

Le dynamisme du coté client : Javascript et Ajax

Nous voyons ici les technologies qui permettent d'apporter du dynamisme au niveau du client. JavaScript est un langage de script (interprété par le navigateur). Ajax est une démarche mettant en oeuvre Java et permettant de ne réduire la quantité d'information échangée avec le serveur. Ceci nous conduira aussi à présenter DOM, le Document Object Model.

1 Javascript

JavaScript est un langage de scripts essentiellement utilisé pour enrichir les pages HTML de codes exécutables par le navigateur. Attention ! Il n'a guère qu'une partie de son nom en commun avec le langage Java !

Ce langage permet d'apporter du dynamisme au niveau des pages Web. Nous percevons immédiatement qu'on semble parvenir à une contradiction entre le fait qu'un navigateur doive essentiellement réaliser l'affichage de pages et que l'on va lui demander de faire exécuter du code. On veillera donc à limiter au maximum l'utilisation de code JavaScript pour :

- éviter d'alourdir les documents html,
- faciliter leur maintenance,
- ne pas surcharger de traitements les postes de travail.

Ceci est d'autant plus important que les différents navigateurs proposent des extensions personnelles au langage qui, si elles sont utilisées, risquent de limiter la portabilité des pages.

De manière générale, JavaScript est utilisé pour¹ :

- permettre de réaliser des **validations de données** saisies dans des formulaires,
- pour éviter d'augmenter inutilement le nombre de requêtes soumises au serveur Web,
- pour créer les cookies², ...
- Il peut aussi permettre d'enrichir la présentation de la page (menus dynamiques par exemple).

1.1 Syntaxe de Javascript

Comme à chaque fois, nous ne présentons que les aspects essentiels du langage et vous renvoyons sur le site de la W3School pour un tutoriel plus complet (<http://www.w3schools.com/js/default.asp>)

Position

Un script JavaScript devra apparaître entre les balises :

- 1 Il peut aussi être utilisé pour faire clignoter une guirlande, faire traverser l'écran par un canard ou demander à un poisson de suivre votre souris (comme quoi il faut savoir ne pas abuser des bonnes choses...)
- 2 *Cookie*: petit fichier texte, envoyé au navigateur par le serveur Web ou généré par du code JavaScript et stocké par le navigateur sur la machine cliente. Ces fichiers permettent aux serveurs de retrouver des informations sur l'utilisateur d'une visite à l'autre.

```
<script type="text/javascript">
<!--
    code du script
//-->
</script>
```

Un tel code peut apparaître soit dans le header de la page soit dans le body. Si le script est dans le body, il sera exécuté au chargement de la page. *Les scripts devant s'exécuter à la survenance d'événements devront être placés dans le header.*

Remarques :

- Les `<!--` et `-->` sont là pour que les anciens browsers ne prennent pas le code en compte.
- Le `//` est nécessaire pour que l'interpréteur JavaScript ne prenne pas `-->` en tant que code : (`//` permet de placer une ligne en commentaire)

On peut également (c'est plus propre) faire appel à des scripts externes en fournissant :

```
<script src="nomduscript.js" />
```

Un exemple simple

Plaçons ceci dans le body d'un de nos précédents documents Html (nous avons omis le code nécessaire à la compatibilité avec les anciens navigateurs) :

```
<script type="text/javascript">
document.write("Mon premier code JavaScript génère ceci ! ")
document.write("<b>Heb-Esi</b> Applications distribuées ")
</script>
```

Nous pouvons aussi sauvegarder ce script à part dans un fichier **MonScript.js** placé dans le même répertoire que la page html. Nous pourrons le référencer comme suit :

```
<script src="MonScript.js" />
```

Un exemple de validation

Un cas classique est celui de la validation d'un champ de saisie devant être numérique. Nous allons réaliser la validation lors de la perte du focus¹ par le contrôle.

```
<script type="text/javascript">
function validateNumeric(champ) {
    with (champ) {
        if ( isFinite(value) ) {
            return true
        }
        else {
            alert("Vous ne pouvez entrer que des caractères numériques!")
            return false
        }
    }
}
</script>
...
<form name="InfoPersonnelle">
<b> Age : </b> <input type="text" name="saisie" size=3 onblur="validateNumeric(this)">
</form>
```

Remarque : si nous désirons réaliser une validation lors de l'envoi du formulaire nous serons obligés d'écrire un script spécifique au formulaire.

¹ Voir http://www.w3schools.com/tags/ref_eventattributes.asp pour une liste des événements gérés.

Un exemple de création de cookie

Vous trouverez sur http://www.w3schools.com/js/tryit.asp?filename=tryjs_cookie_username un exemple de gestion de cookie en Javascript.

Varia

Voici quelques éléments supplémentaires de syntaxe :

- Les instructions sont délimitées par le point-virgule mais celui-ci n'est obligatoire que si l'on désire reprendre plusieurs instructions sur la même ligne.
- Les commentaires sont repris, sur une ligne derrière les caractères //
- Les noms de variables sont sensibles à la casse.
- Les instructions de contrôle de séquence (if, while, ...) sont sensiblement les mêmes qu'en Java.

1.2 Exercice

Écrivez un document html présentant un formulaire demandant de saisir un n° d'étudiant, un nom, un prénom, une année, une section, un sexe et un email.

Veillez à ce que le n° d'étudiant soit numérique (validation à la sortie du champ) et que tous les champs, sauf l'email, soient remplis, lors du submit. La page appelée par le formulaire doit présenter le nom saisi (cette dernière exigence vous obligera à travailler de manière aberrante dans la mesure où nous ne réalisons encore aucun traitement du côté serveur).

2 Manipuler un document HTML grâce à DOM

DOM (Document Object Model)¹

Le DOM, ou modèle objet de document, est une API pour les documents HTML et XML. Le DOM fournit une représentation structurelle du document, permettant de modifier son contenu et sa présentation visuelle. Fondamentalement, il relie les pages Web aux scripts et langages de programmation.

Concrètement, cela signifie qu'au niveau de javascript on dispose d'une représentation du document HTML sous forme d'un « arbre », chaque « noeud » correspondant à une balise du document. Cet arbre peut être parcouru mais surtout **modifié**.

2.1 Quelques éléments de syntaxe

Nous ne présentons que la toute petite partie qui sert à notre propos. Nous en dirons un peu plus lorsque nous introduirons les technologies XSL et, comme d'habitude, vous trouverez plus d'informations sur le site de la W3 school (<http://www.w3schools.com/html/dom/default.asp>)².

- L'objet **document** représente la racine de l'arbre.
- La méthode **getElementById**(« id ») donne le noeud correspondant à cet id unique (attribut « id » de la balise).
- La méthode **getElementsByTagName**(« tag ») donne la liste des noeuds porteurs de ce tag.
- L'attribut **innerHTML** correspond au code HTML inclus dans un noeud.

¹ Tiré de http://developer.mozilla.org/fr/docs/%C3%80_propos_du_Document_Object_Model

² Voyez aussi <http://code.google.com/edu/client/samples/dhtmltest.html> pour des exemples pratiques.

- Le signe = correspond à l'assignation.
- On peut aussi, à partir d'un noeud, demander un tableau avec tous les fils ou remonter au père.

Exemple récapitulatif

Si le document contient

```
<span id="titre"></span>
```

alors le bout de code suivant insère un titre.

```
document.getElementById(« titre »).innerHTML = « Application Distribuée »
```

Autre exemple

Le code suivant affiche un message avec le nombre de paragraphes.

```
function comptePara() {
  var mesParagraphes = document.getElementsByTagName("p");
  alert( mesParagraphes.length )
}
```

Pour qu'il soit exécuté au chargement de la page, on l'indique comme attribut de la balise **<body>**.

```
<body onload="comptePara();">...
```

2.2 Exercices

1. Faire apparaître dans le message le nom du premier formulaire présent dans le document.
2. Vous avez écrit une page proposant un formulaire étudiant à remplir. Pour le moment une alerte est affichée lorsqu'un champ est invalide. Apportez les modifications nécessaires pour que un message adapté apparaisse (en rouge) à côté de chaque champ invalide.
3. Écrivez une page html présentant une question pour laquelle le répondant a le choix entre 5 possibilités et s'il choisit la 5ème (autre) une zone de saisie de texte apparaît (pensez à l'attribut **type** de la balise **input**)

Quelle couleur préférez-vous?

<input type="radio"/> rouge	<input type="radio"/> noir
<input type="radio"/> vert	<input type="radio"/> autre
<input type="radio"/> bleu	

3 Ajax – Asynchronous JavaScript And XML

Ajax n'est pas spécifiquement un langage ou une technologie particulière mais une démarche permettant de diminuer la limitation imposée par http et html aux applications Web.

La lacune essentielle des applications Web était leur manque d'interactivité : toute modification de la page affichée par le navigateur imposait un réaffichage complet. Nous avons déjà aperçu que JavaScript permet d'assouplir ce fait (les applets encore plus), mais les actions apparaissant dans le navigateur sont réalisées sans aucun contact avec le serveur. Si, par exemple, nous voulons afficher des informations à la demande de l'utilisateur, soit l'ensemble des infos doit être chargé initialement (transfert prohibitif) soit la page complète doit être rechargée à chaque action !

Ajax va nous permettre de nous libérer de ces contraintes ! Attention, à nouveau, comme pour JavaScript, il s'agira de ne pas abuser de cette démarche pour ne pas rendre nos pages trop complexes à concevoir et surtout à maintenir.

3.1 Préambule

Ajax met en œuvre des requêtes http au travers de l'objet JavaScript XMLHttpRequest. Malheureusement, les navigateurs permettant de manipuler ce type d'objet proposent des syntaxes différentes d'instanciation ! Lorsque nous désirerons utiliser AJAX, nous aurons à prévoir un code semblable¹ à celui-ci qui crée et retourne un objet **XMLHttpRequest** en fonction du navigateur :

```
function getXmlHttpRequestObject() {
    var xmlhttp;
    try { // Firefox, Opera 8.0+, Safari
        xmlhttp=new XMLHttpRequest();
    } catch (e) { // Internet Explorer
        try {
            xmlhttp=new ActiveXObject("Msxml2.XMLHTTP");
        } catch (e) {
            try {
                xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
            } catch (e) {
                xmlhttp=null;
            }
        }
    }
    return xmlhttp; // null si navigateur incompatible
}
```

3.2 Une requête

L'objet **XMLHttpRequest** permet d'envoyer une requête à un serveur qui retournera la réponse sous forme d'un objet (cf. DOM). En pratique, on spécifie une URL et le serveur retourne un bout de code HTML ou du XML (cf. leçons suivantes). La réponse peut alors être manipulée par le code Javascript (pour être insérée dans le document en cours par exemple).

Ajax fonctionne de préférence en mode **asynchrone** :

- D'abord on envoie la requête en spécifiant une méthode à appeler lorsque la réponse sera prête.
- Quand la page est prête cela déclenche l'exécution de la méthode spécifiée.

```
...
xmlhttp.onreadystatechange=stateChanged; // stateChanged sera appelée quand la réponse sera prête
xmlhttp.open("GET", « adresse.html », true ); // Prépare une requête GET asynchrone pour l'URL indiquée
xmlhttp.send( null ); // L'envoi au serveur (null pour « localhost »)
...

function stateChanged() {
    if (xmlhttp.readyState==4) { // la réponse est bien prête (pas d'erreur)
        xmlhttp.responseText; // contient la réponse
    }
}
```

3.3 Exemple

Actuellement nous ne travaillons pas encore avec de serveur Web et donc pour pouvoir percevoir les avantages d'AJAX nous allons accéder à des fichiers repris dans le même dossier que le document html.

Imaginons un exemple - tiré par les cheveux - dans lequel une page web nous présente un formulaire avec une combobox nous permettant de sélectionner un article à présenter. Sans Ajax, à chaque sélection faite au niveau de la combo, une nouvelle page complète est demandée au serveur

¹ A placer par exemple dans un fichier « ajax.js » pour pouvoir y faire référence dans les documents HTML.

et une page complète, analogue à la précédente, est transférée et affichée ! De plus, chacune de ces pages doit être préécrite ou générée par le serveur.

Avec Ajax nous allons pouvoir nous contenter de requérir du serveur la partie de page à modifier et le code JavaScript l’affichera au bon endroit.

```
<script type="text/javascript">
var xmlhttp

function presenteLivres( str ) {
    xmlhttp = getXmlHttpRequestObject();
    if (xmlhttp==null) {
        alert ("Your browser does not support AJAX!");
        return;
    }
    if ( str != null ){
        var url=str+".html";
        xmlhttp.onreadystatechange=stateChanged;
        xmlhttp.open("GET",url,true);
        xmlhttp.send(null);
    }
    else {
        document.getElementById("Description").innerHTML="";
    }
}

function stateChanged() {
    if (xmlhttp.readyState==4) { // la réponse est bien prête (pas d'erreur)
        document.getElementById("Description").innerHTML=xmlhttp.responseText;
    }
}
</script>
```

Remarque : en pratique la requête ajax serait faite au serveur en déclenchant un traitement d’extraction des données désirées à partir d’un gestionnaire de données persistantes (BD, fichier XML, ERP, ...)

3.4 Avantages / inconvénients

Ajax est une technologie qui a le vent en poupe (les sites Google sont basés là-dessus par exemple). Ajax permet de fluidifier les échanges navigateur-serveur et de rendre l’interface utilisateur moins rigide (client « riche »). Il faut toutefois rester attentifs à certains points :

- Attention à préserver la fonctionnalité du bouton ‘Back’ du navigateur.
- Les robots ne référenceront pas les données obtenues au travers d’Ajax.
- A priori, pas de visualisation du fait que des données sont en chargement.
- Il faut assurer la sécurité des requêtes via Ajax aussi sérieusement que les requêtes « classiques ».
- Un abus de code risque de ralentir l’application.
- Ne pas oublier que certains navigateurs ne sont pas compatibles Ajax.
- Attention de ne pas sortir des conventions usuelles d’interface graphique. Cela risquerait de désorienter l’utilisateur ou pire de lui faire réaliser des choses qu’il ne voulait pas faire
- La notion de lien perd de son sens : fournir un lien à quelqu’un donne accès à la page mais pas nécessairement à l’info que l’on désire faire visualiser.

3.5 Exercice

Écrivez un formulaire de sélection d’un employé dans lequel une sélection d’un département fournit dans une deuxième combobox la liste des employés du département.