p class="text-primary">.text-primary</p>
    <p class="text-secondary">.text-secondary</p>
    <p class="text-success">.text-success</p>
    <p class="text-danger">.text-danger</p>
    <p class="text-warning">.text-warning</p>
    <p class="text-info">.text-info</p>
    <p class="text-light bg-dark">.text-light</p>
    <p class="text-dark">.text-dark</p>
    <p class="text-white bg-dark">.text-white</p>
    <p class="text-black-50">.text-black-50</p>
</div>
```

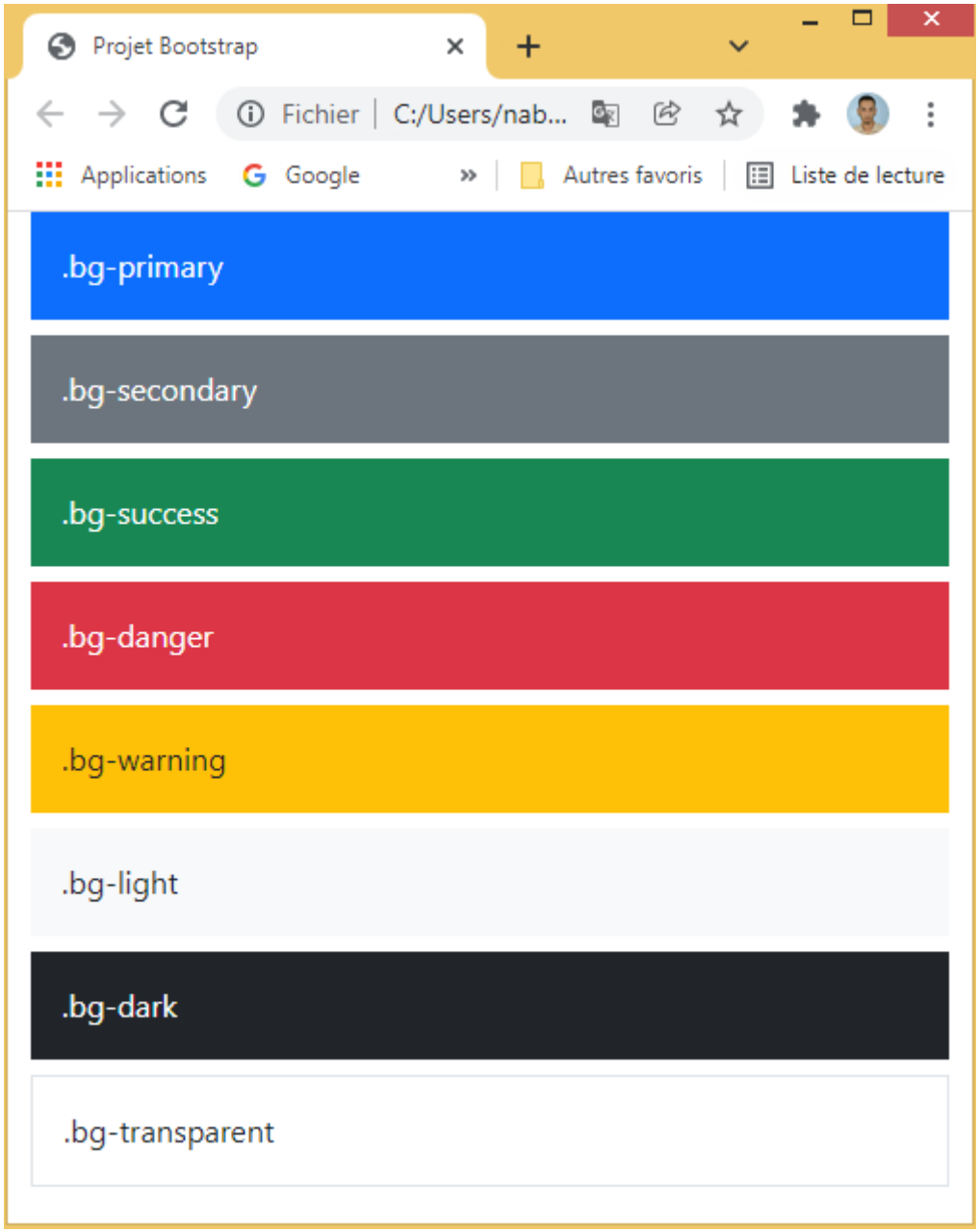


Modifier couleur de fond d'un élément

Nous allons pouvoir modifier la couleur de fond de nos éléments en utilisant le même système de « couleurs contextuelles » fourni par Bootstrap que pour nos textes.

La seule différence est que nous allons cette fois-ci utiliser le préfixe .bg- (pour background) plutôt que .text-.

```
<div class="container">

    <div class="p-3 mb-2 bg-primary text-white">.bg-primary</div>

    <div class="p-3 mb-2 bg-secondary text-white">.bg-secondary</div>

    <div class="p-3 mb-2 bg-success text-white">.bg-success</div>

    <div class="p-3 mb-2 bg-danger text-white">.bg-danger</div>

    <div class="p-3 mb-2 bg-warning text-dark">.bg-warning</div>

    <div class="p-3 mb-2 bg-light text-dark">.bg-light</div>

    <div class="p-3 mb-2 bg-dark text-white">.bg-dark</div>

    <div class="p-3 mb-2 bg-transparent text-dark border">.bg-
transparent</div>

</div>
```


Projet Bootstrap
Fichier | C:/Users/nab...
Applications
Google
Autres favoris
Liste de lecture
.bg-primary
.bg-secondary
.bg-success
.bg-danger
.bg-warning
.bg-light
.bg-dark
.bg-transparent

Ajouter des bordures aux éléments avec Bootstrap

Bootstrap va nous permettre d'ajouter simplement toutes sortes de bordures à nos éléments HTML. Pour cela, nous utiliserons les classes de type border-*.

Ajouter une bordure classique à un élément en utilisant Bootstrap

Nous allons déjà pouvoir ajouter une bordure carrée autour d'un élément ou sur un côté spécifique d'un élément. Pour cela, nous allons utiliser les classes suivantes :

- .border : va ajouter une bordure de couleur grise par défaut tout autour de l'élément ;
- .border-top : va ajouter une bordure de couleur grise par défaut uniquement sur le côté haut de l'élément ;
- .border-right : va ajouter une bordure de couleur grise par défaut à droite uniquement de l'élément ;
- .border-bottom : va ajouter une bordure de couleur grise par défaut uniquement sur le côté bas de l'élément ;
- .border-left : va ajouter une bordure de couleur grise par défaut à gauche uniquement de l'élément.

Enlever certaines bordures d'un élément

Bootstrap nous propose une fonctionnalité relativement intéressante qui va être de pouvoir supprimer toutes les bordures d'un élément ou seulement une bordure d'un côté en particulier. Nous allons pouvoir faire cela en ajoutant « 0 » en fin de classe border-*. Cela va donc nous permettre de gérer très simplement quelles bordures doivent s'afficher autour d'un élément. Pour cela, nous allons utiliser les classes suivantes :

- .border-0 : supprime toutes les bordures de l'élément ;
- .border-top-0 : supprime la bordure supérieure de l'élément ;
- .border-right-0 : supprime la bordure droite de l'élément ;
- .border-bottom-0 : supprime la bordure inférieure de l'élément ;
- .border-left-0 : supprime la bordure gauche de l'élément.

Notez ici que j'ai également utilisé une classe border simple pour ajouter des bordures tout autour de mes éléments par défaut. En effet, pour supprimer certaines bordures, il faut déjà que des bordures aient été ajoutées à un élément !

Créer des bordures de couleur avec Bootstrap

Nous allons pouvoir réutiliser le système de couleurs contextuelles de Bootstrap afin de personnaliser la couleur de nos bordures. Les classes disponibles dans la thème de base sont les suivantes :

- .border-primary : la bordure créée sera colorée en bleu ;

- .border-secondary : la bordure créée sera colorée en gris foncé ;
- .border-success : la bordure créée sera colorée en vert ;
- .border-danger : la bordure créée sera colorée en rouge ;
- .border-warning : la bordure créée sera colorée en jaune ;
- .border-info : la bordure créée sera colorée en une autre teinte de bleu ;
- .border-light : la bordure créée sera colorée en gris clair ;
- .border-dark : la bordure créée sera colorée en noir ;
- .border-white : la bordure créée sera colorée en blanc.

Créer des bordures arrondies avec Bootstrap

Nous allons finalement pouvoir changer le comportement et notamment arrondir des bordures avec Bootstrap en utilisant les classes suivantes. Notez bien encore une fois que ces classes ne vont pas créer de bordures mais simplement modifier l'apparence d'une bordure déjà existante. Pour créer notre bordure, nous devrons également utiliser une classe border-*.

- rounded : arrondit légèrement une bordure ;
- rounded-top : arrondit légèrement les coins supérieurs d'une bordure ;
- rounded-right : arrondit légèrement les coins droits d'une bordure ;
- rounded-bottom : arrondit légèrement les coins inférieurs d'une bordure ;
- rounded-left : arrondit légèrement les coins gauche d'une bordure ;
- rounded-circle arrondit complètement la bordure ;
- rounded-pill : arrondit la bordure jusqu'à obtenir un demi-cercle ;
- rounded-0 : annule l'arrondi d'une bordure ;

Notez qu'on va également pouvoir exercer un certain contrôle sur l'importance de l'arrondi des bordures avec les classes .rounded-sm et .rounded-lg qui vont nous permettre de définir des arrondis moins prononcés ou au contraire plus prononcés.

Ajouter des ombres aux éléments avec Bootstrap

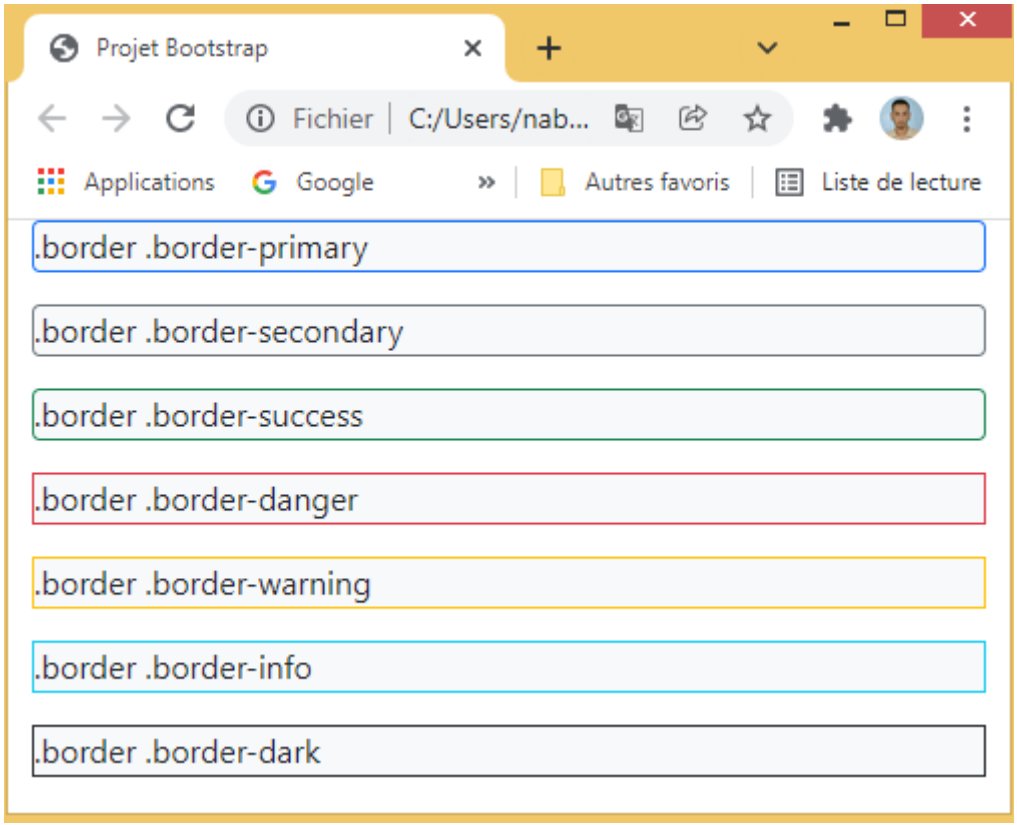
On va pouvoir ajouter des ombres à nos éléments en utilisant les classes de type .shadow-* Bootstrap nous laisse choisir parmi les quatre classe suivantes :

- .shadow-none : aucune ombre ;
- .shadow-sm : ombre peu étendue ;
- .shadow : ombre moyennement étendue ;
- .shadow-lg : ombre étendue.

Notez que les ombres seront grises et qu'on ne dispose pas de classe permettant de modifier leur couleur.

Exemple :

```
<div class="container">

    <p class="bg-light border border-primary rounded">.border .border-
primary</p>
    <p class="bg-light border border-secondary rounded">.border .border-
secondary</p>
    <p class="bg-light border border-success rounded">.border .border-
success</p>
    <p class="bg-light border border-danger">.border .border-danger</p>
    <p class="bg-light border border-warning">.border .border-warning</p>
    <p class="bg-light border border-info">.border .border-info</p>
    <p class="bg-light border border-dark">.border .border-dark</p>
</div>
```



Gérer les dimensions des éléments avec Bootstrap

Les classes Bootstrap ne nous permettent pas d'avoir un contrôle total sur la taille des différents éléments d'une page. Cela s'explique en partie par le fait que le système des grilles permet déjà de créer des zones pour les éléments.

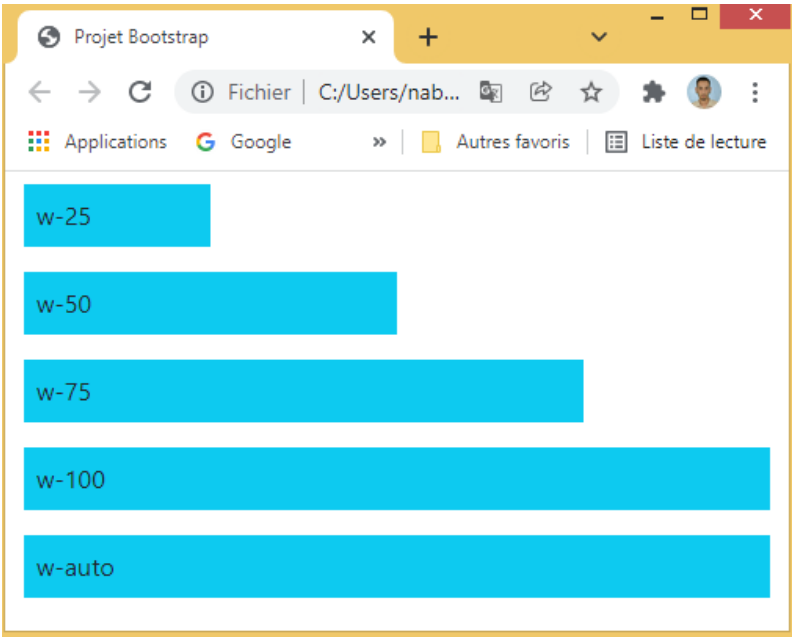
Définir la taille d'un élément relativement à celle de son parent

Bootstrap va déjà nous permettre de définir la taille de nos éléments en fonction de celle de leur parent.

Pour modifier la largeur d'un élément, on va pouvoir utiliser les classes suivantes :

- .w-25 : l'élément a une largeur égale à 25% de celle de son parent ;
- .w-50 : l'élément a une largeur égale à 50% de celle de son parent ;
- .w-75 : l'élément a une largeur égale à 75% de celle de son parent ;
- .w-100 : l'élément a une largeur égale à celle de son parent ;
- .w-auto : la largeur de l'élément est définie automatiquement.

```
<div class="container">
    <p class="w-25 bg-info mt-2 p-2">w-25</p>
    <p class="w-50 bg-info mt-2 p-2">w-50</p>
    <p class="w-75 bg-info mt-2 p-2">w-75</p>
    <p class="w-100 bg-info mt-2 p-2">w-100</p>
    <p class="w-auto bg-info mt-2 p-2">w-auto</p>
</div>
```



Pour modifier la hauteur d'un élément, on utilisera les classes suivantes :

- .h-25 : l'élément a une hauteur égale à 25% de celle de son parent ;
- .h-50 : l'élément a une hauteur égale à 50% de celle de son parent ;
- .h-75 : l'élément a une hauteur égale à 75% de celle de son parent ;
- .h-100 : l'élément a une hauteur égale à celle de son parent ;
- .h-auto : la hauteur de l'élément est définie automatiquement.

Ajouter des marges aux éléments avec Bootstrap

Bootstrap met à notre disposition une série de classes qui vont nous permettre de définir des marges intérieures (le padding) ou extérieures (margin) tout en conservant l'aspect responsive de notre page.

Utiliser les notations de padding ou de margin de Bootstrap

Les classes Bootstrap nous permettant d'appliquer des marges intérieures ou extérieures à nos éléments vont toutes être construites sur le même modèle qui est le suivant : {type de marge}{côté}-{taille}.

Dans le cas où l'on souhaite appliquer des marges différentes à nos éléments selon la taille de l'écran de nos visiteurs (marges responsive), nous devrons également ajouter un breakpoint entre les notations relatives au côté d'application de la marge et sa taille. Le schéma de la classe sera alors le suivant : {type de marge}{côté}-{breakpoint}-{taille}.

Pour le premier élément de notre classe, c'est-à-dire le type de marge, nous allons pouvoir choisir entre deux valeurs :

- **m** va nous servir à définir une marge extérieure (margin) ;
- **p** va nous servir à définir une marge intérieure (padding).

Au niveau du côté de l'élément auquel doit s'appliquer la marge, nous avons le choix entre différentes valeurs :

- **t** va nous permettre d'appliquer une marge (margin ou padding) au niveau du côté supérieur à un élément. Cela va donc servir à définir une margin-top ou un padding-top ;
- **b** va nous permettre d'ajouter une marge (margin ou padding) basse à un élément ;
- **r** va nous permettre d'ajouter une marge (margin ou padding) droite à un élément ;
- **l** va nous permettre d'ajouter une marge (margin ou padding) gauche à un élément ;
- **x** va nous permettre de définir des marges (margin ou padding) gauche et droite à notre élément ;
- **y** va nous permettre de définir des marges (margin ou padding) haute et basse à notre élément.

Notez que si on omet le paramètre « côté » dans notre classe, alors la marge s'appliquera à tous les côtés de l'élément à la fois.

En termes de breakpoint, nous allons pouvoir utiliser les notations habituelles à savoir sm, md, lg et xm. Notre ici que si on ne précise pas de breakpoint alors cela revient à définir une marge pour un breakpoint xs c'est-à-dire pour toutes les tailles d'écran.

Finalement, nous allons pouvoir choisir parmi 7 valeurs différentes pour définir la taille de nos marges :

- **0** va supprimer toutes les marges (margin ou padding) appliquées à un élément ;
- **1** va définir des marges (margin ou padding) de 0.25rem par défaut (0,25 * la valeur de la variable SASS $spacer qui est définie par défaut à 1rem) ;
- **2** va définir des marges (margin ou padding) de 0.5rem par défaut (0.5 * $spacer);
- **3** va définir des marges (margin ou padding) de 1rem par défaut (1 * $spacer);
- **4** va définir des marges (margin ou padding) de 1.5rem par défaut (1.5 * $spacer);
- **5** va définir des marges (margin ou padding) de 3rem par défaut (3 * $spacer);
- La valeur auto va être exclusive aux marges extérieures et définir une margin : auto.

L'affichage des éléments : Bootstrap display et display : flex

Bootstrap nous fournit un ensemble de classes nous permettant de changer la valeur de la propriété display (c'est-à-dire le type d'affichage) de nos éléments.

Les classes permettant de gérer le type d'affichage des éléments

Les classes permettant de définir le type d'affichage d'un élément sont responsives et vont pouvoir être construites en utilisant le schéma suivant :

- .d-{valeur} pour appliquer un type d'affichage à toutes les tailles d'écran ;
- .d-{breakpoint}-{valeur} pour appliquer un type d'affichage à une taille d'écran en particulier. Les breakpoints vont être comme d'habitude représentés par : sm, md, lg et xl.

Les valeurs que l'on va pouvoir utiliser sont les suivantes :

- none : l'élément aura display: none ;
- inline : l'élément aura display: inline ;
- block : l'élément aura display: block ;
- inline-block : l'élément aura display: inline-block ;
- table : l'élément aura display: table ;
- table-cell : l'élément aura display: table-cell ;
- table-row : l'élément aura display: table-row ;
- flex : l'élément aura display: flex ;
- inline-flex : l'élément aura display: inline-flex.

Gérer le type de positionnement des éléments avec Bootstrap

Bootstrap nous permet d'appliquer une propriété position à des éléments grâce aux classes suivantes :

- .position-static ;
- .position-relative ;
- .position-absolute ;
- .position-fixed ;
- .position-sticky.

Créer des tableaux stylisés avec Bootstrap

Dans cette leçon, nous allons voir comment créer des tableaux stylisés en utilisant les classes Bootstrap mises à notre disposition.

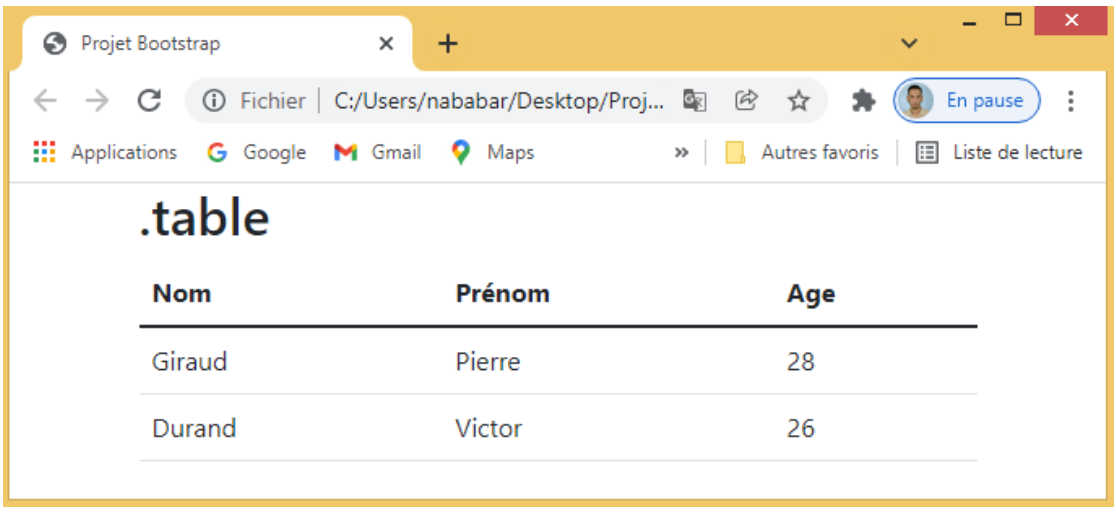
La classe .table

La classe .table est la classe Bootstrap de base pour styliser des tableaux. Nous allons passer cette classe à un élément table. Celle-ci va appliquer une première mise en forme relativement basique à notre tableau.

Inverser les couleurs d'un tableau

On va pouvoir inverser les couleurs par défaut d'un tableau Bootstrap, c'est-à-dire avoir une couleur de fond noire et une couleur de texte blanche en ajoutant la classe .table-dark en plus de la classe .table à notre élément HTML table.

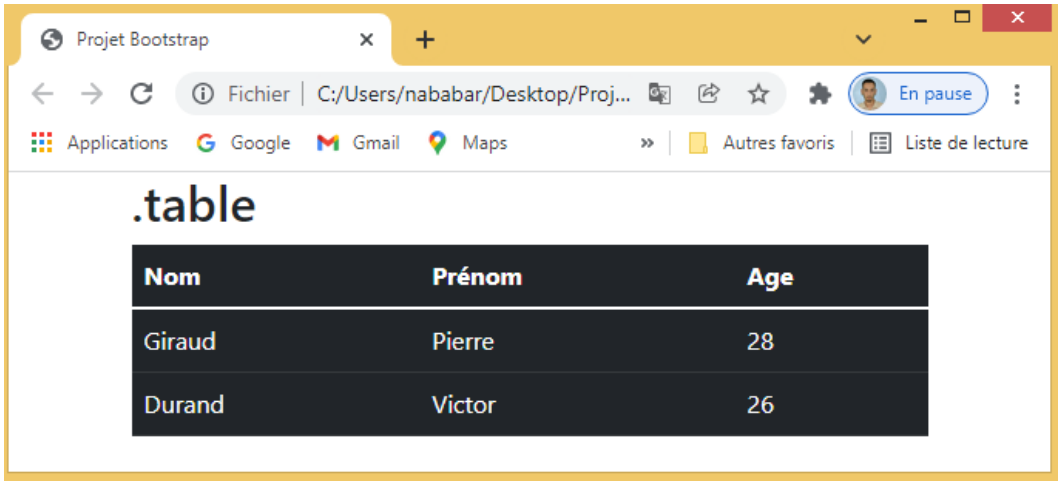
```
<div class="container">
    <h1>.table</h1>
    <table class="table">
        <thead>
            <th>Nom</th>
            <th>Prénom</th>
            <th>Age</th>
        </thead>
        <tbody>
            <tr>
                <td>Giraud</td>
                <td>Pierre</td>
                <td>28</td>
            </tr>
            <tr>
                <td>Durand</td>
                <td>Victor</td>
                <td>26</td>
            </tr>
        </tbody>
    </table>
</div>
```



Personnaliser l'en-tête d'un tableau

Bootstrap met à notre disposition deux classes qu'on va pouvoir appliquer à un élément thead pour personnaliser l'en-tête de nos tableaux : les classes .thead-light (ligne d'en-tête avec un fond gris clair) et .thead-dark (ligne d'en-tête avec un fond noir).

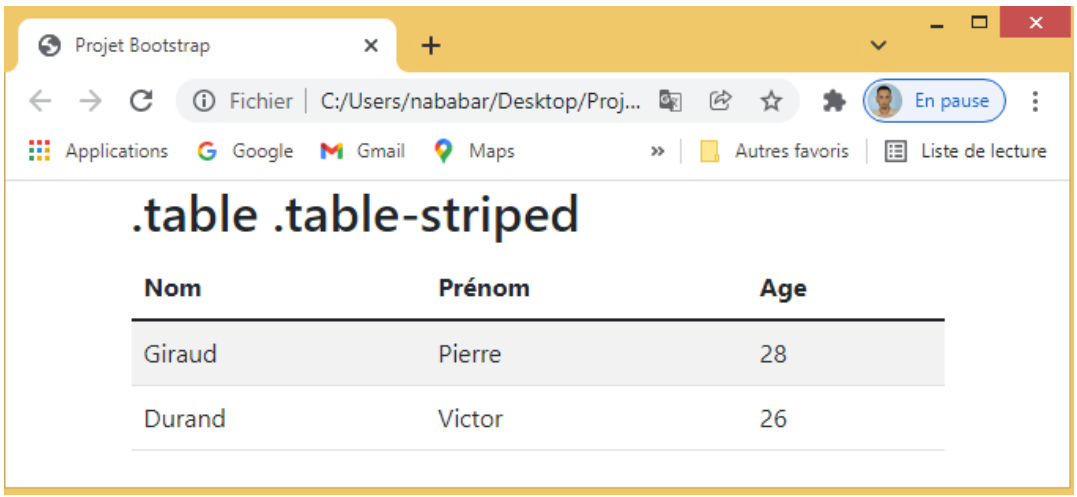
```
<div class="container">
    <h1>.table .table-dark</h1>
    <table class="table table-dark">
        ...
    </table>
</div>
```



Créer un tableau avec alternance de couleurs entre les lignes

La classe Bootstrap .table-striped qu'on va pouvoir appliquer à notre élément table va nous permettre de créer des tableaux zébrés avec une ligne au fond blanc et une ligne au fond gris clair en alternance.

```
<div class="container">
    <h1>.table .table-striped</h1>
    <table class="table table-striped">
        ...
    </table>
</div>
```



Ajouter ou supprimer des bordures d'un tableau

Nous allons pouvoir ajouter des bordures autour de chaque cellule et autour de notre tableau grâce à la classe .table-bordered.

Ajouter des effets lors du passage de la souris

Nous allons encore pouvoir ajouter un peu d'interactivité à notre tableau en changeant la couleur de fond d'une ligne lors du survol de la souris de celle-ci en appliquant la classe .table-hover à notre élément table.

Utiliser les couleurs contextuelles avec les tableaux

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.

Notez que ces couleurs ne vont pas fonctionner avec un tableau possédant une classe .table-dark. Un hack connu va alors consister à utiliser plutôt les classes de couleurs contextuelles .bg-* pour arriver à des résultats similaires.

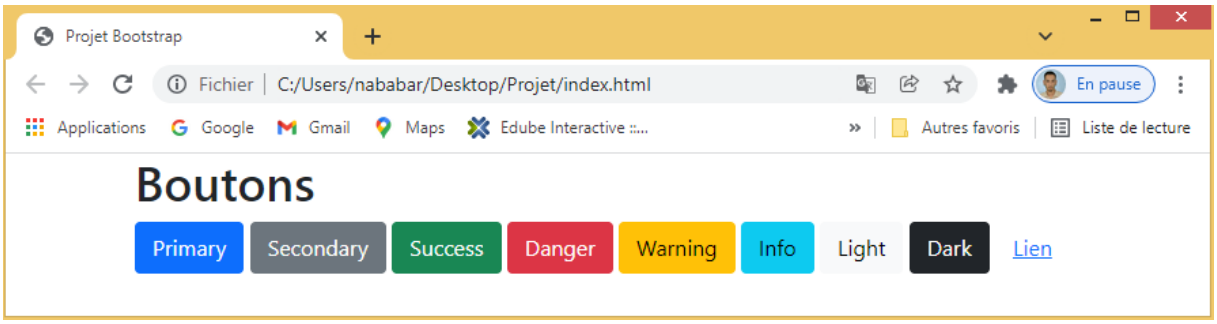
Créer des boutons stylisés avec Bootstrap

Bootstrap va nous permettre de styliser des boutons.

Pour personnaliser l'aspect de nos boutons avec Bootstrap, nous allons utiliser la classe de base .btn et des classes de type .btn-*.

Vous pouvez déjà noter que les classes .btn vont également fonctionner avec des éléments a (liens) et input (champ de formulaire).

```
<div class="container">
    <h1>Boutons</h1>
    <button type="button" class="btn btn-primary mb-2">Primary</button>
    <button type="button" class="btn btn-secondary mb-2">Secondary</button>
    <button type="button" class="btn btn-success mb-2">Success</button>
    <button type="button" class="btn btn-danger mb-2">Danger</button>
    <button type="button" class="btn btn-warning mb-2">Warning</button>
    <button type="button" class="btn btn-info mb-2">Info</button>
    <button type="button" class="btn btn-light mb-2">Light</button>
    <button type="button" class="btn btn-dark mb-2">Dark</button>
    <button type="button" class="btn btn-link mb-2">Lien</button>
</div>
```



Gérer les bordures des boutons

Bootstrap nous donne également la possibilité de créer des boutons avec des bordures colorées et un fond blanc.

Pour cela, nous allons utiliser les classes .btn-outline-* avec les couleurs contextuelles.

```
<div class="container">
    <h1>Boutons btn btn-outline-*</h1>
    <button type="button" class="btn btn-outline-primary mb-
2">Primary</button>
    <button type="button" class="btn btn-outline-secondary mb-
2">Secondary</button>
    <button type="button" class="btn btn-outline-success mb-
2">Success</button>
    <button type="button" class="btn btn-outline-danger mb-2">Danger</button>
    <button type="button" class="btn btn-outline-warning mb-
2">Warning</button>
    <button type="button" class="btn btn-outline-info mb-2">Info</button>
    <button type="button" class="btn btn-outline-light mb-2">Light</button>
    <button type="button" class="btn btn-outline-dark mb-2">Dark</button>
</div>
```

Modifier la taille et le type d'affichage d'un bouton

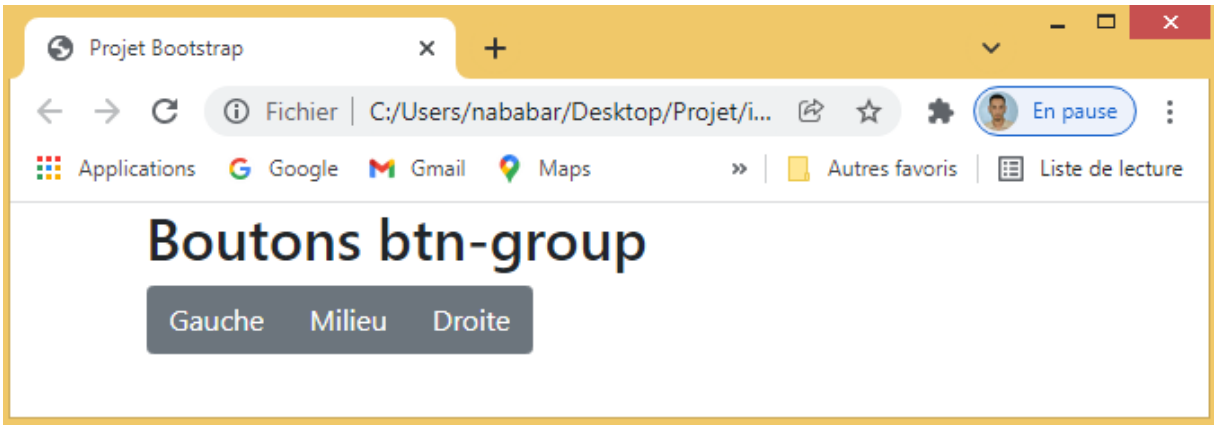
Bootstrap va également nous permettre de créer des boutons plus petits ou plus grands que la taille par défaut. Pour cela, on va utiliser les classes .btn-sm (bouton de petite taille) et .btn-lg (bouton de grande taille).

Nous allons également pouvoir changer le type de display d'un bouton et le transformer en élément de type block grâce à la classe .btn-block.

Les groupes de boutons

Nous allons également pouvoir grouper plusieurs boutons entre eux afin d'homogénéiser l'apparence d'un ensemble de boutons. Pour cela, nous allons utiliser la classe Bootstrap .btn-group que l'on va appliquer à l'élément HTML qui va contenir la série de boutons à grouper (ce sera souvent un élément div).

```
<div class="container">
    <h1>Boutons btn-group</h1>
    <div class="btn-group" role="group" aria-label="Un groupe de boutons">
        <button type="button" class="btn btn-secondary">Gauche</button>
        <button type="button" class="btn btn-secondary">Milieu</button>
        <button type="button" class="btn btn-secondary">Droite</button>
    </div>
</div>
```



Gérer la taille des groupes de boutons

De manière similaires aux boutons simples, nous allons pouvoir créer des groupes de boutons de différentes tailles avec les classes .btn-group-sm et .btn-group-lg

Créer des listes de boutons en colonne

On va pouvoir créer des listes de boutons verticales ou en colonne grâce à la classe .btn-group-vertical.

Créer des groupes de listes et appliquer des styles à des listes avec Bootstrap

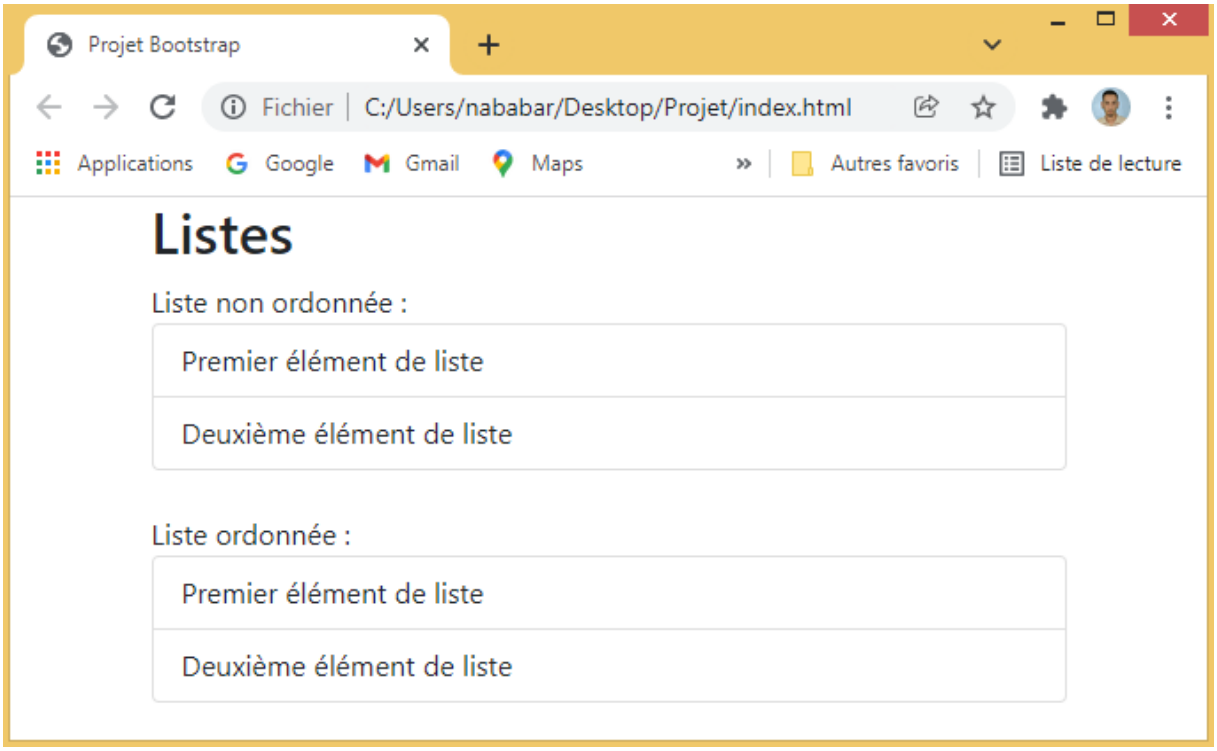
Bootstrap nous permet d'ajouter des styles à nos listes et de créer des groupes de listes ou « list groups » qui correspondent à des ensembles d'éléments affichés sous forme de liste.

Les classes .list-group et .list-group-item et la mise en forme de listes HTML

Pour appliquer une première mise en forme relativement basique à nos listes HTML, on va pouvoir utiliser les classes .list-group sur l'élément représentant la liste et .list-group-item sur les éléments de la liste.

Comme vous pouvez le remarquer, Bootstrap va appliquer exactement les mêmes styles aux listes non-ordonnées (élément HTML ul) et aux listes ordonnées (élément HTML ol).

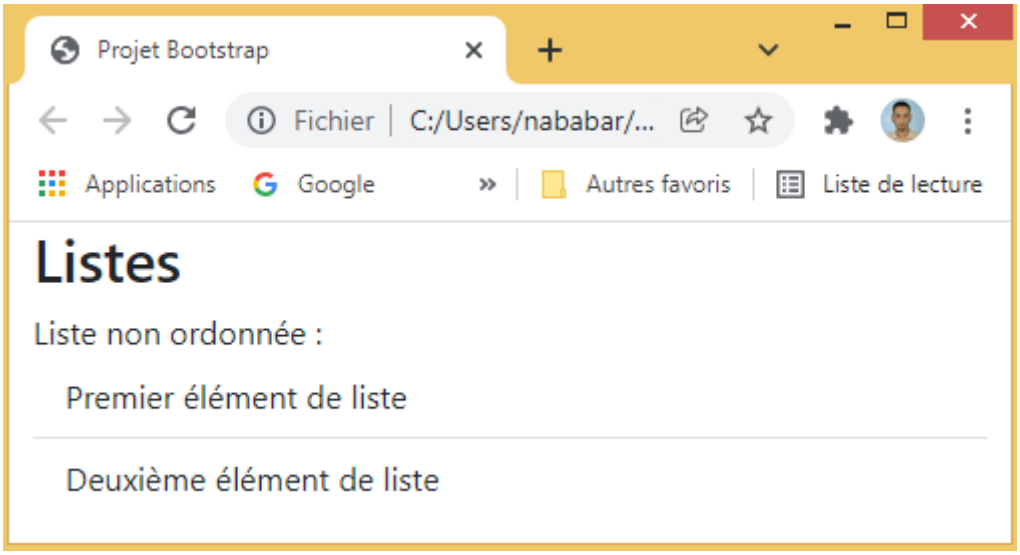
```
<div class="container">
    <h1>Listes</h1>
    <ul class="list-group">Liste non ordonnée :
        <li class="list-group-item">Premier élément de liste</li>
        <li class="list-group-item">Deuxième élément de liste</li>
    </ul>
    <br>
    <ol class="list-group">Liste ordonnée :
        <li class="list-group-item">Premier élément de liste</li>
        <li class="list-group-item">Deuxième élément de liste</li>
    </ol>
</div>
```



Supprimer les bordures des listes

Nous allons pouvoir supprimer les bordures autour d'une liste Bootstrap en ajoutant la classe .list-group-flush à notre élément représentant la liste en soi.

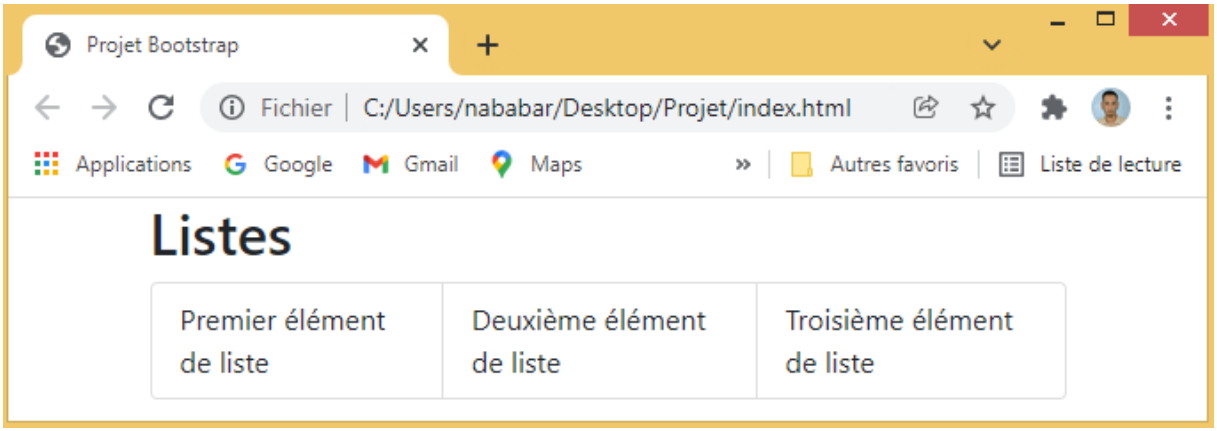
```
<div class="container">
    <h1>Listes</h1>
    <ul class="list-group list-group-flush">Liste non ordonnée :
        <li class="list-group-item">Premier élément de liste</li>
        <li class="list-group-item">Deuxième élément de liste</li>
    </ul>
</div>
```



Créer des listes horizontales

On va pouvoir changer la présentation de nos listes en faisant en sorte que les éléments de celles-ci s'affichent en ligne plutôt qu'en colonne en utilisant la classe .list-group-horizontal sur l'élément représentant la liste.

```
<div class="container">
    <h1>Listes</h1>
    <ul class="list-group list-group-horizontal">
        <li class="list-group-item">Premier élément de liste</li>
        <li class="list-group-item">Deuxième élément de liste</li>
        <li class="list-group-item">Troisième élément de liste</li>
    </ul>
</div>
```



Créer une liste de liens ou de boutons avec Bootstrap

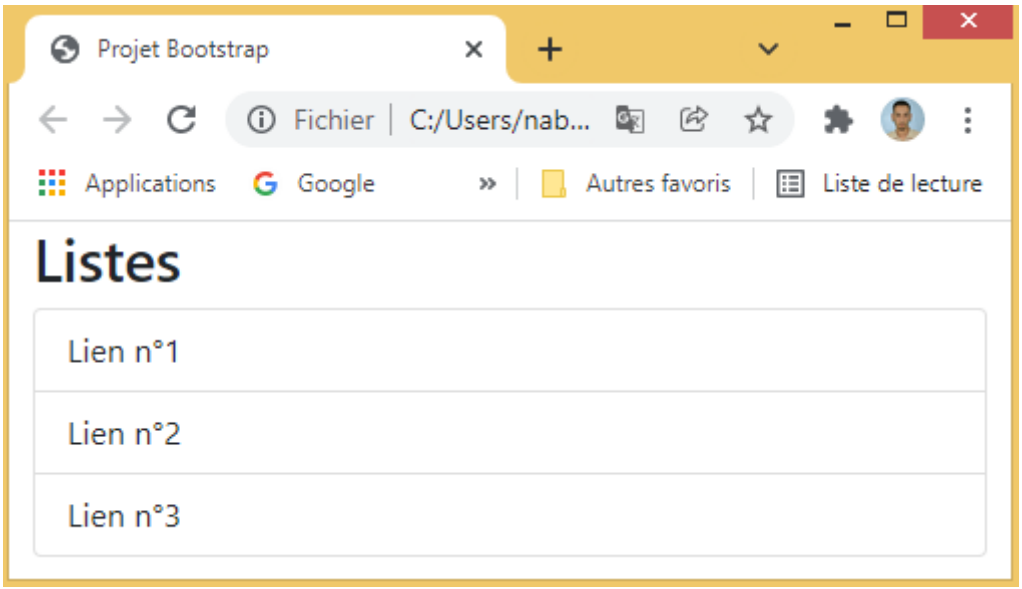
Bootstrap va nous permettre de créer des listes de boutons ou de liens ou plus exactement de styliser des groupes de boutons ou de liens afin de simuler l'apparence d'une liste.

Pour cela, nous allons simplement devoir utiliser un élément conteneur pour notre groupe de boutons ou de liens et lui attribuer une classe .list-group.

Ensuite, nous allons devoir réutiliser la classe .list-group-item pour chaque élément de notre « liste ».

Note : dans le cas d'une liste de boutons, n'utilisez pas de classe .btn.

```
<div class="container">
    <h1>Listes</h1>
    <div class="list-group">
        <a href="#" class="list-group-item list-group-item-action">Lien
n°1</a>
        <a href="#" class="list-group-item list-group-item-action">Lien
n°2</a>
        <a href="#" class="list-group-item list-group-item-action">Lien
n°3</a>
    </div>
</div>
```



Structurer et styliser des formulaires avec Bootstrap

Bootstrap nous offre de nombreuses classes pour nous permettre de structurer nos formulaires et d'ajouter des styles aux différents éléments des formulaires.

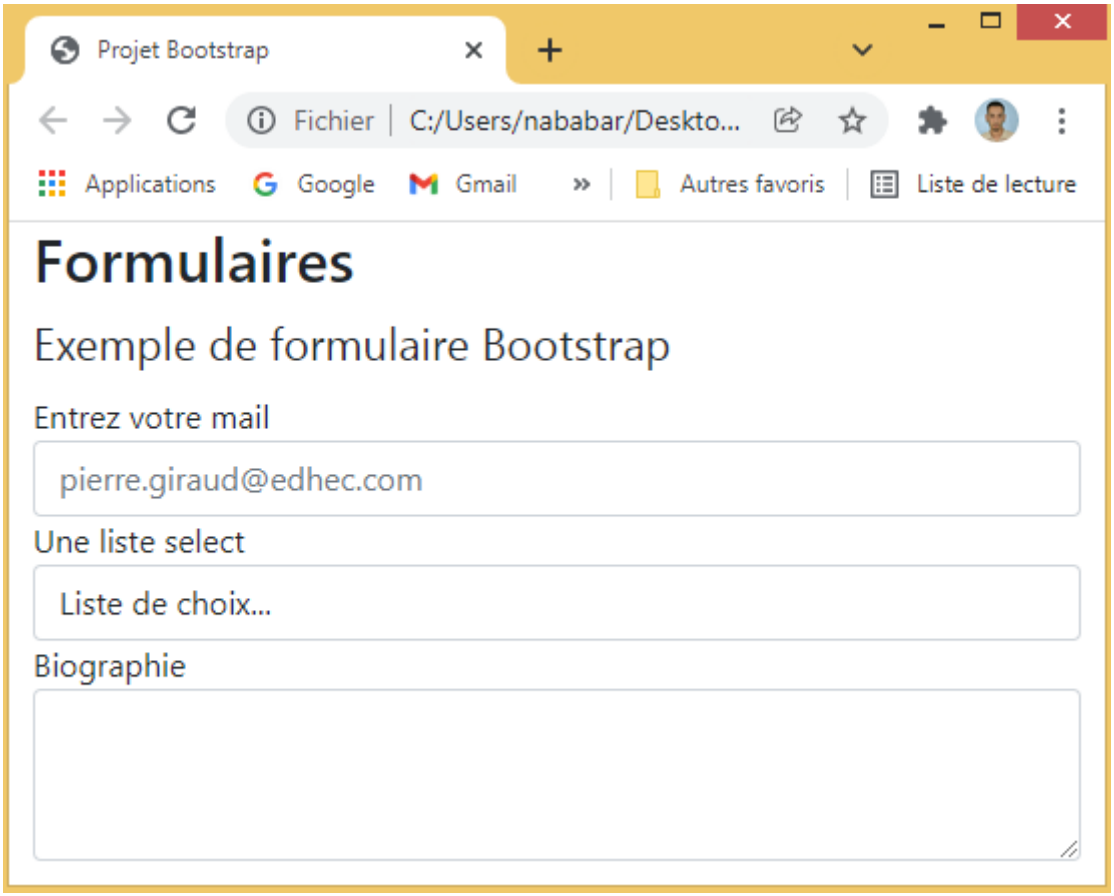
Styliser les champs de formulaire

Bootstrap applique un display : block à la plupart des éléments de formulaire par défaut, ce qui signifie que la plupart des éléments occuperont leur propre ligne.

On va utiliser la classe .form-control pour mettre en forme la majorité des champs de formulaire et notamment les éléments input, select et textarea.

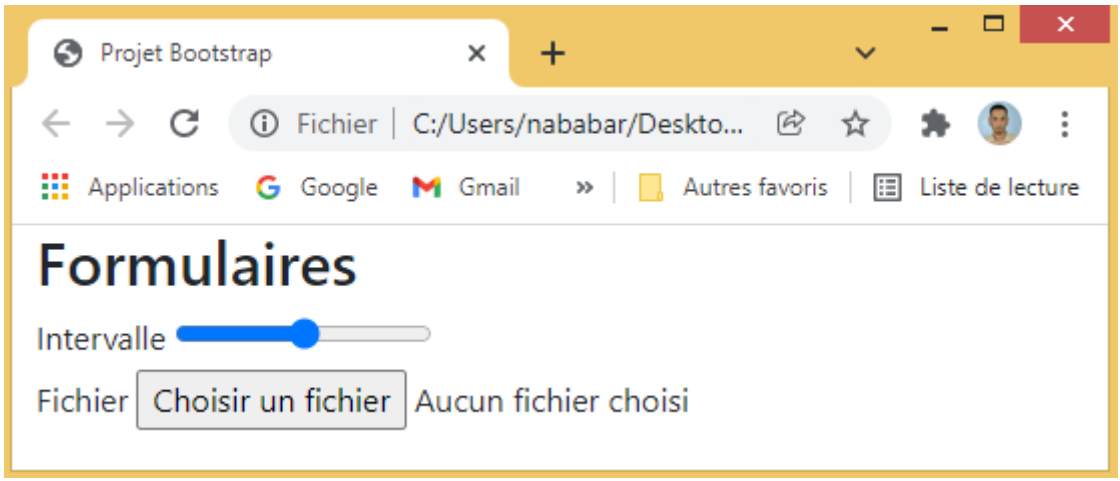
Nous allons également utiliser une classe .form-group pour grouper des label avec le champ de formulaire correspondant. Nous reparlerons de cette classe plus tard.

```
<div class="container">
    <h1>Formulaires</h1>
    <form>
        <fieldset>
            <legend>Exemple de formulaire Bootstrap</legend>
            <div class="form-group">
                <label for="email">Entrez votre mail</label>
                <input type="email" class="form-control" id="email"
placeholder="pierre.giraud@edhec.com">
            </div>
            <div class="form-group">
                <label for="selection">Une liste select</label>
                <select id="selection" class="form-control">
                <option value="">Liste de choix...</option>
                <optgroup label="Groupe d'options 1">
                    <option value="">Option 1</option>
                    <option value="">Option 2</option>
                    <option value="">Option 3</option>
                </optgroup>
                <optgroup label="Groupe d'options 2">
                    <option value="">Option 4</option>
                    <option value="">Option 5</option>
                </optgroup>
                </select>
            </div>
            <div class="form-group">
                <label for="bio">Biographie</label>
                <textarea class="form-control" id="bio" rows="3"></textarea>
            </div>
        </fieldset>
    </form>
</div>
```



Note : on utilisera la classe .form-control-file plutôt que .form-control avec un input type = "file" (champ servant à un envoi de fichier). De même, on utilisera .form-control-range avec un input type = "range" (champ d'intervalle).

```
<div class="form-group mb-1">
    <label for="intervalle">Intervalle</label>
    <input type="range" class="form-control-range" id="intervalle">
</div>
<div class="form-group">
    <label for="fichier">Fichier</label>
    <input type="file" class="form-control-file" id="fichier">
</div>
```



Valider les données des formulaires avec Bootstrap

Nous allons pouvoir utiliser Bootstrap et ses composantes pour effectuer une validation de nos formulaires HTML côté client. Cette validation ne devrait en aucun cas se substituer à une validation puissante effectuée côté serveur.

En effet, la validation côté client va plutôt servir à traiter les erreurs d'utilisateurs négligents / étourdis et leur fournir des indications sur les données attendues mais n'est pas là pour prévenir certains comportements dangereux d'utilisateurs mal intentionnés.

Côté client, Bootstrap s'appuie sur les deux pseudo-classes CSS :invalid et :valid pour la validation des formulaires. Ces deux pseudo-classes s'appliquent aux éléments input, select et textarea.

Bootstrap étend les styles des pseudo-classes :invalid et :valid à la classe .was-validated.

Si on souhaite fournir des informations sur la validité des champs avant qu'un utilisateur ait envoyé le formulaire, on ajoutera la classe .was-validated à l'élément form.

Les groupes d'input

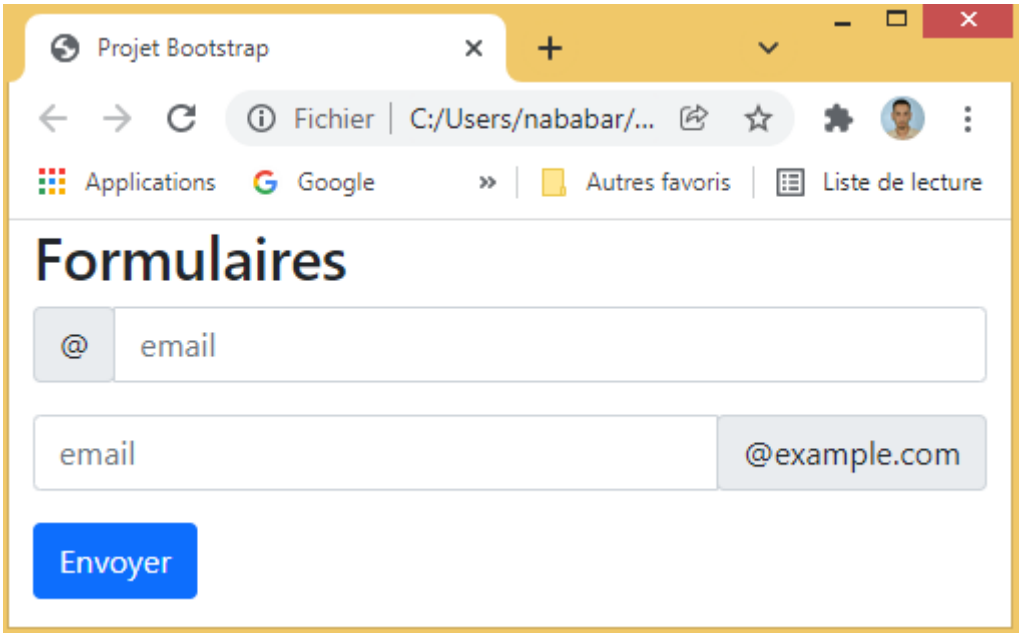
Bootstrap va nous permettre de personnaliser nos formulaires en nous donnant la possibilité de « grouper » des éléments de formulaires ensemble. En pratique, nous allons pouvoir ajouter du texte, des boutons, etc. de chaque côté de nos différents éléments input, textarea ou select pour donner des précisions sur les données attendues à nos utilisateurs ou simplement pour rendre nos formulaires plus esthétiques.

Pour cela, nous allons devoir utiliser la classe Bootstrap .input-group que l'on va généralement appliquer à un div qui contiendra un élément de formulaire ainsi que le(s) texte(s) et / ou le(s) bouton(s) qu'on va vouloir grouper avec l'élément de formulaire.

Notez déjà que lorsqu'on voudra ajouter du texte de part et d'autre d'un élément de formulaire au sein d'un groupe d'input, nous utiliserons la classe .input-group-text qu'on appliquera à l'élément contenant le texte.

Pour décider de l'emplacement du texte ou du bouton, nous utiliserons cette fois-ci les classes .input-group-prepend pour placer le texte ou le bouton avant l'élément de formulaire et .input-group-append pour placer le texte ou le bouton après l'élément de formulaire.

```
<div class="container">
    <h1>Formulaires</h1>
    <form>
        <div class="input-group mb-3">
            <div class="input-group-prepend"><span class="input-group-text"
aria-label="arobase">@</span></div>
            <input type="email" class="form-control" placeholder="email"
id="email">
        </div>
        <div class="input-group mb-3">
            <input type="text" class="form-control" placeholder="email"
id="debut-email">
            <div class="input-group-append"><span class="input-group-text"
aria-label="fin de mail">@example.com</span></div>
        </div>
        <button class="btn btn-primary" type="submit">Envoyer</button>
    </form>
</div>
```



Créer un menu de navigation avec Bootstrap

Dans cette leçon, nous allons voir comment utiliser les classes de type .nav-* pour construire des menus de navigation avec Bootstrap.

On va également utiliser les propriétés du flexbox pour créer des menus responsive.

Créer un menu de navigation avec Bootstrap

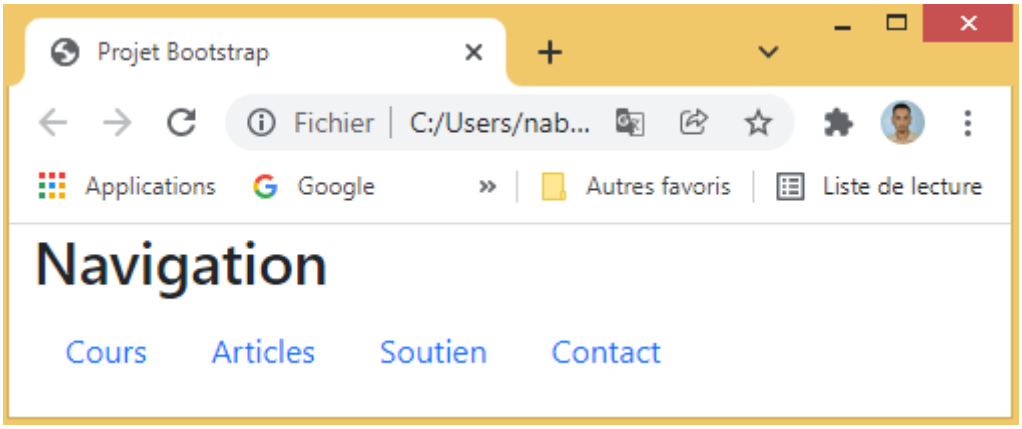
La classe .nav est la classe Bootstrap de base pour la navigation. Cette dernière est construite avec le flexbox et va par défaut appliquer les styles suivants :

```css
.nav {
  display: flex;
  flex-wrap: wrap;
  padding-left: 0;
  margin-bottom: 0;
  list-style: none;
}
```

Nous allons appliquer la classe .nav à l'élément représentant notre barre de navigation. Ce sera généralement un élément de liste ul. On pourra entourer cet élément ul d'un élément nav pour indiquer que le composant créé est bien un composant de navigation.

On va ensuite pouvoir ajouter les classes .nav-item et .nav-link à nos éléments li (éléments ou onglets de navigation) et a (lien de navigation) pour styliser ces éléments.

```html
<div class="container">
    <h1>Navigation</h1>
    <nav>
        <ul class="nav">
            <li class="nav-item">
                <a class="nav-link" href="#">Cours</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="#">Articles</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="#">Soutien</a>
            </li>
            <li class="nav-item">
                <a class="nav-link " href="#">Contact</a>
            </li>
        </ul>
    </nav>
</div>
```



Aligner notre menu de navigation

La classe .nav s'appuie sur le modèle des boites flexibles. On va donc pouvoir aligner nos éléments de navigation horizontalement en utilisant les classes .justify-content-center et .justify-content-end.

Changer la direction de la navigation

Pour obtenir un menu de navigation vertical plutôt qu'horizontal, il suffit d'ajouter la classe .flex-column à l'élément qui possède la classe .nav.

Modifier la taille des éléments de navigation

On va pouvoir demander aux éléments de navigation d'occuper tout l'espace disponible en ajoutant une classe .nav-fill ou .nav-justified à l'élément possédant la classe .nav.

Dans le cas où on utilise .nav-fill, les différents éléments occuperont une place différente dans le menu en fonction de leur taille. En utilisant .nav-justified, en revanche, on demande à chaque élément d'occuper la même largeur.

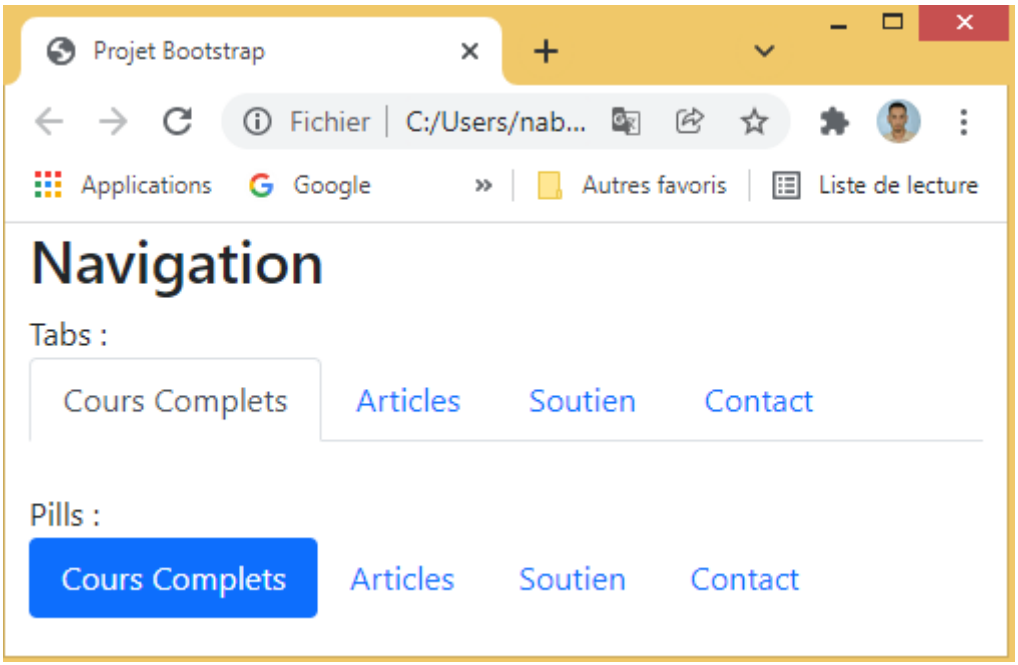
Changer l'apparence des éléments : tabs et pills

On va pouvoir changer l'apparence de nos éléments de menu et créer des menus à onglets ou à pills avec les classes .nav-tabs et .nav-pills.

Ces classes vont notamment nous permettre d'utiliser la classe .active qui n'a aucun effet visible avec la classe de base .nav.

```
<div class="container">
    <h1>Navigation</h1>
    <nav>Tabs :
        <ul class="nav nav-tabs">
          <li class="nav-item"><a class="nav-link active" href="#">Cours
Complets</a></li>
          <li class="nav-item"><a class="nav-link" href="#">Articles</a></li>
          <li class="nav-item"><a class="nav-link" href="#">Soutien</a></li>
          <li class="nav-item"><a class="nav-link " href="#">Contact</a></li>
```

```
        </ul>
    </nav>
    <br>
    <nav>Pills :
    <ul class="nav nav-pills">
        <li class="nav-item"><a class="nav-link active" href="#">Cours
Complets</a></li>
        <li class="nav-item"><a class="nav-link" href="#">Articles</a></li>
        <li class="nav-item"><a class="nav-link" href="#">Soutien</a></li>
        <li class="nav-item"><a class="nav-link " href="#">Contact</a></li>
    </ul>
    </nav>
</div>
```



Créer des éléments et des menus déroulants avec Bootstrap

Dans cette nouvelle leçon, nous allons nous intéresser au composant « dropdown » de Bootstrap qui va nous servir à dérouler des listes d'éléments et à les replier en cliquant sur un autre élément contrôlant le déroulement.

Nous allons également voir comment utiliser ce composant conjointement avec .nav pour créer des menus déroulants.

Création d'un élément déroulant ou dropdown

Pour créer un élément déroulant, on commence par utiliser la classe .dropdown sur un élément conteneur.

Deux types d'éléments HTML vont pouvoir servir à déclencher un déroulement : les éléments HTML a (lien) ou button (bouton). On va leur appliquer une classe .dropdown-toggle qui permet de déplier (d'afficher) ou de replier (de cacher) le contenu déroulant.

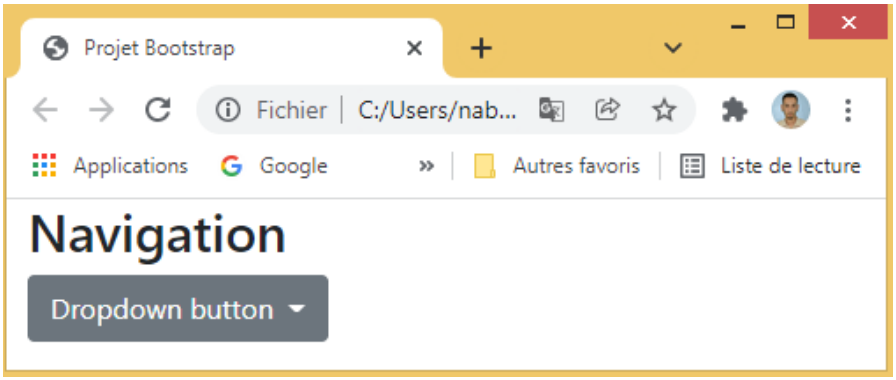
Nous allons ensuite créer le contenu déroulant en soi et le mettre en forme. Ici, nous allons déjà créer un conteneur générique div et lui ajouter une classe .dropdown-menu.

Dans ce div, on va pouvoir définir deux types d'éléments déroulants : des éléments interactifs et des éléments non interactifs.

On crée des éléments déroulants interactifs en utilisant soit des éléments button, soit des éléments a auxquels on ajoute des classes .dropdown-item.

Pour créer un élément non interactif, on peut par exemple utiliser un élément span auquel on ajoutera une classe .dropdown-item-text.

```
<div class="container">
    <h1>Navigation</h1>
    <div class="dropdown">
        <button class="btn btn-secondary dropdown-toggle" type="button"
id="dropdownMenuButton1" data-bs-toggle="dropdown" aria-expanded="false">
            Dropdown button
        </button>
        <ul class="dropdown-menu" aria-labelledby="dropdownMenuButton1">
            <li><a class="dropdown-item" href="#">Action</a></li>
            <li><a class="dropdown-item" href="#">Another action</a></li>
            <li><a class="dropdown-item" href="#">Something else here</a></li>
        </ul>
    </div>
</div>
```



Changer la direction du déroulement

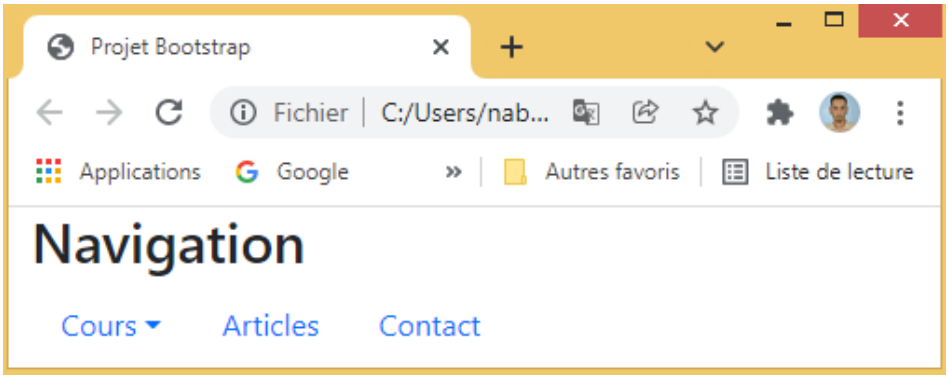
On va pouvoir utiliser les classes .dropup, .dropright et .dropleft à la place de .dropdown afin que nos éléments déroulants soient déroulés au-dessus, à droite ou à gauche de l'élément déclenchant le déroulement.

Notez que ce comportement ne fonctionnera que si les éléments ont effectivement la place de se dérouler.

Créer un menu déroulant avec .nav et .dropdown

On va bien évidemment pouvoir utiliser les classes .dropdown-* avec les classes .nav-* pour créer des menus déroulants.

```
<div class="container">
    <h1>Navigation</h1>
    <nav>
        <ul class="nav">
        <li class="nav-item dropdown">
            <a class="nav-link dropdown-toggle" href="#" data-
toggle="dropdown" aria-haspopup="true" aria-expanded="false">Cours</a>
            <div class="dropdown-menu">
            <a class="dropdown-item" href="#">HTML et CSS</a>
            <a class="dropdown-item" href="#">JavaScript</a>
            <a class="dropdown-item" href="#">PHP et MySQL</a>
            </div>
        </li>
        <li class="nav-item"><a class="nav-link " href="#">Articles</a></li>
        <li class="nav-item"><a class="nav-link " href="#">Contact</a></li>
        </ul>
    </nav>
</div>
```



Présentation des barres de navigation Bootstrap et de la classe .navbar

Pour créer une barre de navigation avec Bootstrap, on va déjà devoir ajouter une classe .navbar à un élément qui va servir de conteneur à notre barre de navigation (généralement un élément nav ou div).

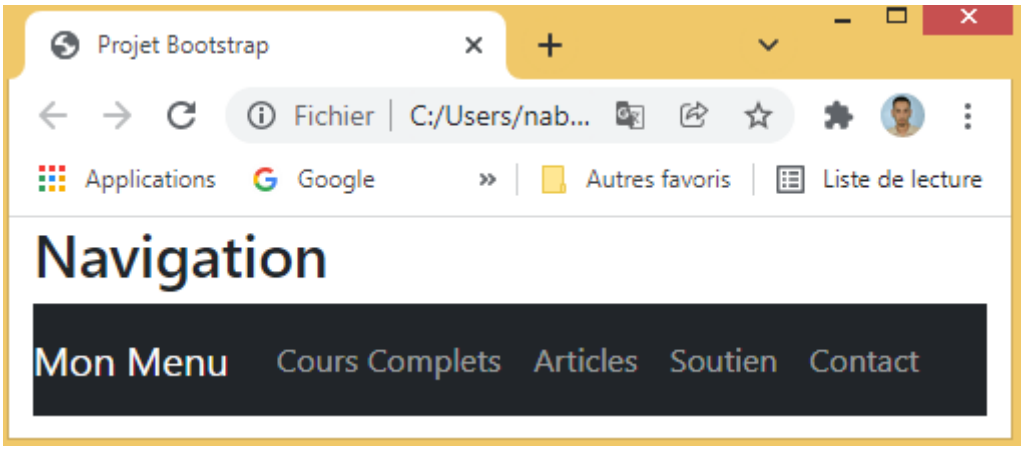
Les barres de navigation Bootstrap ont une taille fluide par défaut, c'est-à-dire qu'elles vont occuper tout l'espace disponible et se redimensionner en même temps que la fenêtre.

Nous allons ensuite pouvoir ajouter différents composants à notre barre de navigation comme un nom de marque, un menu, un champ de recherche, etc. Ces composants vont être mis en forme grâce aux classes suivantes :

- La classe .navbar-brand pour le nom de la marque ;
- La classe .navbar-nav pour le menu de navigation ;
- La classe .form-inline pour ajouter un champ de formulaire ;
- La classe .navbar-text pour ajouter tout autre texte.

En plus de cela, les classes .navbar-expand{-sm|-md|-lg|-xl} utilisées avec .navbar-toggler, .collapse et .navbar-collapse vont nous permettre de changer l'apparence de notre barre de navigation en fonction de la taille de la fenêtre pour proposer une meilleure ergonomie.

```
<div class="container">
    <h1>Navigation</h1>
    <nav class="navbar navbar-expand navbar-dark bg-dark">
        <a class="navbar-brand" href="#">Mon Menu</a>
        <ul class="navbar-nav">
            <li class="nav-item"><a class="nav-link" href="#">Cours
Complets</a></li>
            <li class="nav-item"><a class="nav-link"
href="#">Articles</a></li>
            <li class="nav-item"><a class="nav-link" href="#">Soutien</a></li>
            <li class="nav-item"><a class="nav-link" href="#">Contact</a></li>
        </ul>
    </nav>
</div>
```



Le composant nom de marque et la classe .navbar-brand

On va utiliser la classe .navbar-brand pour appliquer des styles à un nom de marque dans une barre de navigation. On va pouvoir ajouter cette classe à la plupart des éléments HTML.

Le composant menu et la classe .navbar-nav

La classe .navbar-nav va avoir un rôle similaire à la classe .nav vue précédemment. Par défaut, le menu créé va s'afficher en colonne.

On va pouvoir ajouter une classe .navbar-expand à l'élément portant la classe .navbar pour afficher le menu en ligne. Notez que cette classe possède des variations responsive avec .navbar-expand{-sm|-md|-lg|-xl} pour appliquer différentes mises en forme à notre barre de navigation en fonction de la taille de la fenêtre.

Le composant champ de formulaire et la classe .form-online

On va encore pouvoir ajouter des champs de formulaire et particulièrement un champ de recherche dans notre barre de navigation grâce à la classe .form-online.

Pour que ce champ de recherche s'affiche à droite de la barre de navigation, on va pouvoir ajouter .mr-auto à notre menu.

Le composant texte et la classe .navbar-text

Enfin, on va également pouvoir ajouter du texte dans notre barre de navigation en utilisant la classe .navbar-text.

Créer une barre de navigation responsive

Par défaut, les barres de navigation possèdent une taille fluide, ce qui fait qu'elles rétrécissent ou s'étendent en même temps que la fenêtre.

On voudra cependant souvent aller plus loin dans l'adaptabilité de la barre et par exemple cacher le texte du menu sur des petits écrans et proposer à la place une icône permettant d'ouvrir ou de fermer le menu.

Pour faire cela, on va déjà devoir ajouter une classe .navbar-expand{-sm|-md|-lg|-xl} à l'élément portant la classe .navbar. La classe .navbar-expand-* va nous permettre d'indiquer à partir de quelle taille de fenêtre tous les textes de notre barre de navigation doivent être affichés.

Ensuite, on va utiliser les classes .collapse et .navbar-collapse autour des textes qu'on souhaite cacher pour les petites fenêtres et .navbar-toggler et .navbar-toggler-icon pour afficher / cacher ces textes sur petit écran lors d'un clic sur l'icône « burger » affichée grâce à la classe .navbar-toggler-icon.

Les autres composants de navigation Bootstrap : breadcrumb et pagination

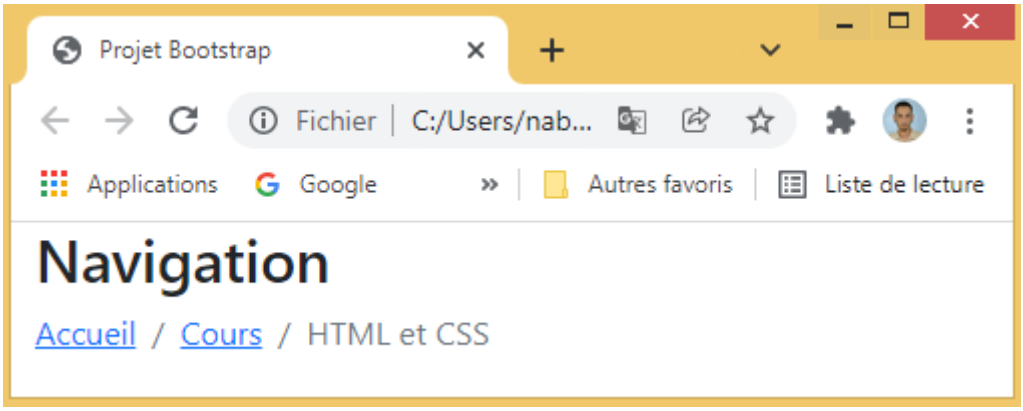
Bootstrap nous permet de créer davantage de composants aidant à la navigation comme un fil d'Ariane, un système de pagination, ou encore une aide au défilement pour nos pages web.

Le fil d'Ariane ou breadcrumb

Le fil d'Ariane ou breadcrumb sert aux utilisateurs et moteurs de recherche à se repérer dans l'arborescence d'un site. Il permet de rapidement comprendre où se situe une page par rapport aux autres.

On va pouvoir créer un fil d'Ariane avec Bootstrap en ajoutant la classe .breadcrumb à un élément nav et à un élément de liste ainsi que la classe .breadcrumb-item aux différents éléments composant notre fil d'Ariane.

```
<div class="container-fluid">
    <h1>Navigation</h1>
    <nav aria-label="breadcrumb">
        <ol class="breadcrumb">
        <li class="breadcrumb-item"><a href="#">Accueil</a></li>
        <li class="breadcrumb-item"><a href="#">Cours</a></li>
        <li class="breadcrumb-item active" aria-current="page">HTML et
CSS</li>
        </ol>
    </nav>
</div>
```

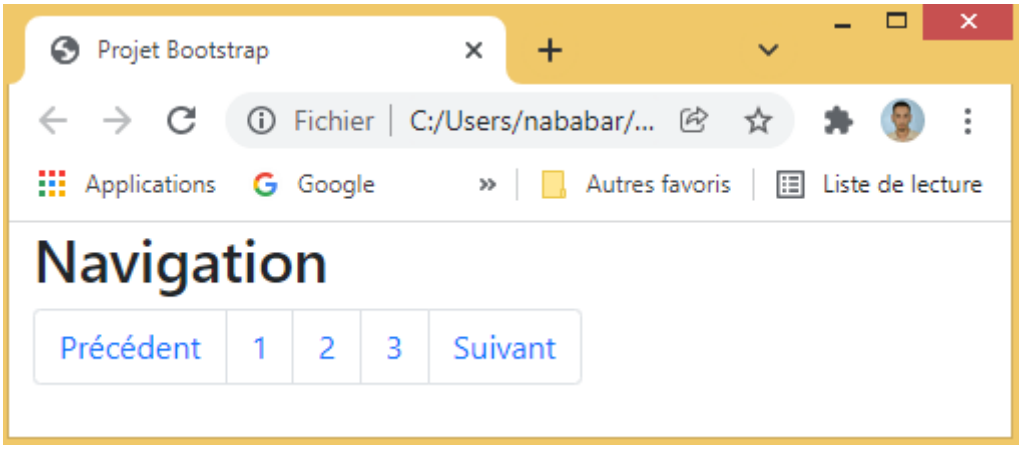


La pagination

Pour mettre en place un système de pagination, on va déjà utiliser un élément de liste ul auquel on va ajouter une classe .pagination.

Ensuite, nous allons rajouter des classes .page-item à chaque élément de liste et des classes .page-link pour chaque lien contenu dans les éléments de liste.

```
<div class="container-fluid">
    <h1>Navigation</h1>
    <nav aria-label="Exemple de pagination">
        <ul class="pagination">
            <li class="page-item"><a class="page-link"
href="#">Précédent</a></li>
            <li class="page-item"><a class="page-link" href="#">1</a></li>
            <li class="page-item"><a class="page-link" href="#">2</a></li>
            <li class="page-item"><a class="page-link" href="#">3</a></li>
            <li class="page-item"><a class="page-link"
href="#">Suivant</a></li>
        </ul>
    </nav>
</div>
```



Alertes, boites modales et notifications toast Bootstrap

Dans cette nouvelle leçon, nous allons voir comment créer et mettre en forme avec Bootstrap 3 composants informatifs incontournables que sont les boites d'alertes, boites modales et les notifications toast.

Les boites d'alertes Bootstrap

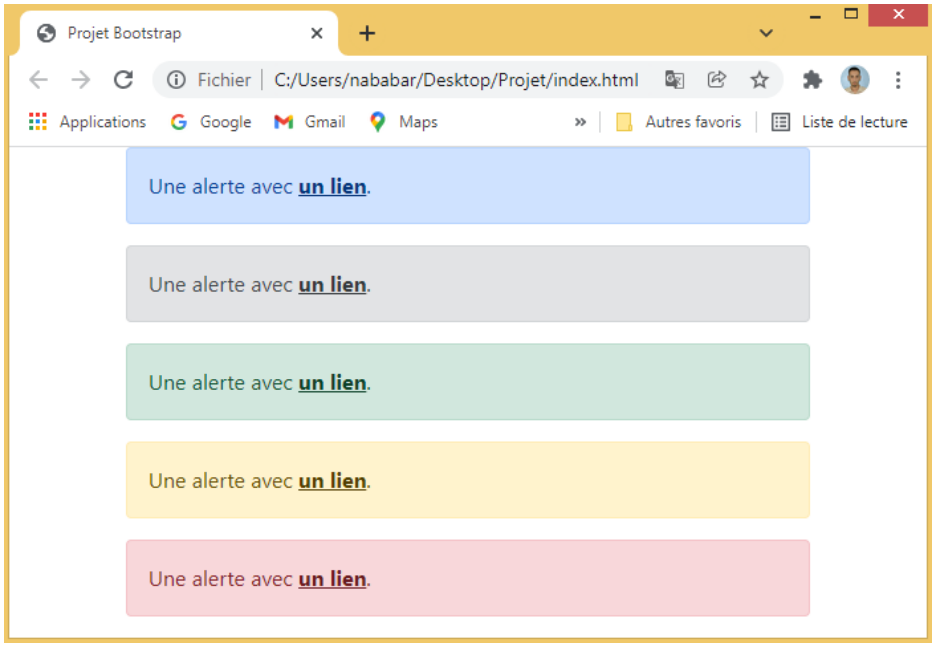
Les alertes Bootstrap sont des boites qui permettent d'afficher des informations contextualisées relatives à certaines actions de la part d'utilisateurs, comme par exemple lors du remplissage d'un formulaire.

Pour définir une alerte Bootstrap, on va ajouter une classe .alert à un conteneur générique comme un élément div. On va ensuite pouvoir changer la couleur des boites et des textes grâce aux classes .alert-{couleur-contextuelle}.

On va également pouvoir laisser la possibilité à nos utilisateurs de fermer une alerte pour la faire disparaitre en ajoutant un bouton avec une classe .close et un attribut data-dismiss="alert".

On ajoutera aussi des classes .alert-dismissible, .fade et .show pour positionner notre bouton et ajouter des effets lors de la disparition de l'alerte.

```
<div class="container">
    <div class="alert alert-primary" role="alert">Une alerte avec <a href="#"
class="alert-link">un lien</a>.</div>
    <div class="alert alert-secondary" role="alert">Une alerte avec <a
href="#" class="alert-link">un lien</a>.</div>
    <div class="alert alert-success" role="alert">Une alerte avec <a href="#"
class="alert-link">un lien</a>.</div>
    <div class="alert alert-warning" role="alert">Une alerte avec <a href="#"
class="alert-link">un lien</a>.</div>
    <div class="alert alert-danger" role="alert">Une alerte avec <a href="#"
class="alert-link">un lien</a>.</div>
</div>
```



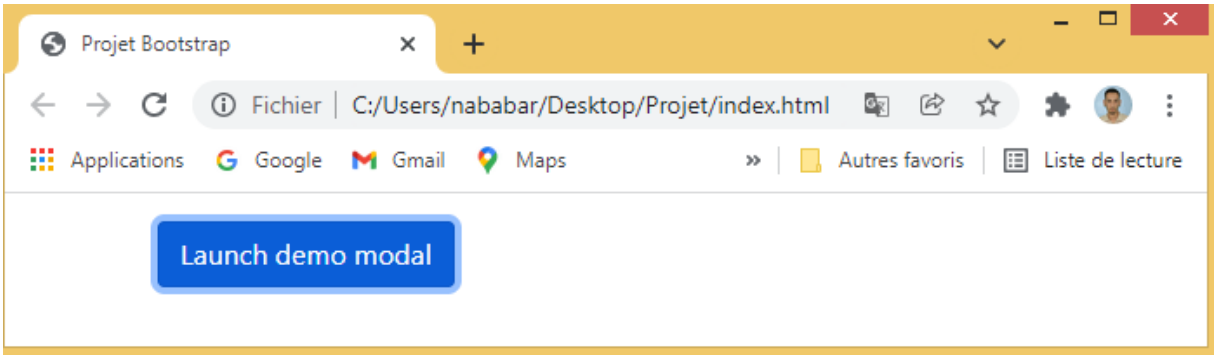
Les boites modales

Les boites modales sont des boites de dialogue qui vont apparaitre devant le reste du contenu de la page et qui vont toujours rester visible même lors d'un défilement dans la page grâce à leur position : fixed.

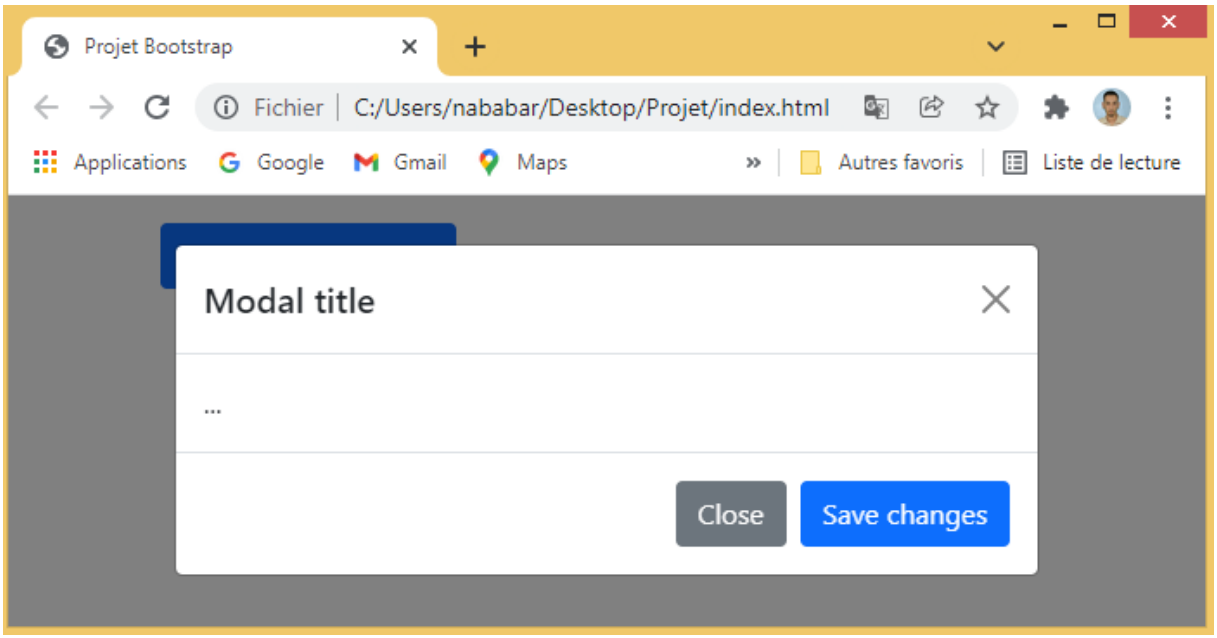
Pour créer un modal, on va déjà devoir définir 3 conteneurs génériques imbriqués les uns dans les autres auxquels on va ajouter des classes .modal, .modal-dialog et .modal-content.

On va ensuite pouvoir découper notre fenêtre modale en trois parties avec .modal-header, .modal-body et .modal-footer.

```
<div class="container mt-3">
    <!-- Button trigger modal -->
    <button type="button" class="btn btn-primary" data-bs-toggle="modal" data-
bs-target="#exampleModal">
        Launch demo modal
    </button>

    <!-- Modal -->
    <div class="modal fade" id="exampleModal" tabindex="-1" aria-
labelledby="exampleModalLabel" aria-hidden="true">
        <div class="modal-dialog">
        <div class="modal-content">
            <div class="modal-header">
            <h5 class="modal-title" id="exampleModalLabel">Modal title</h5>
            <button type="button" class="btn-close" data-bs-dismiss="modal"
aria-label="Close"></button>
            </div>
            <div class="modal-body">
            ...
            </div>
            <div class="modal-footer">
            <button type="button" class="btn btn-secondary" data-bs-
dismiss="modal">Close</button>
            <button type="button" class="btn btn-primary">Save
changes</button>
            </div>
        </div>
        </div>
    </div>
</div>
```



En cliquant sur le bouton, la boîte modale suivante s'affiche :



Les notifications toast

Les notifications toast sont des notifications créées pour imiter le comportement des notifications push popularisées notamment par les OS mobile et les applications.

Pour créer une notification toast, on va utiliser une classe .toast pour définir le toast en soi et des classes .toast-header et .toast-body pour définir l'en tête et le corps de notre notification.

Barres de progression et spinners Bootstrap

Les barres de progression et les spinners vont permettre d'indiquer aux utilisateurs qu'une action est en cours et de les informer dans le cas des barres de progression sur l'avancement de cette action.

Les barres de progression

Pour créer une barre de progression avec Bootstrap, nous allons déjà devoir ajouter une classe .progress à un élément conteneur (un div, par exemple).

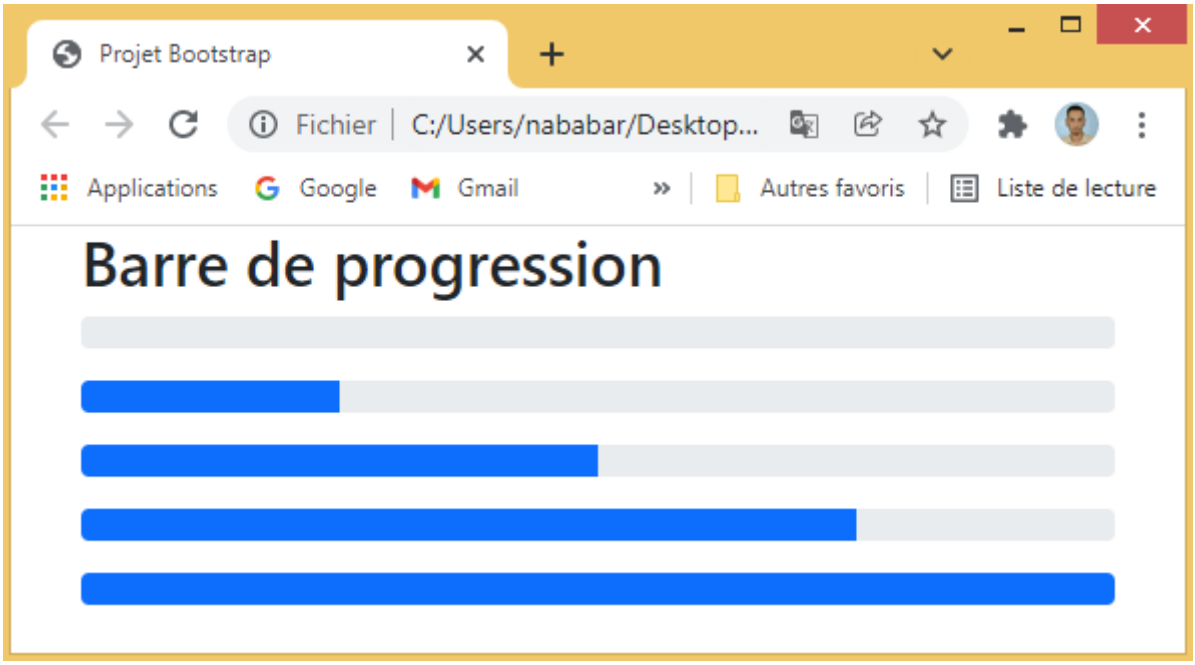
Ensuite, nous ajouterons une classe .progress-bar à un autre élément conteneur enfant pour matérialiser la barre de progression en soi.

Pour finalement définir le taux de remplissage de la barre de progression, c'est-à-dire la taille de la partie colorée de la barre nous allons tout simplement utiliser une propriété CSS width. Pour obtenir une barre remplie à 25%, par exemple, on utilisera width : 25% via un attribut style ou en utilisant la classe .w-25

Pour modifier la hauteur, il suffit de définir une propriété height pour le conteneur possédant la classe .progress.

Pour modifier la couleur de fond, on ajoutera une classe bg-* à l'élément possédant la classe .progress-bar.

```
<div class="container">
    <h1>Barre de progression</h1>
    <div class="progress mb-3">
        <div class="progress-bar" role="progressbar" aria-valuenow="0" aria-
valuemin="0" aria-valuemax="100"></div>
    </div>
    <div class="progress mb-3">
        <div class="progress-bar w-25" role="progressbar" aria-valuenow="25"
aria-valuemin="0" aria-valuemax="100"></div>
    </div>
    <div class="progress mb-3">
        <div class="progress-bar w-50" role="progressbar" aria-valuenow="50"
aria-valuemin="0" aria-valuemax="100"></div>
    </div>
    <div class="progress mb-3">
        <div class="progress-bar w-75" role="progressbar" aria-valuenow="75"
aria-valuemin="0" aria-valuemax="100"></div>
    </div>
    <div class="progress">
        <div class="progress-bar w-100" role="progressbar" aria-valuenow="100"
aria-valuemin="0" aria-valuemax="100"></div>
    </div>
</div>
```



On va également pouvoir obtenir des barres de progression zébrées en ajoutant une classe .progress-bar-striped à l'élément possédant la classe .progress-bar.

Pour animer les zébrures, on pourra encore rajouter une classe .progress-bar-animated.

Les spinners

On va utiliser le composant spinner pour indiquer aux utilisateurs qu'une action est en cours (document en train de charger, etc.).

Pour créer un spinner avec Bootstrap, on va pouvoir utiliser soit la classe .spinner-border soit la classe .spinner-grow en fonction du type d'effet qu'on recherche.

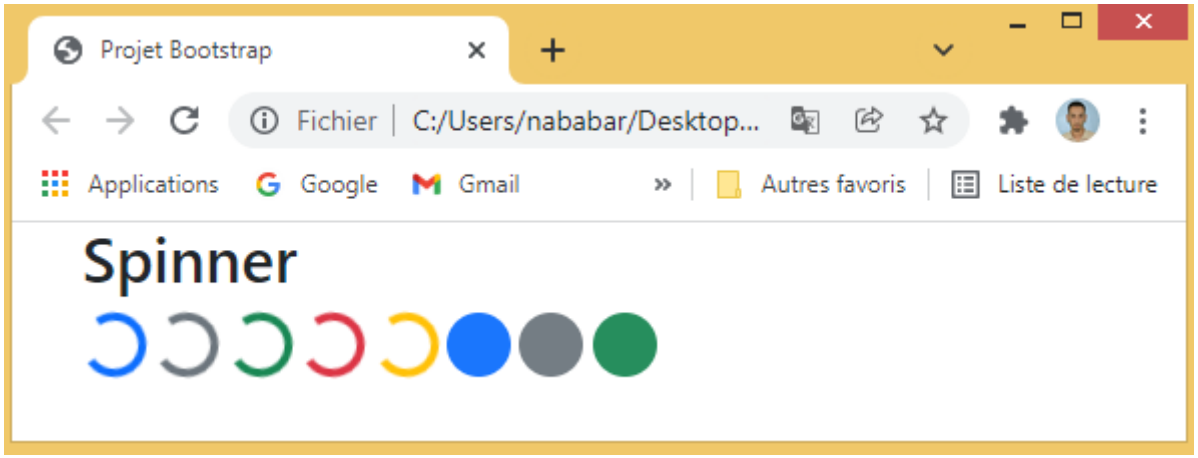
Par défaut, les spinners utilisent la couleur courante. On peut modifier cette couleur en utilisant une classe text-*.

On va inclure un attribut rôle et un message qui ne sera affiché que pour les liseuses d'écran grâce à la classe .sr-only pour des raisons d'accessibilité.

On va également pouvoir réduire la taille de nos spinners grâce aux classes .spinner-border-sm et .spinner-grow-sm, ce qui va nous permettre d'incorporer nos spinners dans d'autres éléments comme des boutons par exemple.

```
<div class="container">
    <h1>Spinner</h1>
    <div class="spinner-border text-primary" role="status">
        <span class="visually-hidden">Loading...</span>
      </div>
      <div class="spinner-border text-secondary" role="status">
        <span class="visually-hidden">Loading...</span>
      </div>
      <div class="spinner-border text-success" role="status">
        <span class="visually-hidden">Loading...</span>
      </div>
      <div class="spinner-border text-danger" role="status">
```

```
        <span class="visually-hidden">Loading...</span>
      </div>
      <div class="spinner-border text-warning" role="status">
        <span class="visually-hidden">Loading...</span>
      </div>
      <div class="spinner-grow text-primary" role="status">
        <span class="visually-hidden">Loading...</span>
      </div>
      <div class="spinner-grow text-secondary" role="status">
        <span class="visually-hidden">Loading...</span>
      </div>
      <div class="spinner-grow text-success" role="status">
        <span class="visually-hidden">Loading...</span>
      </div>
</div>
```



Badges Bootstrap

Les badges sont des composants visuels qui vont nous permettre d'attirer l'attention sur certains contenus particuliers en les faisant ressortir par rapport au reste de la page.

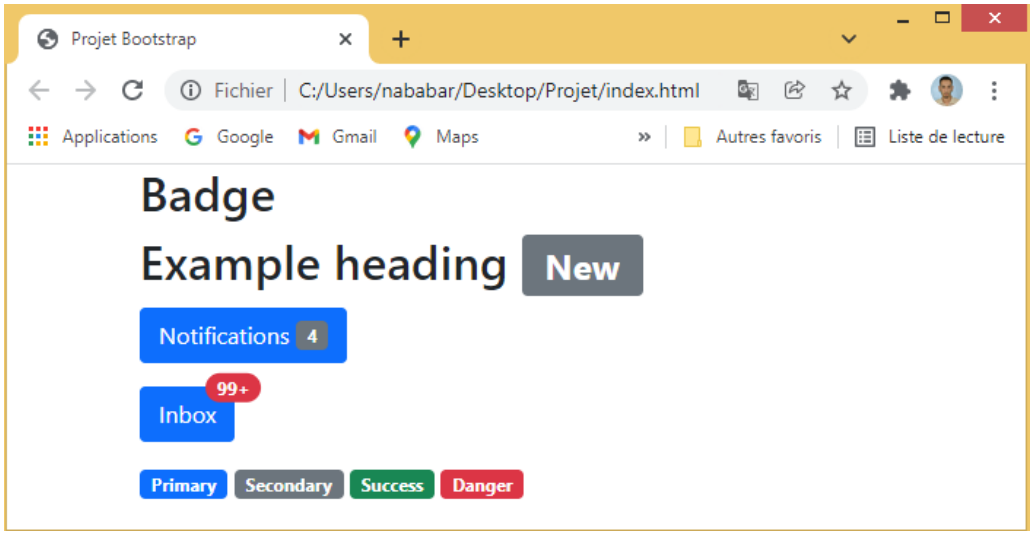
Les badges

On va utiliser les badges pour ajouter une information précise à un certain contenu. Les badges vont notamment être utilisés pour ajouter un label ou un compteur à un élément.

Pour créer un badge avec Bootstrap, nous allons utiliser la classe .badge qu'on va ajouter à un élément inline comme un span.

On va généralement également ajouter une deuxième classe .badge-{primary|success...} pour définir la couleur de fond du badge.

```
<div class="container">
    <h1>Badge</h1>
    <h1>Example heading <span class="badge bg-secondary">New</span></h1>
    <button type="button" class="btn btn-primary">
        Notifications <span class="badge bg-secondary">4</span>
    </button>
    <div class="mt-3">
        <button type="button" class="btn btn-primary position-relative">
            Inbox
            <span class="position-absolute top-0 start-100 translate-middle
badge rounded-pill bg-danger">99+<span class="visually-hidden">unread
messages</span>
            </span>
        </button>
    </div>
    <div class="mt-3">
        <span class="badge bg-primary">Primary</span>
        <span class="badge bg-secondary">Secondary</span>
        <span class="badge bg-success">Success</span>
        <span class="badge bg-danger">Danger</span>
    </div>
</div>
```


Projet Bootstrap
Fichier | C:/Users/nababar/Desktop/Projet/index.html
Applications
Google
Gmail
Maps
Autres favoris
Liste de lecture
Badge
Example heading
New
Notifications 4
99+
Inbox
Primary
Secondary
Success
Danger

Les cartes ou cards Bootstrap

Le composant carte ou « card » de Bootstrap est un conteneur flexible et extensible. Concrètement, il s'agit d'une boite rectangulaire avec bordure qui peut être composée d'un en-tête, d'un corps et d'un pied.

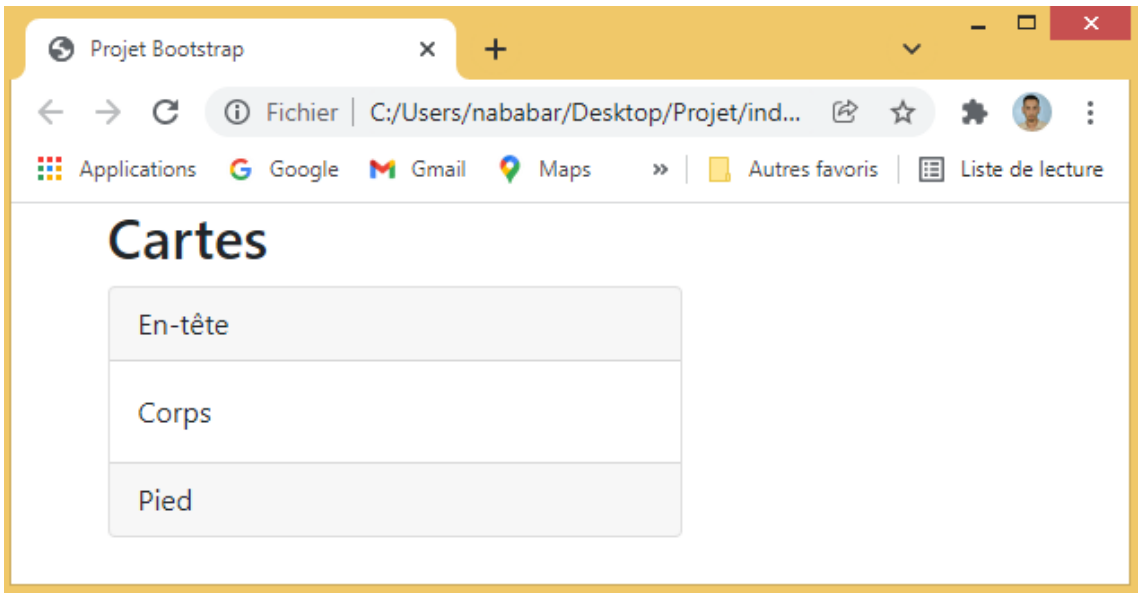
Les cartes n'ont pas de dimension intrinsèque, ce qui fait qu'elles tenteront d'occuper tout l'espace dans leur parent par défaut. Elles ne possèdent pas non plus de marges externes.

Structure d'une carte : header, body et footer

Pour définir une card, on va déjà devoir ajouter une classe .card à un élément conteneur comme un div par exemple.

On va ensuite pouvoir découper notre carte en trois parties distinctes : un en-tête avec la classe .card-header, un corps avec la classe .card-body et un pied avec la classe card-footer.

```
<div class="container">
    <h1>Cartes</h1>
    <div class="card" style="width: 20rem;">
        <div class="card-header">En-tête</div>
        <div class="card-body">Corps</div>
        <div class="card-footer">Pied</div>
    </div>
</div>
```



Les éléments des cartes et les classes associées

On va pouvoir placer plus ou moins n'importe quel contenu HTML dans une carte : du texte, des images, les listes, des liens, etc. Pour que certains de ces éléments soient convenablement mis en forme, on va devoir leur ajouter des classes de type card-*.

On va ainsi pouvoir ajouter une classe .card-title aux éléments de titre et une classe .card-subtitle pour les sous titres. Les autres textes vont pouvoir être mis en forme avec la classe .card-text.

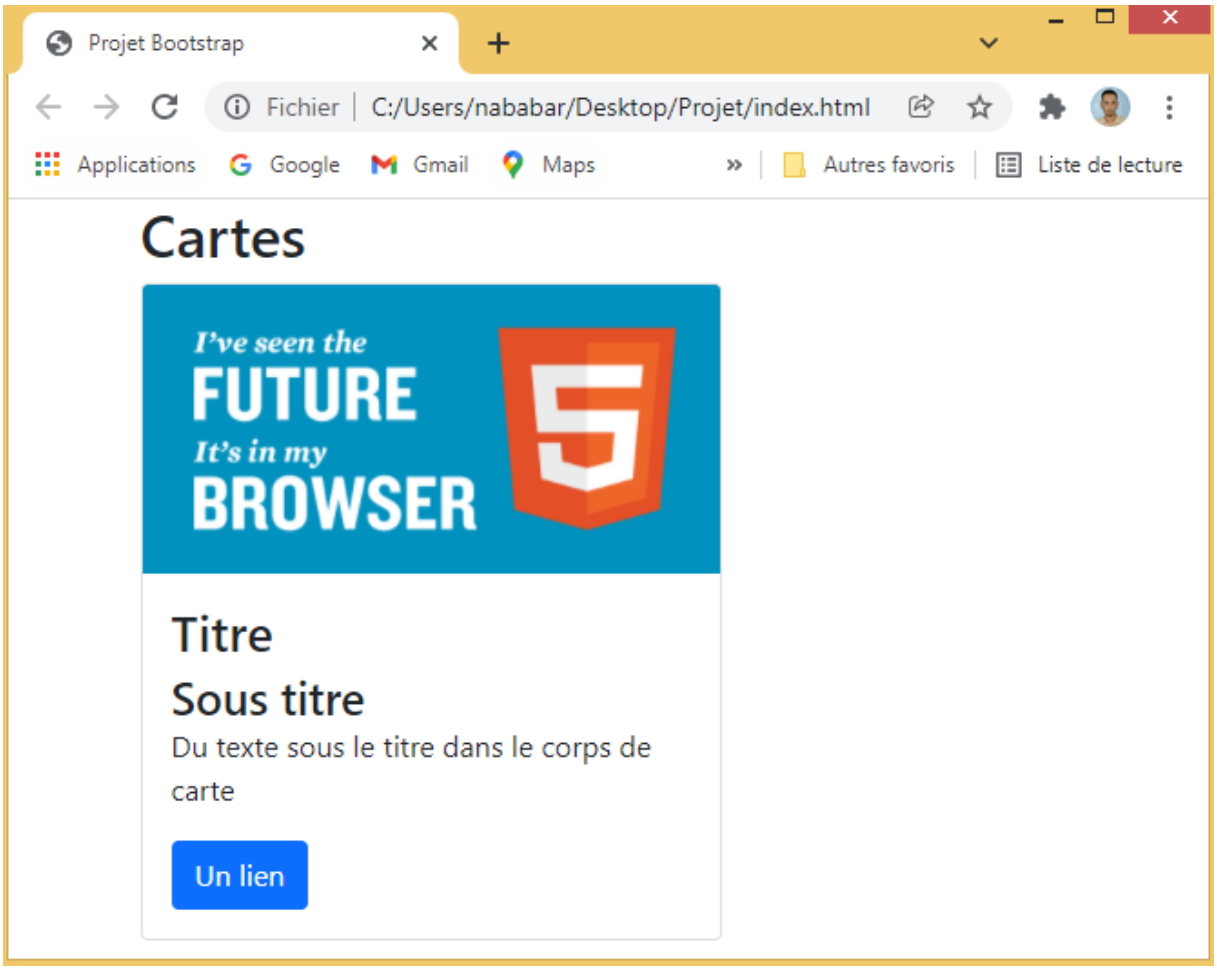
Pour les éléments de lien, on utilisera la classe .card-link.

Enfin, pour intégrer une image dans notre carte, on va pouvoir choisir entre les classes .card-img, .card-img-top et .card-img-bottom.

La classe .card-img va nous permettre d'ajouter une image qu'on va ensuite pouvoir utiliser comme overlay, c'est-à-dire en fond pour une certaine partie de la carte à laquelle on devra ajouter la classe .card-img-overlay.

Les classes .card-img-top et .card-img-bottom vont nous permettre de placer une image soit tout en haut d'une carte, soit tout en bas.

```
<div class="container">
    <h1>Cartes</h1>
    <div class="card" style="width: 20rem;">
        <img src="https://www.w3.org/html/logo/downloads/HTML5_sticker.png"
class="card-img-top" alt="Accroche HTML">
        <div class="card-body">
            <h2 class="card-title">Titre</h2>
            <h3 class="card-subtitle">Sous titre</h3>
            <p class="card-text">Du texte sous le titre dans le corps de
carte</p>
            <a href="#" class="btn btn-primary">Un lien</a>
        </div>
    </div>
</div>
```



TRAVAUX PRATIQUES

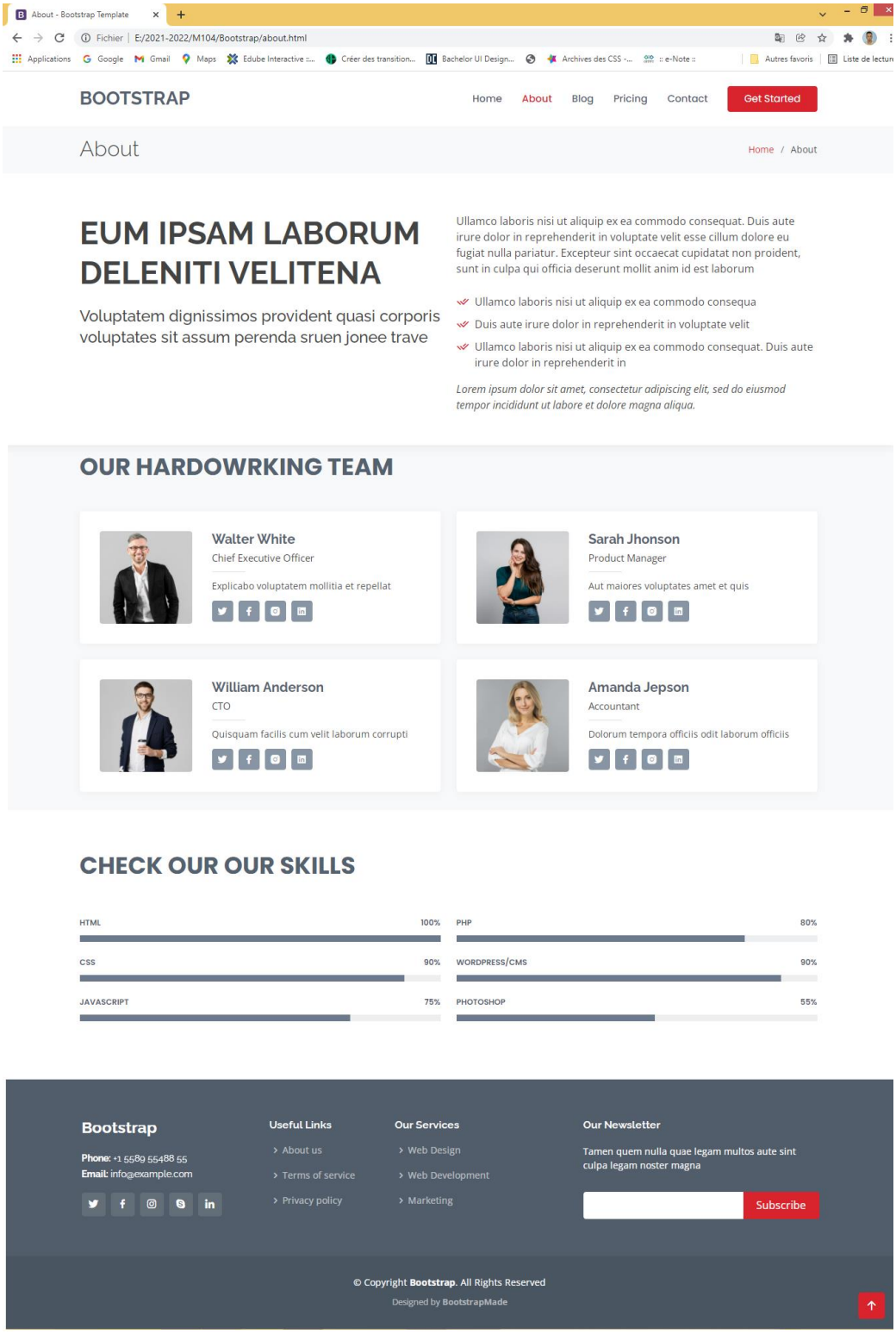
TP N° 1 : INTEGRER UN TEMPLATE BOOTSTRAP 5

Note : Vous trouvez les interfaces et les ressources de Template dans le lien suivant :

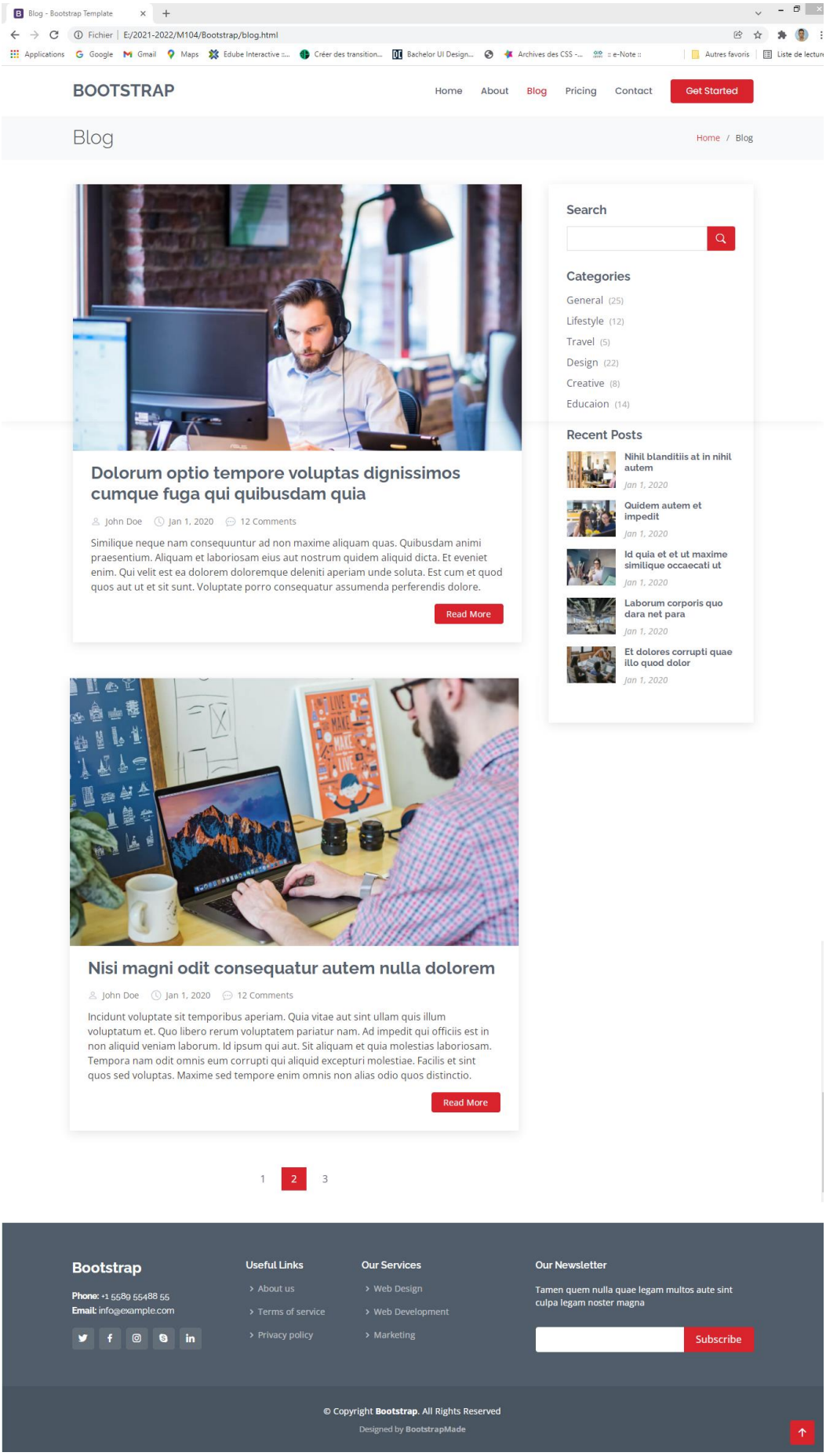
Télécharger les ressources

Interface 1 : « index.html »

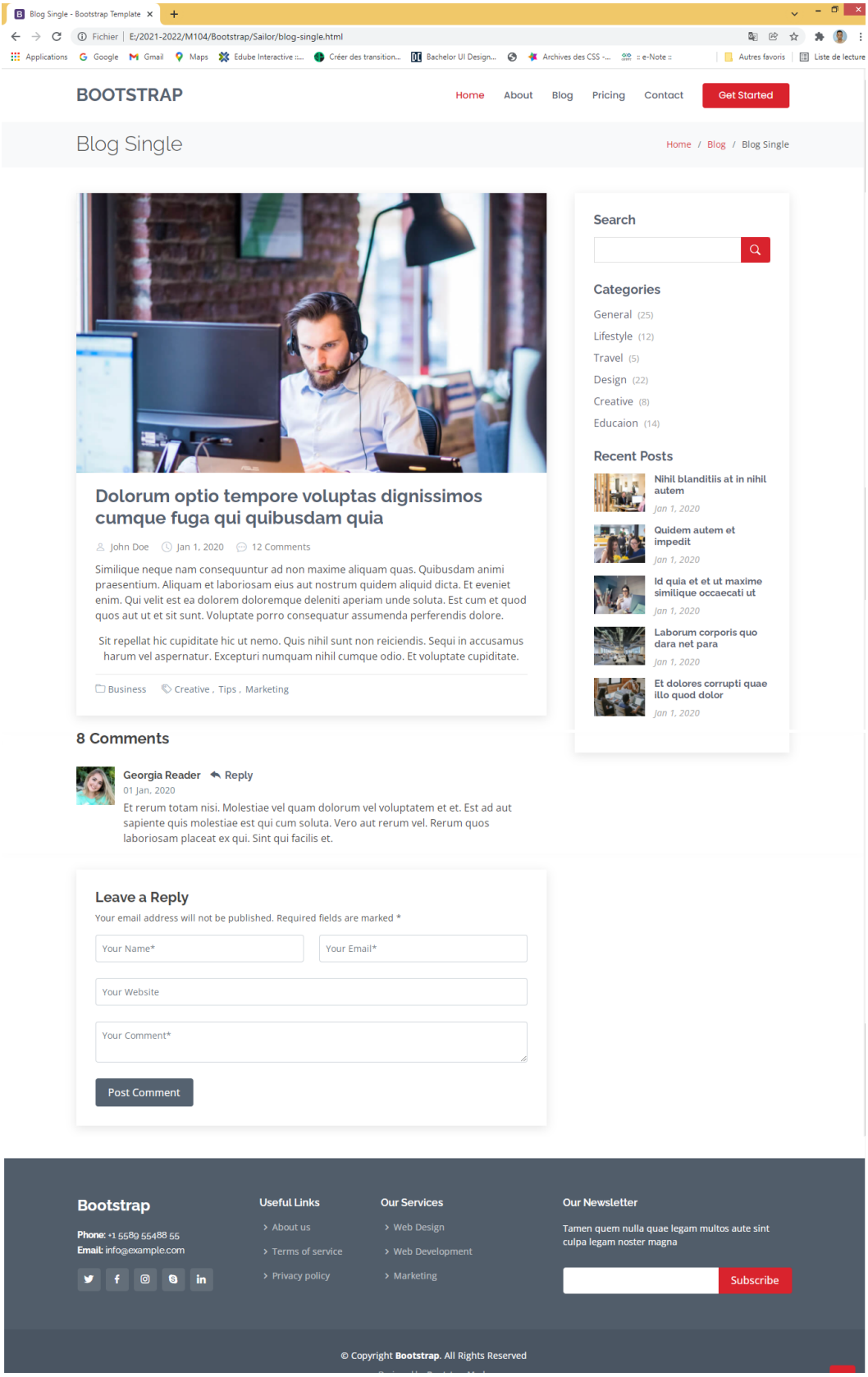
Interface 2 : « about.html »



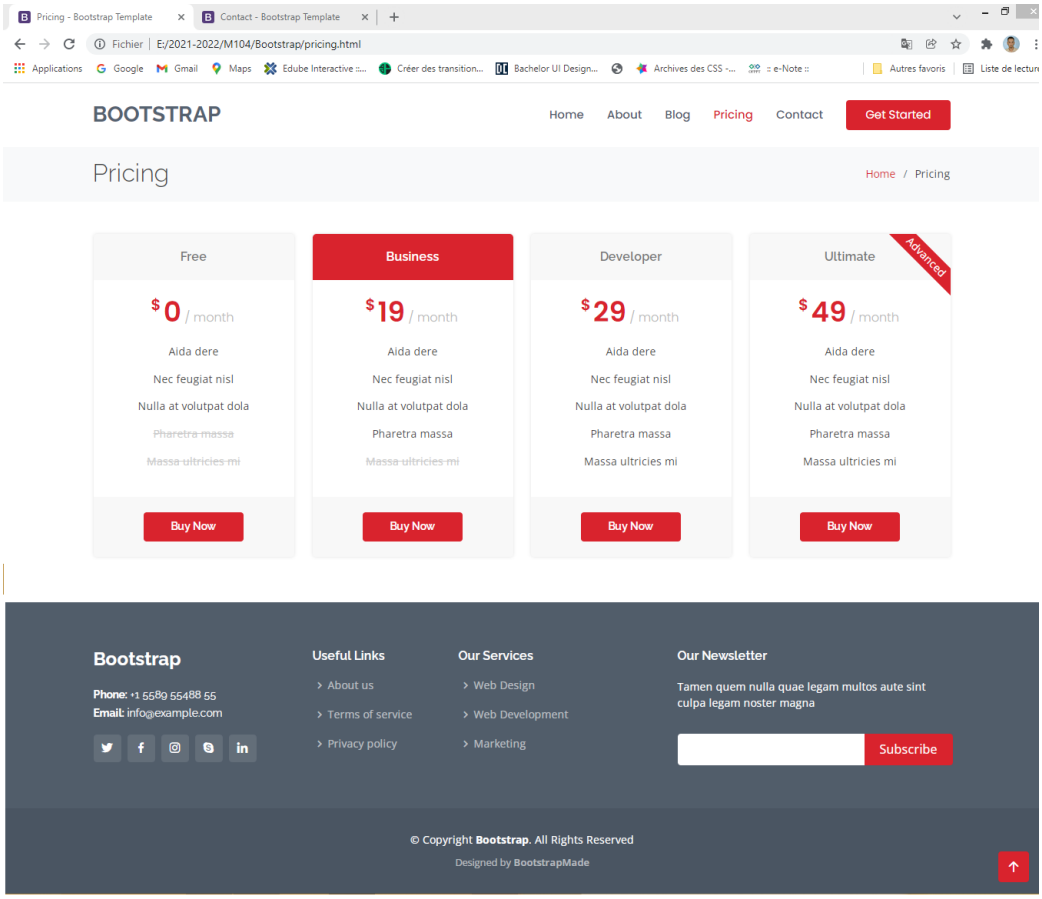
Interface 3 : « blog.html »



Interface 4 : « blog-single.html »



Interface 5 : « princing.html »



Interface 6 : « contact.html »