Rédiger le meilleur code possible

Il ya beaucoup à apprendre - différents éléments, attributs, propriétés, valeurs, etc. - pour écrire du HTML et du CSS. Chaque leçon a jusqu'à présent eu comme objectif principal d'expliquer ces différentes composantes de HTML et CSS, dans l'espoir de vous aider à comprendre les fondamentaux fondamentaux des deux langues. Cette leçon fait un pas en arrière et regarde une image plus abstraite de HTML et CSS.

Plus précisément, cette leçon se concentre sur les meilleures pratiques de codage pour HTML et CSS. Ces pratiques de codification servent de cadre général pour l'écriture HTML et CSS. Ils s'appliquent à chaque leçon et devraient toujours être gardés à l'esprit lors de la programmation.

Lorsque vous examinez ces meilleures pratiques, réfléchissez à la manière dont elles peuvent être utilisées dans d'autres domaines ou dans des langages de programmation. Par exemple, l'utilisation de commentaires pour organiser le code (comme nous le couvrons dans cette leçon) est bénéfique dans tous les langages de programmation. Gardez un état d'esprit ouvert et envisagez comment vous pouvez utiliser pleinement chaque pratique.

Pratiques de codage HTML

Beaucoup de bonnes pratiques de codage insistent sur le maintien du code maigre et bien organisé. Les pratiques générales en HTML ne sont pas différents. L'objectif est d'écrire un balisage bien structuré et conforme aux normes. Les lignes directrices décrites ici fournissent une brève introduction aux pratiques de codage HTML; Ce n'est en aucun cas une liste exhaustive.

Écriture d'un balisage conforme aux normes

HTML, par nature, est un langage pardonnant qui permet à un code pauvre d'exécuter et de rendre à des niveaux variables d'exactitude. Le rendu réussi, cependant, ne signifie pas que notre code est sémantiquement correct ou garantie qu'il valide comme conforme aux normes. En outre, le code pauvre est imprévisible, et vous ne pouvez pas être sûr ce que vous allez obtenir quand il rend. Nous devons prêter une

attention particulière lors de l'écriture HTML et être sûr de nicher et de fermer tous les éléments correctement, d'utiliser les ID et les classes de manière appropriée, et de toujours valider notre code.

Le code qui suit a plusieurs erreurs, y compris l'utilisation de la valeur de l'attribut ID d'intro plusieurs fois quand il doit être une valeur unique, fermer les éléments <p> et dans l'ordre incorrect dans le premier paragraphe et ne pas fermer le <p > Dans le deuxième paragraphe.

Mauvais code

```
<p id="intro">Les nouveaux éléments du menu aujourd'hui
sont <strong>caramel au cidre de pomme et crêpes aux
marrons</p>.</strong>
<p id="intro">Le caramel au cidre de pomme est délicieux.
```

Bon code

```
<p class="intro">Les nouveaux éléments du menu aujourd'hui
sont <strong>caramel au cidre de pomme et crêpes aux
marrons</strong>.</p>
<p class="intro">Le caramel au cidre de pomme est
délicieux.</p>
```

Utiliser des éléments sémantiques

La bibliothèque d'éléments en HTML est assez grande, avec plus de 100 éléments disponibles pour l'utilisation. Décider quels éléments utiliser pour décrire le contenu différent peut être difficile, mais ces éléments sont l'épine dorsale de la sémantique. Nous avons besoin de la recherche et double-vérifier notre code pour s'assurer que nous utilisons les éléments sémantiques appropriés. Les utilisateurs vont nous remercier à long terme pour la construction d'un site Web plus accessible, et votre HTML sera sans doute plus facile à style. Si vous n'êtes jamais sûr de votre code, trouver un ami pour aider et effectuer des examens de code de routine.

Ici, le HTML n'utilise pas les éléments de titre et de paragraphe appropriés; Au lieu de cela, il utilise des éléments sans signification pour le style et le contenu de groupe.

Mauvais code

```
<span class="heading"><strong>Bon retour</span></strong>
<br>
Cela fait un bon moment... Qu'avez-vous fait récemment ?
<br>
```

Bon code

```
<h1>Bon retour</h1>
<p>Cela fait un bon moment... Qu'avez-vous fait récemment
?</p>
```

Utiliser une structure de document appropriée

Comme mentionné précédemment, le HTML est un langage indulgent et, par conséquent, les pages seront rendues sans l'utilisation des éléments `<!DOCTYPE html>` doctype ou `<html>`, `<head>` et `<body>`. Sans un `doctype` et ces éléments structurels, les pages ne s'affichent pas correctement dans tous les navigateurs.

Nous devons toujours être sûr d'utiliser une structure de document appropriée, y compris le doctype `<!DOCTYPE html>` et les éléments `<html>`, `<head>` et `<body>`. Ce faisant, nous gardons nos pages conformes aux normes et totalement sémantiques, et nous garantissons qu'elles seront rendues comme nous le souhaitons.

Mauvais code

```
<html>
  <h1>Hello world !</ h1>
  <p>Ceci est une page Web.</ p>
</html>
```

Bon code

```
<!DOCTYPE html>
<html>
  <head>
    <title>Hello world !</title>
  </head>
  <body>
    <h1>Hello world !</h1>
    <p>Ceci est une page Web.</p>
  </body>
</html>
```

Garder la syntaxe organisée

Au fur et à mesure que les pages augmentent, la gestion du HTML peut devenir une tâche complexe. Heureusement, il existe quelques règles rapides qui peuvent nous aider à garder notre syntaxe propre et organisée. Il s'agit notamment des éléments suivants:

- Utiliser des lettres minuscules dans les noms d'éléments, les attributs et les valeurs
- Indenter les éléments imbriqués
- Utiliser des guillemets doubles, pas simple ou complètement citations omises
- Enlever la barre oblique à l'extrémité des éléments autofermant
- Ne pas mettre les attributs avec des valeurs booléenne par défaut

L'observation de ces règles aidera à garder notre code propre et lisible. En regardant les deux ensembles de HTML ici, le bon code est plus facile à digérer et à comprendre.

Mauvais code

```
<aside>
<h3>Paris</h3>
<H5 CLASS='hidden'>Capitale de la France</h5>
<img src=paris.jpg alt="Paris, la capitale" />
<Ul>
<li>105 km carrés</li>
<li>2.2 millions d’habitants</li>
</ ul>
</ aside>
```

Bon code

```
<aside>
  <h3>Paris</h3>
  <h5 class="hidden">Capitale de la France</h5>
  <img src="paris.jpg" alt="Paris, la capitale">
  <ul>
    <li>105 km carrés</li>
    <li>2.2 millions d'habitants</li>
  </ul>
</aside>
```

Utilisez ID & classe

La création d'ID et de valeurs de classe peut être l'une des parties les plus difficiles de l'écriture HTML. Ces valeurs doivent être pratiques, liées au contenu lui-même, et non au style du contenu. L'utilisation d'une valeur "rouge" pour décrire le texte rouge n'est pas idéale, car elle décrit la présentation du contenu. Si le style du texte doit jamais être changé en bleu, non seulement le CSS doit être modifié, mais il en est de même pour le HTML dans chaque instance où la classe "rouge" existe.

Le HTML ici suppose que le message d'alerte sera rouge. Cependant, si le style de l'alerte change en orange, le nom de classe "rouge" ne sera plus logique et causera probablement de la confusion.

Mauvais code

```
<p class="rouge">
Erreur! S'il vous plaît essayer à nouveau.
</p>
```

Bon code

```
<p class="alert">
Erreur! S'il vous plaît essayer à nouveau.
</p>
```

Utilisez l'attribut de texte alternatif sur Images

Les images doivent toujours inclure l'attribut `alt`. Les lecteurs d'écran et autres logiciels d'accessibilité s'appuient sur l'attribut `alt` pour fournir un contexte aux images.

La valeur de l'attribut `alt` doit être très descriptive de ce que l'image contient. Si l'image ne contient rien de pertinent, l'attribut `alt` doit toujours être inclus, cependant, la valeur doit être laissée vierge pour que les lecteurs d'écran l'ignorent plutôt que de lire le nom du fichier image.

De plus, si une image n'a pas de valeur significative - peut-être qu'elle fait partie de l'interface utilisateur, par exemple - elle devrait être incluse si possible dans une image de fond CSS et non comme un élément `<img>`.

Mauvais code

```
<img src="koala.jpg">
```

Bon code

```
<img src="koala.jpg" alt="Un koala dans les feuilles">
```

Séparer le contenu du style

Ne jamais, jamais, utiliser des styles en ligne dans le HTML. Cela crée des pages qui prennent plus de temps à charger, sont difficiles à entretenir et causent des maux de tête pour les concepteurs et les développeurs. Utilisez plutôt des feuilles de style externes, utilisez des classes pour cibler des éléments et appliquez des styles si nécessaire.

Ici, tous les changements souhaités à des styles dans le mauvais code doit être faite dans le HTML. Par conséquent, ces styles ne peuvent pas être réutilisés, et la cohérence des styles va probablement en souffrir.

Mauvais code

```
<p style="color:#393;font-size:24px;"!>Merci</ p>
```

Bon code

```
<p class="alert-succes">Merci!</ p>
```

Évitez les cas de "Divitisme"

Lors de l'écriture de HTML, il est facile de se laisser emporter en ajoutant un élément `<div>` ici et un élément `<div>` là, pour construire tous les styles nécessaires. Bien que cela fonctionne, il peut ajouter un peu de complexité à une page, et avant trop longtemps, nous ne serons pas sûr de savoir ce que chaque élément `<div>` fait.

Nous devons faire de notre mieux pour garder notre code “maigre” et de réduire le marquage, en liant plusieurs styles à un seul élément si possible.
De plus, nous devons utiliser les éléments structurels HTML5 là où cela convient.

Mauvais code

```
<div class="container">
  <div class="article">
    <div class="titre">Articles sur le monde</div>
  </div>
</div>
```

Bon Code

```
<div class="container">
  <article>
    <h1>Articles sur le monde</h1>
  </article>
</div>
```

Continuellement revoir le code

Au fil du temps, les sites Web et les bases de codes continuent d'évoluer et de se développer. N'oubliez pas de supprimer l'ancien code et les styles non nécessaires lors de l'édition d'une page. Prenons également le temps d'évaluer et de refactoriser notre code après l'avoir écrit, en cherchant des moyens de le condenser et de le rendre plus gérable.

Pratiques de codage CSS

Semblables à ceux pour HTML, les pratiques de codage pour CSS se concentrer sur le maintien de code maigre et bien organisé. CSS a également quelques principes supplémentaires sur la façon de travailler avec certaines des subtilités de la langue.

Organiser le code avec des commentaires

Les fichiers CSS peuvent devenir assez gros, couvrant des centaines de lignes. Ces gros fichiers peuvent rendre la recherche et l'édition de nos styles presque impossible. Gardons nos styles organisés en groupes logiques. Ensuite, avant chaque groupe, fournissons un commentaire indiquant à quoi correspondent les styles suivants.

Si nous le souhaitons, nous pouvons également utiliser les commentaires pour construire une table des matières en haut de notre dossier.
Cela nous rappelle - et d'autres - exactement ce qui est contenu dans le fichier et où les styles sont situés.

Mauvais code

```
tête { ... }
article { ... }
.btn { ... }
```

Bon Code

```
/* Entête primaire */
header { ... }

/* Article  */
article { ... }

/* Boutons */
.btn { ... }
```

Ecrire CSS en utilisant plusieurs lignes et espaces

Lors de l'écriture de CSS, il est important de placer chaque sélecteur et déclaration sur une nouvelle ligne. Ensuite, au sein de chaque sélecteur, nous voulons indenter nos déclarations.

Après un sélecteur et avant la première déclaration vient la parenthèse d'ouverture, {, qui devrait avoir un espace avant.
Dans une déclaration, nous avons besoin de mettre un espace après le deux-points,:, qui suit une propriété et de terminer chaque déclaration avec un point-virgule ;.

Cela rend le code facile à lire ainsi qu'à modifier.
Lorsque tout le code est empilé dans une seule ligne sans espaces, il est difficile de chercher une valeur et d'apporter des modifications.

Mauvais code

```
a,.btn{background:#aaa;color:#f60;font-size:18px;padding:6
px;}
```

Bon Code

```
a,
.btn {
  background: #aaa;
  color: #f60;
  font-size: 18px;
  padding: 6px;
}
```

Commentaires et Espacement

Ces deux recommandations, l'organisation du code avec des commentaires et l'utilisation de plusieurs lignes et des espaces, ne sont pas seulement applicables à CSS, mais aussi au HTML ou toute autre langue. Dans l'ensemble, nous devons garder notre code organisé et bien documenté. Si une partie spécifique de notre code est plus complexe,

expliquons comment cela fonctionne et ce qu'il s'applique aux commentaires. Fournissez ainsi de l’aide aux 'autres personnes travaillant sur la même base de code, mais aussi bien pour nous-mêmes quand nous revisiterons notre propre code sur la route.

Utiliser des noms de classe appropriés

Les noms de classes (ou valeurs) doivent être modulaires et doivent porter autant que possible sur le contenu d'un élément, et non sur l'aspect. Ces valeurs doivent être écrites de manière à ressembler à la syntaxe du langage CSS. En conséquence, les noms de classe doivent être tous en minuscules et doivent utiliser des délimiteurs de trait d'union.

Mauvais code

```
.Rouge_Box{ ... }
```

Bon Code

```
alert-message{ ... }
```

Construire des sélecteurs efficaces

Les sélecteurs CSS peuvent être hors de contrôle s'ils ne sont pas soigneusement entretenus. Ils peuvent facilement devenir trop longs et trop spécifiques à l'emplacement.

Plus un sélecteur est long et plus il contient de préqualifiants, plus la spécificité qu'il contient est élevée. Et plus la spécificité est élevée, plus un sélecteur est susceptible de rompre la cascade CSS et de provoquer des problèmes indésirables.

Toujours avec l’idée de maintenir la spécificité de nos sélecteurs aussi basse que possible, n'utilisons pas d'ID dans nos sélecteurs. Les ID sont trop spécifiques, augmentent rapidement la spécificité d'un sélecteur et cassent assez souvent la cascade dans nos fichiers CSS. Les

inconvénients dépassent de loin les avantages avec les ID, et nous allons sagement les éviter.

Utilisons des sélecteurs plus courts et principalement directs. Ne les imbriquez que de deux à trois niveaux de profondeur et retirez autant de sélecteurs de sélection basés sur l'emplacement que possible.

Mauvais code

```
#aside #featured ul.news li a{ ... }
#aside #featured ul.news li a em.special { ... }
```

Bon Code

```
.news a{ ... }
.news .special { ... }
```

Utiliser des classes spécifiques au besoin

Il ya des moments où un sélecteur CSS est si long et spécifique qu'il n'a plus de sens. Il crée un problème de performance et est difficile à gérer. Dans ce cas, il est conseillé d'utiliser une seule classe. Tout en appliquant une classe à l'élément ciblé peut créer plus de code dans HTML, il permettra au code de rendre plus rapide et supprimer tous les obstacles de gestion.

Par exemple, si un élément `<em>` est imbriqué dans un élément `<h1>` à l'intérieur d'un élément `<aside>` et que tout cela est imbriqué dans un élément `<section>`, le sélecteur peut ressembler à `h1 em`. Si l'élément `<em>` est déplacé de l'élément `<h1>`, les styles ne s'appliquent plus. Un sélecteur meilleur et plus souple utiliserait une classe, par exemple `text-offset`, pour cibler l'élément `<em>`.

Mauvais code

```
section aside h1 em { ... }
```

Bon Code

```
.text-offset { ... }
```

Utiliser les propriétés et valeurs abrégées

Une caractéristique de CSS est la capacité d'utiliser des propriétés et des valeurs abrégées. La plupart des propriétés et des valeurs ont des options abrégées acceptables. A titre d'exemple, plutôt que d'utiliser quatre déclarations de propriétés et de valeurs basées sur la marge pour définir les marges autour des quatre côtés d'un élément, utilisez une propriété de marge unique et une déclaration de valeur qui définit les valeurs pour les quatre côtés à la fois. L'utilisation de l'option abrégée nous permet de définir et d'identifier rapidement des styles.

Toutefois, lorsque nous ne fixons qu'une seule valeur, nous ne devrions pas utiliser d'abréviations. Si une zone ne nécessite qu'une marge inférieure, utilisez uniquement la propriété `margin-bottom`. Cela garantit que les autres valeurs de marge ne seront pas écrasées, et nous pouvons facilement identifier à quel côté la marge est appliquée sans beaucoup d'effort cognitif.

Mauvais code

```
img {
  margin-top: 5px;
  margin-right: 10px;
  margin-bottom: 5px;
  margin-left: 10px;
}
button {
  padding: 0 0 0 20px;
}
```

Bon Code

```
img {
  margin: 5px 10px;
}
button {
  padding-left: 20px;
}
```

Utiliser des valeurs de couleur hexadécimales abrégées

Si disponible, utilisez la valeur de couleur hexadécimale à trois caractères et utilisez toujours des caractères minuscules dans n'importe quelle valeur de couleur hexadécimale. L'idée, encore une fois, est de rester cohérent, d'éviter toute confusion et d'embrasser la syntaxe du langage dans lequel le code est écrit.

Mauvais code

```
.module {
  background: #DDDDDD;
  color: #FF6600;
}
```

Bon Code

```
.module {
  background: #ddd;
  color: #f60;
}
```

Supprimer les unités de mesure pour les éléments de valeurs zéro

Une façon de réduire facilement la quantité de CSS que nous écrivons consiste à retirer l'unité de toute valeur nulle. Quel que soit l'unité de longueur utilisée - pixels, pourcentages, em, etc., zéro est toujours zéro. L'ajout de l'unité est inutile et n'offre aucune valeur supplémentaire.

Mauvais code

```
div {
  margin: 20px 0px;
  letter-spacing: 0%;
  padding: 0px 5px;
}
```

Bon Code

```
div {
  margin: 20px 0;
  letter-spacing: 0;
  padding: 0 5px;
}
```

Grouper & aligner les préfixes fournisseur

Avec CSS3, les préfixes de fournisseur ont gagné une certaine popularité, et surtout ajouté un peu de code aux fichiers CSS. Le travail supplémentaire de l'utilisation de préfixes fournisseurs vaut souvent par les styles générés, cependant, ils doivent être organisés. Conformément à l'objectif d'écrire un code facile à lire et à modifier, il est préférable de regrouper et d'aligner les préfixes individuels des fournisseurs afin que les noms de propriété soient alignés verticalement, tout comme leurs valeurs.

Selon l'endroit où le préfixe du fournisseur est placé, sur la propriété ou la valeur, l'alignement peut varier. Par exemple, le bon code suivant maintient la propriété d'arrière-plan alignée à gauche, tandis que les fonctions `linear-gradient()` préfixées sont indentées pour aligner correctement leurs valeurs verticalement empilées. Ensuite, la propriété de `box-sizing` préfixée est décalée comme nécessaire pour conserver l'alignement vertical des propriétés.
Comme toujours, l'objectif est de rendre les styles plus facile à lire et à modifier.

Mauvais code

```
div {
  background: -webkit-linear-gradient(#a1d3b0, #f6f1d3);
  background: -moz-linear-gradient(#a1d3b0, #f6f1d3);
  background: linear-gradient(#a1d3b0, #f6f1d3);
  -webkit-box-sizing: border-box;
  -moz-box-sizing: border-box;
  box-sizing: border-box;
}
```

Bon Code

```
div {
background: -webkit-linear-gradient(#a1d3b0, #f6f1d3);
background:    -moz-linear-gradient(#a1d3b0, #f6f1d3);
background:         linear-gradient(#a1d3b0, #f6f1d3);
-webkit-box-sizing: border-box;
   -moz-box-sizing: border-box;
        box-sizing: border-box;
}
```

Préfixes de fournisseurs -navigateurs-

Lorsque vous utilisez des préfixes de fournisseur, nous devons nous assurer de placer une version non préfixée de notre propriété et la dernière valeur, après toutes les versions préfixées. Cela garantit que les navigateurs qui prennent en charge la version non préconfigurée rendent ce style en fonction de son emplacement dans la cascade, en lisant les styles du haut du fichier vers le bas.

Les bonnes nouvelles sont que les navigateurs sont en grande partie loin d'utiliser les préfixes des fournisseurs. Avec le temps, cela deviendra moins une préoccupation; Cependant, pour le moment nous sommes bien avisés de vérifier quels sont les styles qui exigent un préfixe fournisseur et de garder ces préfixes bien organisés.

Modulariser les styles pour une réutilisation maximale

CSS est conçu pour permettre la réutilisation des styles, en particulier avec l'utilisation des classes. Pour cette raison, les styles affectés à une classe doivent être modulaires et disponibles pour partage entre les éléments si nécessaire.

Si une section de nouvelles, classe `news`, est présentée dans une boîte qui comprend une bordure, une couleur d'arrière-plan et d'autres styles, la classe de `news` pourrait sembler une bonne option. Cependant, ces mêmes styles peuvent également être appliqués à une section d'événements, classe `events`. La classe des nouvelles ne convient pas

dans ce cas. Une classe elt-box, pour “boîte d’éléments” aurait plus de sens et peut être largement utilisé sur l'ensemble du site web.

Mauvais code

```
.news {
  background: #eee;
  border: 1px solid #ccc;
  border-radius: 6px;
}
.events {
  background: #eee;
  border: 1px solid #ccc;
  border-radius: 6px;
}
```

Bon Code

```
.elt-box {
  background: #eee;
  border: 1px solid #ccc;
  border-radius: 6px;
}
```

Ressources utiles

Vous trouverez ci-dessous une liste plus longue de ressources, ainsi que des liens utiles.

HTML & CSS

- W3Schools
- Mozilla Developer Network
- Caniuse
- CCSreference.io

Conception / Inspiration

- Dribbble

Icônes

- Ionicons
- FontAwesome

Fonts

- Google Fonts

Images

- Stocksnap.io
- Stock-up
- Burst

Couleurs

- Kuler
- ColorHexa
- Open-color
- Color Supplyyy