12

L'accessibilité en HTML5

L'accessibilité des applications web est un problème majeur. Non seulement le nombre de personnes handicapées utilisant Internet croît chaque jour, mais le contexte dans lequel elles naviguent sur le Web s'agrandit également. Remplir des formulaires web au clavier et non à la souris, glisser-déposer des objets sur écran tactile, ou utiliser un élément multimédia depuis une connexion UMTS faible (**http://fr.wikipedia.org/wiki/Universal_Mobile_Telecommunications_System**) sur un téléphone portable sont autant de barrières auxquelles les personnes handicapées sont confrontées. Les raisons poussant à la création d'applications accessibles et les techniques pour y parvenir ne concernent pas seulement les handicaps (visuels, auditifs, physiques, liés à la parole, cognitifs ou neurologiques) qui limitent l'accès au Web. Elles visent plutôt l'amélioration de l'accès au Web pour toute personne qui rencontre des difficultés.

Les quatre principes de l'accessibilité

Les conseils et les critères de succès s'organisent autour de quatre principes, qui posent les fondations nécessaires pour que chacun accède au contenu web et l'utilise (**http://www.w3.org/TR/UNDERSTANDING-WCAG20/intro.html#introduction-fourprincs-head**). Toute personne qui veut naviguer sur le Web doit pouvoir obtenir un contenu :

- **Perceptible.** Les informations et les composants de l'interface utilisateur doivent être présentés de telle façon que les utilisateurs puissent les percevoir d'une manière ou d'une autre, ils ne peuvent être invisibles à tous leurs sens.

- **Utilisable.** Les composants de l'interface utilisateur doivent être utilisables : les utilisateurs doivent pouvoir faire fonctionner l'interface (laquelle ne doit pas imposer une interaction qu'un utilisateur ne pourrait pas effectuer).
- **Compréhensible.** Les informations et le fonctionnement de l'interface utilisateur doivent être compréhensibles. Les utilisateurs doivent être à même de comprendre l'information, mais aussi le fonctionnement de l'interface (le contenu et/ou le fonctionnement ne doivent pas dépasser leur compréhension).
- **Robuste.** Le contenu doit être suffisamment fiable pour pouvoir être interprété de manière robuste par un large champ de navigateurs, y compris les technologies auxiliaires. Cela signifie que les utilisateurs doivent être capables d'accéder au contenu qui doit rester accessible au fur et à mesure que la technologie évolue.

Il existe plusieurs organismes qui travaillent à ce problème: W3C, WCAG, WAI, ATAG (*Authoring Tool Accessibility Guidelines*) et UAAG (*User Agent Accessibility Guidelines*). Nous allons les clarifier un peu. En effet, les choses peuvent parfois apparaître plus complexes qu'elles ne le sont vraiment.

Le W3C (*World Wide Web Consortium*) et le WAI (*Web Accessibility Initiative*) ont sorti le 11 décembre 2008 la seconde version du WCAG (*Web Content Accessibility Guidelines*), des conseils pour l'accessibilité au contenu web.

Le but de ces conseils est d'expliquer comment faire du contenu web qui soit accessible aux personnes handicapées, en définissant plusieurs niveaux. En pratique, le consortium veut permettre à tous les utilisateurs du Web d'obtenir la même information et d'appliquer les mêmes fonctions.

Le but du WCAG

Les recommandations d'accessibilité au contenu web du WCAG font partie d'une série de conseils destinés à aider les développeurs à produire du contenu accessible, principalement pour les personnes handicapées, sur tous les user-agents, même les plus limités comme les périphériques mobiles.

Ce groupe de recommandations est entretenu par le W3C (voir Figure 12.1).

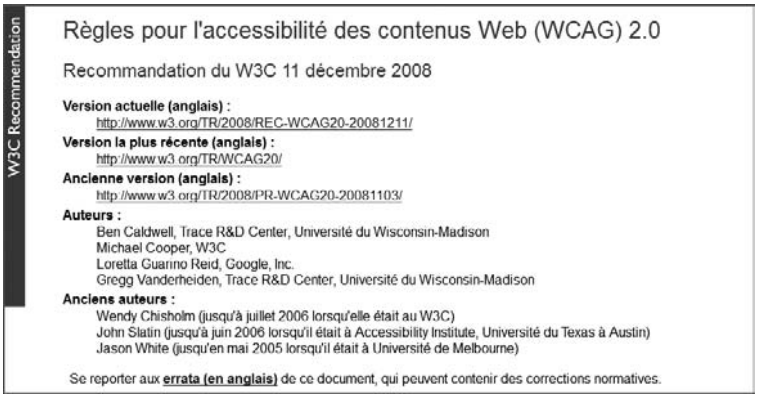

Figure 12.1
La page de spécification WCAG sur le site du W3C.

Trois niveaux de recommandations ont été établis.

Niveau 1. Vue d'ensemble des principes de conception, recommandations et critères de succès

Ce niveau propose une introduction au WCAG 2 (la deuxième génération des recommandations pour l'accessibilité), les principes de l'accessibilité, treize recommandations qui ne sont pas liées à une technologie, et les réglementations et définitions d'exemple pour chaque recommandation, avec un index final.

Niveau 2. Listes de vérifications spécifiques des technologies

Conçu pour appuyer les recommandations générales, il inclut une série de documents et de listes de vérifications qui aident à définir l'information concernant les technologies à utiliser pour respecter les spécifications du WCAG 2.

Niveau 3. Informations d'application spécifique des technologies

Il inclut des exemples de code, des captures d'écran et d'autres informations techniques.

Techniques générales

- les techniques pour le HTML et le XHTML ;
- les techniques pour les feuilles de style en cascade (CSS) ;
- les techniques pour le codage côté serveur ;
- les techniques pour le codage côté client ;
- les techniques pour les SVG (*Scalable Vector Graphics*) ;
- les techniques pour le SMIL (*Synchronized Multimedia Integration Language*) ;
- les techniques pour le XML (*eXtensible Markup Language*).

Par conséquent, c'est par l'étude et l'application des bonnes pratiques décrites dans le WCAG que vous pourrez réaliser des applications web accessibles.

Il y a un autre problème que nous devons évoquer qui est celui du logiciel utilisé pour la mise à jour du contenu web. La plus grande partie du contenu présent sur le Web est créée à l'aide d'outils de développement (comme les CMS). Ce logiciel détermine souvent comment et jusqu'à quel point le contenu peut être accessible. Il joue donc un rôle fondamental dans la création de contenu conforme avec le WCAG. Par conséquent, il est essentiel que les développeurs de ces outils suivent les spécifications WCAG afin de rendre leurs logiciels plus polyvalents et plus appropriés à la création de contenu accessible.

L'ATAG (*Authoring Tool Accessibility Guidelines*) édite des guides à destination des auteurs de logiciels et ses recommandations ont été établies avec les critères exposés ci-dessus comme base.

D'autre part, l'UAAG (*User Agent Accessibility Guidelines*) guide le développement des user-agents, entraînant une accessibilité en fonction de différents types de handicaps. Les user-agents incluent les navigateurs, les lecteurs de médias et tous les logiciels qui exécutent le rendu et le traitement du contenu web.

Solution 12.1 – Créer des *skip links* avec l'élément *nav*

Si l'on se réfère à Wikipédia (**http://fr.wikipedia.org/wiki/Lecteur_d%27%C3%A9cran**), un lecteur d'écran est un logiciel destiné aux personnes aveugles ou fortement malvoyantes. Il retranscrit par synthèse vocale et/ou sur un afficheur braille ce qui est affiché sur l'écran d'un ordinateur. Les lecteurs d'écran sont une forme de technologie d'assistance potentiellement utile pour les personnes aveugles ou à la vision limitée, illettrées ou souffrant de troubles d'apprentissage. Ils sont souvent utilisés combinés avec d'autres technologies d'assistance comme les magnificateurs d'écran.

Les personnes handicapées, qui naviguent sur le Web à l'aide d'un lecteur d'écran, ont des besoins différents les unes des autres. Certaines peuvent vouloir sauter rapidement à certaines parties d'une page web, alors que d'autres préfèrent attendre que le lecteur d'écran lise la page de manière séquentielle.

Le premier type d'utilisateurs peut tirer avantage de raccourcis (*skip links*) proposés par le lecteur d'écran afin de déterminer rapidement le contenu à sauter. Le lecteur d'écran propose aussi certains éléments sur demande, comme les éléments de navigation, les contenus, les en-têtes, les pieds de page, etc. Le développeur doit donc concevoir la page qui pourra passer ces possibilités au lecteur d'écran.

Voyons comment créer ces fonctions en HTML5.

Tour d'horizon

La plupart des pages web se réduisent à la structure suivante : en-tête, pied de page, menu de navigation, body, colonne, etc.

En HTML4, cette structure était souvent créée à l'aide de la balise `<div>` associée avec un attribut `id` :

```
<body>
<div id="header"></div>
<div id="body">
  <div id="menu"></div>
  <div id="page content"></div>
  <div id="right_sidebar"></div>
</div>
<div id="footer"></div>
</body>
```

Dans cet exemple, nous avons créé la structure d'une page avec en-tête, body, menu, contenu, colonne latérale droite et pied de page. Le contenu est intégré dans cette structure puis formaté avec des déclarations CSS.

L'utilisation massive de `div` était nécessaire du fait du manque de sémantique en HTML4 pour décrire la répartition logique de la page.

Plusieurs techniques permettaient d'offrir une navigation avec des skip links. Ils sont utilisés pour permettre à une personne utilisant les technologies d'assistance de ne pas écouter tout le

contenu de la page, mais seulement ce qui l'intéresse. Cette approche est également vue dans le WCAG au Checkpoint 13.6 "Groupez les liens liés entre eux, identifiez le groupe (pour les user-agents) et, jusqu'à ce que les user-agents le fassent, proposez un moyen d'éviter les groupes".

Une méthode utilise les skip links visibles, alors qu'une autre préfère des skip links cachés. Quelle que soit la solution adoptée, le concept du skip link est basé sur l'attribut `name` de la balise `<a>`. En fait, on créait un lien vers une ancre dans la page, ancre qui, une fois sélectionnée, amenait l'utilisateur à la partie spécifique de cette page.

Voici un exemple pratique de skip link qui permet à un lecteur d'écran de sauter le menu de navigation pour passer directement au contenu principal :

```
<div id="body">
<div id="menu">
<ul>
  <p><a href="#main_content">passer au contenu principal</a></p>
  <li>Liens de Menu</li>
  <li>Liens de Menu</li>
  <li>Liens de Menu</li>
...
</ul>
</div>
<div id="page content">
  <a name="main_content" id="contentjump"></a>
  <p>Contenu</p>
</div>
</div>
<div id="right_sidebar"></div>
</div>
```

Un autre exemple de navigation skip link, implémenté par de multiples portails qui disposaient d'un contenu très riche, crée un vrai menu afin de naviguer rapidement vers des portions de la page :

```
<div class="skiplinks">
<strong>Bienvenu sur le portail XYZ </strong>
Passer directement à: <a href="#main">Contenu principal</a>,
<a href="#nav">Menu de Navigation</a>,
<a href="#news">Section Actualités</a>.
<a href="#video">Section Vidéo</a>.
</div>
```

Avec le HTML5, cette approche a changé de manière radicale. En effet, il introduit tout un groupe de nouveaux éléments qui permettent de structurer les pages web beaucoup plus facilement.

Vous avez vu les éléments de structure et de sémantique au Chapitre 3. Vous allez maintenant les utiliser pour créer des fonctions de navigation rapide dans une page web, sans avoir besoin de skip links.

Pour le moment, tous les lecteurs d'écran ne reconnaissent pas les nouveaux éléments sémantiques du HTML5, par exemple le marquage nav. Néanmoins, les sociétés qui proposent ce logiciel travaillent en étroite collaboration avec le W3C pour proposer cette prise en charge.

Mise en œuvre

Avec la nouvelle balise <nav>, les navigateurs disposent maintenant d'un élément HTML standard pour spécifier un contenu de navigation. De fait, l'ancienne technique – les skip links – est devenue obsolète.

Avec le nouvel élément <nav>, on peut maintenant spécifier du contenu de navigation à l'intérieur de la page pour les lecteurs d'écran, comme l'explique une note des spécifications HTML5.

Les user-agents (tels que les lecteurs d'écran) ciblant les personnes ayant la possibilité de bénéficier de l'omission d'informations de navigation lors du rendu initial, ou encore celles pouvant bénéficier d'informations de navigation immédiatement disponibles, peuvent utiliser cet élément [l'élémen nav] pour déterminer quel contenu de la page sauter dès le début, et/ou le proposer sur demande.

Lorsque le logiciel du lecteur d'écran reconnaît l'élément <nav>, il devient capable de proposer à l'utilisateur une manière de sauter la section navigation.

Il est possible d'utiliser plus d'un élément <nav> dans la même page. Dans l'exemple suivant, nous en avons deux. Le premier permet à l'utilisateur de se connecter aux pages web externes alors que le second est conçu pour la navigation au sein de la page.

```
<!DOCTYPE html>
<html>
<head>
<title>
Solution 12-1: Créer des skip links avec l'élément nav
</title>
</head>
<body>
<header>
<h1>Solution 12-1: Créer des skip links avec l'élément nav
</h1>
</header>
<nav>
<ul>
  <li><a href="http://casario.blogs.com">My Blog</a></li>
  <li><a href="http://it.linkedin.com/in/marcocasario">Mon Profil LinkedIn</a></li>
  <li><a href="http://twitter.com/#!/marcocasario">Mon Twitter</a></li>
</ul>
</nav>
<header>
<h1>Arrêtez d'utiliser des skip links personnalisés!</h1>
<p>et utilisez les balises sémantiques HTML5.</p>
```

```
</header>
<nav>
<ul>
  <li><a href="#navtag">La balise NAV</a></li>
</ul>
</nav>
<section id="navtag">
  <h1> La balise NAV </h1>
  <h3>D'après les spécifications W3C NAV</h3>
  <p>
Une section avec des liens de navigation
  </p>
</section>
</body>
</html>
```

Tous les groupes de liens d'une page n'ont pas à être dans un élément `<nav>`. Seules les sections qui sont des blocs majeurs de navigation sont appropriées pour cet élément. Il est courant, en particulier dans les pieds de page, d'avoir une courte liste de liens vers diverses pages du site, comme les mentions légales, l'accueil et la page de copyright. L'élément `footer` est dans ce cas suffisant sans l'élément `<nav>`.

Conseil d'expert

Dans le registre des menus et des skip links, il existe un nouvel élément à prendre en considération dans les spécifications HTML5 : `<menu>`.

L'élément `<menu>` peut remplacer l'élément `<nav>` pour la navigation. `Il` est utilisé pour une liste de commandes. C'est un élément interactif qui dispose de plusieurs possibilités pour un emploi exclusif dans des applications web.

```
<menu type='toolbar'>
  <menu type='list' label='File' >
    <command label='New'/>
    <command label='Save'/>
    <command label='Close'/>
  </menu>
  <menu type='list' label='Edit'>
    <command label='Cut'/>
    <command label='Copy'/>
    <command label='Paste'/>
  <menu>
</menu>
```

Vous pouvez penser à l'élément `menu` comme à un système de menu dans une application classique sur poste de travail.

Solution 12.2 – Créer des données tabulaires accessibles

Les tableaux servent pour représenter des données au format tabulaire. Certes, ils n'ont été que trop souvent utilisés par le passé pour contrôler l'aspect d'une page et cela a généré de multiples problèmes d'accessibilité aux pages. En effet, les technologies d'assistance obtenaient des résultats très déroutants. Les utilisateurs de lecteurs d'écran, en particulier, éprouvaient beaucoup de difficultés pour naviguer dans des pages conçues à l'aide de tableaux.

D'un autre côté, ce format est nécessaire au HTML pour afficher des données ayant plus d'une dimension (les spécifications W3C le précisent).

Afin qu'un tableau soit très accessible, nous devons nous assurer qu'il dispose du formatage nécessaire pour être transformé de manière élégante et aussi être prêt pour les périphériques tels que les lecteurs d'écran.

C'est là que certaines propriétés peuvent vous venir en aide. Si elles sont utilisées intelligemment, elles rendent les données d'un tableau accessibles à tous.

Tour d'horizon

Concevoir un tableau accessible nécessite de porter une attention particulière à une série d'attributs, qui permettent aux user-agents de comprendre et d'interpréter le contenu et la structure du tableau lui-même de manière optimale.

Chaque tableau doit procurer les informations concernant les données qu'il contient. Elles seront essentielles pour les utilisateurs de technologies d'assistance afin de naviguer dans ces tableaux, suivant leur lecture et leur interprétation.

C'est pourquoi le HTML5 dispose de la balise `<caption>` qui associe une étiquette au tableau, ainsi qu'un attribut `summary` qui n'est pas affiché par un navigateur web standard, mais qui procure un texte lisible par un user-agent et décrit le contenu, la structure et le but du tableau.

```
<table summary="Scores de la dernière saison">
<caption>
  <strong>Liste des équipes et leurs scores</strong>
  <p>Les résultats pour 2010</p>
</caption>
```

Chaque tableau représente par définition des données d'une à plusieurs dimensions. Il est donc nécessaire d'associer la lecture de chaque ligne avec la colonne correspondante. C'est la seule manière de garantir une lecture correcte de l'information pour les personnes qui activent par exemple des outils de lecture d'écran.

En fait, les attributs `scope` et le couple `id` et `headers` permettent l'identification des `headers` et la relation qui existe entre eux et les cellules, ce qui entraîne une consultation plus aisée des données :

```
<tr>
  <th id="year">Year</th>
  <th id="team">Team</th>
  <th id="record" abbr="Record">Scores de la saison actuelle</th>
```