

## Bloc-note 'HTML'

---

Le 30 décembre 2005, à 9:20 par *Ulhume...*

Les tags HTML à l'origine permettaient un formatage des documents. Avec le temps et l'apparition des feuilles de styles CSS, HTML devient un langage plus sémantique. C'est à dire qu'il permet de définir les types de contenus qu'il contient, le formatage de ces contenus étant délégué aux feuilles de style.

### Gestion des modifications

Un peu comme l'activation du suivi de modification dans un traitement de texte les balises `ins` et `del` (définition [W3C](#) [1]) permettent de marquer des portions de texte ayant été respectivement ajoutées ou supprimées du document d'origine. Les attributs `datetime` et `cite` permettent de spécifier la date de la correction et la source (URL) ayant motivée cette dernière.

Les défauts de ces tags tiennent plus à sa gestion dans les navigateurs. Par exemple l'auteur n'y est pas pris en compte (pas d'attribut `author`), on ne peut pas masquer ces modifications pour avoir la version finale, pas non plus de bulle d'aide sur la modification pour connaître les informations `datetime` et `cite`. En gros, pas très utilisable hors en tant que suivi de modification. Il s'agit plus d'un outil d'honnêteté intellectuelle à utiliser pour montrer sa bonne foi 😊. Ceci dit, rien n'empêche de créer une feuille de style prenant en charge les manques cités ici.

Je pense avoir commis quelques de nombreuses erreurs.

### Mise en avant de portions de code

Déjà pour clarifier la problématique, il y a deux types de rendu de code. Un rendu au format bloc, un listing en gros. Et un rendu en tant que citation dans une phrase. Pour le rendu de type citation nous disposons des balises suivantes :

#### **Balise `code`**

Définition d'une portion de code. ex: `for (int value; valueList)`

#### **Balise `var`**

Définition d'une variable du code. ex: `int rowWidth`. A noter que cette balise hérite de `code`.

#### **Balise `kbd`**

Représente une saisie clavier. ex: `# dd`

#### **Balise `samp`**

Définit le résultat de sortie d'une commande, fonction, etc.. ex: `Syntax Error`

Maintenant que nous savons cela, il nous manque malgré tout un élément. Comment définir un paragraphe de type "portion de code". La solution préconisée par le W3C consiste à utiliser les balises `pre` et `code`. La première étant le conteneur de la seconde. Ce qui nous donne un code du genre :

```
<pre><code>
Mon code source
</code></pre>
```

Ce qui est assez désagréable avec cette solution c'est que cela devient assez lourd à relire dès que l'on utilise du code XML ou HTML car tous les `>` et `<` doivent être remplacés par des `>` et `<`. Maintenant, un exemple reprenant l'ensemble des idées de ce chapitre :

```
<pre>
<kbd># ls *.sh</kbd>
<samp>test.sh</samp>
<kbd># cat test.sh</kbd>
<code>
export <var>VAR</var>=value
echo $VAR
</pre></code>
```

## Gestion des ellipsis

Les ellipsis sont les `...` que l'on trouve à la fin d'une phrase tronquée lorsque il n'y a pas assez de place pour l'afficher entièrement. J'ai longtemps cherché un moyen de gérer ce fonctionnement en CSS car c'est à peu près impossible à programmer (il faudrait connaître la taille de la font au niveau client). Et j'ai trouvé une solution qui semble fonctionner [http://www.hedgerwow.com/360/dhtml/text\\_overflow/demo2.php](http://www.hedgerwow.com/360/dhtml/text_overflow/demo2.php).

Cela donne donc le code CSS suivant :

```
.ellipsis {
  white-space: nowrap
  -o-textoverflow: ellipsis
  textoverflow: ellipsis
  overflow: hidden
  display: block
  position: relative
  zoom: 1
}

.ellipsis:after {
  content: ''
  display: block
  clear: both
}

.ellipsis span {
  white-space: nowrap
  -o-textoverflow: ellipsis
  textoverflow: ellipsis
  width: 100%

  float: left
  overflow: hidden
```

