Architectures CSS: BEM et OOCSS

Lisa Michallon et Estefania Vila

Pourquoi suivre une méthode d'organisation du CSS ?

- ➔ **meilleure compréhension du code** quand on revient dessus après plusieurs mois sans y toucher
- ➔ **modulation du site beaucoup plus facile** grâce à la façon de nommer et d'utiliser les classes

Qu’est-ce que OOCSS ?

Object-Oriented Cascading Style Sheets

➔ un objet c’est un élément HTML ou quelque chose qui lui est associé

Une méthodologie basée sur deux principes :

- ➔ la séparation de la structure et de l'apparence
- ➔ la séparation du conteneur et du contenu

Donc on part du **design** pour repérer des **répétition visuelles** ou patterns que l'on va **nommer**. Cela permettra de créer des classes qui sont **réutilisables** et que l'on peut mettre en relation.

= Bootstrap

Séparer la **structure** et l'**apparence** :

```css
.btn {
    border-radius: 15px;
    max-width: 100px;
    font:Bold 13px Arial;
    padding:6px 0 6px 0;
    background-color: #767676;
}

.btn-alert {
    background-color: red;
}

.btn-primary {
    background-color: deepskyblue;
}
```

On évite de dupliquer des styles similaires grâce à :

Une classe générale `.btn` contenant ce qui est commun à tous les boutons

Des classes plus spécifiques `.btn-alert` et `.btn-primary` pour modifier leur apparence

Séparer le **conteneur** du **contenu** :

Cela signifie qu'**un objet doit avoir la même apparence quelle que soit sa position.**

Donc il est préférable d'**éviter les cascades** comme `.links-box .title` qui couple l'apparence du contenu `.title` au conteneur `.links-box`.

Privilégier l'utilisation d'une classe `.box-title` qui sera plus réutilisable.

Les avantages de cette méthodologie :

- ➔ possibilité de modifier ou styliser un site **sans avoir à toucher le fichier CSS** en réutilisant des classes déjà existantes
- ➔ fichier CSS **plus léger** donc un site plus performant

<Qu’est-ce que BEM?>

BEM est une façon de nommer les classes en CSS, une méthodologie en d'autres termes voire une convention de nommage.

BEM est un sigle pour Blocks - Elements - Modifiers, en français Blocs - Éléments - Modificateurs.

Trois entités qui vont permettre d’organiser et découper tous les composants d'une page web.

<Le concept qui se cache derrière ses 3 lettres>

BEM envisage la composition d'une page web en deux types de «composants» :

Le bloc est un composant «parent» contenant un ou plusieurs éléments. Le bloc peut être indépendant, et lorsque pris hors contexte d'une page spécifique, garde du sens.

par exemple : un menu, un footer, une sidebar.

L'élément est un composant appartenant à un bloc. Il faut considérer un élément comme l'enfant d'un bloc.

par exemple : le titre d'un bloc, une page d'un menu.

Enfin BEM introduit le concept de modificateur.

Le modificateur introduit une notion de comportement. Il se modifie en fonction du contexte de la page ou d'une action de l'utilisateur.

Le modificateur peut aussi bien être appliqué à un bloc qu'à un élément.

> par exemple : un fond de couleur différente pour une page spécifique, un élément rendu visible (ou caché) après un clic de l'utilisateur.

En BEM, toutes les classes CSS sans exception commencent nécessairement par le nom du bloc. Par exemple : `.menu{ }`

Si on souhaite ajouter dans notre bloc .menu un élément item, on va le nommer .menu_item. On sépare le bloc de l'élément par deux underscores:

`.menu__item{ }`

Le code HTML correspondrait à cela :

```
<div class="menu">
    <div class="menu_item">Page1</div>
</div>
```

Pour le modificateur, on sépare le bloc ou l'élément par deux tirets:

```
.menu--modifier { }
```

```
.menu__item--modifier { }
```

Le code HTML correspondrait à cela :

```
<div class=“menu”>
    <div class=“menu__item”>Page1</div>
    <div class=“menu__item--modifier”>Page2</div>
</div>
```

<Les trois principes>

Trois principes s'articulent ici :

1. Commencer le nom de la classe par le nom du Block parent
2. Précéder directement le nom d'un Élément par deux underscores
3. Précéder directement le nom d'un Modifier par deux tirets

<BEM réduit les risques de conflits de nommage>

Un des problèmes que l'on rencontre rapidement en rédigeant du CSS, c'est de se retrouver avec des noms de classes similaires. La méthodologie BEM impose un nommage des classes qui élimine quasiment les risques de conflits CSS.

Sans BEM

```
/* titre de la page nommée simplement .titre /*
.titre {
      font-size: 15px;
      color: red;
}

/* titre d'un article dans la page également appelé .titre voici
donc un conflit, deux classes qui se superposent ! */

.titre {
      font-size: 30px;
      color: blue;
}
```

Avec BEM

```
/* titre de la page nommée .page__titre /*
.page__titre {
      font-size: 15px;
      color: red;
}

/* titre d'un article dans la page appelé .article__titre . On a éliminé le conflit en nommant suivant une convention nos éléments. */
.article__titre {
      font-size: 30px;
      color: blue;
}
```

BEM impose une méthodologie qui rend la rédaction de CSS plus rigide mais aussi plus sûre. Il est facile d'utiliser les différentes classes produites dans toutes les pages sans se soucier des conflits.

On envisage alors le code comme un assemblage de briques réutilisables, modulaires et extensibles. Il est même possible de réutiliser certaines briques, d'un projet à un autre, sans rencontrer de conflits.

Documentation:

https://en.bem.info/methodology/