Ouverture vers d'autres frameworks CSS

Table des matières

I. Contexte

Durée : 1 h

Environnement de travail : VSCode

Pré-requis : Bases de CSS

Contexte

Il existe de très nombreux frameworks sur le marché, le plus populaire est probablement Bootstrap, mais d'autres options sont disponibles. Chaque framework propose généralement les mêmes composants, mais chacun possède un visuel qui lui est propre. Certains frameworks ont une approche fondamentalement différente et posent la question de l'architecture des styles dans l'application. Nous verrons plusieurs frameworks et soulignerons leurs similarités autant que leurs différences, pour déterminer plus facilement quel framework est adapté pour un projet.

II. Choisir un framework

Objectifs

- Comprendre les différences de philosophie entre les frameworks
- Trouver le framework qui convient le mieux à nos besoins

Mise en situation

Utilisez la documentation pour choisir un framework.

Un framework CSS est un outil qui permet de gagner du temps dans l'application de styles sur notre projet. Il propose le plus souvent un ensemble de styles préconçus et dogmatiques. Pour pouvoir s'y retrouver, il sera nécessaire de passer du temps sur la documentation.

La clarté et l'accessibilité de celle-ci seront des arguments dans le choix d'un framework par rapport à un autre.

La documentation peut également mettre en évidence les fonctionnalités du framework :

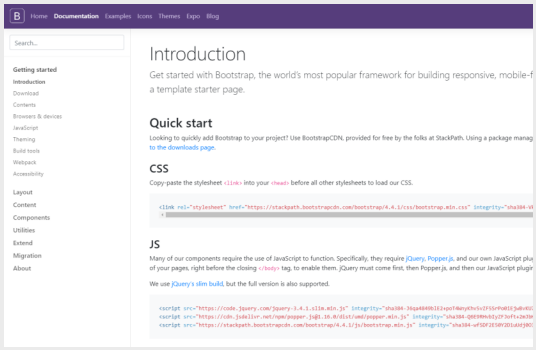
- Possède-t-il un système de grille ? Sur quelle technologie repose cette grille, `float` ou `flexbox` ?
- Quels sont les composants mis à disposition pour construire une interface (Éléments de formulaires, Navbar, Modale...) ?
- L'apparence des composants est-elle cohérente par rapport au résultat attendu ?
- Le style est-il paramétrable par le développeur ?
- Les composants sont-ils pensés pour être adaptatifs sur les petits écrans ?

Toutes ces notions sont à prendre en compte dans le choix du framework par rapport aux besoins du projet.

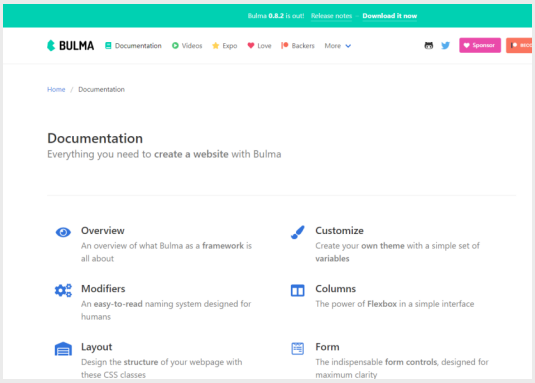
La documentation mettra généralement en évidence le système de grille dans une section `layout` et les composants utilisables dans `components`, avec pour chacun une démonstration de leur utilisation et un exemple visuel.

Exemple Bootstrap

Bootstrap[1] propose un système de positionnement d'éléments via une grille `flexbox` depuis la version 4, accessible dans la section *layout*. Les fonctionnalités sont réparties dans les sections *content* et *components*. le style graphique est orienté flat, assez simple, mais convient généralement pour les applications web.



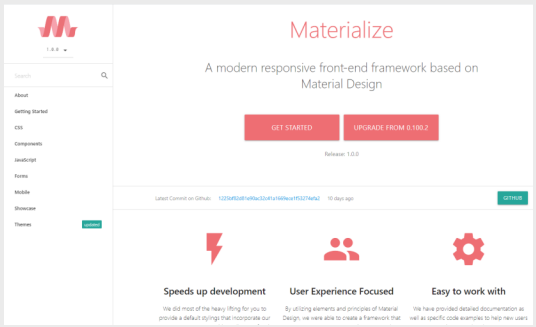
Exemple Bulma



Bulma[2] est un cousin de Bootstrap, son style est assez similaire, lui aussi orienté flat. Il propose comme Bootstrap une customisation via l'édition de variables pour modifier le comportement par défaut.

Exemple Materialize

Materialize[3] est un framework CSS dont le style est inspiré de Material Design de Google. Il est responsive et propose un ensemble de composants d'interface, tout comme Bootstrap et Bulma.



[1] https://getbootstrap.com/
[2] https://bulma.io/
[3] https://materializecss.com/

Fondamental **UI Kit vs Classes utilities**

Les trois frameworks présentés plus tôt (Bootstrap, Bulma et Materialize) ont un point commun : ce sont tous trois des frameworks de type **UI Kit**.

Cette philosophie privilégie le gain de temps à la personnalisation, bien qu'il soit tout de même possible d'agir sur l'apparence via l'édition de variables. Des composants sont mis à disposition du développeur et utilisables en quelques lignes de code par l'utilisation d'une ou plusieurs classes CSS réservées par le framework.

Cette philosophie est opposée à celle de frameworks comme TailwindCSS[1], qui sont spécialisés dans la mise à disposition de classes `utilities` plutôt que de composants entiers. Le développeur doit alors assembler ces classes `utilities` qui ont souvent une responsabilité unitaire (par exemple, appliquer une couleur de fond, appliquer un padding, etc.), comme des briques pour construire le style qu'il souhaite mettre en place.

C'est une approche qui donne plus d'importance à la personnalisation qu'au gain de temps : ici, il n'y a pas de composants tout prêts.

En plus de proposer une forte personnalisation via la combinaison de ses classes `utilities`, Tailwind permet également la personnalisation par l'édition de variables, pour gérer les couleurs ou les familles de polices de caractères par exemple.

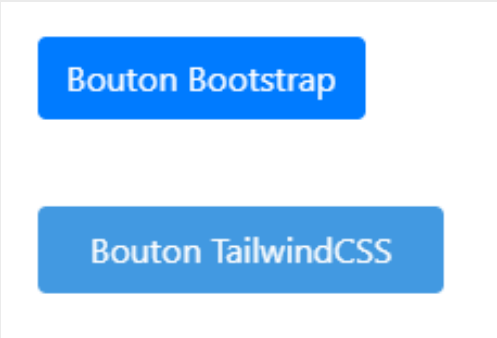
Exemple **La création d'un bouton**

La simple création d'un bouton illustre bien la différence de philosophie entre les deux types de frameworks.

Là où Bootstrap requiert seulement les classes `.btn` qui applique tous les styles du bouton et `.btn-primary` qui donne la couleur bleue, avec Tailwind il faut construire nous-mêmes notre bouton, on lui donne un fond bleu (`bg-blue-500`), un texte blanc (`text-white`), un padding sur les axes x et y (`px-4` et `py-2`), etc.

La version Bootstrap est plus concise et plus rapide à écrire, mais il est plus difficile de modifier drastiquement les styles du bouton par défaut.

```
1 <!--bootstrap-->
2 <button type="button" class="btn btn-primary">My button</button>
3
4 <!--tailwindcss-->
5 <button type="button" class="bg-blue-500 hover:bg-blue-700 text-white px-4 py-2 rounded">My
  other button</button>
6
```



[1] https://tailwindcss.com/

Complément **Documentation de Tailwind**

Concernant sa documentation[1], elle est un peu différente de celle des frameworks UI Kit, dans la mesure où il n'y a pas de composants.

À la place, Tailwind divise ses classes `utilities` en catégories, comme la taille d'un élément ou les styles typographiques. Il y a beaucoup d'éléments et, la plupart du temps, il est plus simple de se servir du champ de recherche pour obtenir les classes liées à un style particulier.

Syntaxe **À retenir**

- Il faut distinguer deux catégories de framework CSS : les framework UI Kit prêts à l'emploi, mais moins personnalisables ; et les frameworks `utility-first` qui se basent sur une combinaison de styles très atomiques, ce qui peut être un peu plus long à prendre en main.
- Pour choisir un framework, il faut se poser la question de la philosophie qu'on souhaite suivre, mais également de l'apparence générale qu'on souhaite donner à la page, ainsi que des composants proposés.
- La documentation de l'outil offre généralement ces informations.

III. Exercice : Appliquez la notion

Question 1

À partir de la documentation, déterminez si Bulma possède un composant de menu. Si oui, quelle est la classe CSS de base pour le définir ?

Question 2

L'application qui est développée demande un carrousel pour présenter des images. Quelles sont les options de framework possibles parmi Bootstrap, Bulma, Materialize et Tailwind ?

IV. Présentation de quelques frameworks

Objectif

- Découvrir les frameworks

Mise en situation

Bien qu'ils se ressemblent beaucoup les différents frameworks CSS présentent tout de même quelques différences. Dans cette partie, nous allons passer en revue quelques frameworks.

Materialize, Foundation, Bulma

Materialize est, avec Bootstrap, un framework CSS populaire. Il a l'avantage d'être à la fois très complet et simple d'utilisation. Il inclut un grand nombre de composants JavaScript permettant de créer de nombreux effets, et opte pour un style graphique très dynamique.

Foundation est considéré comme le principal concurrent à Bootstrap. Il est très similaire à celui-ci. Comme Materialize, il possède un grand nombre de composants JavaScript. Foundation est toutefois un peu plus complet que Bootstrap, mais plus difficile à prendre en main. Il s'adresse aux développeurs front-end plus expérimentés et offre plus de liberté.

[1] https://tailwindcss.com/docs/installation

Bulma, comme Foundation, est très similaire à Bootstrap dans la conception et la mise en œuvre. Toutefois, ce framework est plus léger et ne propose pas de composants JavaScript.

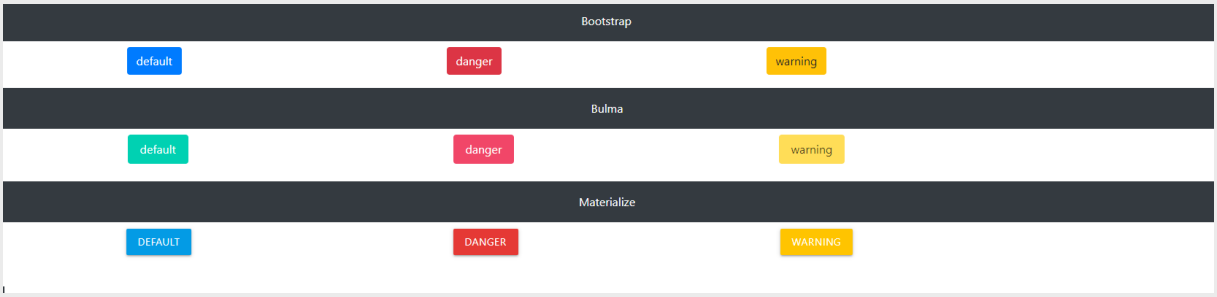
Exemple **Des boutons avec les différents frameworks**

Pour démontrer la similarité de Materialize, Bulma et Bootstrap, voici un exemple de code très simple qui divise la page en 3 colonnes, chaque colonne contenant un bouton.

```
<!--  Materialize -->
<div class="container">
    <div class="row">
        <div class="col s4">
            <a href="#" class="waves-effect lightblue darken-1 btn">default</a>
        </div>
        <div class="col s4">
            <a href="#" class="waves-effect red darken-4 btn">danger</a>
        </div>
        <div class="col s4">
            <a href="#" class="waves-effect amber accent-3 btn">warning</a>
        </div>
    </div>
</div>
```

```
<!--  Bulma -->
<div class="block">
    <div class="container">
        <div class="columns">
            <div class="column">
                <button class="button is-primary">default</button>
            </div>
            <div class="column">
                <button class="button is-danger">danger</button>
            </div>
            <div class="column">
                <button class="button is-warning">warning</button>
            </div>
        </div>
    </div>
</div>
```

```
<!--  Bootstrap -->
<div class="container">
    <div class="row">
        <div class="col">
            <button class="btn btn-primary">default</button>
        </div>
        <div class="col">
            <button class="btn btn-danger">danger</button>
        </div>
        <div class="col">
            <button class="btn btn-warning">warning</button>
        </div>
    </div>
</div>
```



TailwindCSS

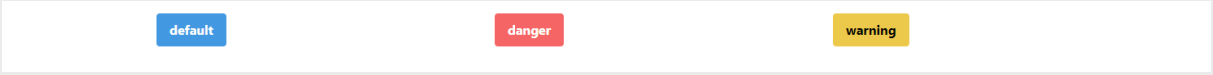
Tailwind se démarque des autres frameworks par son approche `utility-first` : il ne propose aucun composant préconçu, mais un ensemble de classes qui, combinées, vont permettre la création de styles complexes. Cette approche permet de créer des styles plus maintenables et plus évolutifs.

Tailwind propose ainsi une très grande variété de classes permettant de composer un style en appliquant une combinaison de classes CSS dans le HTML.

Exemple

Voici comment reproduire l'exemple précédent avec Tailwind. Chaque classe apporte une partie du design final du bouton :

```
<!--  TailwindCSS -->
<div class="container mx-auto p-4">
    <div class="flex mb-4">
        <div class="w-1/3 object-center">
            <button class="bg-blue-500 hover:bg-blue-700 text-white font-bold py-2 px-4 rounded">default</button>
        </div>
        <div class="w-1/3 object-center">
            <button class="bg-red-500 hover:bg-red-700 text-white font-bold py-2 px-4 rounded">danger</button>
        </div>
        <div class="w-1/3 object-center">
            <button class="bg-yellow-500 hover:bg-yellow-700 text-black font-bold py-2 px-4 rounded">warning</button>
        </div>
    </div>
</div>
```



Syntaxe **À retenir**

- Il n'y a pas de framework qui se distingue totalement. Chaque framework possède des caractéristiques différentes et va être plus ou moins adapté à un projet.
- Ils sont malgré tout relativement similaires structurellement. Quel que soit le niveau de complexité, lorsqu'on a compris comment fonctionne un framework, on s'adapte très vite à un autre.

V. Exercice : Appliquez la notion

Question

En vue de la mise en place d'un framework CSS pour la réalisation d'un site de vente en ligne, nous allons comparer les solutions Bulma et Materialize. Nous allons mettre en place le composant de *breadcrumb* (fil d'Ariane) dans chaque framework.

Nous allons créer deux projets HTML séparés pour ne pas que les deux frameworks CSS interfèrent l'un avec l'autre. Pour plus de simplicité, nous utiliserons la version CDN des frameworks :

- Documentation de Bulma[1] pour l'installation du framework via le CDN **jsDelivr**.
- Documentation de Materialize[2] pour l'installation du framework via le CDN **cdnjs**. Attention, il est nécessaire de lier également la librairie Material Icons pour bénéficier de l'affichage des icônes de chevron.

Les deux breadcrumbs qu'on va mettre en place représenteront l'enchaînement de pages suivant :

Accueil > Produits > Informatique > Claviers & Souris > Claviers mécaniques

Import de Material Icons :

```
<link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
```

VI. Auto-évaluation

A. Exercice final

Exercice

Exercice

Les frameworks CSS qui fournissent des composants clés en main sont souvent appelés...

- ○ Préprocesseurs
- ○ Utility-first
- ○ All-in-one
- ○ UI Kit

Exercice

Un framework dit `utility-first`...

- ○ Fournit des classes `utilities`
- ○ Ne contient que les composants CSS essentiels

Exercice

Parmi ces déclarations, laquelle correspond à un framework proposant des composants clés en main ?

- ○ <button type="button" class="btn btn-primary">My button</button>
- ○ <button type="button" class="bg-blue-500 hover:bg-blue-700 text-white px-4 py-2 rounded">My other button</button>

Exercice

Lequel de ces frameworks est le plus similaire à Bootstrap en termes fonctionnel et graphique ?

[1] https://bulma.io/documentation/overview/start/
[2] https://materializecss.com/getting-started.html

- ○ Bulma
- ○ Materialize
- ○ TailwindCSS

Exercice

On choisira un framework orienté `utility-first`...

- ○ Pour le gain de temps
- ○ Pour les possibilités de personnalisation qu'il offre

Exercice

À quoi sert un système de grille sur un framework CSS ?

- ○ À afficher les bordures des éléments HTML
- ○ À placer facilement les élément sur la page
- ○ À créer un tableau

Exercice

Materialize se base sur la notion de Material Design de Google.

- ○ Vrai
- ○ Faux

Exercice

Quels frameworks proposent nativement des composants couplés à du JavaScript ?

- ☐ Bootstrap
- ☐ Bulma
- ☐ Materialize
- ☐ TailwindCSS

Exercice

À quel type de composant le code ci-dessous pourrait-il être associé ?

```
<div>
   <div class="nav-wrapper">
     <a href="#" class="brand-logo right">Logo</a>
     <ul id="nav-mobile" class="left hide-on-med-and-down">
       <li><a href="sass.html">Sass</a></li>
       <li><a href="badges.html">Components</a></li>
       <li><a href="collapsible.html">JavaScript</a></li>
     </ul>
   </div>
 </div>
```

- ○ Un carrousel
- ○ Un média représenté sous forme de carte
- ○ Une barre de navigation

Exercice

On souhaite vérifier si un framework propose un composant permettant de mettre en place un fil d'Ariane. Comment sont fréquemment nommés ces composants ?

- Card
- Breadcrumb
- Nav
- Dropdown

B. Exercice : Défi

Vous travaillez sur le site d'une agence de voyage. En utilisant le framework TailwindCSS, vous allez devoir intégrer la maquette qui vous a été fournie par le webdesigner.

Pour cet exercice, même si ce n'est pas recommandé, nous utiliserons la version CDN de Tailwind[1] pour accélérer le développement.

Cette version CDN ne permet pas d'utiliser des fonctionnalités avancées ni de personnaliser le framework via des variables, la maquette à intégrer prend en compte cette limitation et a été réalisée avec les paramètres par défaut du framework en tête.

[1] https://tailwindcss.com/docs/installation/#using-tailwind-via-cdn

Idée voyage

DÉCOUVREZ

Oahu - Hawaï

2590 €

Question 1

Commençons par initialiser la structure de notre page :

- Incluez TailwindCSS au sein d'une nouvelle page HTML,
- Indiquez que le corps de la page disposera d'un fond de couleur[1] jaune (200) et que celui-ci prendra toute la hauteur[2] et toute la largeur[3] de l'écran,
- Tout élément contenu dans le corps de la page sera centré[4] et justifié[5], le contenu sera centré grâce à flex, on ajoutera donc également la classe `.flex`.

Question 2

Notre fond de page étant initialisé, attaquons-nous à la structure de la carte. Il s'agira d'un élément placé en position relative[6] qui disposera d'un fond blanc et de larges coins arrondis[7].

Les éléments contenus dans cette carte seront organisés grâce à `flexbox` en colonnes[8], espacées[9] avec le niveau 20.

On appliquera une large ombre[10] et un padding[11] de niveau 10 sur cet élément.

Par défaut, les valeurs de Tailwind ne permettent pas de paramétrer une largeur suffisante (`w-64` est encore trop petit). Pour obtenir un résultat qui correspond à la maquette sur plusieurs tailles d'écran, il sera nécessaire de paramétrer la taille avec ces classes :

```
w-2/3 md:w-3/6 lg:w-2/6 xl:w-2/6
```

Question 3

Notre carte est prête, nous allons pouvoir intégrer son contenu. À ce stade, à vous de chercher comment intégrer les différents éléments, dans la documentation si besoin.

Intégrez un titre de niveau 3 "Idée voyage" au sein de cette carte, puis, à la suite, créez un conteneur pour lequel :

- Celui-ci occupera toute la largeur disponible,
- Les éléments contenus seront organisés grâce à `flexbox` en colonnes et centrés.

Au sein de ce conteneur, ajoutez un titre de niveau 2 contenant "Découvrez" et un titre de niveau 1 contenant "Oahu - Hawaï".

Question 4

Vous aurez remarqué que ces différents titres ne sont pas du tout stylés. Qu'à cela ne tienne, sans utiliser de CSS, mais uniquement avec les classes proposées par Tailwind :

- Qu'importe leur niveau, les titres seront affichés en gras et utiliseront la classe `.font-sans`
- Les titres de niveau 1 et 3 seront colorés en gris (900) et celui de niveau 2 en gris (300)
- Les titres de niveau 1 et 2 seront centrés
- Le titre de niveau 2 sera en majuscules
- Le titre de niveau 1 sera affiché en taille 4 xl, ceux de niveau 2 et 3 en taille 2 xl

---

[11] https://tailwindcss.com/docs/background-color/#app
[11] https://tailwindcss.com/docs/height/#app
[11] https://tailwindcss.com/docs/width/#app
[11] https://tailwindcss.com/docs/align-content/#app
[11] https://tailwindcss.com/docs/justify-content/#app
[11] https://tailwindcss.com/docs/position/#app
[11] https://tailwindcss.com/docs/border-radius/#app
[11] https://tailwindcss.com/docs/flex-direction/#app
[11] https://tailwindcss.com/docs/space/#app
[11] https://tailwindcss.com/docs/box-shadow/#app
[11] https://tailwindcss.com/docs/padding/#app

Question 5

Il est temps de finaliser notre carte en y intégrant l'image et le bouton.

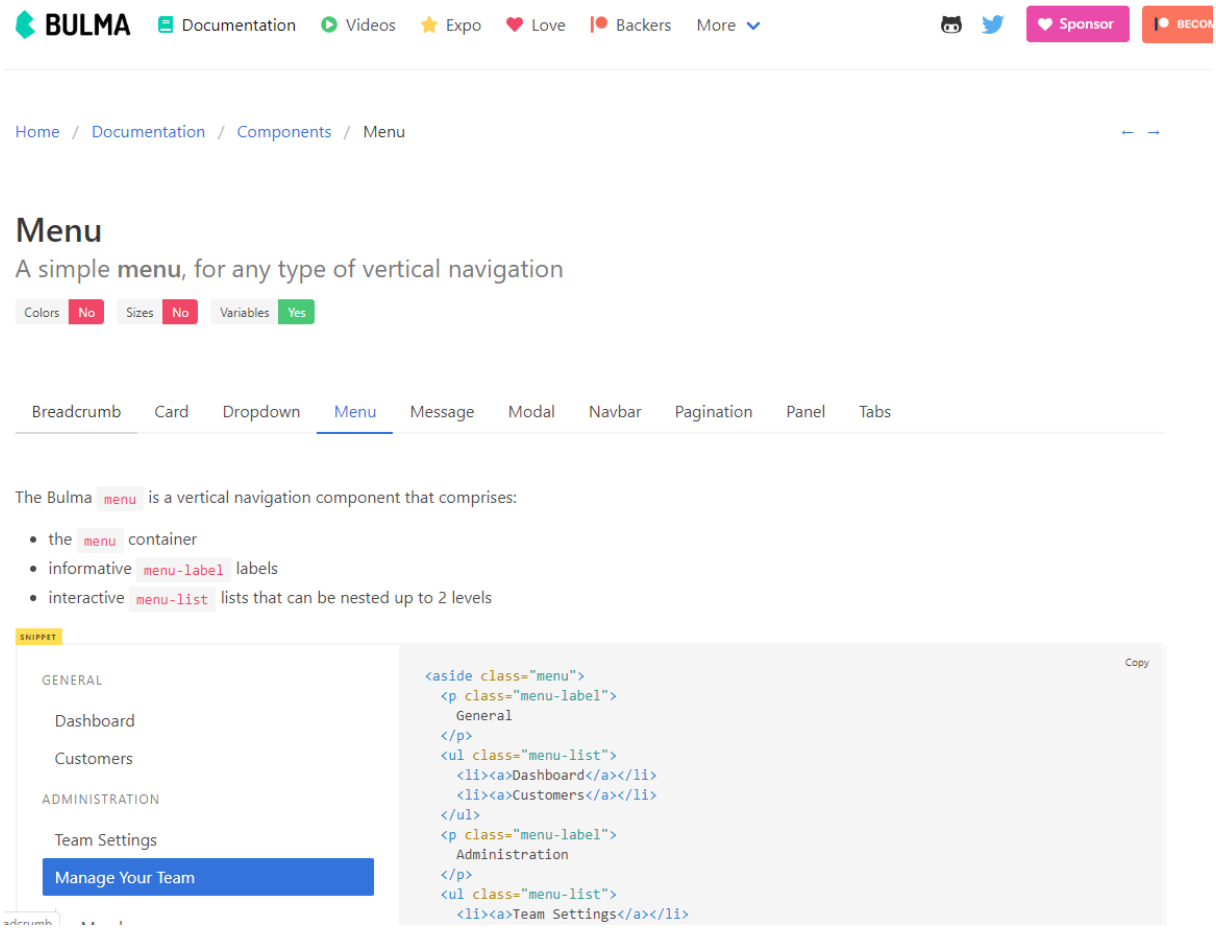
L'image devra disposer de coins arrondis et de marges verticales de niveau 6. Elle sera située en haut du titre de niveau 2.

Quant au bouton, il devra être affiché en bleu (600) avec un texte blanc 2 xl et en gras. On y appliquera un padding vertical de niveau 4 et des coins arrondis. Le texte contenu sera "2590 €".

Solutions des exercices

Exercice p. Solution n°1

D'après la documentation[1], Bulma possède en effet un composant *menu*, la classe à ajouter pour démarrer un menu est `.menu`.

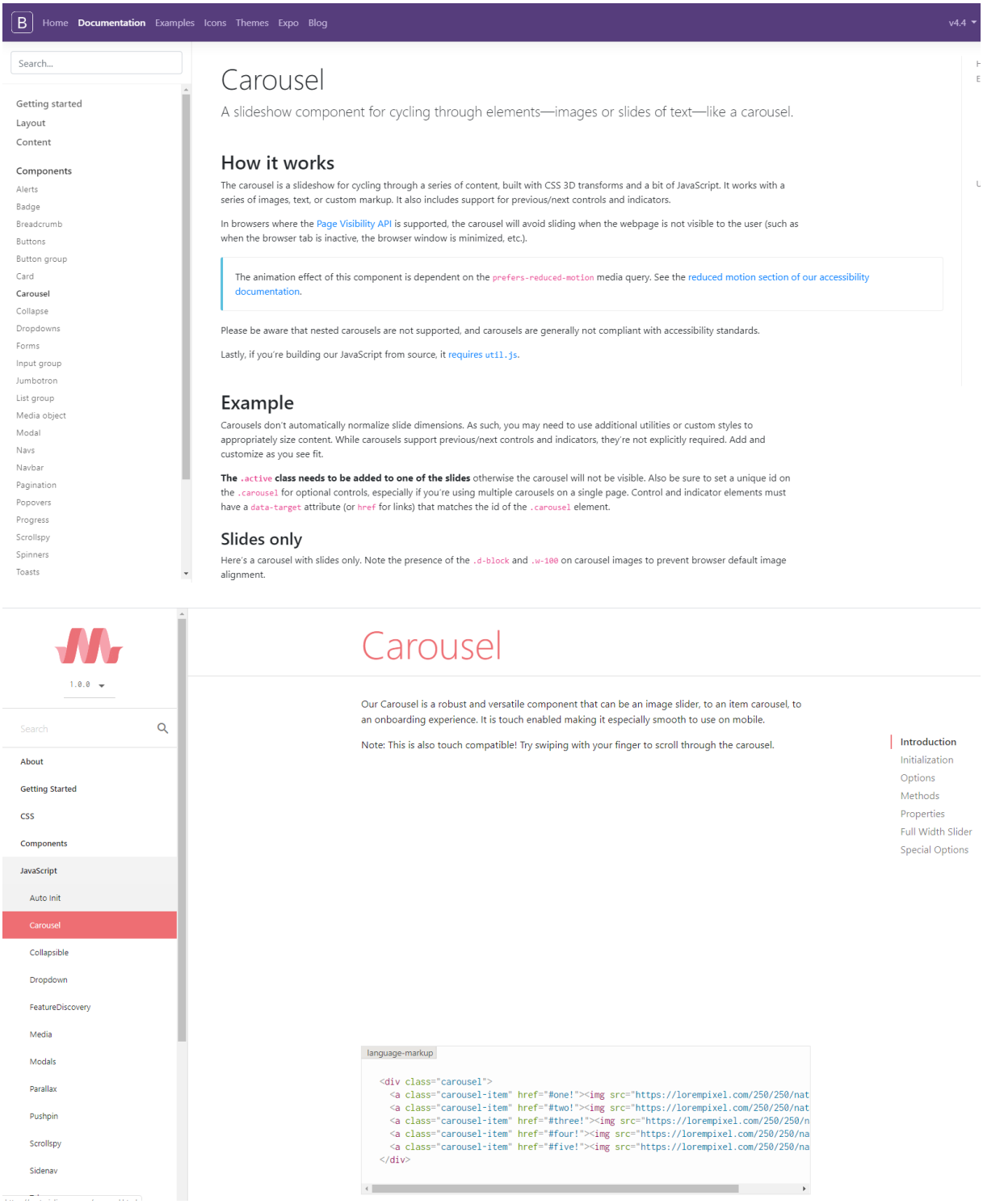


Exercice p. Solution n°2

[1] https://bulma.io/documentation/components/menu/

Parmi les quatre frameworks CSS, seuls Bootstrap[1] et Materialize[2] fournissent un composant carrousel.

Une partie JavaScript est nécessaire pour faire fonctionner le carrousel, Bulma n'est composé que de CSS et ne fournit pas cela nativement. Tailwind de son côté ne fournit aucun composant et n'embarque pas non plus de code JavaScript.



[1] https://getbootstrap.com/docs/4.5/components/carousel/
[2] https://materializecss.com/carousel.html

Exercice p. Solution n°3

Implémentation du breadcrumb avec Materialize :

```
<!DOCTYPE html>
  <html>
    <head>
      <!--Import Google Icon Font-->
      <link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
      <!--Import materialize.css-->
      <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/css/materialize.min.css">

      <!--Let browser know website is optimized for mobile-->
      <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
    </head>

    <body>
      <nav>
        <div class="nav-wrapper">
          <div class="col s12">
            <a href="#!" class="breadcrumb">Accueil</a>
            <a href="#!" class="breadcrumb">Produits</a>
            <a href="#!" class="breadcrumb">Informatique</a>
            <a href="#!" class="breadcrumb">Claviers & Souris</a>
            <a href="#!" class="breadcrumb">Claviers mécaniques</a>
          </div>
        </div>
      </nav>
      <!--JavaScript at end of body for optimized loading-->
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/js/materialize.min.js"></script>
    </body>
  </html>
```

Implémentation du breadcrumb avec Bulma :

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Hello Bulma!</title>
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bulma@0.8.2/css/bulma.min.css">
  </head>
  <body>
    <nav class="breadcrumb has-succeeds-separator" aria-label="breadcrumbs">
      <ul>
        <li><a href="#">Accueil</a></li>
        <li><a href="#">Produits</a></li>
        <li><a href="#">Informatique</a></li>
        <li><a href="#">Claviers & Souris</a></li>
        <li class="is-active"><a href="#" aria-current="page">Claviers mécaniques</a></li>
      </ul>
    </nav>
  </body>
</html>
```

Exercice p. 9 Solution n°4

Exercice

Les frameworks CSS qui fournissent des composants clés en main sont souvent appelés...

- ○ Préprocesseurs
- ○ Utility-first
- ○ All-in-one
- ◉ UI Kit

Exercice

Un framework dit `utility-first`...

- ◉ Fournit des classes `utilities`
- ○ Ne contient que les composants CSS essentiels

Exercice

Parmi ces déclarations, laquelle correspond à un framework proposant des composants clés en main ?

- ◉ <button type="button" class="btn btn-primary">My button</button>
- ○ <button type="button" class="bg-blue-500 hover:bg-blue-700 text-white px-4 py-2 rounded">My other button</button>

Exercice

Lequel de ces frameworks est le plus similaire à Bootstrap en termes fonctionnel et graphique ?

- ◉ Bulma
- ○ Materialize
- ○ TailwindCSS

Exercice

On choisira un framework orienté `utility-first`...

- ○ Pour le gain de temps
- ◉ Pour les possibilités de personnalisation qu'il offre

Exercice

À quoi sert un système de grille sur un framework CSS ?

- ○ À afficher les bordures des éléments HTML
- ◉ À placer facilement les élément sur la page
- ○ À créer un tableau

Exercice

Materialize se base sur la notion de Material Design de Google.

- ◉ Vrai
- ○ Faux

Exercice

Quels frameworks proposent nativement des composants couplés à du JavaScript ?

- [x] Bootstrap
- [ ] Bulma
- [x] Materialize
- [ ] TailwindCSS

Exercice

À quel type de composant le code ci-dessous pourrait-il être associé ?

```
<div>
   <div class="nav-wrapper">
     <a href="#" class="brand-logo right">Logo</a>
     <ul id="nav-mobile" class="left hide-on-med-and-down">
       <li><a href="sass.html">Sass</a></li>
       <li><a href="badges.html">Components</a></li>
       <li><a href="collapsible.html">JavaScript</a></li>
     </ul>
   </div>
 </div>
```

- ○ Un carrousel
- ○ Un média représenté sous forme de carte
- ◉ Une barre de navigation

Exercice

On souhaite vérifier si un framework propose un composant permettant de mettre en place un fil d'Ariane. Comment sont fréquemment nommés ces composants ?

- ○ Card
- ◉ Breadcrumb
- ○ Nav
- ○ Dropdown

Exercice p. Solution n°5

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link href="https://unpkg.com/tailwindcss@^1.0/dist/tailwind.min.css" rel="stylesheet">
    <link rel="stylesheet" href="style.css">
    <title>Voyage</title>
  </head>
  <body class="bg-yellow-200 w-screen h-screen flex items-center justify-center">

  </body>
</html>
```

Exercice p. Solution n°6

Voici notre code à ce stade :

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link href="https://unpkg.com/tailwindcss@^1.0/dist/tailwind.min.css" rel="stylesheet">
    <link rel="stylesheet" href="style.css">
    <title>Voyage</title>
  </head>
  <body class="bg-yellow-200 w-screen h-screen flex items-center justify-center">
     <div class="relative bg-white w-2/3 md:w-3/6 lg:w-2/6 xl:w-2/6 p-10 rounded-lg flex flex-col space-y-20 shadow-xl">
    </div>
  </body>
</html>
```

Exercice p. Solution n°7

Voici votre code à ce stade :

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link href="https://unpkg.com/tailwindcss@^1.0/dist/tailwind.min.css" rel="stylesheet">
    <link rel="stylesheet" href="style.css">
    <title>Voyage</title>
  </head>
  <body class="bg-yellow-200 w-screen h-screen flex items-center justify-center">
    <div class="relative bg-white w-2/3 md:w-3/6 lg:w-2/6 xl:w-2/6 p-10 rounded-lg flex flex-col space-y-20 shadow-xl">
      <h3>Idée voyage</h3>

      <div class="w-full flex flex-col items-center">
        <h2>Découvrez</h2>
        <h1>Oahu - Hawaï</h1>
      </div>
    </div>
  </body>
</html>
```

Exercice p. Solution n°8

Voici votre code à ce stade :

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link href="https://unpkg.com/tailwindcss@^1.0/dist/tailwind.min.css" rel="stylesheet">
    <link rel="stylesheet" href="style.css">
    <title>Voyage</title>
  </head>
  <body class="bg-yellow-200 w-screen h-screen flex items-center justify-center">
```

```
    <div class="relative bg-white w-2/3 md:w-3/6 lg:w-2/6 xl:w-2/6 p-10 rounded-lg flex flex-col space-y-20 shadow-xl">
      <h3 class="text-2xl font-bold text-gray-900">Idée voyage</h3>

      <div class="w-full flex flex-col items-center">
        <h2 class="text-2xl font-bold font-sans uppercase text-gray-300 text-center">Découvrez</h2>
        <h1 class="text-4xl font-bold font-sans text-gray-900 text-center">Oahu - Hawaï</h1>
      </div>
    </div>
  </body>
</html>
```

Exercice p. Solution n°9

Voici votre code une fois finalisé :

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link href="https://unpkg.com/tailwindcss@^1.0/dist/tailwind.min.css" rel="stylesheet">
    <link rel="stylesheet" href="style.css">
    <title>Voyage</title>
  </head>
  <body class="bg-yellow-200 w-screen h-screen flex items-center justify-center">
    <div class="relative bg-white w-2/3 md:w-3/6 lg:w-2/6 xl:w-2/6 p-10 rounded-lg flex flex-col space-y-20 shadow-xl">
      <h3 class="text-2xl font-bold font-sans text-gray-900">Idée voyage</h3>

      <div class="w-full flex flex-col items-center">
        <img
          class="rounded-full my-6"
          src="oahu.jpg"
          alt="Oahu island, Hawaï">
        <h2 class="text-2xl font-bold font-sans uppercase text-gray-300 text-center leading-none">Découvrez</h2>
        <h1 class="text-4xl font-bold font-sans text-gray-900 text-center leading-none">Oahu - Hawaï</h1>
      </div>

      <button
        type="button"
        class="bg-indigo-600 text-white text-2xl font-bold font-sans py-4 rounded-lg">
        2590 €
      </button>
    </div>
  </body>
</html>
```