HowDoWeDoCSS

Conseils et recommandations dans la rédaction ou refactorisation de code CSS.

Commentaires

La compilation sass du code source n'effectue pas de compression du code CSS compilé. Il est donc déconseillé d'utiliser les blocs commentaires CSS /* commentaire */.
À la place, utiliser des commentaires Sass // commentaire.

Plusieurs fichiers

Évitez les fichiers CSS à rallonge.
Il est judicieux de scinder des fragments de code distincts dans leurs propres fichiers, qui sont concaténés lors d'une étape de construction.
Isoler chaque composant/bloc dans un fichier propre permet de les faire évoluer distinctement des autres sans craindre les conflits ou d'éventuelles régressions sur les autres.

Tous les styles d'un composant seront donc de préférence définis dans son fichier.
Il faut éviter de styler plusieurs éléments provenant de plusieurs composants / blocs au sein d'une même déclaration.

Ne pas faire :

```
.composant-1 h2,
.bloc-2 h2,
.composant-3 h3 {
    text-transform: uppercase
}
```

Sélecteurs CSS

Liste des bonnes pratiques pour sélectionner et nommer un élément en CSS.

Éviter les sélecteurs d'id

Les sélecteurs par `#id` ont une spécificité trop lourde dans le calcul de résolution de la cascade, empêchant ainsi l'écriture efficace d'un code modulaire.

Éviter les sélecteurs de tag

Utiliser le sélecteur de tag doit être considéré comme une **très mauvaise pratique**.
Cela revient à définir le style d'un bloc en rapport à sa sémantique et rend impossible la mise à jour de l'HTML.

Exemple et conséquence de cette mauvaise pratique. **Ne pas faire :**

```
div.composant {
    background-color: gray;
}
```

Quand, dans le fichier *template* du composant, il s'avérera judicieux de remplacer la balise `div` non sémantique par une balise `section`, le style ne s'appliquera plus. Ce sélecteur empêche la mise à jour du *template*.

No DOM

Cette règle suit logiquement les recommandations précédentes.
Évitez les sélecteurs descendants par tag ou #id, **Ne pas faire :**

```
div.composant > ul > li {
    padding-left: 1em;
}
```

Dans cet exemple, les modifications suivantes du *template* HTML généreront des régressions de styles :

• La modification des balises div, ul ou li par des choix sémantiques différents.
• L'ajout des balises intermédiaires, comme ajouter une balise `nav` entre `div` et `ul`.

Ne pas ajouter d'informations relatives au DOM dans vos sélecteurs.

First class

Il est donc préférable de cibler les éléments uniquement par leur class.
Pour permettre une sélection sécurisée d'un élément par une class unique, une convention de nommage excluant les conflits est nécessaire.

Convention `block--element`

Le modèle `.app-block--element` est préconisé pour l'écriture de tout nouveau composant ou bloc.

- `app` : préfixe optionnel mais utile pour indiquer l'application générant le bloc.
- `block`: identifiant unique de la racine du bloc ou composant.
- `element`: élément/enfant d'un bloc

block

Règles à suivre pour la définition du nom de class du bloc.

- La class d'un bloc doit être unique.
- La class d'un bloc peut se composer du préfixe de l'application qui le construit
- Si préfixe `app`, il est séparé de l'identifiant par un trait d'union (abusivement appelé tiret du 6): `.app-block`.
- L'identifiant peut être composé de plusieurs parties séparées par 1 trait d'union: `.app-composant-extra`, `latest-updated-pages-cell`. Pour des raisons de lisibilité, l'utilisation de traits d'unions est recommandé:
  - `latestupdatedpagescell` NON
  - `latest-updated-pages-cell` OUI
- L'utilisation du CamelCase n'est autorisé **que dans le cas** où l'identifiant du composant nécessiterait plus d'un trait d'union: `.wcs-cell-currentDrafts`

element

- La class d'un élément est composé de l'identifiant de son bloc en préfixe, suivi de l'identifiant de l'élément, séparé par 2 tirets `.block--element`
- L'identifiant de l'élément peut-être composé d'un trait d'union: `.bloc--sub-title`, préférable à l'utilisation du camelCase.
- On ne peut pas coller 2 éléments au nom d'un bloc pour définir un sous element: `.nav--sub-nav--link` n'est pas autorisé. Dans ce cas, définir un nouveau bloc `.sub-nav`.

Dictionnaire d'identifiants d'éléments

- --wrapper
- --list
- --item
- --title
- --body
- --link
- --metas
- --img

Élément ou sous-bloc ?

Lorsqu'un composant présente une arborescence profonde, comme c'est souvent le cas avec les listes, il est préférable d'imbriquer les blocs plutôt que de définir des éléments trop complexes.

```
.main-nav {}
.main-nav--list {}
.main-nav--item {}
.main-nav--link {}

.sub-main-nav {}
.sub-main-nav--list {}
.sub-main-nav--item {}
```

```
.sub-main-nav--link {}
```

Exemple

Exemple d'un fichier scss `wcs/_steps.scss` avec le composant 'étapes d'une demande'.

Il est composé de 2 blocs

- `.wcs-steps` identifiant racine du composant
- `.wcs-step` identifiant du composant

```
.wcs-steps              { /* styles du bloc */ }
.wcs-steps--list        { /* styles pour la liste (ol) */ }

.wcs-step               { /* styles pour l'item (li) */ }
.wcs-step--marker       { /* styles pour le marker (abbr) */ }
.wcs-step--marker-nb    { /* styles pour le num du marker (span) */ }
.wcs-step--marker-total { /* styles pour le num total du marker (span) */ }
.wcs-step--label        { /* styles pour le label (p) */ }
```

Avec le *nesting SASS*, il est possible de ne pas répéter l'identifiant du bloc dans les class des éléments tout en représentant graphiquement l'arborescence DOM des balises du *template*.

```
.wcs-step {
    /* styles pour l'item (li) */
    &--marker {
        /* styles pour le marker (abbr) */
        &-nb {
            /* styles pour le num du marker (span) */
        }
        &-total {
            /* styles pour le num total du marker (span) */
        }
    }
    &--label {
        /* styles pour le label (p) */
    }
}
```