INFORMATIQUE
Sciences
Université Paris Cité

Programmation Web
TP n° 4 : Bootstrap

Le but de ce TP est d'apprendre à utiliser Bootstrap, une bibliothèque CSS et jQuery permettant de simplifier et d'améliorer la mise en page des sites. Nous utilisons ici la version 4 de Bootstrap.

1. **Media queries en CSS.** La version actuelle de CSS permet déjà de choisir un style parmi plusieurs en fonction du média utilisé pour visualiser une page web – *c.f.* ce lien pour une introduction à la notion de "media query", et celui-ci pour des exemples. Dans l'élément `style` d'une page web, le code suivant permet par exemple de choisir un style selon la taille de l'écran :

```
@media screen and (max-width: 600px) {
  <!-- style pour les écrans de largeur au plus 600px   -->
}
```

```
@media screen and (min-width: 600px) {
  <!-- style pour les écrans de largeur au moins 600px  -->
  <!-- et au plus 1000px (a cause de la règle suivante) -->
}
```

```
@media screen and (min-width: 1000px) {
  <!-- style pour les écrans de largeur au moins 1000px -->
}
```

   Tester ce code dans une vraie page web en choisissant, selon la largeur de la fenêtre du navigateur, une couleur de texte différente pour chaque intervalle de largeur. Redimensionnez la fenêtre dynamiquement et observez le résultat.

2. **Un blog avec Bootstrap.** La classe `container-fluid` de Bootstrap permet disposer les éléments d'une page web sur une grille dont chaque ligne (`row`) contient 12 colonnes s'étalant sur toute la largeur de l'écran. Il est possible de spécifier explicitement le nombre de colonnes sur lequel s'étale chaque élément d'une ligne :

```
<div class="container-fluid">
  <div class="row">
    <div class="col-sm-2">
      sur deux colonnes
    </div>
    <div class="col-sm-3 offset-sm-2">
      sur trois colonnes, décalé de deux colonnes
   </div>
   <div class="col-sm-5">
      sur les cinq colonnes restantes
   </div>
</div>
div>
```

   Les trois éléments `div` internes seront disposés horizontalement sur des écrans au moins de la largeur d'une tablette ("`sm`" pour "small devices"). Sur des écrans plus petits (*e.g.*, un smartphone), ils seront disposés verticalement.

   Lorsque les éléments sont affichés verticalement sur un écran trop petit, l'ordre d'affichage des éléments internes d'une ligne est celui de leur écriture dans le code HTML. On peut modifier l'ordre d'affichage horizontal des éléments, en spécifiant pour chaque élément un priorité (`order-sm-...`). Peu importe le choix exact des priorités (`1`, `2`,...) pourvu qu'elles forment une suite croissante :

```html
<div class="container-fluid">
  <div class="row">
    <!-- affichés verticalement dans l'ordre d'écriture -->
    <div class="col-sm-2 order-sm-2">
      <!-- affiché horizontalement après le suivant -->
      sur deux colonnes
    </div>
    <div class="col-sm-3 order-sm-1">
      <!-- affiché horizontalement avant le précédent -->
      sur trois colonnes
    </div>
    <div class="col-sm-7">
      sur les sept colonnes restantes
    </div>
  </div>
</div>
```

a. Créer une page web de type "blog", avec un titre `jumbotron`. Sous le titre et sur un écran de taille au moins `sm`, la page sera affichée de la manière suivante :

- A gauche on trouvera, sur les trois-quarts de la largeur de l'écran, une suite verticale d'articles rerésentant les posts du blog.
- A droite, sur le quart de la largeur restante, on trouvera un menu vertical de navigation de classe `nav` prenant le quart de la page contenant des liens internes vers les posts (`#`).

Sur les écrans taille inférieure à `sm`, la barre de navigation doit venir se placer verticalement *au-dessus* des articles.

b. En utilisant javascript/jQuery, alimentez votre blog avec huit posts factices de titres et de contenus générés aléatoirement. Chaque nouvel article ajouté à la page recevra un identifiant (`"post0"`, `"post1"`, ...). La barre de navigation sera mise à jour à chaque nouvelle entrée, en lui ajoutant un lien interne vers cette entrée (`<a class="nav-link" href="#post...">`).

c. Écrire un fichier CSS auxiliaire pour personnaliser l'apparence de votre blog (couleurs, etc.).

3. Trombinoscope

Le fichier `images.zip` sur Moodle contient de nombreuses images, de taille variable. Le but de cet exercice est d'écrire avec Bootstrap une page liée à un fichier CSS et un fichier javascript dont la forme finale est décrite ci-desssous.

a. Au sommet de la page se trouve une barre de navigation de classe `navbar`, de position fixe (utilisez `sticky-top` pour éviter que la barre ne chevauche le reste de la page). Lorsque la taille de lécran est trop petite, cette barre est remplacée par un bouton permettant d'afficher les liens verticalement sous ce bouton (*c.f.* le premier exemple de la documentation de `navbar` – observez l'évolution de l'affichage quand la largeur de la fenêtre est réduite).

b. Sous la barre se trouve le titre `jumbotron` de la page. Sous le titre, se trouvent : à gauche, sur un sixième de la page, un texte arbitraire ; à droite, sur les cinq sixièmes de la page, une mosaïque de photos. Cette mosaïque affiche 6 images par ligne sur écran large (`lg`), 4 sur un écran médium (`md`), 3 sur un écran petit (`sm`) et 2 sur écran extra-petit (`xs`). Dans le CSS, on pourra utiliser `max-width=100%` pour autoriser la taille des images à varier sans dépasser leur taille d'origine.

c. La barre de navigation permet l'accès direct aux images de rangs 0, 10, 20, 30, ..., 90. Ajouter l'option `Scrollspy` permettant de suivre dans la barre de navigation la zone consultée (`<body data-spy="scroll" data-target=".navbar">`).

Dans le fichier javascript, remplissez le trombinoscope avec les images de l'archive, ajoutez à la barre ses liens, et vérifiez le résultat.