Institut Supérieur d'Informatique

JAVASCRIPT, DOM ET AJAX

Dr. Fahem KEBAIR – kebairf@gmail.com

INTRODUCTION

- JavaScript (JS) est un langage de programmation interprété léger pour le développement web.
- La dernière version de JS est la 1.8.5.
- JS a pour rôle principal de rendre des pages web statiques écrites en HTML interactives et dynamiques.
- JS s'exécute généralement côté client (dans le navigateur), mais il existe récemment des versions pour serveur.
- JS permet plusieurs fonctionnalités utiles :
  - La validation des formulaires,
  - La création des menus dynamiques,
  - La gestion des événements : clic d'un bouton, passage de la souris, …
  - Apporter de l'animation,
  - …

SYNTAXE

<script language="javascript" type="text/javascript">

code script JS

</script>

Le script JS peut être placé partout dans une page web, mais de préférence entre les balises <head></head>

La balise <script> peut prendre deux attributs :
language : type du script et type : type de média internet (MIME)

Exemple :

```
<html>
<head>
<script language="javascript" type="text/javascript">
   document.write("Hello World!");
<* Ceci est un commentaire : il est recommandé d'utiliser les
points virgule à la fin de chaque ligne de code JS *>
// Ceci un autre commentaire
 </script>
</head>
<body>
</body>
</html>
```

Le résultat de cette page web : Hello World !

VARIABLES

- JS manipule trois types primitifs :
  - Les nombres : 123, 12.5, …
  - Les chaines de caractères : "ceci est une chaine…"
  - Les booléens : *true* et *false*
- JS défini aussi deux types : *null* et *undefined*.
- JS manipule aussi les objets, ex : {"name":"John"}
- JS est un langage qui n'est pas typé, c'est-à-dire qu'une variable JS peut avoir plusieurs types de valeur au cours de l'exécution.

Exemple :

```
var myVariable = "John";
myVariable = 12.5;
```

VARIABLES

Portée

- *Variable globale* : portée globale, car la variable est définie partout dans le code.
- *Variable locale* : la variable n'est visible que dans la fonction où elle est définie.

Exemple :

```
var myVar = "global"; // variable globale
function checkscope() {
   var mayVar = "local";
   document.write(myVar); // variable locale
}
```

- Le résultat de ce script est « local ».

OPÉRATEURS

- JS supporte les opérateurs suivants :
  - Opérations arithmétiques : + , - , * , / , % , ++ , --
  - Opérateurs de comparaison : == , != , > , < , >= , <=
  - Opérateurs logiques : && , || , !
  - Opérateurs d'affectation : = , += , -= , *= , /= , %=
  - Opérateurs bità-bit

Exemple :

```
var a= 10;
var b = 20;
document.write("a < b : ");
var resultat = a < b;
document.write(resultat);
```

- Le résultat de ce script est « a < b : true ».

INSTRUCTIONS CONDITIONNELLES

- *Exemple :* if .. else

```
if (expression 1) {
   instrcution
} else if (expression 2) {
   instruction
} else {
   // instruction
}
```

- *Exemple :* switch case

break permet de sortir du swtich, s'il est enlevé le programme continue à exécuter la suite des instructions.

```
Switch (expression)
{
   case condition 1 : instruction;
                      break;
   case condition 2 : instruction;
                      break;
   default : instruction;
}
```

BOUCLES

- *Exemple :* while

```
while (expression) {
   instruction
}
```

- *Exemple :* do

```
do(expression) {
   instruction
} while (expression)
```

- *Exemple :* for

```
var count;
for(count = 0; count < 10; count++) {
   instruction
}
```

- *Exemple :* for .. in : permet de parcourir des propriétés d'un objet

```
var os = {"windows":"os","linux":"os"};
for(systeme in os) {
   document.write(systeme);
   document.write("<br />");
}
```

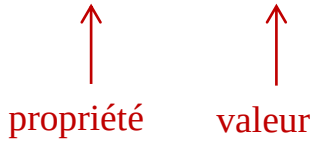
Le résultat de ce code :
windows
linux

OBJETS

- Un objet JS est une variable ayant des propriétés et des méthodes.

Exemple : création d'un objet

```
var person = {firstname:"jhon", lastname:"Doe", age:50};
```



- Accès aux propriétés d'un objet

```
person.lastname;
person["lastname"];
```

- Accès aux méthodes d'un objet

```
person.setNom(); // une méthode est définie dans le constructeur
document.write("hello"); // document est un objet JS natif
```

- Un tableau JS est un objet contenant une liste de strings ou d'entiers

```
var cars = {"BMW", "VW", "Mercedes"};// déclaration d'un tableau
cars[1]; // accès à une valeur du tableau
```

FONCTIONS

- Une fonction JS est un bloc de code réutilisable.

Exemple 1 : fonction sans paramètres ni retour (se comportant comme procédure)

```
function sayHello() {
   alert("hello"); // afficher une boite à message
}
sayHello(); // appel de la fonction
```

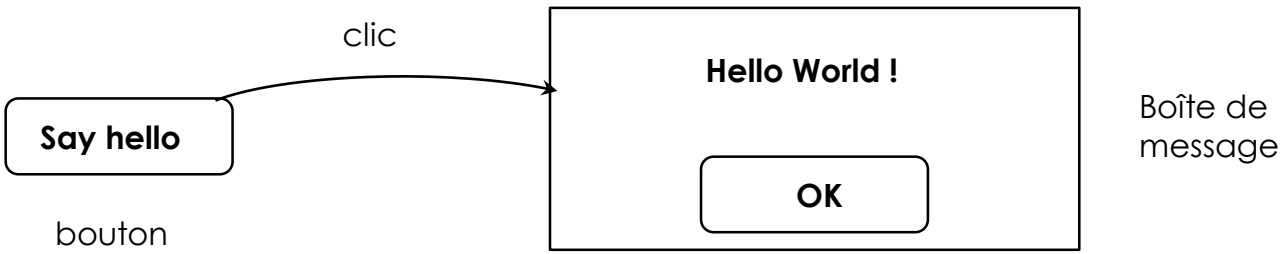
Exemple 2 : fonction avec paramètres et retour

```
function myAddition(a, b) {
   return a+b;
}
var x = myAddition(4, 3); // appel de la fonction
```

ÉVÈNEMENTS

- JS permet l'interaction avec HTML à travers les évènements. Par exemple : click sur un bouton, chargement d'une page, passage du curseur de la souris, …
- Exemple : évènement issu d'un clic sur un bouton

```
<head>
<script …>
function sayHello() {
   alert("hello");
}
</script>
</head>
<body>
<input type="button" onclick="sayHello()"; value="say hello" />
</body>
```



COOKIES

- Un cookie est un ensemble de données stockés sous la forme d'un fichier texte dans la machine du client afin de garder des informations bien utiles tels que : les achats, les préférences, …
- Fonctionnement :
  - Le serveur envoi des données sous la forme d'un cookie
  - Le navigateur peut accepter ce cookie et le stocke en tant que fichier texte sur le disque dur de l'ordinateur.
  - Chaque requête établie par l'utilisateur est accompagné du même cookie pour qu'il puisse être retrouvé.
- Attributs du cookie :
  - *Expires* : la date d'expiration du cookie
  - *Domain* : le nom de domaine du site
  - *Path* : le chemin vers le répertoire ou la page qui va éditer le cookie
  - *Name/value* (obligatoire) : un ensemble de couples nom/valeur identifiant un cookie

bouton

COOKIES

- Les cookies sont manipulés par des langages de script. JS peut aussi le manipuler en utilisant l'objet **document**.

Stocker un cookie

- L'objet cookie permet de stocker la valeur du cookie

```
document.cookie = "key1=value1";key2=value2;expires=date";
```

bouton

COOKIES

Exemple :

```
<html>
<head>
<script type="text/javascript">
function WriteCookie()
{
   if( document.myform.customer.value == "" ){
      alert("Enter some value!");
      return;
   }

   var cookievalue= document.myform.customer.value + ";";
   document.cookie="name=" + cookievalue;
   alert("Setting Cookies : " + "name=" + cookievalue );
}
</script>
</head>
<body>
<form name="myform" action="">
Enter name: <input type="text" name="customer"/>
<input type="button" value="Set Cookie" onclick="WriteCookie();"/>
</form>
</body>
</html>
```

COOKIES

Lire un cookie

```
<html>
<head>
<script type="text/javascript">
function ReadCookie()
{
   var allcookies = document.cookie;
   alert("All Cookies : " + allcookies );

   // stocker tous les cookies dans un tableau
   cookiearray  = allcookies.split(';');

   // lire les cookies un par un
   for(var i=0; i<cookiearray.length; i++){
      name = cookiearray[i].split('=')[0];
      value = cookiearray[i].split('=')[1];
      alert("Key is : " + name + " and Value is : " + value);
   }
}
</script>
</head>
<body>
<form name="myform" action="">
<input type="button" value="Get Cookie" onclick="ReadCookie()"/>
</form>
</body>
</html>
```

bouton

COOKIES

Supprimer un cookie

```
<html>
<head>
<script type="text/javascript">
function DeleteCookie()
{
   var now = new Date();
   now.setMonth( now.getMonth() - 1 );
   cookievalue = escape(document.myform.customer.value) + ";"
   document.cookie="name=" + cookievalue;
   document.cookie = "expires=" + now.toUTCString() + ";"
   alert("Setting Cookies : " + "name=" + cookievalue );
}
</script>
</head>
<body>
<form name="formname" action="">
Enter name: <input type="text" name="customer"/>
<input type="button" value="Set Cookie" onclick="DeleteCookie()"/>
</form>
</body>
</html>
```

Affecter une date passée à l'attribut *expires*

bouton

JSON

- JSON (JavaScript Object Notation) est une forme d'écriture de données en JavaScript.
- Il fournir un support d’écriture au format texte plus simple et plus légère que celui de XML.
- JSON est actuellement le format d’échange privilégié des web services REST et d’autres technologies comme AJAX et WebSocket.
- JSON peut être traité aisément par des bibliothèques JavaScript et des langages évolués tels que PHP, JAVA, C#,…

bouton

PRINCIPE

- JSON support les types de JS tels que le Boolean, Number, String, Array, Object et null.
- Il est composé par des notations de la forme : "clé":"valeur"
- Exemple complet :

```
var courses = {
        "fruits": [
           { "kiwis": 3,
            "mangues": 4,
            "pommes": null
           },
           { "panier": true }
        ],
        "legumes":
           { "patates":
             "amandine",
             "poireaux": false
           }
};
```

- La lecture d'une valeur : `courses.fruits[0].kiwis;` => retourne 3

bouton

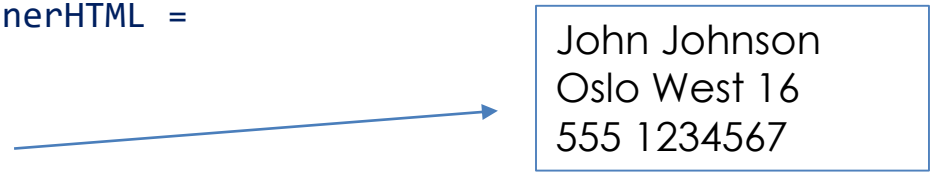
TRAITEMENT DE JSON

- JSON permet de fournir un texte ou un fichier qu'on peut l'exploiter directement avec la fonction eval() de JS afin de construire un objet. eval evalue d'une manière générale une chaine de caractère en l'occurrence une chaine formatée en JSON.

- Exemple :

```
var text = '{"name":"John Johnson","street":"Oslo West 16","phone":"555 1234567"}';
var obj = eval('(' + text + ')');

document.getElementById("demo").innerHTML =
obj.name + "<br>" +
obj.street + "<br>" +
obj.phone;
```



- Il est possible (**recommandé**) d'utiliser JSON.parse(str) pour la conversion de JSON en un objet JS :

```
var obj = JSON.parse(text);
```

- La conversion inverse, d'un objet JS en une chaine JSON est effectuée par JSON.stringify(obj) :

```
var textejson = JSON.stringify({"kiwis":3,"mangues":4})
```

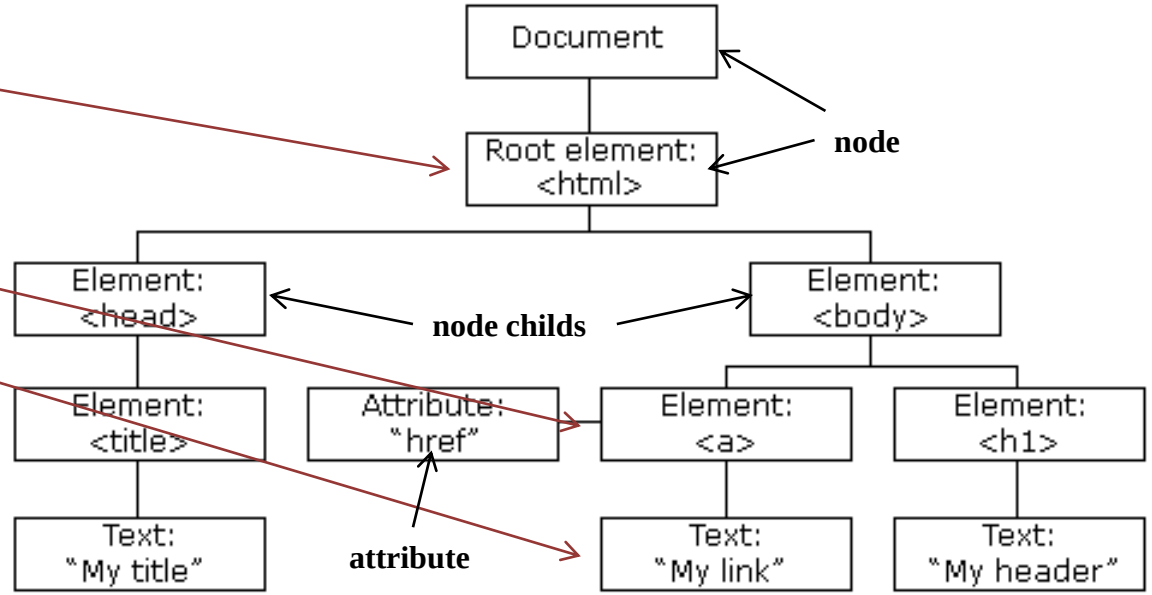
bouton

HTML DOM

- DOM (Document Object Model) est un standard permettant l'accès et la modification d'une interface indépendamment de tout langage de programmation.
- Le DOM HTML est le document HTML affiché par le navigateur.
- JS accède et change les éléments d'une page HTML grâce au DOM.
- DOM HTML est un arbre où tout composant est un nœud :
  - Le document lui-même est un nœud (la racine de l'arbre)
  - Les éléments HTML sont des éléments nœuds
  - Les attributs HTML sont des attributs nœuds
  - Le texte à l'intérieur des éléments HTML sont des textes nœuds
  - Les commentaires sont des commentaires nœuds.

HTML DOM
LES ÉLÉMENTS

```
<html>
<head>
</head>
<body>
<a href="http://w3c.org"/>
w3c web site</a>
</body>
</html>
```



- Un élément nœud peut avoir des nœud fils (*child nodes*) de type : élément, texte ou commentaire.
- Un élément peut avoir aussi des attributs de type noeud attribut (mais pas en tant que nœud fils).

HTML DOM
LES ÉLÉMENTS

- Chaque nœud a des **méthodes** et des **propriétés**.

Exemple :

```
<body>
<script>
document.getElementById("demo").innerHTML = "Hello World !";
</script>
<p id="demo"> </p>
</body>
```

- Dans cet exemple, *getElementById* est une méthode, alors que *innerHTML* est une propriété.
- *getElementById* permet de récupérer l'élément « p » ayant la valeur « demo » de son attribut « id ».
- *innerHTML* permet d'affecter la valeur « Hello World ! » au contenu de la balise de l'élément « p ».

HTML DOM
LES ÉLÉMENTS

Quelques méthodes de document

Method	Description
document.getElementById()	Find an element by element id
document.getElementsByTagName()	Find elements by tag name
document.getElementsByClassName()	Find elements by class name
document.createElement()	Create an HTML element
document.removeChild()	Remove an HTML element
document.appendChild()	Add an HTML element
document.replaceChild()	Replace an HTML element
document.write(*text*)	Write into the HTML output stream

Quelques attributs d'un élément

Method	Description
element.innerHTML=	Change the inner HTML of an element
element.attribute=	Change the attribute of an HTML element
element.setAttribute(*attribute,value*)	Change the attribute of an HTML element
element.style.*property*=	Change the style of an HTML element

HTML DOM
QUELQUES EXEMPLES

Exemple 1 :

```
var bouton = document.createElement("BUTTON");
var text = document.createTextNode("Click me");
btn.appendChild(text);
```

- Créer un bouton avec un texte « click me ».

Exemple 2 :

```
<script>
function getAllParaElems(){
var allParas = document.getElementsByTagName("p");
var num = allParas.length;
alert("Il y a" + num + "éléments <p> dans ce document");
}
</script>
<body>
<p> Un premier texte.</p>
<p> Un deuxième texte.</p>
<input type="button" onclick="getAllParasElems()"; value="nombre Params" />
</body>
```

- *document.getElementsByTagName* retourne une liste de nœuds.
- Le click permet d'afficher un message contenant le nombre des éléments « p ».

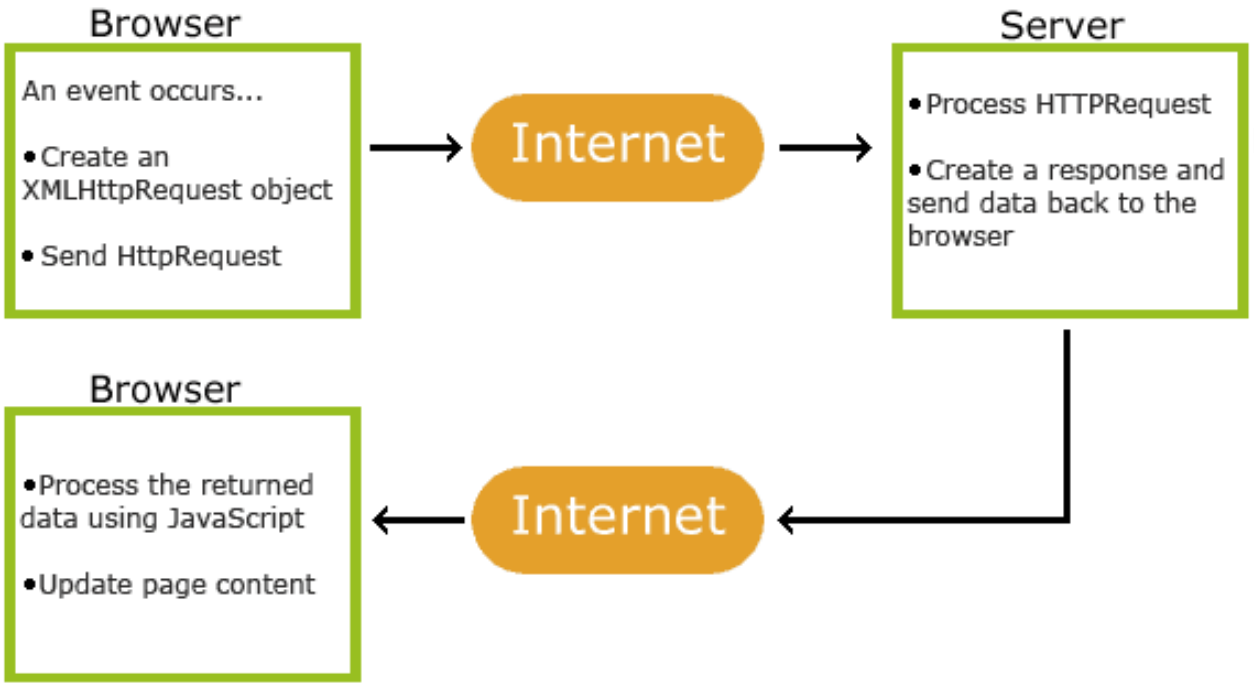
AJAX
INTRODUCTION

- AJAX (Asynchronous JavaScript and XML) est une nouvelle manière d'utiliser des standards existants, permettant d'échanger des données avec un serveur et mettant à jour des parties d'une page web sans recharger toute la page.
- AJAX est la technologie phare pour rendre les pages web encore plus dynamiques et riches, on note des fonctionnalités telles que :
  - La validation d'un formulaire; ex : vérification du login et du mot de passe
  - Récupération et affichage des données en temps réel sans envoi explicite d'une requête HTTP.
  - Autocomplete
  - ….

AJAX
TECHNOLOGIES

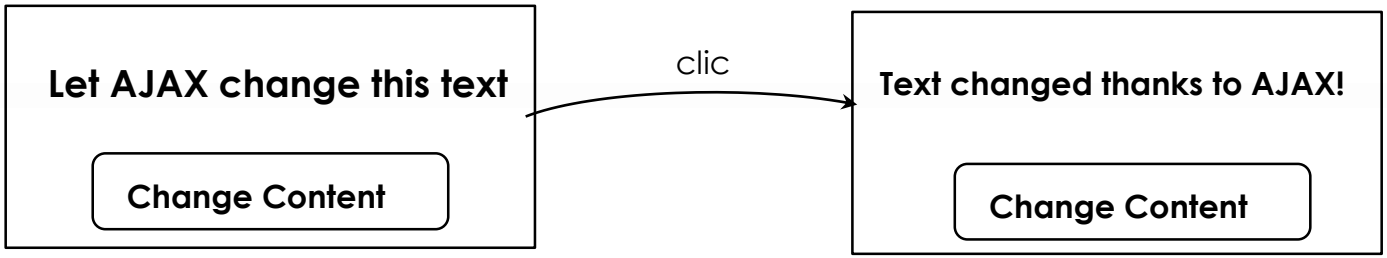
- Ajax repose sur les standards internet suivant :
  - L'objet XMLHttpRequest : échanger des données de façon asynchrone avec le serveur
  - JavaScript/DOM : afficher/interagir avec les informations
  - CSS : gérer le style des données
  - XML : format de transfert des données

AJAX
ARCHITECTURE ET FONCTIONNEMENT



1. L'utilisateur déclenche un événement (clic d'un bouton, saisi d'un texte, …)
2. Un objet XMLHTTPRequest est crée
3. La connexion est ouverte et une requête est envoyée
4. Le serveur HTTP reçoit la requête, la traite puis envoi la réponse
5. La réponse est reçue par le navigateur
6. La page web est partiellement mise à jour avec le DOM.

AJAX
UN EXEMPLE




```
<head>
<script>
function loadXMLDoc()
{
   ..AJAX script goes here..
}
</script>
</head>
<body>
<div id="myDiv"><h2>Let AJAX change this text</h2></div>
<button type="button" onclick="loadXMLDoc()">Change Content</button>
</body>
```

Le contenu du div change dynamiquement, suite à l'appel de la fonction loadXMLDOC, dans laquelle AJAX envoi une requête pour charger un fichier XML.

AJAX
L'OBJET XMLHTTPREQUEST

- The XMLHTTPRequest est l'objet central de la technologie AJAX qui permet d'échanger des données avec le serveur en *background*, ce qui permet de mettre à jour la page sans recharger la totalité.

Propriétés de XMLHTTPRequest

- *onreadystatechange* : appelé automatiquement quand la propriété readyState change.
- *readyState* : contient l'état du XMLHTTPRequest, de 0 à 4 :
  - 0 : connexion avec le serveur non initialisée
  - 1 : connexion établie
  - 2 : requête reçue
  - 3 : requête traitée
  - 4 : requête terminée et réponse prête
- *status* : 200 : « OK » et 404 : Page not found.

AJAX
L'OBJET XMLHTTPREQUEST

Création : `var xhr = new HMLHttpRequest();`

Envoi d'une requête : Ouvrir une connexion avec *open* (méthode, url, async) :

- *méthode* : GET ou POST, GET est plus rapide que POST, cependant POST permet d'envoyer une grand nombre de données et plus robuste et sécurisé.
- *url* : l'emplacement du fichier dans le serveur
- *async* : true (asynchrone) ou false (synchrone)

Exemple avec GET sans arguments :

```
xhr.open("GET","ajax_info.txt",true);
xhr.send();
```

Exemple avec GET avec arguments :

```
xhr.open("GET","ajax_info.txt?fname=Henry&lname=Ford ",true);
xhr.send();
```

Exemple avec POST sans arguments :

```
xhr.open("POST","ajax_info.txt",true);
xhr.send();
```

Exemple avec POST avec arguments :

```
xhr.open("POST","ajax_info.txt",true);
xhr.send("fname=Henry&lname=Ford");
```

Réception d'une réponse

- La propriété responseText (si la réponse n'est pas XML) ou responseXML (si la réponse est XML) pour récupérer une réponse du serveur.

Exemple avec *responseText* :

```
xhr.onreadystatechange=function() {
    if (xhr.readystate==4 && xhr.status== 200) {
     document.getElementById("myDiv").innerHTML=xhr.responseText;
 }
}
```

- Ce script permet de récupérer une réponse texte et de l'ajouter au contenu de la balise *div*.

Réception d'une réponse (suite)

Exemple avec *responseXML* :

```
xhr.onreadystatechange=function() {
   if (xhr.readystate==4 && xhr.status== 200) {
       xmlDoc = xhr.responseXML;
       Txt = "";
       X = xmlDoc.getElementByTagName("ARTIST");
       for (i=0; i < x.length; i++) {
          txt = txt + x[i].childNodes[0].nodeValue + "<br>";
       }
       ("myDiv").innerHTML=txt;
    }
}
```

- Ce script permet de récupérer une réponse xml et d'ajouter son contenu à une balise *div*.

AJAX
EXEMPLE COMPLET

```
<!DOCTYPE html>
<html>
<head>
<script>
function loadText()
{
   var xhr = new XMLHTTPRequest();
   // waiting for response
   xhr.onreadystatechange=function() {
   if (xhr.readystate==4 && xhr.status== 200) {
        document.getElementById("myDiv").innerHTML=xhr.responseText;
   }
   }
   // opening connexion and sending request
   xhr.open("GET","ajax_info.txt",true);
   xhr.send();
}
</script>
</head>
<body>
<div id="myDiv"><h2>Let AJAX change this text</h2></div>
<button type="button" onclick="loadText()">Change Content</button>
</body>
</html>
```