Tuto installation et utilisation de Bootstrap.

1. Télécharger le zip de bootstrap sur le site : https://getbootstrap.com/docs/4.2/getting-started/download/
2. Décompresser l'archive dans ton dossier Becode et dis-toi que ce dossier deviendra ton meilleur ami pour les mois à venir.
3. Dans tes futurs projets, tu crées ta base incluant le **Sass** comme tu l'as fais jusqu'à présent, sauf que tu inclus dans ton dossier « **./assets/css/ »** le fichier **« Bootstrap.min.css »** contenu dans l'archive du framework.
4. Sur tes pages **.html**, tu n'oublies pas de déclarer le **« bootstrap.min.css »** dans ton header, avant de déclarer ton « **style.css** »

```
    <link rel="stylesheet" type="text/css" media="screen"
href="./assets/css/style.css" />
```

5. Bootstrap va nous permettre de gérer des container. Deux types de container :
    a. Les container-fluid. Container qui va prendre toute la largeur de la page.
    b. Les container : container qui va prendre +- 80% de la largeur de la page, il y aura une marge entre les bords de la page et le contenu. Ce container sera centré sur la page. (voir point 1 & 2 sur la page d'exercice que j'ai construite (https://guyvil1.github.io/decouverte-boostrap/)
    c. Je peux créer un container dans un container-fluid sans aucun soucis (voir point 3)
    d. Attention, on tape un « container-fluid » par section, si on met un second « container-fluid » dans la même section, le second « container-fluid » sera un peu moins grand !!!
6. Avec Boostrap, la largeur de nos pages sont divisées en 12 grilles que nous allons pouvoir utiliser pour positionner notre contenu. Le contenu se réparti :
    a. En lignes : on utilise la classe « row » pour faire comprendre au navigateur qu'on formatte le contenu sur une ligne.
        i. On décide de la largeur de nos contenus avec la classe « col-x » « x » étant une valeur comprise en 1 et 12 définissant ainsi la place occupée sur la largeur de notre page.
        Dans la partie 4 de la page d'exercices, j'ai créé trois contenus et je réparti chaque contenu sur 4 grilles, les trois contenus vont donc se répartir sur tout la largeur de ma section qui fait 12 grilles :

```
7.      <section>
8.          <div class="container-fluid black">
9.              <div class="container yellow">
10.                 <div class="row">
11.                     <article class="col-md-4">
12.                         <h1>Ceci est mon titre</h1>
13.                         <p>ceci est mon texte réparti sur 4 grilles</p>
14.                     </article>
15.                     <article class="col-md-4">
16.                         <h1>Ceci est mon titre</h1>
17.                         <p>ceci est mon texte réparti sur 4 grilles</p>
18.                     </article>
19.                     <article class="col-md-4">
20.                         <h1>Ceci est mon titre</h1>
```

```
                        <p>ceci est mon texte réparti sur 4 grilles.</p>
                    </article>
                </div>
            </div>
        </div>
    </section>
```

i. Je vais ensuite créer une nouvelle ligne.
ii. Cette fois-ci, je vais créer 6 contenus que je vais répartir sur 2 grilles (partie 5) essaie d'imaginer le code html utilisé pour le réaliser, pas besoin de css, puisque c'est bootstrap, notre feuille css précalculée qui s'en charge, il suffit juste d'utiliser la bonne classe.
iii. Et dans la partie 6, je vais garder ces 6 contenus, mais je vais les étendre sur 3 grilles cette fois, observe comment réagit ton navigateur…
Tu remarques que le contenu qui dépasse les 12 grilles de la lignes sont placé en dessous.
iv. Niveau responsive maintenant, nous avons une valeur à utiliser pour que tout s'imbrique parfaitement selon l'écran sur lequel s'affiche notre site.
Les valeurs sont « xs » pour les smartphones (max 768px), « sm (du calme Nikita, ce n'est qu'une classe #balanceTonPorc)» pour les tablettes (>768px) et enfin « md » pour les pc (>992px).Tu remarqueras que dans le code de la partie 4, j'ai utilisé la valeur « md ».
Pour un smartphone on utilisera donc la propriété « col-xs-x »
Pour une tablette « col-md-x »
et pour les pc « col-sm-x »
ps : pour les très grand écran, on utilisera la valeur « lg ».

v.L'avantage de Bootstrap, c'est que nous allons pouvoir, pour chaque contenu, délimiter la place qu'il occupera selon l'écran utilisé, voici un petit exemple

```
        <section>
            <div class="container-fluid black">
                <div class="container red">
                    <div class="row">
                        <article class="col-xs-12 col-sm-6 col-md-1">
                            <h1>Partie 6 du tuto</h1>
                            <p>contenu </p>
                        </article>
                    </div>
                </div>
            </div>
        </section>
```

v. Si tu utilises l'option « inspecter » de ton navigateur sur la partie6 de la page, tu observeras que sur un écran classique, le contenu se gère sur une colonne, sur une tablette, il recouvre la moitié de la page, et sur un smartphone, il s'étends sur toute la largeur.
vi. Tout est géré en une fois, de quoi alléger grandement notre feuille de style.css.

7. Nous avons la possibilité d'influer sur les boutons grâce à notre nouveau meilleur ami, avec la classe « btn », on va pouvoir s'économiser là aussi sur nos lignes de code CSS. Il y a 7 valeurs(tag) que nous pouvons utiliser pour colorer instantanément nos boutons. Pour déclencher l'appel à ces valeurs, on utilises la classe <button class=''btn btn-x''> « x » étant une des valeurs suivantes :

- defaut
- primary
- success
- info
- warning
- danger
- link

regarde la partie 7 de la page pour un aperçu de ces valeurs.
On peut aussi personnaliser la couleur de nos boutons en incluant un peu de code css :

exemple :

```
.btn-perso{
    background-color: aquamarine;
}
```

En appelant la classe <button class=''btn btn-perso''>, mon bouton prendra la couleur que j'ai décidée.

Si on rajoute l'élément « outline » entre le btn et la valeur, on obtient un bouton dont la couleur la valeur s'activera en mode hover

```
                            <article class="col-md-2">
                                <h1>Partie 7 du tuto</h1>
                                <button class=" btn btn-outline-
defaut">Bouton</button>
                            </article>
```

(voir partie 7)

On peut avoir un bouton plus large ou plus petit en rajoutant la classe ''btn-lg'' ou ''btn-sn''

```
                                <article class="col-md-2">
                                    <h1>Partie 8 du tuto</h1>
                                    <button class=" btn btn-outline-primary
btn-lg">Bouton large</button>
                                </article>
```

Voir partie 8

Avec un btn-block ; le bouton va prendre toute la largeur de la page :

```
                                    <article class="col-md-12">
                                        <h1>Partie 9 du tuto</h1>
                                        <button class=" btn btn-outline-
primary btn-block">Bouton large</button>
                                    </article>
```

Voir partie 9

Il y a encore quelques fonctions liées aux boutons que je t'invites à découvrir dans la doc de boostrap : https://getbootstrap.com/docs/4.2/components/buttons/ (ben oui, t'as vraiment cru que je connaissais l'outil ?? :p)

Les infos pratiques pour le défi en équipe.

Avec les éléments introductifs, on a déjà pas mal d'infos pour nous faciliter la vie, on va voir maintenant les petits trucs qui vont nous être utiles pour nous lancer le cœur léger dans notre défi, on va reprendre point par point ce qu'on nous demande et voir comment bootstrap va nous aider.

Attention, je ne vais pas rappeler les élémentaires que j'ai développé plus haut, c'est à toi de t'en rappeler, si y'a quelque chose qui cloche dans ton rendu, demande-toi toujours si tu as appliqué les infos précédentes avant de désespérer et d'éteindre ton flambeau hein !!

« Réalise le site vitrine d'une franchise de restaurants. La franchise est fictive mais le type de nourriture (burger, pizza, asiatique,…) est laissé au choix de l'apprenant.

*Le site doit être **responsive** : au moins pour les petits écrans (xs) et les écrans moyens (md) »*

Bon pour ça, on est paré, ce sera maintenant une formalité !

« Minimum 5 pages accessibles par une barre de navigation (navbar) présente sur toutes les pages et menant aux différentes rubriques : Accueil, Carte, Photos, Restaurants, Contact. »

Ok, une navbar, c'est tout simplement les menus qu'on a tous créé en « s'amusant » (hum hum) sur Turlutu et Active.collab.
Si tu as mon niveau de codage, tu as probablement bien galéré sur ce sujet et tu seras heureux d'apprendre qu'avec Bootstrap, ça se gère les doigts dans le nez.

On va faire d'une pierre deux coups, et on va s'amuser à faire un header et son menu, et on va apprendre quelques petits raccourcis utilise avec VSCode.

Pour un header qui comprends un background image, un titre, un contenu texte et un logo, je vais avoir besoin de créer :

<header> →la balise générale dans lequel mon contenu sera présent et on lui collera une classe ''header'' pour pouvoir le mettre en forme.

 qui va être notre block général dans lequel on va poser tous nos éléments, et on lui donnera la classe ''container-fluid'' pour qu'il s'étende sur toute la surface.

 une autre div qui aura la class ''row'' pour nous permettre d'utiliser notre grille de 12 colonnes.

 une autre div qui va nous permettre de gérer le responsive, et on va le paramétrer pour les écrans xs – sm et md.

 une dernière div a qui on donnera la class ''content''

<h1> pour le titre de notre site, on lui donnera la classe ''title''

<img> pour afficher notre logo, on lui donnera la classe ''logo''

<p> pour le texte d'intro du site à qui on donnera la class ''tgeneral''.

Si tu es une paillasse comme moi, t'es déjà fatigué juste à lire ce qu'il y a à créer, on peut économiser quelques frappes clavier en utilisant cette technique. Voici ce que je t'invite à taper pour créer tous ces élèments automatiquement :

header.header>div.container-fluid>div.row>div.col-xs-4.col-sm-8.col-md-12>div.content>h1.title>img.logo>p.tgeneral>

Quand tu as tapé ça, appuie sur « tab » de ton clavier, et VSCode se charge de te créer tous les composants avec les bonnes classes.

Ça n'a pas marché ? C'est que tu es encore plus paillasse que moi, et que tu as fais un « copié coller » de ce document vers VSCode, recommence et tapes tout toi-même nondijou !!

C'est un petit raccourci intéressant quand tu connais les différents éléments que tu dois créer.
le nom de la balise suivi d'un point et du nom de la classe que tu veux lui coller, tu passes à la balise suivante avec un > et tu enchaînes. Si c'était un « id » que tu voulais mettre, tu remplaces le point pas un dièse.

Avec la ligne de commande que tu viens de tapper, tu dois obtenir ceci :

```
<header class="header">
    <div class="container-fluid">
        <div class="row">
            <div class="col-xs-12 col-sm-8 col-md-4">
                <div class="content">
                    <h1 class="title"></h1>
                        <img src="" alt="" class="logo">
                            <p class="tgeneral"></p>
                </div>
            </div>
        </div>
    </div>
</header>
```

Tu n'as plus maintenant qu'à claquer ton contenu entre les différentes balises créées.

En scss, tu dois maintenant paramétrer le design de ton contenu. Alors les valeurs utilisées ici notamment en termes de row, ne sont pas spécialement adéquates pour faire une entête de site (pour l'alignement surtout), mais tu peux voir dans la partie 10 de la page de test ce que ça donne à ce niveau-ci de la partie avec un peu de paramétrage css.

Pour les balises h1, on a la possibilité de faire appel à des grandeurs prédéfinie dans boostrap, en rajoutant la classe ''display-x'' x étant une valeur comprise entre 1 et 6, 1 étant la taille de police la plus grande.

J'ai utilisé un display-2 pour la partie 9 que tu viens de voir :

```
<h1 class="display-2 title">partie 9 du tuto</h1>
```

Pour centrer mon contenu, j'ai ajouté un padding-top en %tage dans la classe adéquate.

J'ai rajouté aussi un text-shadow pour que le contenu de mon <p> ressorte bien de mon background, sans cela, c'était juste illisible.

On a le header, c'est le moment de se coller au menu

Le menu, … On a tous pu s'y essayer avec turlumachin et active.collab, et si t'es comme moi, t'as surement du bien en chier pour arriver à quelque chose de potable… ou pas.
Avec Bootstrap, on va pouvoir se faciliter la tâche, grâce à la fonction « navbar », la barre à tout faire, tout se fait en quelques clics.

On va inclure les éléments de notre menu dans une liste « ul » de manière classique.
J'ai besoin de 5 élèments dans ma liste, et je vais quand même rajouter le logo de l'entreprise, je n'oublie pas que chaque éléments renvoie vers une page, j'aurai donc besoin de la balise « a » aussi.
Je tape le raccourci :

```
section>div>ul>li*6>a
```

pour obtenir

```
<section>
    <div>
        <ul>
            <li><a href=""></a></li>
            <li><a href=""></a></li>
            <li><a href=""></a></li>
            <li><a href=""></a></li>
            <li><a href=""></a></li>
            <li><a href=""></a></li>
        </ul>
    </div>
</section>
```

Alors n'essaie pas de suite, parce que tu vas voir qu'il y a beaucoup de choses qui vont changer pour programmer notre navbar

Je vais poser un texte « bouton x » dans chaque « li », sauf dans la première ou je vais poser mon logo.

Je vais donner à la div qui contient ma liste la classe ''container'', et je vais l'inclure, dans une div qui aura la class ''container-fluid''

```
            <div class="container-fluid">
                <div class="container">
                    <ul>
                        <li><img src="./img/logo.png" class="logo50px"></li>
                        <li><a href="#">bouton 1</a></li>
                        <li><a href="#">bouton 2</a></li>
                        <li><a href="#">bouton 3</a></li>
                        <li><a href="#">bouton 5</a></li>
                        <li><a href="#">bouton 5</a></li>
                    </ul>
                </div>
            </div>
```

J'étends donc mon premier « div » sur toute la largeur de ma page, et ma liste elle sera étendue sur +-80% de la surface, puisqu'elle est inclue dans la div.container.

Je vais donner à «ul » la class ''nav'', et je vais inclure mon div.container dans une balise « nav » qui aura la classe ''navbar'' et je rajoute des classe de couleurs pour visualiser comment le schmilblick réagit :

```
        <section>
            <article class="col-md-12">
                <h1>Partie 10 du tuto</h1>
            </article>
            <nav class="navbar">
                <div class="container-fluid blue">
                    <div class="container red">
                        <ul class="nav">
                            <li><img src="./img/logo" class="logo50px"></li>
                            <li><a href="#">bouton 1</a></li>
                            <li><a href="#">bouton 2</a></li>
                            <li><a href="#">bouton 3</a></li>
                            <li><a href="#">bouton 5</a></li>
                            <li><a href="#">bouton 5</a></li>
                        </ul>
                    </div>
                </div>
            </nav>
        </section>
```

Tu peux aller voir la partie 10 de la page d'exercices pour voir le résultat….

Qu’est-ce que tu observes ?

Question bonus : pourquoi est-ce que mon container-fluid ne s’étends pas sur toute la surface, laissant un peu de blanc aux extrémités de ma page ?

Si on a compris ça, c’est bon, on a les bases, mais comme les bases ne suffisent jamais pour être expert, va falloir qu’on creuse un peu.

Les options à appliquer dans notre classe ‘’nav’’

1. Mon site doit être responsive, pour ça, on va pouvoir appliquer deux classes à « nav »

    a. la classe « expend » : va permettre à notre menu de s’étendre sur toute la largeur de notre page. On réutilise les tags md, sm, lg ou xl déjà vue plus haut pour définir la réactivité de notre menu sur les différents écrans.
    ex :

```
<nav class="navbar navbar-expand-lg">
```

Le menu s’étends sur les largeur d’écran lg et +.
Sur les tailles inférieures, le menu va se placer en rangée. Pour déclencher la réaction, on ajout à notre balise « ul » qui contient notre liste, la classe ‘’navbar-nav’’.
Je vais modifier mon code du menu précédent :

```
    <nav class="navbar navbar-expand-sm">
        <div class="container red">
            <ul class="nav navbar-nav">
                <li><img src="./img/logo.png" class="logo50px"></li>
                 <li><a href="#">bouton 1</a></li>
                 <li><a href="#">bouton 2</a></li>
                 <li><a href="#">bouton 3</a></li>
                 <li><a href="#">bouton 4</a></li>
                 <li><a href="#">bouton 5</a></li>
            </ul>
        </div>
    </nav>
</div>
```

Si tu regardes sur la partie 11 de la page d’exercices, et que tu joues avec la largeur de ton écran, tu verras que le menu va se placer en rangée quand tu passeras sous la valeur « sm ».

IMPORTANT : a partir de maintenant, tu vas devoir explique à ton navigateur que les balises « li » et « a » font partie de ton menu, pour ça, tu leur ajoutes la classe ‘’nav-item’’ pour li, et ‘’nav-link’’ pour les a.

Avec la classe ‘’justify-content-center’’, je peux centrer ma barre nav sur mon écran, et pour terminer avec les class que je peux ajouter à ma classe ‘’nav’’,
J’ai la possibilité de donner une couleur de fond avec « bg-x » x représentant une couleur (tips, les extensions de couleurs des boutons qu’on a vu plus haut fonctionnent), et j’ai la

possibilité d'expliquer à mon navigateur que j'ai choisi une couleur sombre ou claire et de demander d'adapter automatiquement la couleur de mon texte en conséquence. J'utiliserai les class ''navbar-dark'' ou ''navbar-light'' et mon texte s'ajustera.

Pfff même moi je me saoule, allez un petit bout de code pour clarifier tout ça:

```
<article class="col-md-12">
      <h1>Partie 12 du tuto</h1>
</article>
<div class="container-fluid">
      <nav class="navbar navbar-expand-sm bg-dark justify-content-center
navbar-dark">
      <div class="container">
            <ul class="navbar-nav">
                  <li class="nav-item"><img src="./img/logo.png"
class="logo50px"></li>
      <li class="nav-item"><a class="nav-link" href="#">bouton 1</a></li>
      <li class="nav-item"><a class="nav-link" href="#">bouton 2</a></li>
      <li class="nav-item"><a class="nav-link" href="#">bouton 3</a></li>
      <li class="nav-item"><a class="nav-link" href="#">bouton 4</a></li>
      <li class="nav-item"><a class="nav-link" href="#">bouton 5</a></li>
            </ul>
</div>
</nav>
</div>
```

En gros, mon nav est une navbar qui va prendre toute la largeur de l'écran jusqu'en taille « sm », la couleur de fond sera noire, la barre s'affichera au centre (jusqu'en taille « sm ») et comme mon fond est noir, mon texte sera en clair.

J'ai expliqué que li et a étaient des items pour les uns, des liens pour les autres, allez « go » sur la partie 12 du tuto pour voir tout ça en action.

Oufti on a parlé de la classe extend, mais je t'ai dis qu'il y avait un autre type de classe pour gérer l'étendue de notre menu :

b. le « collapse » : c'est la classe qui va nous permettre d'aller planquer tout ou une partie de notre menu dans un « réservoir » quand la taille de notre écran ne nous permettra plus d'afficher tous les éléments sur la largeur de notre écran.

Quand je dis « tout ou une partie », cela inclus qu'on va devoir scinder notre menu en deux parties. Un petit bout de code pour bien comprendre, accroche toi, on va voir plein de trucs nouveaux:

```
<nav class="navbar navbar-expand-sm navbar-light bg-light">
      <a href="#"><img src="./img/logo.png" class="logo50px" /> </a>
      <a class="navbar-brand" href="#">Accueil</a>
      <button class="navbar-toggler" type="button" data-
toggle="collapse" data-target="#hiddencontent" aria-
controls="hiddencontent" aria-expanded="false" aria-label="Toggle
navigation"> <span class="navbar-toggler-icon"></span>
      </button>
      <div class="collapse navbar-collapse" id="hiddencontent">
            <ul class="navbar-nav mr-auto">
                  <li class="nav-item active">
                        <a class="nav-link" href="#">Carte </a>
                  </li>
                  <li class="nav-item">
                        <a class="nav-link" href="#">Photos</a>
                  </li>
                  <li class="nav-item">
                        <a class="nav-link" href="#">Restaurant</a>
                  </li>
                  <li class="nav-item">
                        <a class="nav-link" href="#">Contact</a>
                  </li>
            </ul>
         </div>
</nav>
```

C’est ici que les romains devinrent chauves, mais je vais essayer de te faire comprendre ce que j’ai appris.

On commence à construire notre menu avec une balise nav à laquelle on donne les attributs que l’on a vu.

Ensuite je place les éléments que je veux garder visible, quoi qu’il arrive, cad, mon logo, et mon lien vers la page d’Accueil.

Ensuite, je vais créer un bouton que je mets dans mes éléments visibles puisqu’il va me permettre de dévoiler mon contenu quand j’appuierai dessus. Je reprends le code de mon bouton et je t’explique :

```
<button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#hiddencontent" aria-controls="hiddencontent" aria-expanded="false"
aria-label="Toggle navigation">
                        <span class="navbar-toggler-icon"></span>
                      </button>
```

on donne à ce bouton la classe ‘’navbar-toggler’’, on défini son type ‘’button’’ (pas benjamin hein Vanessa, du calme). Le data-toggle, c’est le type de bascule qu’il va y avoir quand le menu va se dérouler, on le tape sur ‘’collapse’’ on se demande pas pourquoi, on le fait.
le data-target, c’est juste un ID qu’on donnera à la div dans laquelle on tapera notre contenu à cacher.
Tous les élements contenu dans la div portant l’id ‘’hiddencontent’’ seront rangé dans notre bouton.

Aria-control=''id'' va chercher le contenu se trouvant dans notre ID

Aria-expended=''false'', explique que par défaut, le contenu est caché à l'intérieur de ce bouton

Aria-label : c'est le texte qu'on affichera en passant la souris sur notre bouton.

Tu remarqueras que dans mon bouton, j'ai un span qui a une classe ''navbar-togler-icon'', c'est juste l'attribut pour aller chercher l'icône des 3 barres parrallèles de mon menu.

Allez on enchaîne avec la dernière partie du code :

```
      <div class="collapse navbar-collapse" id="hiddencontent">
            <ul class="navbar-nav mr-auto">
                  <li class="nav-item active">
                        <a class="nav-link" href="#">Carte</a>
                  </li>
                  <li class="nav-item">
                        <a class="nav-link" href="#">Photos</a>
                  </li>
                  <li class="nav-item">
                        <a class="nav-link" href="#">Restaurant</a>
                  </li>
                  <li class="nav-item">
                        <a class="nav-link" href="#">Contact</a>
                  </li>
            </ul>
         </div>
</nav>
```

Dans cette div, je place tout le contenu que je veux envoyer dans mon bouton.
Je donne la classe ''collapse'' et ''navbar-collapse'' à ma div, et je lui donne notre fameux id ''hiddencontent'' pour faire le lien avec mon bouton.

Dans le menu « carte », tu remarqueras que la balise « li » a un attribut ''active''. Tu utilises cet attribut pour désigner la page du site sur lequel tu es. Donc dans ce-cas-ci, mon menu est affiché sur la page « Carte » de mon site.
Si j'avais été sur la page « Contact », j'aurai placé mon attribut ''active'' sur ce « li ».

Tiens j'y pense, si tu reprends le code complet que je t'ai donné en premier tu aurais du remarquer que j'ai placé l'attribut ''brand'' sur le menu « Acceuil ». Soit tu as été trop poli.e pour me le faire remarquer, et dans ce cas, je te rappelle que tu peux le faire, je suis là pour t'aider… quand j'en ai la possibilité, faut pas avoir peur de venir, soit tu ne l'as pas remarqué, et je suis alors colère et déception de ton manque d'inattention :p

Plus sérieusement, l'attribut ''brand'' va juste te permettre de mettre en avant un élément de ton menu, en lui donnant une taille légèrement plus importante que le reste de ta liste.

La classe ''mr-auto'' et sa classe parente ''ml-auto'' ; vont tout simplement te permettre de créer des marge automatiquement pour agencer visiblement ton contenu.

Tu peux maintenant enfin regarder la partie 13 du tuto pour voir le résultat final.

C'est le moment ou tu dois crier à l'arnaque. Je sais il nous reste un élément important à voir pour fixer notre menu, ça va se jouer dans la balise « nav », tu as juste à rajouter la class ''fixed-top'' à la longue liste de classe qu'on a déjà fourée dans notre balise « nav », et le tour est joué.

…

Quoi ? y'a un truc qui marche pas ??? regarde mon code html directement sur le site, y'a peut-être un truc de filou que je ne t'ai pas dit (uniquement pour l'ouverture du menu, tout le reste est explicite)

Les sources interéssantes pour la suite du projet collectif :

https://o7planning.org/fr/11745/bootstrap

tu vas tout y trouver, chaque chose que demande BeCode s'y trouve, faut juste s'accrocher un peu.

Et si vous avez 10 heures devant vous pour vraiment bien caler tout ça :

https://openclassrooms.com/fr/courses/1885491-prenez-en-main-bootstrap