Cours d'intégration

3W Academy

CSS 9. Conventions et bonnes pratiques

9. Conventions et bonnes pratiques

9.1. Conventions de codage

Un ensemble de règles et de principes ont été publiés par Nicolas Gallagher, auteur de normalyse.css: https://github.com/necolas/idiomatic-css

Certains principes sont commun à tous les langages, à savoir:

- Écrire un code lisible et compréhensible par vous et par d'autres.
- Choisir une nomenclature et s'y tenir. Garder son code cohérent.
- Utiliser si possible une convention largement utilisées par d'autres.
- Choisir un mode d'indentation (tabulation ou espaces) et s'y tenir.
- Commenter le code avec si possible, des sections et des blocs de commentaires pour décrire votre code sur plusieurs lignes. Utilisez par exemple le style de Doxygen.

Exemple de commentaires

```
/* ================================================
   Section comment block
   ================================================ */

/* Sub-section comment block
   ================================================ */

/**
 * Short description using Doxygen-style comment format
 *
 * The long description is ideal for more detailed explanations and
 * documentation.
 * It can include example HTML, URLs, or any other information
 * that is deemed necessary or useful.
 *
 * @tag This is a tag named 'tag'
 *
 * TODO: This is a todo statement that describes a task to be completed
 *   at a later date. It wraps after 80 characters and following lines are
 *   indented by 2 spaces.
 */

/* Basic comment */
```

9.2. Déclarations CSS

Quelques règles simples:

- Déclarez un sélecteur par ligne avec une virgule en fin de ligne si plusieurs sélecteurs.
- Préférez une déclaration par ligne à l'intérieur d'un bloc de déclarations sans oublier le point-virgule à la fin de chaque ligne.
- Indentez toutes les déclarations d'un même bloc avec un niveau d'indentation.
- laissez un espace après chaque virgule dans les listes de valeurs ou dans les paramètres de fonctions.
- Préférez les doubles guillemets.
- Fermez chaque bloc de déclaration au même niveau que la première lettre du sélecteur de celle-ci.

```
.selector-1,
.selector-2,
.selector-3[type="text"] {
    -webkit-box-sizing: border-box;
    -moz-box-sizing: border-box;
    box-sizing: border-box;
    display: block;
    font-family: helvetica, arial, sans-serif;
    color: #333;
    background: #fff;
    background: linear-gradient(#fff, rgba(0, 0, 0, 0.8));
}

.selector-a,
.selector-b {
    padding: 10px;
}
```

9.3. Ordre des déclarations

Il existe plusieurs façons d'organiser vos déclarations au sein d'un bloc. Par exemple en suivant l'ordre alphabétique ou suivant votre propre convention. L'une des meilleurs pratiques est de décomposer les déclarations en 3 ou 4 parties:

- Les déclarations concernant le positionnement
- Les déclarations concernant le modèle de boite et son display
- Les déclarations concernant le texte
- Les déclarations concernant la décoration, les animations et le reste

Exemple de déclarations bien ordonnée

```
.selector {
    /* Positioning */
    position: absolute;
    top: 0;
    right: 0;
    z-index: 10;

    /* Display & Box Model */
    display: flex;
    justify-content: center;
    overflow: hidden;
    box-sizing: border-box;
    width: 20em;
    min-height: 20em;
    margin: 0;
    padding: 1em;
    border: 1px solid #000;

    /* Text */
    font-family: sans-serif;
    font-size: 1.2em;
    text-align: right;

    /* Decoration */
    background-color: #333;
    color: #fff;
}
```

9.4. Les conventions de nommage

9.4.1. BEM

https://en.bem.info/method/naming-convention/

BEM offre une ensemble de règles de nommage des classes CSS afin de mieux identifier les classes dans un document et de les regrouper en ensembles cohérents. ces règles définissent 3 types de classes:

- les blocs
- les éléments de ces blocs
- les modificateurs associés

Pour ces trois types, il est possible d'utiliser les minuscules, majuscules, chiffres aisni que le tiret simple

Les éléments de blocs sont écrits à partir du bloc parent suivi par un double trait de soulignement (__) et d'un nom.

Les modificateurs indique un changement d'état du bloc ou de l'élément concerné. Ils sont écrits à partir du nom du bloc ou de l'élément concerné suivi par un double tiret (--) et d'un nom.

Exemple

```
<div class="block block--modifier">
    <p class="block__element block__element--modifier">...</p>
</div>
```

9.4.2. Title CSS

http://www.sitepoint.com/title-css-simple-approach-css-class-naming/

Il s'agit d'une alternative à BEM dans laquelle les blocs sont simplement écrits avec une majuscule. Les éléments des blocs et les modificateurs sont écrits en camelCase.

Exemple de code HTML:

```
<div class="Title isModified">
    <p class="descendant">...</p>
</div>
```

Et les déclarations CSS associées:

```
.Title {}
    .Title.isModified {}
    .Title .descendant {}
```

9.4.3. OOCSS - Object Oriented CSS

https://www.smashingmagazine.com/2011/12/an-introduction-to-object-oriented-css-oocss/

Il s'agit surtout de principes afin d'éviter toute redondance dans les déclarations CSS, comme en programmation Orientée Objets.

Les classes CSS doivent être pensées comme réutilisables le plus possible.

9.4.4. SMACSS - Scalable and Modular Architecture for CSS

https://smacss.com/book/

Il s'agit d'un ensemble de principes pour permettre de d'organiser les différentes déclarations en fonction de leur rôle, à savoir:

- Les déclarations de bases communes à l'ensemble du document, comme par exemple les déclarations de reset.css ou de normalyse.css.
- Les déclarations relatives au modèle (layout) principal de la page
- Les déclarations relatives aux modules (les blocs)
- Les déclarations relatives aux états (les modificateurs)
- Les déclarations relatives au thème (couleurs, fonts….)
- Les déclarations relatives aux éléments variables comme par exemple les media-queries

9.5. Quelques bonnes pratiques

Voici quelques bonnes pratiques organisées par catégorie

9.5.1. Organisation du travail

- Réfléchir, faire un plan ou un schéma sur papier avant de commencer à coder.
- Bien organiser ses fichiers, regrouper les types de fichiers similaires dans des dossiers (img, css, js, etc...).
- Bien nommer les fichiers (en minuscules uniquement, sans espace et sans accents).
- Toujours commencer par ce que l'on sait faire.
- Utiliser un logiciel de versionning (GIT).
- Sauvegarder régulièrement son travail.

9.5.2. Le code

- Bien formater le code, indenter
- Commenter son code avec une nomenclature et s'y tenir.
- Bien structurer le code HTML avant de commencer le CSS.
- Utiliser les balises sémantiques (sans abuser: un article dans une section dans un article ne veut rien dire).

- Éviter autant que possible de rajouter des éléments HTML uniquement dédiés à la décoration. Préférer les pseudo-éléments ::before et ::after.
- Faire le bon choix pour les images: balise IMG pour du contenu ou la propriété background pour la décoration.
- Valider le code HTML sur https://validator.w3.org/

9.5.3. Le CSS

- Ne pas réinventer la roue, utiliser des outils existants, par exemple une CSS de base comme normalize.css ou un framework tel que Bootstrap. On peut également utiliser un préprocesseur CSS comme SASS.
- Utiliser si possible une convention de nommage et s'y tenir.
- Réserver les ID aux ancres et au JavaScript, NE PAS les utiliser comme sélecteur.
- Préférer le fluide au fixe: pourcentage pour la disposition et em/rem/vw pour le reste. Réserver les pixels pour la largeur principale du "container".
- Ne jamais utiliser la propriété height sauf pour les éléments qui avaient des dimensions avant leur intégration sur le site (autrement dit les images et des sprites), préférer line-height ou min-height.
- Éviter de mettre du style sur la balise <form>, elle peut être déplacée par le développeur.
- Vérifier la compatibilité des navigateurs sur https://caniuse.com/ et utiliser leurs prefixes: -moz- -webkit-... (à automatiser avec autoprefixer)

9.5.3. Performances

- Limiter le nombre de ressources externes, fusionner les fichiers CSS et JS.
- Limiter le poid (minifier) des fichiers CSS et JS.
- Limiter le nombre de fonts Web
- Optimiser et compresser les images. Bien choisir leur taille.

9.6. Ressources

Un exemple de bonnes pratiques et recommandations internes.

Guidelines de l'agence web Alsacréations:
https://github.com/alsacreations/guidelines