Sémantique et bonnes pratiques

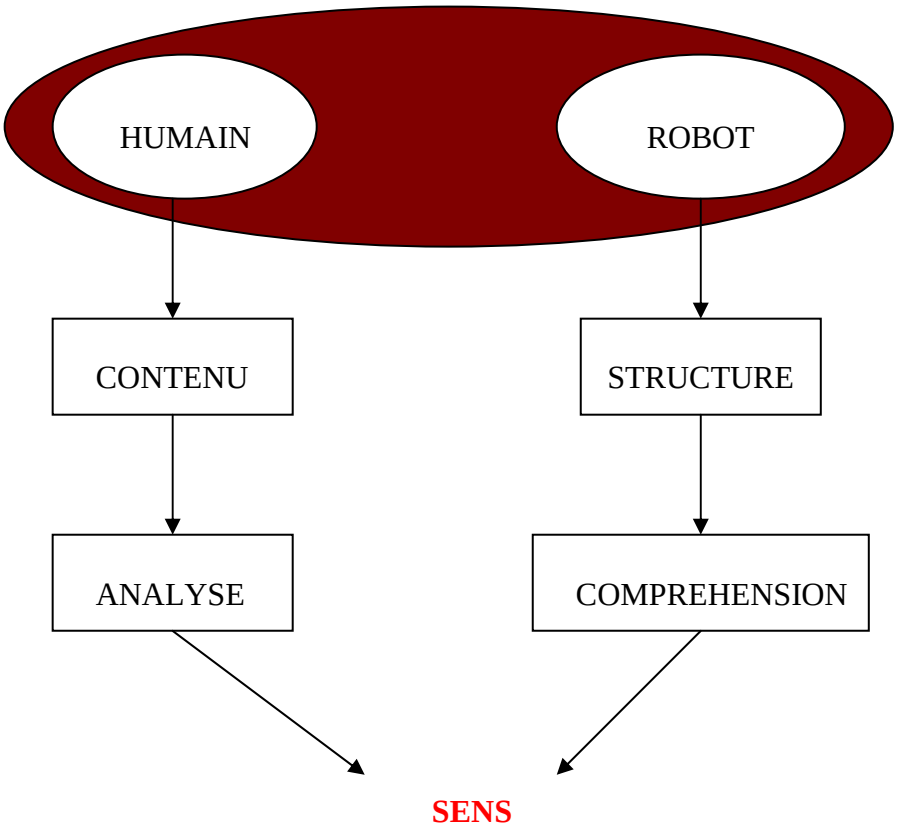
1 – CONTEXTE

Depuis plusieurs années le web est surchargé, surtout depuis 2 ou 3 ans avec l'arrivée du web 2.0. Le web 2.0 consiste à faire créer du contenu par les utilisateurs, quelque chose d'important car plus un site a de contenu plus celui-ci est susceptible d'intéresser les utilisateurs. Comme vous l'aurez compris, le contenu se place aujourd'hui au cœur du débat, le problème c'est que la structure de ce contenu à tendance à être laissée de côté par les développeurs. Il s'agit pourtant d'une notion cruciale permettant de mettre en valeur son contenu.

Il devient de plus en plus important de mettre en valeur son contenu afin de pouvoir être visible sur les principaux moteurs de recherche et que ceux-ci comprennent notre contenu. En effet les moteurs de recherche bien que de plus en plus évolués ne réalisent pas encore d'analyses graphiques, ils se contentent donc d'analyser le code de vos pages pour en extraire le contenu, et ainsi cibler votre site internet.

Afin de bien comprendre la façon de voir d'un robot de moteur de recherche, il faut le considérer comme un aveugle qui analyserait votre page en braille, c'est-à-dire sans aucun style ni image, juste le contenu mis en valeur par vos balises. C'est ainsi que nous sommes amenés à produire du contenu sémantique ou plus précisément semi-sémantique.

Un contenu sémantique est un contenu structuré à l'aide de balises descriptives (méta-données), ces données sont le seul moyen pour un robot de donner du sens à votre contenu. A l'inverse pour un utilisateur non développeur c'est incompréhensible.



2- LA SEMANTIQUE INTRODUITE DANS VOS PAGES INTERNET

Comme expliqué précédemment la sémantique consiste à faire comprendre le sens d'un texte grâce à sa structure. Nous allons dans cette partie, aborder des notions qui devraient être acquises pour tous les développeurs internet.

La structure réelle de la page XHTML est très proche de celle du corps humain. En effet, il est difficile d'imaginer une tête sans corps, tout comme il est difficile d'imaginer l'inverse. C'est pareil pour du code HTML il est impossible d'imaginer une balise head sans balise body, où l'inverse. Les informations contenues dans la balise head (la tête) n'apparaissent pas pour l'utilisateur, mais par contre tout comme pour le corps humain elles commandent le corps. Le mental influe sur votre comportement corporel, tout comme le head influe sur le body.
Ainsi il faut considérer le header d'un code HTML comme coordinateur et informateur permettant d'identifier et de mieux analyser le corps de la page.

A- ENTETES HTML

Beaucoup de développeurs se demandent l'utilité du header, et plus grave encore l'utilité du doctype, ce sont pourtant les bases du développement web, on ne peut se permettre de développer des sites internet sans se soucier de cette partie.

Le doctype (bien que extérieur au header au niveau du code) :
Dans les normes du HTML et du XHTML, un DOCTYPE (contraction pour "Document Type Declaration") permet d'informer votre navigateur et les validateurs de la version de (X)HTML que vous utilisez. Il doit absolument apparaitre en première position dans chaque page web. Les DOCTYPEs sont des composants primordiaux pour les pages conformes aux normes : ni le balisage ni la CSS ne passeront une validation sans DOCTYPE.

Les éléments importants du header :
Tout d'abord il ne faut surtout pas négliger la balise title, elle permet d'informer l'utilisateur de votre contenu et d'en informer les robots par la même occasion.
Logiquement le title doit être différent sur chaque page car il permet de cibler vos pages.

Ensuite il y a les balises meta, ces balises laissées de côté par beaucoup de développeurs, ont encore du sens contrairement à ce qu'on aurait put penser.
La balise keyword et la balise description sont des éléments essentiels à mettre dans une page. Sans ces éléments vous ne pouvez espérer que les moteurs de recherche comprennent votre contenu.

Puis il y a les balises LINK, ces balises permettent de faire appel à des fichiers extérieurs, cf. documentation html.

B- COMMENT STRUCTURER SON CONTENU

En XHML il faut savoir qu'il existe plusieurs types d'éléments, les principaux sont les éléments de type block, de type inline, et ceux de type liste.

La différence entre ces éléments est relativement simple, les éléments de type block comme les éléments de type liste s'affichent les uns en dessous des autres alors que les éléments de type inline s'affichent les uns à la suite des autres.

Les principaux éléments de type block : div, h1, h2, h3, h.., p, ...
Les principaux éléments de type inline : span, small, strong, em, …
Les principaux éléments de type liste : dt, dd, li, …

Il découle de ces différents types des règles essentielles sensées êtres connues de tout développeur web :

- Les éléments de type block ne peuvent pas contenir d'autres éléments de type block (sauf les div qui eux peuvent contenir tout type d'éléments).
- Les éléments de type inline peuvent être imbriqués les uns dans les autres.
- Les éléments inline peuvent être imbriqués dans des éléments de type block mais jamais l'inverse.
- On ne peut en aucun cas imbriquer un élément de type block dans un élément de type liste.
- Il est correct d'imbriquer des éléments de type inline dans des listes.
- les éléments de type liste ne doivent être imbriqués que dans des éléments div.
- Il est possible modifier le comportement de ces éléments avec la CSS (propriété display : block, ou display : inline), mais en respectant toujours les notions de sémantique énoncées précédemment qui sont essentielles pour la compréhension d'un robot.

Voici un code XHTML de contenu tel qu'il devrait être sur tous les sites :

```
<div>
<h1>MON SITE</h1>
<p>C'est le plus beau de tous</p>
  <h2>BONNE SEMANTIQUE</h2>
    <h3>LES TITRES</h3>
      <p>les titres sont très importants pour la compréhension</p>
    <h3>LES BLOCK</h3>
      <p>Si ils sont bien interprétés par les robots peuvent vous faire remonter sur des recherche thématiques</p>
</div>
```

Comme vous pouvez le voir dans l'exemple valide ci-dessus, le div joue un rôle d'encadreur il est avant tout là pour positionner son contenu plus que pour le structurer. La structure est donnée par les titres et leur contenu.

Voici un code de menu tel qu'on devrait le voir partout :

```
<ul>
        <li>Ma Rubrique 1</li>
        <li>Ma Rubrique 2</li>
        <li>Ma Rubrique 3</li>
        <li>Ma Rubrique 4</li>
</ul>
```

Voici un code de formulaire tel qu'on devrait le voir partout :

```
<form method="post">
  <fieldset>
    <legend>MON FORMULAIRE</legend>
    <label for="lastname">Name :</label>
    <input type="text" name="lastname" />
    <label for="firstname">Prénom :</label>
    <input type="text" name="firstname" />
   </fieldset>
</form>
```

C- COMMENT INTEGRER LE JAVASCRIPT ET LA CSS

Nous avons vu comment structurer une page XHTML, mais on ne peut se permettre de limiter une page web au XHTML, en effet il est important d'étudier le positionnement du javascript ainsi que du style CSS au milieu de toutes ces notions.

-> JAVASCRIPT

En se replaçant au niveau du corps humain on comprend qu'un comportement javascript, qui commande un élément n'a aucune raison de se retrouver dans le corps, puisque le cerveau est dans la tête. C'est ainsi qu'on arrive à la notion de **javascript non intrusif**, il s'agit d'une notion essentielle pour réaliser un code propre.

En effet un évènement qui se produirait dans la vie influerait sur votre tête qui commanderait votre corps en retour. Pour le javascript c'est pareil afin qu'il soit non intrusif il ne doit sous aucun prétexte se trouver dans le corps d'une page XHTML. Le javascript doit venir s'adapter à la page et s'auto-commander.

Une question risque de vous venir à l'esprit : comment peut-on appeler une fonction sur un évènement si on ne peut plus mettre de javascript dans une page.

C'est très simple, en manipulant le DOM et affectant les évènements aux éléments importants.

EXEMPLE :

Exemple.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Mon superbe exemple</title>
<meta name="description" content="Superbe exemple par arthur Tarlay." />
<meta name="keywords" content="superbe, exemple, superbe exemple, arthur, tarlay, arthur tarlay" />
<script type="text/javascript" src="exemple.js"></script>
</head>

<body>
<a href="#" id="monId">toto le roi du velo</a>
</body>
</html>
```

exemple.js

```
/*
 *  DEFINITION DE LA FONCTION APPELEE AU CHARGEMENT
 */
function attacherAction() {
   if (document.getElementById)
       if (document.getElementById('monId')) {
           leLien = document.getElementById('monId');
           leLien.onclick = function() {
               maFonction(this);
           }
       }
}


/*
 *  DEFINITION DE L'APPEL DE FONCTION AU CHARGEMENT
 */
function addLoadEvent(func) {
  var oldonload = window.onload;
  if (typeof window.onload != 'function')
    window.onload = func;
  else {
    window.onload = function() {
      if (oldonload)
        oldonload();
      func();
    }
  }
}

addLoadEvent(attacherAction); // APPEL DE LA FONCTION AFFECTANT LE CHARGEMENT DE LA PAGE
```

Dans cet exemple le javascript se gère entièrement seul et vient se greffer sur le code HTML. Ainsi il est enfin possible de rayer les onclick, onfocus, onmouseover et autre événement de son code html et d'obtenir un code XHTML propre.

-> CSS

Il en va de même pour la feuille de style CSS qui doit être extérieure au fichier, il faut absolument éviter de mettre du code CSS au milieu du code XHTML. Minimiser les classes et id aussi afin d'assainir le code. Et surtout pour la propreté du code CSS, éviter de se baser sur des id afin de pouvoir réutiliser les classes.

EXEMPLE :

Exemple.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Mon superbe exemple</title>
<meta name="description" content="Superbe exemple par arthur Tarlay." />
<meta name="keywords" content="superbe, exemple, superbe exemple, arthur, tarlay, arthur tarlay" />
<link href="exemple.css" rel="Feuille de style principale" type="text/css" />
</head>

<body>
<h1>Ma jolie page</h1>
<p>Ma jolie page exposes les notions importantes de <span class="underline">sémantique</span>.</p>
</body>
</html>
```

Exemple.css

```css
/* Surcharge */
p {  color: blue; font-weight: bold; font-size: 1em; border: 1px solid red; }
h1 { color: red; font-weight: bold; font-size: 1.2em; }

/* Classes utiles */
span.underline { text-decoration: underline; }
```

3- CONCLUSION

Un schéma valant mieux que de longs discours, voici un schéma récapitulatif des notions abordées dans cet article (vous noterez la ressemblance avec le corps humain) :