Urbilog
Solutions Accessibles

Ajax et Accessibilité

Présentation

- Michel HOËL :
  - Responsable Technique d'Urbilog

- 

En 2001, création d'Ocawa : Outil de validation automatique de "règles d'accessibilité" pour France Télécom.
  - Les règles sont externes et rattachées à un Référenciel (WCAG 1.0, DGME/ADAE,... et bientôt RGAA).

Préalable

- Ce dont je vais vous parler :
  - Savoir si on peut faire de l'Ajax accessible.
  - Si oui sous quelles conditions.
  - Peut-on espérer « industrialiser » l'utilisation de la/des technique/s.
- Ce dont je ne parlerai pas :
  - Si Ajax est la solution du Web 2.0.
  - De la comparaison des Frameworks utilisant Ajax.

Ajax

- Asynchronous Javascript And XML.
- En français : Javascript et XML asynchrones.
- Attention : ce n'est pas une technique, mais plutôt l'association conjointe de techniques.

Ajax

- Page « classique » :
  - Les informations sont recueillies (par le biais des formulaires, des liens, ...) <->Le serveur fournit des informations au travers une nouvelle page web.
- Page avec Ajax :
  - L'ensemble de ces traitements sont effectués dans la même page et intégrés à l'aide du DOM de Javascript.

Ajax - Bases

- XmlHttpRequest (XHR)
  - Requête HTTP faite par le poste client en vue de recevoir un flux d'informations.
  - Sous IE 5 et 6 : ActiveX.
  - Pour les autres pouvant faire cette requête : intégré à Javascript.
- Javascript
  - Elabore la requête,
  - Attend la réponse,
  - Analyse et traite la réponse.

Ajax – Premières remarques

Pas d'Ajax :

- XmlHttpRequest : ActiveX
  - Si le navigateur interdit les ActiveX.
- Javascript
  - Si le navigateur n'interprète pas le javascript.
  - Si l'on bloque son interprétation.

Ajax – Premières conclusions

Se poser les questions suivantes pour chaque fonction :

- Doit-on utiliser absolument Ajax?
- Dans l'affirmative, doit-on prévoir une fonctionnalité alternative?

Ajax – Le pour, le contre

- Le pour:
  - Mise à jour partielle de la page, donc:
    - Une limitation des échanges avec le serveur,
    - La modification ne concerne qu'un nombre limité d'éléments dans la page.

Ajax – Le pour, le contre

- Le contre:
  - Problème pour le référencement,
  - Problème pour les fonctions de base d'un navigateur:
    - Précédent, Suivant, Raffraichissement
    - La mise en Favoris.

Ajax – Les choix sont faits

On suppose que le choix de l'utilisation des techniques Ajax est validé (qu'il soit bon ou mauvais)

Il faut donc maintenant trouver la meilleure façon d'intégrer la technique en essayant d'être :

- Accessible,
- Testable,
- Maintenable.

Mode « dégradé » et « rehaussé »

Graceful Degradation & Progressive Enhancement, par Tommy Olsson sur Accessites.org (Fev 2006).

- Cet article a été traduit en français sur :

  http://www.yoyodesign.org/doc/accessites/graceful-degradation-progressive-enhancement/

- Il décrit deux méthodes de conception de page Web intégrant la problématique de l'accessibilité.

Mode « dégradé »

Mode dégradé (« dégradation gracieuse »),

- Une page web est d'abord fabriquée en considérant que la majorité aura les options permettant d'exécuter tout ce que l'on veut.
- Puis compléter avec mes éléments pour permettre une meilleure accessibilité (comme le préconise souvent les WCAG), permettant ainsi de réagir en mode dégradé :
  - Les éléments alternatifs (alt, title...)
  - Les balises noscript, noframe...

Mode « rehaussé »

- Mode réhaussé:
  - Ici la logique est inverse, on développe en s'imposant d'assurer le service en concevant la structure sémantique et cela :
    - Sans javascript,
    - Sans CSS.

Mode « rehaussé »

- Mode réhaussé (suite) :
  - Puis on « réhausse » le rendu en ajoutant et/ou en améliorant le service en :
    - Ecrivant les feuilles de styles externes,
    - Intégrant le Javascript en « non-intrusif » (un-obstructive),

Mode « rehaussé »

- Exemple : rendre accessible la notion de Popup,
  - <a href ="xxxx.htm" onclick="return popup(xxxx.htm);">
  - Ou mieux :
    - <a class="popup" href ="xxxx.htm"> et dans un script :

```
var maregle = {
  'a.popup' : function(element){
    element.onclick = function(){
      return popup(this.href);
    }
  }
};

Behaviour.register(maregle);
```

  - Référence : Behaviour.js - version 1.1 - Copyright (c) Ben Nolan and Simon Willison sur http://www.bennolan.com

Mode « rehaussé »

- Mode réhaussé (suite) :
  - Maintenant en intégrant les fonctions demandant des requêtes XmlHttpRequest.

Hijax

- Cette façon de procéder a été décrite également par Jeremy Keith et a même reçu un nom le « Hijax ».
- Exemple d'application de la méthode sur

  http://elsewhere.adactio.com/

Plan de développement

- L'idée forte est celle-ci:
  - Un plan pour l'utilisation d'Ajax au démarrage du projet.
  - Une intégration d'Ajax à la fin conformément au mode réhaussé:

    HTML->CSS->Javascript->Ajax

Maintenabilité d'une application Ajax

- D'abord assurer la maintenabilité du code en assurant la séparation :
  - De la structure des informations (HTLM XHTML),
  - De la représentation visuelle : CSS
  - La partie dynamique : Javascript Un-obstructive.

Maintenabilité d'une application Web

- Séparation HTML-CSS : beaucoup de littérature.
- Javascript Un-Obstructive :
  - On évite l'écriture des attributs HTML: onclick, onload, onsubmit....
  - On centralise le code dans des fichiers .js et intégrés par balise <script>

Maintenabilité d'une application Web

- Les fonctions de traitements javascript utilisent les fonctions de base de traitement du DOM
  - getElementById,
  - getElementsByTagName,
  - ...

Maintenabilité d'une application Web

- Séparation CSS-Javascript :
  - On a tendance dans les fonctions Javascript d'aller transformer le rendu en modifiant les attributs de style du DOM directement.
  - Cela peut devenir très rapidement non maintenable.
  - En modifiant seulement les attributs class ou id et en reportant le rendu à effectuer dans des règles CSS, on règle le problème.

Maintenabilité d'une application Ajax

- Il faut maintenant parler de la maintenabilité des fonctions qui intègrent les informations reçues de requêtes Ajax.
- Type de réception :
  - XML / XHTML.
  - Texte
  - JSON
- Injection des informations : innerHTML -> non standard. (mais très efficace).

Utilisation de bibliothèques Javascript

- Prototype (version 1.6.0) :
  - Ajax prend en charge
    - l'échange en fonction du navigateur utilisé.
    - l'intégration de la réponse si elle est faite en xhtml à partir d'un noeud du DOM.
  - La manipulation des Array, String.. enrichi.
  - Gestion des événements : souris, clavier...
- Avantages :
  - Réduction du code à développer,
  - La simplification de l'écriture.

Utilisation de bibliothèques Javascript

- Inconvénient de Prototype:
  - La taille de la bibliothèque : 120 ko pour la version 1.6.0.

```
/*  Prototype JavaScript framework, version 1.6.0
 *  (c) 2005-2007 Sam Stephenson
 *
 *  Prototype is freely distributable under the terms of an MIT-style license.
 *  For details, see the Prototype web site: http://www.prototypejs.org/
 *
 *--------------------------------------------------------------------------*/

var Prototype = {
  Version: '1.6.0',

  Browser: {
    IE:     !!(window.attachEvent && !window.opera),
    Opera:  !!window.opera,
    WebKit: navigator.userAgent.indexOf('AppleWebKit/') > -1,
    Gecko:  navigator.userAgent.indexOf('Gecko') > -1 && navigator.userAgent.indexOf('KHTML') == -1,
    MobileSafari: !!navigator.userAgent.match(/Apple.*Mobile.*Safari/)
  },
```

Utilisation de bibliothèques Javascript

- Solution : utiliser la version « compressée » < 25 ko

```
eval(function(p,a,c,k,e,r){e=function(c){return(c<a?'':e(parseInt(c/a)))+((c=c%a)>35?String.fromCharC
o q.1v(/&/g,\'&7H;\').1v(/</g,\'&7I;\').1v(/>/g,\'&7J;\')),7A:7(){o q.1v(/&7I;/g,\'<\').1v(/&7J;/g,\'
p(c))1W;b.N(c);C e=[c,d];e.2f=c;e.1m=d;a(e)});57=1Z.20();M.L(57.O,25);M.L(57.O,{2j:7(a,b,c){q.4o=a;q.
);b=b||\'2i\';B(1b b==\'23\')b=11 D(b);18 D.L(b);a.1M.8r(b,a);b.4j(a);o a),1G:7(d){d=$(d);C e=\'<\'+d
b.3T+\'2q\';B(c.8S)a.V.31=b.3U+\'2q\';o a)};B(!12.5y)12.5y=7(f){7 bI(a){o}7 6P(a){o a.55()?17:"[4A(1r
}7 9b(a){C b;C c={"cj":"ck","9a":"cl","P":"cm","co":"cp","cq":"cr","cs":"ct","cu":"cv","cw":"cx","cy"
),3d:/^\\s/,1h:11 22(\'^\\\\\s*\'+\'(\\\\\'+\'*|[\\\\\w\\\\\-]+)(\\\\\b|$)?\'),2X:11 22(\'^#([\\\\\w\\\\\-\'
b.1H(\',\'),b=[];c.7v(/(([\\w#:.~>+()\\s-]+|\\*|\\[.*?\\])+)\\s*(,|$)/,7(m){b.N(m[1].2D())});C d=[],h
;o D.6L(a)},6M:7(a){7i.7j();o D.6M(a)},dV:D.U.6N,2L:D.U.6O,dW:D.U.5x,2k:7(a,b,c){c=c||{};o D.8O(b,a,c
ter|contentType|Events|get|onreadystatechange|setRequestHeaders|overrideMimeType|readyState|xml|Conte
```

Ajax et Les lecteurs d'écran

- 2 problèmes importants sont à prendre en compte
  - Comment avertir les lecteurs d'un changement dans la page.
  - A l'inverse éviter d'être perturbé par les changements cycliques et automatiques d'une zone de page.

Fonctionnement des lecteurs d'écran

- Les buts d'un lecteur d'écran:
  - Lire le contenu de la page
  - Interagir avec le contenu de la page
- Ils font cela en prenant une « image » instantanée de la page et en la plaçant dans une zone tampon.
    - Dans Jaws, cette zone s'appelle Virtual PC.

Ajax et les lecteurs d'écran

- Quand est mise à jour la zone tampon ?
  - Sur événement clavier ou souris : zone input, lien hypertext...
- Avec Ajax, un nouveau contenu est ajouté à la suite de l'événement 'onreadystatechange' de l'objet XMLHttpRequest et en général Jaws n'est pas alerté.
- Conclusion : Une fois l'intégration du nouveau contenu, il faut s'assurer que le lecteur soit averti.
- Seul Jaws 7.0 avec Firefox prend en charge le onreadystatechange

Ajax et les lecteurs d'écran

- La solution pour faire prendre en compte la modicafication du contenu, est de donner le focus à la zone où le changement a été effectué.
  - Attribut tabindex= "-1"
  - Donner le focus à un lien ou un bouton de la zone.
- Tabindex négatif ne fonctionne pas sous Safari.

Le autres problèmes d'accessibilité

- Avec des solutions framework (Google, Yahoo, Dojo, jQuery ...):
  - Même si un jour on vous dit que les bibliothèques tiennent compte de l'accessibilité, reste à vous de gérer:
    - Changement de navigation (WCAG 2.0)
    - Les problèmes liés aux mouvements, contrastes de couleur, clignotements...
    - ....

Le futur

- Le projet : ARIA

  W3C – ARIA – Roadmap

  http://www.w3.org/TR/2006/WD-aria-roadmap-20060926/

  W3C – ARIA – Rôles

  http://www.w3.org/TR/2006/WD-aria-role-20060926/

  W3C – ARIA – Etats et Propriétés

  http://www.w3.org/TR/2006/WD-aria-state-20060926/

- Une intégration Ajax-ARIA semble le plus prométeur.

Merci

Des questions ?