W1 - Intégration et animations

W-INT-501

Framework CSS

Maquette

2.3

Framework CSS

repository name: frameworkcss
language: HTML, CSS, Javascript

- The totality of your source files, except all useless files (binary, temp files, obj files,...), must be included in your delivery.

Compétences à acquérir

Lors de ce projet vous devrez utiliser et maîtriser les outils suivants:

- SASS/SCSS
- ES6
- Bootstrap

Introduction

Objectif

L'objectif de ce projet est de reproduire une partie des fonctionnalités présentes dans le framework Bootstrap. Vous devez reproduire toute celles décrites dans ce sujet.

Vos fonctionnalités ne doivent pas forcément avoir le même nom que dans Bootstrap, mais leur fonctionnement doit être similaire.

Une fonctionnalité non finie ne sera pas comptée. Ne passez pas à une autre étape avant d'avoir fini toutes celles qui la précèdent, sinon elle ne comptera pas non plus.

Restrictions

vous devrez utiliser SASS/SCSS pour la partie CSS et ES6 pour la partie JavaScript.

L'utilisation de git est obligatoire pour ce projet et sera vérifiée en soutenance. Faites des branches et utilisez des messages de commit explicites.

Projet

I. Grille

Nous allons commencer par réaliser une grille, similaire à celle de Bootstrap. Cela va nous permettre de faire des sites responsives très facilement.

- Classes `.container` et `.container-fluid` qui devra contenir votre grille.
- Classes `.row` pour créer chaque ligne de la grille.
- Classes `.col-*-*` pour remplir chaque `.row`. Ce sont des blocs que vous modifierez en fonction de la résolution.

Assurez-vous que votre grille résiste au responsive.

II. Remplissage

Maintenant que nous avons une grille fonctionnelle, vous allez pouvoir créer des éléments pour la peupler. Chacun de ces éléments pourra être mis n'importe où dans la grille :

- **Boutons**, avec différentes couleurs et tailles via les classes `.btn-`. Vos boutons doivent égalements avoir une apparence différente avec le selector `active` ou s'ils possèdent l'attribut `.disabled`.
- **Images**, qui pourront être responsives avec la classe `.img-responsive`. On pourra modifier les bords via les classes `.img-rounded`, `img-circle` et `.img-thumbnail`. Et comme on trouve que ça manque dans Bootstrap, on veut pouvoir flouter les images avec la classe `.img-blur`.

- **Couleurs**, des classes `.text-muted`, `text-primary`, `.text-success`, `.text-info`, `.text-warning` et `.text-error` . Libre à vous d'en rajouter d'autres si vous trouvez qu'elles manquent. Pour chaque classe de couleur créée, vous devrez créer son équivalent qui modifie la couleur de fond, avec les classes `.bg-*`.
- **Alertes**, des alertes, à la manière de Bootstrap, via des classes `.alert-success`, `alert-info`, `.alert-warning` et `.alter-danger`. Comme pour les couleurs, libre à vous d'en rajouter.

III. Masquer des éléments

On s'est rendu compte que sur mobile, nos pages étaient un peu surchargées et on voudrait que des éléments ne soient visibles que sur un grand écran. On aimerait aussi pouvoir afficher des éléments uniquement sur mobile. Vous allez donc reproduire toute les classes suivantes:

- `.visible-*`, qui diront sur quels devices les éléments seront forcément visibles.
- `.hidden-*`, qui, inversement, seront cachés sur les devices voulus.
- `.visible-print` et `.hidden-print`, qui afficheront ou cacheront des éléments lors de l'impression.

IV. Navbars

On aimerait bien faire une sorte de menu, un peu comme les navbars de Bootstrap, il y a pour cela beaucoup de façons de faire, nous vous laissons choisir la plus adaptée. Cependant nous voulons retrouver toute les fonctionnalités de Bootstrap, y compris les listes déroulantes.

V. Popins

Nous pouvons maintenant faire de jolis sites, responsives, et avec des éléments qui ne s'affichent que sur certains types de devices.

Nous allons à présent mettre à profit les boutons que nous avons créés dans la partie II et allons donc ajouter des popins qui s'afficheront au clic sur un bouton. Cependant, les popins de Bootstrap sont trop verbeux. Nous allons donc les adapter pour qu'ils soient plus simples à utiliser.

On veut donc que les éléments avec une classe `.modal` soient masqués par défaut. Ces éléments, qui correspondent aux contenus de vos popins devront avoir un id pour fonctionner correctement.

Notre bouton aura un attribut `data-target` qui contiendra un sélecteur vers le corps du contenu.
Le popin doit pouvoir s'afficher au clic sur le bouton et se masquer de trois façons différentes

- En appuyant sur échap,
- En cliquant en dehors du popin,
- En cliquant sur un bouton ayant la classe `.popin-dismiss`.

Notre popin doit pouvoir s'afficher et se masquer en JavaScript, comme ceci:

```
Terminal
~/W-INT-501> $("#myModal").modal("show");
$("#myModal").modal("hide");
```

Enfin, on veut pouvoir mettre une grille dans notre popin, n'hésitez pas à rajouter plein d'options afin de rendre votre popin unique.

VI. Onglets

A présent, nous allons améliorer l'organisation de nos pages grâce à des onglets. Ceux-ci seront bien plus proches de Bootstrap que nos popins.
Vous allez créer une classe `.tab-list` qui sera appliquée sur un élément `<ul>`, qui correspondra aux liens vers qui activeront vos onglets. Ces liens seront en réalité des élements `<li>` qui auront un attribut `data-target` qui contiendra un sélecteur vers le corps de l'onglet selectionné. Cet élément sera généralement un élément `<div>`, qui aura la classe `.tab-pane` .
Tous les blocs d'un même ensemble d'onglets seront regroupés dans un même `<div>` qui aura la classe `.tab-content`. L'onglet actif, ainsi que le lien qui y est associé auront la classe `.active`.

Contrairement à Bootstrap, il doit être possible de mettre un ensemble d'onglets dans un onglet déjà existant.

VII. Info-bulles

Certains de nos boutons ont des noms qui ne sont pas suffisamment explicites. Nous allons donc ajouter des info-bulles sur ces boutons. Un bouton avec une classe `.tooltip` verra le contenu de son attribut `title` dans une infobulle au survol. Cette infobulle se trouvera soit à droite, soit à gauche, soit au-dessus, soit en dessous du bouton. Sa position sera définie par l'attribut `data-placement`, qui pourra valoir `right`, `left`, `top`, `bottom`. Si le bouton n'a pas d'attribut `title`, rien ne se passe.

Après avoir réalisé ces infos bulles, on s'est rendu compte qu'ils étaient très pratiques. Tellement qu'on voudrait pouvoir les mettre sur n'importe quel élément. N'hésitez pas à implémenter le plus d'options possibles afin de les rendre plus jolis via des attributs `data-*` (comme le fait Bootstrap).

Bonus

Votre Framework est maintenant complet et fonctionnel ! N'hésitez pas à rajouter des fonctionnalités afin de le rendre encore plus puissant. Vous pouvez pour cela vous inspirer d'autres fonctionnalités présentes dans Bootstrap, ou en inventer d'autres.