Guide des bonnes pratiques

Les bonnes pratiques, quel que soit votre langage de développement, vous permettront de conserver un code lisible et maintenable dans le temps. Prendre de bonnes habitudes maintenant éviteront de trop faire de réécriture de code plus tard...

Indentation

Aujourd'hui, l'indentation standard tend à se faire avec 2 espaces, parfois avec 4 espaces. L'idée est de garder une indentation claire, sans avoir de lignes trop longues.

Le nombre d'espaces dépend de chacun. L'important est de définir un standard à chaque début de projet, et de s'y tenir.

Nombre de caractères par lignes

Généralement, la taille maximum conseillée d'une ligne est de 120 caractères. Comme pour l'indentation, chacun ses préférences, il faut juste le définir en début de projet et s'y tenir tout au long.

Commentaires

Les commentaires se font comme ceci en HTML :

```
<!-- Mon commentaire -->
<h1>Mon titre</h1>
```

Et comme cela en CSS :

```
/* Mon commentaire */
.exemple {
```

```
    color: red;
}
```

Vous pouvez mettre des commentaires dès que vous le jugez utile. Par exemple, si vous récupérez un morceau de code d'ailleurs, vous pouvez le préciser :

```
<!---
Hack for IE from
https://stackoverflow.com/questions/28417056/how-to-target-only-ie-any-version-within-a-stylesheet
--->
```

HTML

- L'indentation se fait à chaque balise de bloc (`p` , `div` etc). Les balises inline, elles, ne bougent pas.

  Les balises de même importance restent au même niveau, tandis que les balises enfants se décalent d'une indentation.

  ```
  <section>
    <h1>Mon titre principal</h1>
    <p>
      Lorem ipsum dolor sit amet, consectetur adipiscing elit.
  <strong>Morbi</strong> quis arcu vitae odio laoreet ullamcorper. Phasellus <a
  href="www.google.fr">feugiat vitae</a> odio sit amet aliquam.
      <ul>
        <li>Premier élément</li>
        <li>Deuxième élément</li>
      </ul>
    </p>
  </section>
  ```

- En cas d'import de script externe, privilégiez les adresses en https, mieux sécurisés

  ```
  <script
  src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.js">
  </script>
  ```

- Tous les attributs, balises, noms d'éléments etc, doivent être écrits en **minuscule**

```
<img src="mon-image.png" alt="Mon texte alternatif" />
```

- Toujours ajouter un texte alternatif sur une image : c'est utile pour les lecteurs d'écran... mais aussi si l'image ne charge pas, car ce texte sera affiché à la place

```
<img src="mon-image.png" alt="Mon texte alternatif" />
```

- Les attributs sont toujours entourés de guillemets double (`quote`) et pas de guillemets simples (`single quote`)

```
<a href="www.google.fr" title="Lien vers Google">Clic</a>
```

- Un attribut ne doit pas contenir de caractère encodé, du type `&eur;` pour €. Il vaut mieux écrire directement le symbole, pour éviter les erreurs d'encodage :

```
<p>
  Ça m'a coûté 10€
</p>
```

- Un attribut ne doit pas être vide

Eventuellement, dans le cas de balises avec beaucoup d'attributs, vous pouvez sauter à la ligne à chaque attribut (attention à bien respecter l'indentation) :

```
<input
   type="text"
   id="monid"
   name="monnom"
   placeholder="Ceci est un textarea"
   size="30"
>
```

CSS

- Il faut toujours mettre un espace entre le sélecteur et l'accolade

```
.exemple {
  color: red;
}
```

- Les valeurs hexadécimales sont toujours en minuscules, comme les noms de classes

```
.exemple {
  color: #d1d1d1;
}
```

- Tous les attributs ont un saut de ligne

```
.exemple1 {
  color: red;
}
.exemple2 {
  color: red;
  margin: 2rem;
}
```

- Ne pas hésiter à organiser son CSS par éléments en usant et abusant des commentaires.

```
/** === GENERAL === **/
h1,
h2,
h3 {
  font-family: 'Roboto', sans-serif;
}

/** === HEADER === **/
header {
  /* ... styles */
}

/** === MAIN CONTENT == **/
/* ... styles */

/** === SIDEBAR == **/
/* ... styles */
```

  Vous pouvez également faire des @import pour chaque type d'élément (header, sidebar...), et faire un index sur le premier fichier

```
/** index.css **/
@import "global.css";
@import "header.css";
@import "article.css";

/**
  * Global => global.css
  * Header de la home => header.css
  * Page d'un article => article.css
**/
```

- Les noms des classes et des id en CSS ne doivent pas contenir d'underscore (`ma_class`), ou de camelCase (`maClass`). Les mots doivent être séparés par des tirets haut

```
.mon-exemple {
  color: red;
}
```

- Pour le nommage des classes, il faut essayer d'avoir une vision globale et penser à la réutilisation. Par exemple, `.badge` ou `.alert` .

- Il vaut mieux écrire un sélecteur par ligne

```
h1,
h2,
h3 {
  font-family: 'Roboto', sans-serif;
}
```

- Contrairement au HTML, les url s'écrivent entre single quote

```
.exemple {
  background-image: url('../ballon.png');
}
```

- Ne mettez pas d'unités si la valeur est zéro, sauf pour les angles (`deg`) et les unités de temps (`s` , `ms`)

```
.exemple {
  margin: 0 10px;
}
```

- Évitez le style dit "inline", c'est-à-dire directement dans le HTML.

  Au lieu de faire :

  ```
  <p style="text-align: center">
    Mon texte
  </p>
  ```

  Faites plutôt :

  ```
  <p class="text-center">
    Mon texte
  </p>
  ```

- Si possible, évitez de faire des `!important`. Trop d'`important` rend inefficace le système de cascade du CSS, puisque tous les sélecteurs qui auront la propriété `important` seront au même niveau d'importance... Généralement, utiliser `important` signifie que quelque chose n'est pas à sa place dans le code.

  Exception : dans le cas d'utilisation de framework comme Bootstrap ou MaterialUI, qui nécessite parfois de surcharger leurs classes.

- Il est conseillé de mettre les attributs par ordre alphabétique, même si ce n'est pas obligatoire.