9

Ajax, RIA et HTML5

Au sommaire de ce chapitre

- Prise en charge d'Ajax
- Bibliothèques JavaScript
- Extensions CSS de WebKit
- Applications Internet riches mobiles
- HTML5

Ajax, HTML5 et RIA, comment toutes ces technologies trouvent-elles une place sur un appareil mobile ? Puisque nous savons tous qu'Ajax n'est qu'une technique JavaScript, cet aspect est évident. Du point de vue des navigateurs mobiles, HTML5 est également plus ou moins une fonctionnalité JavaScript, avec des éléments de balisage supplémentaires (les technologies empruntées par les navigateurs mobiles au brouillon de la norme HTML5 sont quasiment toutes de nouvelles API JavaScript qui opèrent sur le contenu HTML). Le développement d'applications Internet riches (RIA, *Rich Internet Application*) implique également la mise en œuvre de modèles d'interface utilisateur en JavaScript.

9.1 Prise en charge d'Ajax

Ajax se révèle particulièrement important sur les appareils mobiles. La possibilité de télécharger uniquement les données modifiées et d'éviter ainsi des chargements de pages inutiles constitue un point essentiel pour les navigateurs mobiles. Toutefois, Ajax ne fait pas partie des standards officiels et sa prise en charge varie d'un appareil à l'autre.

INFO

Si votre site web mobile obtient du contenu en utilisant Ajax, vous devez mettre en œuvre les préconisations de Google pour l'optimisation des moteurs de recherche (SEO, *Search*

Engine Optimization). Pour de plus amples informations sur ce sujet, consultez le site **http://code.google.com/web/ajaxcrawling**.

Commençons par vérifier la prise en charge de l'objet `XMLHttpRequest` par les différents navigateurs (voir Tableau 9.1). En raison de problèmes de sécurité supposés, les requêtes interdomaines sont incompatibles avec les navigateurs mobiles actuels. Vous pouvez toutefois passer outre en plaçant un simple proxy sur votre serveur.

Tableau 9.1 : Résultats des tests de compatibilité avec l'objet *XMLHttpRequest*

Navigateur/plateforme	*Prise en charge de XMLHttpRequest*
Safari	Oui
Android	Oui
Symbian/S60	Oui depuis 3e édition
Nokia Series 40	Non avant 6e édition
webOS	Oui
BlackBerry	Non avant 4.6
NetFront	Non avant 3.5
Internet Explorer	Oui depuis Windows Mobile 5
Navigateur Motorola	Non
Opera Mobile	Oui depuis 8.0
Opera Mini	Oui depuis 3.0

Pour considérer que l'objet est disponible, nous exigeons que les propriétés et méthodes suivantes soient pleinement prises en charge :

- `open` ;
- `abort` ;
- `send` ;
- `onreadystatechange` ;
- `readyState` ;
- `status` ;
- `responseText`.

Analyser du contenu XML

Dans la liste précédente, nous avons omis la propriété `responseXML`, pourtant standard dans la technologie Ajax de bureau. L'analyse d'un contenu XML étant un processus complexe pour le navigateur, elle peut présenter quelques problèmes d'interprétation. La première différence concerne la manière dont le navigateur traite les espaces blancs dans les balises. Prenons le simple exemple suivant :

```
<noeud>
    <sousnoeud />
</noeud>
```

Certains navigateurs comprennent que le balisage précédent définit un nœud avec un seul enfant, représenté par le sous-nœud. D'autres analyseurs XML pensent qu'il existe trois enfants : un nœud de texte avec des espaces (et un caractère de passage à la ligne), le sous-nœud et un autre nœud de texte. Cette différence peut se révéler un tantinet complexe à déboguer si vous n'en avez pas conscience. Le Tableau 9.2 indique quels navigateurs savent analyser le contenu XML et leur manière d'interpréter les espaces.

Tableau 9.2 : Résultats des tests de compatibilité avec l'analyse XML

Navigateur/plateforme	*Prise en charge de l'analyse XML*	*Espaces considérés comme des enfants*
Safari	Oui	Oui
Android	Oui	Oui
Symbian/S60	Oui	Oui
Nokia Series 40	Non avant 6e édition	Oui
webOS	Oui	Oui
BlackBerry	Non avant 4.6	Oui
NetFront	Oui depuis 3.6	Oui
Internet Explorer	Oui	Non
Navigateur Motorola	Non	Oui
Opera Mobile	Oui	Oui
Opera Mini	Oui, sur le serveur	Oui

Analyser du contenu JSON

JSON (*JavaScript Object Notation*) constitue la solution de représentation des données la plus légère pour Ajax. En effet, l'accès aux objets se fonde sur une notation pointée, non, à l'instar de XML, par une analyse du DOM. Le contenu JSON est reçu sous forme de texte *via* `responseText` et est converti en objet avec `eval`. La première interrogation concerne donc le fonctionnement d'`eval` sur les appareils mobiles.

Les tests de prise en charge d'`eval` sur les objets JSON doivent prendre en considération le standard strict et le standard *de facto* employés par la plupart des navigateurs. Le code suivant illustre les différences et le Tableau 9.3 résume leur prise en charge :

```
// Standard strict.
var obj = {
   'name': 'Jean',
   'surname': 'Aimard'
}

// Standard de facto.
var obj = {
   name: 'Jean',
   surname: 'Aimard'
}
```

Tableau 9.3 : Résultats des tests de compatibilité avec l'analyse JSON

Navigateur/plateforme	*eval avec JSON (standards strict et* **de facto***)*
Safari	Oui
Android	Oui
Symbian/S60	Oui
Nokia Series 40	Non avant 6e édition
webOS	Oui
BlackBerry	Non avant 4.6
NetFront	Oui
Internet Explorer	Oui
Navigateur Motorola	Oui
Opera Mobile	Oui
Opera Mini	Oui

INFO

Pour les appareils de bas et de milieu de gamme compatibles Ajax, il est déconseillé de créer plus de deux connexions simultanées au serveur. Le nombre de connexions simultanées doit rester aussi faible que possible.

JSONP et chargement paresseux

JSONP (*JSON with Padding*) est une technique moderne pour accéder au contenu d'un domaine tiers sans être confronté au problème des requêtes Ajax interdomaines. De nombreux services web publics proposent cette nouvelle manière de communiquer avec des serveurs tiers.

JSONP se fonde sur une balise de script générée en JavaScript et placée dans une URL avec un paramètre précisé. Ce paramètre correspond généralement à une fonction de rappel locale invoquée lorsque le script (et les données qu'il récupère) a été téléchargé et exécuté.

Une technique comparable est employée avec les scripts : un sous-ensemble des scripts est initialement téléchargé, les scripts requis ultérieurement étant téléchargés par la suite.

Pour que JSONP puisse être exploité dans un navigateur, il doit être en mesure d'insérer dynamiquement un script depuis du code JavaScript. Lorsque cette opération est prise en charge, le navigateur détecte le nouvel élément `script` ajouté au DOM, puis télécharge et exécute automatiquement cette nouvelle ressource. Puisque ce script appelle la fonction indiquée en lui passant les données collectées, nous pouvons recevoir des données depuis un serveur tiers.

INFO

Les bibliothèques JavaScript récentes, comme jQuery, prennent en charge les requêtes JSONP sans se préoccuper du DOM. Vous pouvez invoquer `$.getJSON` avec un argument pour remplacer une requête Ajax par une requête JSONP.

En général, le serveur tiers dispose d'une URL qui permet d'utiliser le format JSONP. Par exemple :

```
http://api.serveur_tiers.domaine/<script_jsonp>?<rappel>=<notre_fonction>
```

Le serveur renvoie une réponse comparable à la suivante :

```
<notre_fonction>( {<données_json>} );
```

Le code suivant détermine si un navigateur mobile détecte la création dynamique des éléments `script` :

```
function doJSONP() {
   var head = document.getElementsByTagName('head')[0];
   var script = document.createElement('script');
   script.type = 'text/javascript';
   script.src = 'http://mobilexweb.com/tests/jsonp?cb=finished';
   head.appendChild(script);
}
```

Le Tableau 9.4 résume le fonctionnement de ce code sur les différents navigateurs. Certains acceptent un événement `onload` sur le script et l'exécutent lorsque le script est prêt. JSONP n'a pas besoin de cet événement car la fonction de rappel est définie dans la même URL. Toutefois, sur les navigateurs compatibles, il permettra de mettre en place un chargement paresseux.

Tableau 9.4 : Résultats des tests de compatibilité avec le chargement dynamique d'un script

Navigateur/plateforme	*Prise en charge du chargement dynamique d'un script*	*Prise en charge d'onload*
Safari	Oui	Oui
Android	Oui	Oui
Symbian/S60	Oui	Oui
Nokia Series 40	Non avant 6e édition	
webOS	Oui	Oui
BlackBerry	Oui depuis 4.6	Non
NetFront	Oui	Non
Internet Explorer	Non	
Navigateur Motorola	Non	Non
Opera Mobile	Oui	Oui et `onreadystatechange`
Opera Mini	Non	

Comet

Comet est un nouveau modèle d'application web qui se veut une alternative à l'interrogation périodique du serveur de manière à obtenir les dernières informations ou pour effectuer des opérations comparables. Il se fonde sur des requêtes HTTP dont la durée

de vie est longue et qui restent ouvertes jusqu'à ce que le serveur ait une réponse à renvoyer au client.

Par exemple, si nous donnons accès à une boîte de réception du courrier électronique, nous pourrions souhaiter vérifier à intervalles réguliers si de nouveaux messages sont arrivés. En utilisant l'approche Comet, nous pouvons émuler une forme de technologie de push dans laquelle les requêtes effectuées toutes les *x* secondes sont remplacées par une seule requête que le serveur garde ouverte jusqu'à ce qu'il ait des données à envoyer. Dans ce cas, des connexions HTTP peuvent rester ouvertes pendant très longtemps. Il s'agit là d'un exemple d'application de Comet. Il en existe d'autres mais elles sont moins fiables.

INFO

Palm, BlackBerry et Apple offrent des services de push aux développeurs. Autrement dit, vous pouvez envoyer des messages ou du contenu depuis vos serveurs vers leurs serveurs, qui relaieront les informations vers les appareils. Malheureusement, ces fonctionnalités ne sont pas disponibles aux applications web dans les solutions de Palm et d'Apple.

Ces techniques ne sont pas encore conseillées dans les navigateurs mobiles. Le principal problème vient des connexions réseau 3G et 2.5G : même si le serveur les accepte, les passerelles Internet ne sont pas conçues pour des connexions HTTP de longue durée et les proxies les fermeront après un certain temps.

ATTENTION

Sur le serveur, si vous utilisez une connexion suspendue pour Comet, sachez que seuls Symbian 5e édition, Safari Mobile, Windows Mobile 6.5 et Android produiront des résultats prévisibles. Avec les appareils de la famille Series 40 6e édition, ainsi que d'autres, le navigateur se bloque sur la requête et le mobinaute ne peut même plus cliquer sur un lien.

Il existe également des solutions Adobe Flash qui utilisent des sockets ouvertes avec le serveur pour recevoir les dernières informations. Elles ne pourront être envisagées qu'au moment où Flash Player 10.1 sera largement répandu et uniquement *via* des connexions WiFi. Les réseaux 3G ne sont pas suffisamment fiables pour cette utilisation. Il faut également prendre en compte la consommation électrique.

9.2 Bibliothèques JavaScript

La vie du programmeur JavaScript a énormément changé depuis 2006, au moment de l'apparition d'Ajax et des centaines de bibliothèques qui lui ont permis de mieux

travailler avec ce langage. Une grande partie de ces bibliothèques modifie ou ajoute des comportements au langage, créant ainsi de nouveaux langages à l'intérieur (ou au-dessus) de JavaScript.

Si les bibliothèques sont fondées sur JavaScript et si les navigateurs mobiles prennent en charge ce langage, pourquoi y aurait-il un problème ? Tout simplement parce que bon nombre de ces bibliothèques utilisent des fonctionnalités mal définies dans les standards et, si elles ont été conçues pour et testées sur les navigateurs de bureaux répandus (Internet Explorer, Firefox, Safari, Chrome, Opera), elles n'ont pas été vérifiées sur tous les navigateurs mobiles. C'est pourquoi, comme nous l'avons vu, certaines fonctionnalités du DOM, par exemple, sont absentes dans de nombreux navigateurs mobiles.

Voilà la première raison pour laquelle nous devons faire preuve de prudence quant à l'utilisation des bibliothèques JavaScript. La seconde raison, tout aussi importante, concerne leur impact sur les temps de téléchargement et d'exécution. Nous l'avons mentionné précédemment, ces bibliothèques modifient le langage et le comportement des objets. Même si nous n'exploitons qu'une partie du code de la bibliothèque, elle doit être chargée intégralement, ce qui prend du temps. Puisque cela peut conduire à des problèmes de performances avec certains navigateurs, nous allons étudier le temps nécessaire à l'initialisation des bibliothèques classiques.

ATTENTION

Dans leur version complète, certaines bibliothèques occupent plus de 600 Ko. Vous devez faire très attention aux performances lorsque vous les utilisez car elles augmentent le trafic réseau, la consommation mémoire et les temps d'exécution. Si vous le pouvez, évitez les bibliothèques volumineuses ou n'utilisez que la partie de code nécessaire.

Le Tableau 9.5 présente les résultats obtenus avec les bibliothèques jQuery, Prototype, Yahoo! UI et Dojo. N'oubliez pas que les temps d'exécution dépendent énormément du matériel et du processeur. Ces tests veulent uniquement vous faire prendre conscience de l'impact temporel d'une bibliothèque.

Tableau 9.5 : Temps moyen d'exécution/chargement de quelques bibliothèques JavaScript (en secondes)

Navigateur/plateforme	*jQuery*	*Prototype*	*Yahoo! UI*	*Dojo*
Safari	1,8	0,2	0	0,1
Android	4	2,5	0,4	4,6
Symbian/S60	1,7	0,9	0,2	0,7

Tableau 9.5 : Temps moyen d'exécution/chargement de quelques bibliothèques JavaScript (en secondes) *(suite)*

Navigateur/plateforme	*jQuery*	*Prototype*	*Yahoo! UI*	*Dojo*
Nokia Series 40	Impossible à calculer			
webOS	0,2	0,5	0,1	0,4
BlackBerry	6,2	5	0,8	7
NetFront	8	13,6	3,7	11,1
Internet Explorer	2	3	0,4	2
Navigateur Motorola	Incompatible			
Opera Mobile	1,4	0,3	0,1	0,4
Opera Mini	Impossible à calculer, exécution sur le serveur			

En conclusion, évitez autant que possible ces bibliothèques. Dans le cas contraire, employez-les uniquement avec les smartphones et sachez que certains plugins et caractéristiques peuvent ne pas fonctionner correctement.

Bibliothèques mobiles

De nombreux développeurs ont publié des bibliothèques alternatives conçues pour les appareils mobiles et plus légères que les précédentes. Il existe également des frameworks complets pour le développement d'applications mobiles (principalement pour l'iPhone), comme jQTouch, iUI, iWebKit et Webapp.Net ; nous les présenterons plus loin. Ces frameworks s'occupent de l'affichage, des événements et des interactions sur nos sites web.

Vous trouverez également des bibliothèques qui peuvent remplacer jQuery et les autres sur les appareils mobiles. Il s'agit de bibliothèques très légères dont la prise en charge du DOM, des événements et d'Ajax est rudimentaire.

baseJS

baseJS est une bibliothèque légère (8 Ko) compatible avec Safari Mobile et d'autres navigateurs WebKit (**http://paularmstrongdesigns.com/projects/basejs**). Elle a été intégralement testée uniquement sur Safari, d'iOS 1.0 à iOS 3.0.

baseJS propose une fonction de sélection comparable à celle de jQuery, `$(selector)`, ainsi que quelques méthodes équivalentes, comme `each`, `addClass`, `hasClass`, `removeClass`, `toggleClass`, `getXY` ou `fire`, et certaines méthodes Ajax.

XUI

XUI est un framework JavaScript simple pour la construction de sites web mobiles. En version compressée, il occupe uniquement 6,7 Ko. Il est disponible gratuitement sur le site **http://xuijs.com** et a été testé sur les navigateurs WebKit et sur Opera Mobile. Les développeurs travaillent à sa compatibilité avec IE Mobile et les BlackBerry.

XUI ressemble également à jQuery, mais il est plus puissant que baseJS. Il utilise `x$` comme sélecteur principal et fournit les méthodes recensées au Tableau 9.6.

Tableau 9.6 : Méthodes de XUI pour une requête de sélection

Méthode	*Description*
`html(`*`code`*`)` ou `html(`*`emplacement`*`, `*`code`*`)`	Définit le contenu HTML intérieur (ou à un autre emplacement dans la seconde version) des éléments. Le paramètre `emplacement` est une chaîne de caractères à choisir parmi les suivantes : `inner`, `outer`, `top`, `bottom`, `before`, `after` ou `remove`.
`on(`*`événement`*`, `*`fonction`*`)`	Enregistre un gestionnaire d'événements. Le nom de l'événement peut également être utilisé directement comme nom de méthode, par exemple `click` à la place de `on('click')`. Voici les événements reconnus : `click`, `load`, `touchstart`, `touchmove`, `touchend`, `touchcancel`, `gesturestart`, `gesturechange`, `gestureend` et `orientationchange`.
`setStyle(`*`propriété`*`, `*`valeur`*`)`	Définit un style CSS.
`getStyle(`*`propriété`*`, `*`rappel_facultatif`*`)`	Lit la valeur d'une propriété. Si le sélecteur retourne plusieurs éléments, la fonction de rappel est invoquée.
`addClass(`*`nom_de_classe`*`)`	Ajoute une classe aux éléments.
`removeClass(`*`nom_de_classe`*`)`	Retire une classe aux éléments.
`css(`*`objet`*`)`	Définit des styles CSS en utilisant un objet de type JSON qui possède des propriétés et des valeurs.
`tween(`*`objet`*`)`	Anime une ou plusieurs propriétés CSS à partir des valeurs définies dans l'objet.

Nous pouvons par exemple capturer `onclick` sur les boutons à l'aide du code suivant :

```
x$('input.button').on('click', function(e){ alert('Aie !') });
```

Le code suivant, qui enchaîne les méthodes à la manière de jQuery, fonctionne également :

```
x$('input.button').click(function(e){ alert('Aie !') })
                  .html('Cliquez !').css({color: 'blue'});
```

Pour Ajax, XUI apporte les fonctions globales `xhr` et `xhrjson` de manière à créer des requêtes avec des options.

9.3 Extensions CSS de WebKit

Au moment de l'écriture de ces lignes, Safari pour iOS est probablement le navigateur mobile le plus complexe. Nous l'avons indiqué au Chapitre 7, depuis la version 2.0 d'iOS, il prend en charge un ensemble d'extensions CSS intéressantes (et étranges) qui nous permettent de mettre en place sur nos sites web des animations, des transitions et même des effets 3D accélérés matériellement. Certaines de ces extensions sont également disponibles dans les navigateurs d'Android et de webOS, selon la version du système d'exploitation.

Fonctions WebKit

De nombreux attributs CSS prennent une fonction en paramètre. Ces fonctions sont des extensions WebKit et font l'objet d'une accélération matérielle.

ATTENTION

Selon la Safari Reference Library, les fonctions de dégradé mentionnées dans cette section ne sont pas officiellement prises en charge par iOS. Toutefois, elles sont pleinement opérationnelles depuis la version 3.0 et les anciens appareils utilisent simplement un arrière-plan de couleur uniforme.

Les fonctions disponibles aux iPhone sont recensées au Tableau 9.7 (il en existe d'autres, mais elles sont réservées au navigateur Safari de bureau). Certaines d'entre elles, comme `scale` et `rotate`, se retrouvent également dans les navigateurs d'Android et de webOS.