Programmation avancée pour le Web

JavaScript+JQuery+AJAX

Réaliser par : Mr Chouha Adel
E-mail : adch05@yahoo.fr

JavaScript

Définition

JavaScript est une extension du langage **HTML** qui est incluse dans le code. Ce langage est un langage de programmation qui permet d'apporter des améliorations au langage HTML en permettant d'exécuter des commandes.

JavaScript a été initialement développé par Netscape et s'appelait à l'époque ***LiveScript***. Adopté à la fin de l'année 1995, par la firme Sun (qui a aussi développé Java), il prit alors son nom actuel de JavaScript.

Les scripts **JavaScript** sont gérés et exécutés par le browser lui-même sans devoir faire appel aux ressources du serveur. Ses instructions seront donc traitées en direct et surtout sans retard par le navigateur.

Définition

La balise **<script>** informe le navigateur que c'est le début d'un script. Et pour préciser que c'est du **JavaScript** il faut ajouter l'attribut **language=''javascript".**

La balise devient alors

<script language=''javascript">

La balise **</script>** informe de la fin du script **JavaScript**.

Il est toute fois possible de préciser au navigateur la version JavaScript à exécuter en ajoutant son identificateur ainsi:

<script language=''javascript1.2">

C'est la version 1.2 que doit exécuter le navigateur dans ce cas.

Masquage d'un script JavaScript?

Les anciens navigateurs ne reconnaissent pas le JavaScript. Alors au lieu d'exécuter le script ils affichent votre code JavaScript comme si c'était du texte HTML.

Pour remédier à ce problème il faut placer les scripts entre les tags de commentaire **HTML**, à savoir **<!-- et -->**. Dans ce cas les anciens navigateurs laisseront aller les script sans pouvoir les exécuter (ce qui n'est, sûrement, pas désiré) mais aussi sans pourvoir les afficher.

Ainsi l'insertion du JavaScript ressemblerait à cela:

```
<script language="'javascript"'>
    <!--
    Votre script
    // -->
</script>
```


Caractéristiques de JavaScript

JavaScript est un langage de programmation (ou de script) coté client. C'est-à-dire que c'est le navigateur qui se charge de l'exécuter tout comme le code HTML.

On appelle souvent le langage **JavaScript** un langage **événementiel**. En effet la plupart de ses scripts sont associés à des événements qui peuvent se produire sur le navigateur tel que le chargement de la page, la fermeture de la page, le clic, le survol, la sélection, la frappe au clavier…

JavaScript est généralement utilisé pour contrôler les formulaires avant envoi au lieu d'attribuer ce travail à un langage coté serveur tel que le PHP.

Les commentaires

Les commentaires en **JavaScript** sont semblables à ceux du langage C. En effet on utilise les doubles slash **(//)** pour un commentaire de fin de ligne. Tandis **que /* et */** délimitent les commentaires sur plusieurs lignes.

Les commentaires sont vivement recommandés dans un programme quelconque pour faciliter sa lisibilité.

Variables et opérateurs

En **JavaScript** un nom de variable doit commencer par une lettre ou le signe « _ » et se composer de lettres, de chiffres et des caractères **_** et **$** (à l'exclusion du blanc). Le nombre de caractères n'est pas précisé.

JavaScript fait la différence entre majuscules et minuscules.

Le **typage** des variable n'est pas obligatoire en **JavaScript**. Ainsi on peut changer le type d'une variable plusieurs fois lors du même script.

Pour déclarer une variable on utilise soit le mot clé « **var** ». On parle alors d'une déclaration **explicite**: ***var*** **a=20;**

Comme on peut directement initier la variable sans le mot clé « **var** » (déclaration **implicite**): **a=20;**

Les opérateurs en **JavaScript** sont les même qu'en langage **C**:

(+ - * / % < > <= >= == != ++ -- += -= *= /= ( ) && ‖)

Les chaînes de caractère et structures conditionnelles

Une chaîne de caractère (**string**) est une suite de caractères délimités par des guillemets (simples ou doubles).

On signale que les chaînes de caractères sont limitées théoriquement à une longueur comprise entre 50 et 80 caractères.

L'objet **string** possède plusieurs propriétés.

Les structures conditionnelles en **JavaScript** sont les même qu'en langage **C**, à savoir:

- **If … else**
- **Boucle for**
- **Boucle while**

Les mots clé **break** et **continue** conservent toujours leurs fonctions connues en JavaScript.

Les fonctions

En **JavaScript** les fonctions sont déclarées et appelées de la même façon qu'en **C**.

Il est courent que toutes les fonctions sont déclarées à l'entête du document, c'est-à-dire entre <**head**> et </**head**>.

L'appel de la fonction peut s'effectuer n'importe où dans le document à condition que la déclaration doit être faite en premier.

Délaration:

```
Function identificateur_de_la_fonction(liste_des_arguments)
{
Liste des instructions;
Return (valeur_de_retour);
}
```

Les fonctions

L'identificateur de la fonction suit les même règles que l'identificateur de la variable:

Les **arguments** de la fonction sont **facultatifs** cependant les **parenthèses** sont **obligatoires** même vides.

Le mot clé **return** permet de passer le résultat de la fonction dans le reste du **script**. Il est facultatif.

D'habitude les fonctions sont déclarées à l'entête du document et sont souvent appelés par les événements.

Les événements

JavaScript est un langage de programmation appelé aussi langage **événementiel** vu que la majorité de ses scripts sont associés à des événements qui se produisent sur le navigateur.

Les événements sont des **actions** de l'uitilsateur qui vont pouvoir donner lieu à une **interactivité**. L' événement le plus utilisé est le clic de la souris, car c'est le seul que le HTML gère.

Grâce à **JavaScript** il est possible d'associer des fonctions ou des méthodes à des événements tel que le passage de la souris au dessus d'une zone, le changement d'une valeur…

Ce sont les **gestionnaires d'événement** qui permettent d'associer une action à un événement.

La syntaxe d'un gestionnaire d'événement est la suivante:

onEvenement="action_JavaScript_ou_fonction()";

Les événements

Liste des événements:

- **Click**: se produit lorsque l'utilisateur clique sur un élément associé à l'événement.
- **Load**: se produit lorsque le navigateur de l'utilisateur charge la page en cours.
- **Unload**: se produit lorsque le navigateur de l'utilisateur quitte la page en cours.
- **MouseOver**: c'est lorsque l'utilisateur survole (met le curseur de la souris sur) l'élément associé à l'événement.
- **MouseOut**: c'est lorsque le curseur de la souris quitte l'élément.
- **Focus**: se produit lorsque l'utilisateur donne le focus à un élément, c'est-à-dire que l'élément devient actif.
- **Blur**: c'est lorsque l'élément perd le focus (il était actif et il ne l'est plus, c'est à ce moment que cet événement est déclanché).
- **Change**: se produit lorsque l'utilisateur modifie le contenu d'un champs donné (souvent une liste de sélection).

Les événements

Liste des événements:

- **Select**: se produit lorsque l'utilisateur sélectionne un texte (ou une partie d'un texte) dans un champs de type 'text' ou 'textarea'.
- **Submit**: se produit lorsque l'utilisateur clique sur le bouton de la soumission d'un formulaire (bouton 'submit').
- **KeyDown**: se produit lorsque l'utilisateur enfonce une touche du clavier.
- **KeyUp**: c'est lorsque l'utilisateur relâche une touche du clavier.
- **DblClick**: se produit lorsque l'utilisateur clique deux (double clique) fois sur un élément du navigateur.
- **Resize**: quand l'utilisateur redimensionne le navigateur.

Les événements

Chaque événement ne peut pas être associé à n'importe quel objet. Par exemple on ne pas appliquer l'événement **onChange** à un lien hypertexte.

Ci-dessous un résumé des évènements qui peuvent être associés aux objets les plus courants:

Lien hypertexte: onClick, onMouseOver, onMouseOut

Page du navigateur: onLoad, onUnload

Boutons: onClick

Liste de selection: onBlur, onFocus, onChange

Champs de texte: onBlur, onFocus, onChange, onSelect, onKeyUp, onKeyDown

Images: onMoseOver, onMouseOut

Objets et méthodes

JavaScript repose sur les notions de **Objet** et **Méthodes**. En effet, **JavaScript** peut exploiter les objets du navigateurs selon les méthodes définies.

Les objets sont classés **hiérarchiquement**, c'est-à-dire que pour accéder à un objet, il faut d'abord passer par l'objet parent.

Pour voir cela de plus prés on vas considérer que nous avons le code **HTML** suivant:

```
<html>
    <head></head>
    <body>
        <form name=''mon_form''>
            <input type=''text'' name=''mon_text''>
        </form>
    </body>
</html>
```

Objets et méthodes

Avec JavaScript on peut accéder à la valeur du champ texte nommé « **mon_text** ». Cependant on ne peut pas y accéder directement car comme on peut le voir nous avons la hiérarchie suivante:

L'objet « **text** » se trouve dans l'objet « **form** » donc, pour arriver au texte on va écrire **mon_form.mon_text** (*l'objet parent.l'objet cible*).

L'objet « **form** » se trouve dans l'objet « **document** ». L'objet document étant le corps du navigateur. Alors l'expression devient: **document.mon_form.mon_text**

L'objet « **document** » se trouve dans l'objet « **window** ». L'objet **window** (ou fenêtre) est l'objet le plus haut en hiérarchie. C'est à partir de là que commence l'appel aux objets.

L'expression précédente redevient: **window.document.mon_form.mon_text**

Objets et méthodes

Puisque le premier objet qui peut être rencontré dans la hiérarchie est l'objet « **window** » alors on peut toujours ne pas le mentionner lors de l'appel hiérarchique aux objets; il est appelé implicitement. La dernière expression devient alors: **document.mon_form.mon_text**

On veut maintenant accéder à la valeur du champs texte. On doit alors mentionner la **méthode** (ou la propriété) à la fin de la hiérarchie. On aura donc: **document.mon_form.mon_text.value**

Pour mieux exploiter cette valeur on peut affecter la chaîne hiérarchique à une variable qu'on nomme par exemple «Mon_text»

Mon_text= document.mon_form.mon_text.value

Objets et méthodes

En général, chaque objet en **JavaScript** possède ses propres méthodes (ou propriétés) commençant de l'objet fenêtre (le plus haut de la hiérarchie) et arrivant jusqu'aux moindres objets de formulaires et constituants du navigateur.

Une méthode déjà très populaires auprès des débutants; c'est la méthode **Alert().**

Alert() est une méthode de l'objet **window**. On peut donc l'écrire de deux façons différentes:

Window.alert(''chaîne_de_caractères ou variable'')

ou bien tout simplement **alert(''chaîne de caractère ou variable'')** (l'objet window étant facultatif puisqu'il est implicite).

Alert permet d'afficher le message en paramètres dans une boite de dialogue (**objet window**).

L'objet Window

L'objet **window** possèdes plusieurs séries de méthodes: **boites de dialogue, minuties, ouverture et fermeture des fenêtres…**

Les boites de dialogues

- La méthode **alert().**
- La méthode **confirm()** qui affiche une boite de dialogue de confirmation qui donne la possibilité de choisir entre '**OK**' ou '**Annuler**'. Cette méthode est très utile pour confirmer les actions de l'utilisateur sur le navigateur.
- La méthode **prompt()** qui affiche une boite de dialogue avec une zone de texte qui peut être remplie par l'utilisateur. La valeur de **prompt()** est ensuite transmise au navigateur en cliquant sur '**ok**'.

L'objet Window

La minuterie

À l'aide de **JavaScript** on peut programmer une minuterie (ou compteur à rebours) qui permettra de déclencher une fonction une fois le temps remonté est écoulé.

La **syntaxe** de la **minuterie** est la suivante:

Compteur=setTimeout(''fonction()", Durée_en_milliseconde)

À la fin de la durée spécifiée en milliseconde la fonction sera exécutée.

La variable '**Compteur**' est l'identificateur du compteur qui permet d'arrêter prématurément celui-là à l'aide de la fonction:

ClearTimeout(Compteur)

L'objet Window

Ouverture et fermeture des fenêtres :

JavaScript propose des méthodes pour l'objet **window** qui permettent d'ouvrir ou de fermer des fenêtres.

La méthode **open()** permet d'ouvrir une nouvelle fenêtre. La syntaxe est la suivante:

Open(''URL","nom_de_la_page","options_de_la_fenêtre")

Options	**Description**
Location=yes/no	Affiche ou non la barre d'adresse
Menubar=yes/no	Affiche ou non la barre de menu
Resizable=yes/no	Permet ou non le redimensionnement de la page
Scrollbar=yes/no	Affiche ou non les ascenseurs
Status=yes/no	Affiche ou non la barre d'état
Toolbar=yes/no	Affiche ou non la barre d'outils
Width=largeur en px	Largeur de la fenêtre
Height=hauteur en px	Hauteur de la fenêtre

L'objet Window

Ouverture et fermeture des fenêtres:

Les options doivent être saisies l'une après l'autre, séparées par des virgules et sans espace. L'ensemble des options doit être encadré par des guillemets.

La méthode **Close()** de l'objet '**Window**' est utilisée pour fermer une fenêtre. Le plus souvent c'est de fermer la fenêtre en cours en utilisant le mot clé '**self**'.

Self renvoie la fenêtre en cours et la syntaxe devient:

self.close(); // Ceci ferme la fenêtre en cours.

Cependant la fermeture d'une fenêtre est souvent associée à un événement. Soit l'événement **onClick** et on aura:

onClick='self.close()";

L'objet String

On signale une limitation théorique de la longueur des strings (chaînes de caractères) à 50 voir 80 caractères. Cette limitation du compilateur **JavaScript** peut toujours être contournée par l'emploi de signes **'+'** de la concaténation.

L'objet **string** possède un ensemble de propriétés et méthodes qui peuvent s'avérer utiles surtout au niveau du traitement des formulaires. En voici les plus importantes:

length: Entier qui indique la longueur de la chaîne de caractères.

charAt(): Méthode qui permet d'accéder à un caractère de la chaîne.

indexOf(): Méthode qui renvoie la position d'une sous-chaîne si elle existe dans la chaîne principale commençant par une position donnée.

substring(): Méthode qui renvoie une sous chaîne commençant d'une position donnée et finissant par une position donnée.

toLowerCase(): Conversion de la chaîne en minuscules.

toUpperCase(): Conversion de la chaîne en majuscules.

L'objet Math

L'objet **math** possède de nombreuses méthodes qui sont en fait des fonctions mathématiques connues:

Math.abs(x) renvoie la valeur absolue de x

Math.ceil(x) renvoie l'arrondi supérieur (entier)

Math.floor(x) renvoie l'arrondi inférieur (entier)

Math.round(x) renvoie l'arrondi le plus proche (entier)

Math.max(x,y) renvoie le plus grand des deux nombres x et y

Math.min(x,y) renvoie le plus petit des deux nombres x et y

Math.pow(x,y) renvoie x à la puissance y

Math.sqrt(x) renvoie la racine carré de x

On trouve aussi: **Math.sin(x), Math.asin(x), Math.cos(x), Math.acos(x), Math.tan(x), Math.atan(x), math.E, Math.PI, Math.log(x), math.exp(y)…**

L'objet Date

JavaScript permet d'évaluer les dates et heures locales (de la machine du client).

La méthode ***new*** **Date()** renvoie toutes les informations « **date** et **heure** » de l'ordinateur de l'utilisateur sous la forme:

Mon Mar 17 10:58:09 UTC 2008

jour_abrégé Mois_abrégé date_du_jour heure base_du_temps_civil Année

Il existe cependant une multitude de méthodes qui permettent de mieux extraire les informations du temps local pour les afficher selon nos choix.

Sois la variable **Temp=*new* Date();**

Temp.getYear(): retourne l'année courante (sur deux chiffres pour les anciens version de JavaScript).

Temp.getMonth(): retourne un entier désignant le mois compris entre 0 et 11 (0 désigne janvier).

L'objet Date

Temp.getDate(): retourne la date du jour comprise entre 1 et 31.

Temp.getDay(): retourne un entier désignant le jour compris entre 0 et 6 (0 désigne Dimanche).

Temp.getHours(): retourne un entier compris entre 0 et 23 désignant l'heure locale

Temp.getMinutes(): retourne un entier compris entre 0 et 59 désignant les minutes

Temp.getSeconds(): retourne un entier compris entre 0 et 59 désignant les secondes

L'objet Array

L'objet **Array** (ou tableau) est une liste d'éléments indexés. De point de vu **JavaScript** (comme en C) un tableau est une variable qui peut contenir plusieurs valeurs qu'on identifie par des indexes.

La méthode ***new*** **Array()** Permet de déclarer un tableau. Elle procède comme paramètre le nombre d'éléments qu'on veut indexer. Les tableaux en **JavaScript** peuvent contenir des éléments de types différents.

Exemple

```
Tab=new Array(3);
Tab[0]=''Marrakech";
Tab[1]=15;
```

L'appel d'un élément est effectué à l'aide de son index:

```
Document.write(''Le deuxième élément est: ''+tab[1]);
```

L'indexation des éléments commence de 0.

L'objet Array

Il existe cependant une autre manière pour initier un tableau. C'est l'initialisation directe:

Tab=[''Lundi","Mardi","Mercredi"];

Dans ce cas on a pas besoin de déclarer l'objet **Array** comme on n'a pas besoin de déterminer le nombre d'éléments indexés.

Méthode agissant sur l'objet array()

length: retourne le nombre d'éléments du tableaux.

join(): retourne une chaîne de caractère constituée des éléments du tableaux et séparés par la chaîne fournie en paramètres.

reverse(): inverse l'ordre d'indexation des éléments.

sort(): tri alphabétique ou numérique des éléments (les éléments dans ce cas doivent être de même type).

L'objet Screen

L'objet **screen** permet de récupérer les propriétés de l'écran chez le client. Les propriétés de cet objet sont les suivantes:

Width: Retourne la largeur de l'écran (résolution horizontale) chez le client en pixels.

Height: Retourne la hauteur de l'écran (résolution verticale) chez le client en pixels.

availWidth: Retourne la largeur de l'écran maximale disponible pour accueillir la fenêtre de navigation (La barre d'état par exemple n'en fait pas partie).

availHeight: Retourne la hauteur de l'écran maximale disponible pour accueillir la fenêtre de navigation.

colorDepth: Permet de récupérer la définition de la palette de couleur chez le client (Nombre de couleurs utilisés: 8bits=256 couleurs, 16 bits = 65536 couleurs…).

L'objet Document

L'objet document se réfère au contenu affiché dans la fenêtre du navigateur. Dans la hiérarchie objet de **JavaScript** il se trouve sous l'objet **window**.

Il existe de nombreuses propriétés, méthodes et sous-objets pour l'objet document mais elles ne sont pas tous fréquemment utilisés.

write(): méthode qui permet d'écrire dans le document.

writeln(): méthode qui permet d'écrire dans le document avec un retour chariot (écriture ligne par ligne).

getElementById(): méthode DOM (Document Object Model) qui permet l'accès aux éléments par l'attribut id.

L'objet Document

getElementByName(): méthode DOM qui permet l'accès aux éléments par l'attribut name.

images: sous objet de l'objet document qui permet d'avoir l'accès aux graphiques (images) du fichier HTML.

cookie: propriété qui permet de déposer ou récupérer un cookie chez le client. Les cookies sont des fichiers que des sites web déposent chez le client en général pour sauvegarder des préférences.

lastModified: propriété qui permet de récupérer la date de la dernière modification de la page en question.

L'objet Document

Sous objet images

Le sous objet **images** est placé hiérarchiquement en dessous de l'objet **document**. Il permet le plus souvent réussir un survol d'image; c'est-à-dire importer une nouvelle image en survolant un élément image dans le document HTML.

Exemple

document.images['image_name'].src='images/im.jpg';

Cette instruction permet de changer l'image dont l'attribut **name** vaut la valeur ***image_name*** par l'image répertoriée sur le chemin relatif ***images/im.jpg***

La bibliothèque JQuery

Introduction

- Une bibliothèque **javascript** open-source
- Elle permet de traverser et manipuler très facilement l'arbre **DOM** des pages web à l'aide d'une syntaxe fortement similaire à celle d'XPath.
- **JQuery** permet par exemple de changer/ajouter une classe **CSS**, créer des animations, modifier des attributs, etc.
- Gérer les événements **javascript**
- Faire des requêtes **AJAX** simplement

Introduction

- **jQuery** est très puissant et très pratique, mais vous devez néanmoins connaitre les bases de **Javascript**, notamment la partie “objet” du langage.
- Disponible sur le site de Jquery

  http://jquery.com/

  <script type="text/javascript" src="jquery.js"></script>

  Ou directement sur Google code

  <script type="text/javascript"

  src="http://ajax.googleapis.com/ajax/libs/jquery/1/jquery.min.js">

  </script>

La fonction jQuery()

- **jQuery** repose sur une seule fonction :

  jQuery() ou $().
- C'est une fonction **JavaScript**, Elle accepte des paramètres et retourne un objet
- $ accepte un ***sélecteur CSS*** en argument
- $ accepte des ID :
  - $('#nomID') retourne **un** élément <-> document.getElementById
- $ accepte des classes :
  - $('.nomClasse') retourne tous les éléments qui correspondent à cette classe
- $ accepte plusieurs sélecteurs
  - $('.article, .nouvelles, .edito')

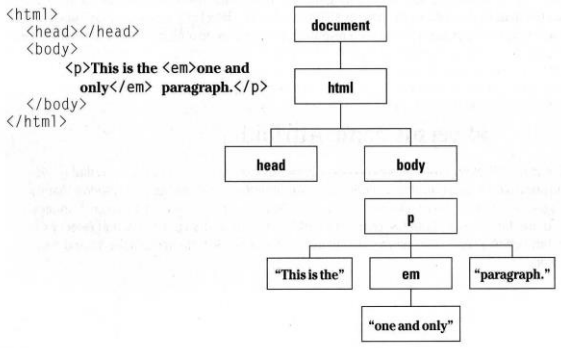
La fonction jQuery()

- **$** accepte des sélecteurs spécifiques :
  - $(':radio'), $(':header'), $(':first-child')
- des sélecteurs en forme de filtres :
  - $(':checked'), $(':odd'), $(':visible')
  - plus fort: $(':contains(du texte)')
- des attributs
  - $('a[href]'), $('a[href^=http://'), $('img[src$=.png]'

Le DOM

- "Le Document Object Model (**DOM**) permet de représenter des objets sous forme d'un arbre dans des documents en HTML, XHTML ou XML.

A simple HTML document node tree.



Exemple d'utilisation des DOM

```
<html>
<script type="text/javascript"
   src="http://ajax.googleapis.com/ajax/libs/jquery/1/jquery.min.js">
</script>
<body>
<div id="monDiv">Bonjour</div>
<a href="#" onClick="$('#monDiv').hide();">
disparition</a>
</body>
</html>
```

jQuery: un objet

- les méthodes s'appliquent généralement à tous les éléments sélectionnés
  - $('.classe').hide();
  - $('.classe').show();
- de nombreuses méthodes utilitaires
  - parcourir le DOM: .parent(), .next(), .children(), .parents()
  - ajouter ou retirer des classes CSS: addClass, removeClass
  - manipuler: append, wrap, prepend
- Intérêt fondamental: la plupart des méthodes de l'objet retournent l'objet lui-même
  - on peut chaîner les appels !
  - $('anything').parent().find('still anything').show();

Example d'utilisation

Determiner si une checkbox est cochée

```
If ($('#total').attr('checked')) {
        //Traitement si cochée
}
else {
        //Traitement si non cochée
}
```

Example d'utilisation

Intercepter le bouton submit d'un formulaire :

```
$(document).ready(function() {
        $('#ok').submit(function() {
                if ($('#login').val() ==‘’) {
                        alert ('Entrer un login');
                        return false;
                }
        })
});
```


Introduction à AJAX

Définition

- AJAX = Asynchronous JavaScript And XML
- Méthode informatique de développement d'applications Web.
- Utilisation ***conjointe*** d'un ensemble de technologies couramment utilisées sur le Web :
  - ♦ HTML (ou XHTML) et CSS pour la mise en forme
  - ♦ DOM et JavaScript pour afficher et interagir dynamiquement avec l'information présentée
  - ♦ XML, XSLT et l'objet XMLHttpRequest pour échanger et manipuler les données de manière asynchrone avec le serveur web

Application WEB traditionnelle :

- Le client envoie une requête HTTP
- Le serveur renvoie une page
- Cela consomme inutilement une partie de la bande passante, car une grande partie du code HTML est commun aux différentes pages de l'application
- Le chargement d'une nouvelle page à chaque requête demande beaucoup de temps et de bande passante.

Approche Asynchrone AJAX

- Envoyer des requêtes au serveur HTTP pour ne récupérer que les données nécessaires
- Utilisation la requête HTTP XMLHttpRequest
- Utilisation la puissance des feuilles de style (CSS) et de Javascript côté client pour interpréter la réponse du serveur
- Permet au navigateur de modifier partiellement la page pour la mettre à jour sans avoir à la recharger
- Applications plus réactives, meilleure ergonomie

Approche Asynchrone AJAX

- quantité de données échangées fortement réduite.
- Nécessite de charger, sur la première page, une bibliothèque **AJAX** volumineuse
- Nécessite un navigateur compatible, autorisant le **Javascritpt** et le compasant **XMLHTTP**
- Nécessite des tests car il existe de grandes différences entre les navigateurs

Fonctionnement

- Ajax utilise un modèle de programmation comprenant d'une part la présentation, d'autre part les évènements.
- Les évènements sont les actions de l'utilisateur, qui provoquent l'appel des fonctions associées aux éléments de la page.
- L'interaction avec l'utilisateur se fait à partir des formulaires ou boutons html.
- Ces fonctions JavaScript identifient les éléments de la page grâce au DOM et communiquent avec le serveur par l'objet XmlHttpRequest.

l'objet XmlHttpRequest

- AJAX se base sur un composant embarqué dans presque tous les navigateurs récents : XmlHttpRequest
- Cet objet a d'abord été développé par Microsoft, en tant qu'objet ActiveX, pour Internet Explorer 5
- Il a ensuite été repris et implémenté sous Mozilla 1 Safari 1.2, Konqueror 3.4 et Opera 8.
- Il n'est pas supporté par les navigateurs dits de « vieille génération ».
- En avril 2006, il a été proposé pour devenir une recommandation du W3C.

l'objet XmlHttpRequest

- Pour recueillir des informations sur le serveur, l'objet **XmlHttpRequest** dispose de deux méthodes:
  - ♦ - open: établit une connexion.
  - ♦ - send: envoie une requète au serveur.
- Les données fournies par le serveur seront récupérées dans les champs **responseXml** ou **responseText** de l'objet **XmlHttpRequest**. S'il s'agit d'un fichier xml, il sera lisible dans **responseXml** par les méthodes de DOM.
- Il faut créer un nouvel objet **XmlHttpRequest**, pour chaque fichier que vous voulez charger.

Exemple

```
<html><head><title>Exemple 1</title></head><body>
<script>
…
function ajax()
{ var xhr=getXMLHttpRequest();
  xhr.open("GET",  "http://localhost/ajax/reponse.txt", false);
  xhr.send(null);
  alert(xhr.responseText);
}
</script>
<p><a href="ajax();">Cliquez-moi !</a></p>
</body></html>
```