

Introduction

- Le Javascript est un langage "de script" simplifié "orienté objet" :
 - ▣ Initialement élaboré par Netscape en association avec Sun Microsystem.
 - ▣ Standardisé par un comité spécialisé, l'ECMA (European Computer Manufactures Association).

- Javascript permet :
 - ▣ De rendre dynamique un site internet développé en HTML :
 - Validation de formulaires, calculs, messages,
 - Modification de la page web,
 - Communication avec un serveur directement (AJAX)
 - ▣ De développer de véritables applications fonctionnant exclusivement dans le cadre d'Internet.

Caractéristiques principales

- Le Javascript est :
 - ▣ Ecrit directement dans le document HTML
 - ▣ Un script encadré par des balises HTML
 - ▣ Exécuté chez le client (pas d'appel réseau)
 - ▣ Interprété (pas compilé)
- Supporté par la plupart des navigateurs web
- Syntaxe proche du C

Plan du cours

- Syntaxe : tests, boucles, fonctions
- Événements / Manipulation de page
- Objets
- Insertion dans une page html
- JSON
- Cookies
- (AJAX)

www.Mcours.com
Site N°1 des Cours et Exercices Email: contact@mcours.com

JAVASCRIPT SYNTAXE

LI288 – web et développement web

HTML et JavaScript

- Deux types d'insertion (comme CSS)
 - ▣ Directement dans le fichier HTML
 - ▣ Dans un fichier externe et inclus en HTML
- Utilisation de balises spécifiques :
 - ▣ `<script language="Javascript">...</script>`

Insertion dans une page HTML

- Dans le corps de la page HTML
 - ▣ Le code s'exécute lors du chargement de la page

```
<html>
<body>
<script language="JavaScript">
  alert('bonjour');
</script>
</body>
</html>
```

Insertion dans une page HTML

- Dans l'entête de la page
 - ▣ Le code s'exécute lors d'un événement venant de l'utilisateur
 - ▣ Le code correspondant à cet événement se trouve dans le corps du document.
- En pratique les deux sont similaires sur la plupart des navigateurs

```
<html>
<head>
<script language="JavaScript">
    function f () { alert('Au revoir'); }
</script>
</head>
<body onUnload=f();>
</body>
</html>
```

Insertion dans une page HTML

- A placer dans le `<head>` ou le `<body>`
 - ▣ Fichier en format texte
 - ▣ Permet de réutiliser les scripts dans plusieurs pages
 - ▣ Inconvénient : requête supplémentaire vers le serveur

```
<html>
<head>
  <script language="javascript" src="fichier.js"></script>
</head>

<body>
  <input type="button" onclick="popup()" value="Clic">
</body>
</html>
```

Structure d'un script

- Similaire à Java ou C
- Règles générales
 - ▣ On peut mettre des espaces n'importe où
 - ▣ On sépare les commandes par des point-virgule ";"
 - ▣ Les réels sont notés avec un "." et pas une virgule ","
 - ▣ Commentaires : `//...` ou `/*...*/`
 - `//` ceci est un commentaire
 - `/*` ceci est aussi un commentaire `*/`

Les variables

- Déclaration et affectation
 - ▣ Déclaration avec le mot clé "var"
 - ▣ Affectation avec le signe d'égalité (=)
- Remarques :
 - ▣ La déclaration est faite par défaut (si affectation sans déclaration préalable)
 - ▣ La lecture d'une variable non déclarée provoque une erreur
 - ▣ Une variable déclarée non affectée est de type undefined (indéfinie)

```
//Déclaration de i, de j et de k.  
var i, j, k;  
  //Affectation de i.  
i = 1;  
//Déclaration et affectation de prix.  
var prix = 0;  
  
//Déclaration et affectation d'un tableau  
var car = ["a", "b", "c"];
```

Les variables

- Contraintes concernant les noms de variables :
 - Les noms de variables ne peuvent contenir que des lettres, chiffres, ou le caractère "_" (underscore)
 - `var Mon_Prenom; // correct`
 - Les caractères spéciaux et accentués sont interdits :
 - `var Mon_Prénom; // incorrect`
 - Attention aux majuscules et minuscules :
 - `MonPrenom` est différent de `Monprenom`.
 - Un nom de variable ne peut contenir d'espaces.
 - `var Mon Prenom; // incorrect`

- Les mots réservés JavaScript ne peuvent être utilisés comme noms de variable.

Les types de variables

- Principaux types :
 - ▣ Chaînes
 - ▣ Nombre (entier ou décimaux) : $10^{-308} > \text{nombre} < 10^{308}$
 - ▣ 3 valeurs spéciales :
 - Positive Infinity ou +Infinity (valeur infini positive)
 - Negative Infinity ou -Infinity (valeur infinie négative)
 - Nan (Not a Number) en général le résultat d'une opération incorrecte
 - ▣ Boolean
 - true (vrai) et false (faux)

- Le type d'une variable dépend de son contenu
 - ▣ `var maVariable = "Philippe"; // type chaîne`
`maVariable =10; // type nombre (entier)`

Les types de variables

- Deux types supplémentaires
- Type Undefined :
 - ▣ Une seule valeur, undefined
 - ▣ Type d'une variable avant qu'elle soit affectée
- Type Null :
 - ▣ Une seule valeur, null
 - ▣ Signale l'absence de données dans une variable

Les objets

- `typeof` permet de connaître le type d'une variable

```
var i = 1;
typeof i; //retourne number

var titre="Les raisins de la colère";
typeof titre; //retourne string

var jour = new Date();
typeof jour; //retourne object

var choix = true;
typeof choix; //retourne boolean

var cas = null;
typeof cas; //retourne object
typeof parseFloat; //retourne function
typeof Math; //retourne object (IE 5.*, NS 6.*, NS 4.78, Opera 6.*, Opera 5.*
typeof Math; //retourne function NS 3.*, Opera 3.*
```

Portée des variables

- Globale :
 - ▣ Variable déclarée en début de script
 - ▣ Accessible à n'importe quel endroit du programme

- Locale :
 - ▣ Variable déclarée à l'intérieur d'une fonction
 - ▣ Accessible uniquement dans la fonction

Syntaxe, les boucles

- Boucle for :
 - ▣ `for (i=0; i<5; i++) {...}`

- Boucle while :
 - ▣ `while (test) {...}`

 - ▣ `do {test} while (test)`

Syntaxe, les conditions

- `if (test) {} else {}`

- Tests possibles :
 - ▣ Egalité : `==, !=`
 - ▣ Inférieur, supérieur : `=<, >=, >, <`
 - ▣ Opérations bit à bit : `&, |`
 - ▣ Identique à : `===, !==` (teste valeur et type)
 - `('1' == 1) // true`
 - `('1' === 1) // false`
 - ▣ Opérations logiques : `&&, ||`

Les fonctions

- Définition :
 - ▣ `function maFonction(arg1,arg2) {instr;}`
- Appel :
 - ▣ `maFonction("1 2",13);`
- Exemple : calcul de la fonction factoriel
 - ▣ Calcul récursif
 - ▣ Choix (if)
- Mais aussi (à venir) :
 - ▣ Appelé sur un événement
 - ▣ Utilisation de `document.getElementById`
 - ▣ Utilisation de `this.value`

Fonction factorielle

```
<html>
<head>
<script language="JavaScript">
function fac(n){
  if (n < 2) {
    return 1;
  } else {
    return n * fac(n-1);
  }
}
</script>
</head>

<body>
<form>
  <input id="facArg" type="text"
    onchange="result = fac(this.value); document.getElementById('facResult').value = result;"/>
  <input id="facResult" type="text" />
</form>
</body>
</html>
```

Arguments

- ❑ Pas de type dans la signature de la fonction
- ❑ La déclaration d'arguments est optionnelle
 - ❑ On peut déclarer une fonction sans arguments
 - ❑ On peut ensuite lui passer des arguments
 - ❑ Et y accéder dans la fonction (via la variable arguments)

```
function test(argument1) {  
    alert("argument : " + argument1);  
    alert("nombre d'args : " + arguments.length);  
    for (var i=0 ; i<arguments.length ; i++) {  
        alert("arg " + i + " : " + arguments[i]);  
    }  
}  
  
test(); // undefined - 0  
test("1"); // 1 - 1 - 1  
test("1","2"); // 1 - 2 - 1 - 2
```

Fonctions d'ordre supérieur

- Une fonction peut prendre une fonction en argument
- Permet plus de flexibilité

```
function carre(x) {  
    return x*x;  
}  
  
function map (a,f) {  
    for (var i=0 ; i<a.length ; i++) {  
        a[i]=f(a[i]);  
    }  
    return a.toString();  
}  
  
map([1,2,3],carre);
```

Quelques fonctions de base

- Eval :
 - ▣ Prend une chaîne de caractère
 - ▣ Interprète cette chaîne comme du code Javascript

```
<html>
<body>
<script language="JavaScript">
  eval('function carre(n){ return n*n;};alert(carre(2));');
</script>
</body>
</html>
```

Quelques fonctions de base

- `isFinite`
 - ▣ Détermine si le paramètre est un nombre fini

- `isNaN`
 - ▣ Détermine si le paramètre n'est pas un nombre
 - ▣ NaN : Not a Number

```
isFinite(240) //retourne true
isFinite("Un nombre") //retourne false

isNaN("un nombre") //retourne true
isNaN(20) //retourne false
```

Quelques fonctions de base

- `parseFloat`
 - ▣ Analyse une chaîne de caractères et retourne un nombre décimal
 - ▣ Si l'argument évalué n'est pas un nombre, renvoie NaN (Not a Number)
- `parseInt(string, base)`
 - ▣ Analyse une chaîne de caractères et retourne un nombre entier de la base spécifiée
 - ▣ La base peut prendre les valeurs 16 (hexadécimal) 10 (décimal), 8 (octal), 2 (binaire)

```
alert(5+"2"); // affiche 52
alert(5+parseInt("2")); // affiche 7

parseInt("10.33") // 10
parseInt("40 years") // 40
parseInt("He was 40") // NaN
parseInt("010") // base 8 (0...)
parseInt("0x10") // base 10 (0x...)
parseInt("10",3) // base 3
```

Quelques fonctions de base

- Number
 - ▣ Convertit l'objet spécifié en valeur numérique
- String
 - ▣ Convertit l'objet spécifié en chaîne de caractères
- Escape
 - ▣ Retourne la valeur hexadécimale à partir d'une chaîne codée en ISO-Latin-1.

```
var jour = new Date("December 17, 1995 03:24:00");  
alert (Number(jour));  
  
jour = new Date(430054663215);  
alert (String(jour));  
  
escape("!&") //retourne %21%26%
```

Opérations sur les chaînes

- La concaténation : utilisation de +
 - ▣ `var chaine = "bonjour" + "FI3/FCD1";`

- Déterminer la longueur d'une chaîne : attribut `length`
 - ▣ `var ch1 = 'bonjour';`
 - ▣ `var longueur = ch1.length;`

- Extraction d'une partie de la chaîne :
 - ▣ `var dateDuJour = "04/04/03"`
 - ▣ `var carac = dateDuJour.charAt(2);`
 - Extrait le caractère en position 2 de la chaîne
 - ▣ `var mois = dateDuJour.substring(3, 5);`
 - 3: est l'indice du premier caractère de la sous-chaîne à extraire
 - 5 : indice du dernier caractère à prendre en considération ; ce caractère ne fera pas partie de la sous-chaîne à extraire

www.Mcours.com

Site N°1 des Cours et Exercices Email: contact@mcours.com

JAVASCRIPT ÉVÉNEMENTS

LI288 – web et développement web

Gestionnaire d'événements

- Les événements servent à interagir avec l'utilisateur
 - ▣ On peut détecter les clics, les modifications de formulaires, ...
- Chaque événement a un identifiant
 - ▣ De la forme onQuelqueChose
 - ▣ Par exemple : onLoad, onClick, onMouseOver, etc.
- Un événement peut exécuter du code javascript
 - ▣ Une ou plusieurs instructions, en général un appel de fonction
- Activation :
 - ▣ `<balise ... onQuelqueChose="code javascript;">`

Les événements de base

□ Événement onLoad

- Se produit lorsque une page web est chargée dans la fenêtre du navigateur
- Toute la page (y compris les images qu'elle contient si leur chargement est prévu) doit avoir été chargée pour qu'il ait lieu
- Cet événement peut être associé à une image seulement (il se produit alors une fois le chargement terminé)

Les événements de base

□ Événement onClick

- Se produit lorsque l'utilisateur clique sur un élément spécifique dans une page, comme un lien hypertexte, une image, un bouton, du texte, etc.
- Ces éléments sont capables de répondre séparément à cet événement
- Il peut également être déclenché lorsque l'utilisateur clique n'importe où sur la page s'il a été associé non pas à un élément spécifique, mais à l'élément body tout entier

Les événements de base

- Événement `onMouseover`
 - ▣ Analogue à `onClick` sauf qu'il suffit que l'utilisateur place le pointeur de sa souris sur l'un des éléments précités (lien hypertexte, image, bouton, texte, etc.) pour qu'il ait lieu
- Événement `onMouseout`
 - ▣ A l'inverse de `onMouseover`, cet événement se produit lorsque le pointeur de la souris quitte la zone de sélection d'un élément.

Une liste plus longue

- Globales :
 - onAbort : chargement d'une image interrompu
 - onError : une erreur durant le chargement de la page
 - onLoad : chargement de la page
 - onUnload : l'utilisateur quitte la page

- Souris :
 - onBlur : un élément perd le focus
 - onClick : clic sur l'élément
 - ondblclick: double clic sur l'élément
 - ondragdrop : drag and drop sur la fenêtre du navigateur
 - onfocus : le focus est donné à un élément
 - onmouseover : la souris passe sur un élément
 - onmouseout : la souris quitte un élément
 - onresize : la fenêtre est redimensionnée

Une liste plus longue

- **Formulaires :**
 - `onChange` : modification d'un champ de données
 - `onReset` : effacement d'un formulaire à l'aide du bouton Reset.
 - `onSelect` : sélection d'un texte dans un champ "text" ou "textarea"
 - `onSubmit` : clic sur le bouton de soumission d'un formulaire

- **Clavier :**
 - `onKeyDown` : appui sur une touche du clavier
 - `onKeyPress` : appui et maintien sur une touche
 - `onKeyUp` : relâchement d'une touche

- Attention, selon les versions de javascript, les événements peuvent ne pas exister.

Fonction factorielle

```
<html>
<head>
<script language="JavaScript">
function fac(n){
  if (n < 2) {
    return 1;
  } else {
    return n * fac(n-1);
  }
}
</script>
</head>

<body>
<form>
  <input id="facArg" type="text"
    onchange="result = fac(this.value); document.getElementById('facResult').value = result;"/>
  <input id="facResult" type="text" />
</form>
</body>
</html>
```

www.Mcours.com
Site N°1 des Cours et Exercices Email: contact@mcours.com

JAVASCRIPT

OBJETS

Les objets

- L'opérateur New : créer une instance d'un objet
 - Objet défini par l'utilisateur
 - Objets prédéfinis, Array, Boolean, Date, Function, Image, Number, Object, ou String.
 - `nouvel_objet = new type_objet(parametres)`
- `texte = new String("Une chaîne de caractère");`

Les objets - création

- Créer un objet et accéder à ses attributs/méthodes
 - ▣ C'est simplement un tableau associatif !

```
var obj = new Object();

obj["attribut"] = "valeur";
// ou obj.attribut = "valeur";

obj["methode"] = function(p) { alert("parametre : " + p);};
// ou obj.methode = ...

alert(obj.attribut);
obj.methode("test");
```

Les objets - création

□ Version avec constructeur

```
function Test(p,q) {
  this.att1=p;
  att2 = q;
  this.al = function() {alert(this.att1);}
  var al2 = function() {alert(att2)};
}

var x = new Test(2,3);
alert(x.att1);
x.al();
alert(x.att2);
try {x.al2();} catch(err) {alert(err);}
```

Les objets - création

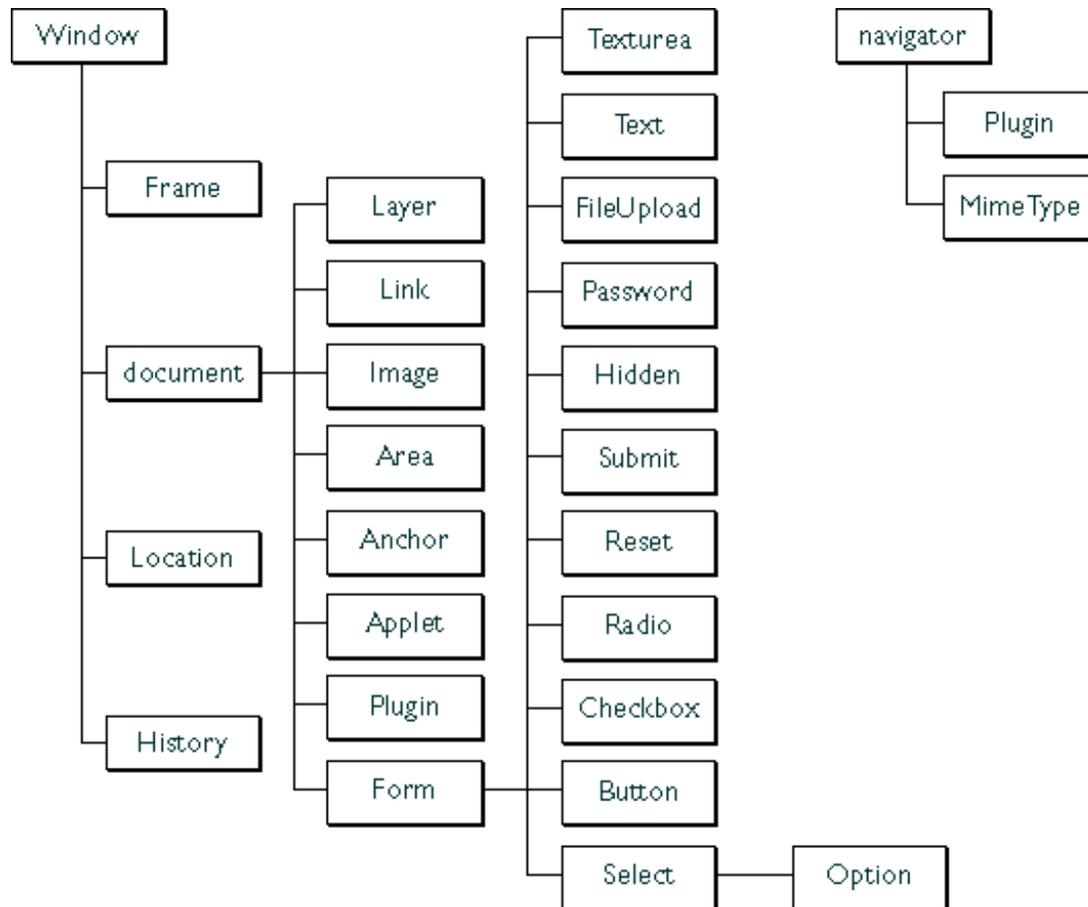
- Création directe sous forme de tableau associatif :
 - ▣ Séparation par des virgules ","
 - ▣ cf JSON plus loin dans le cours.

```
var obj = {  
  attribut: "valeur",  
  methode: function(p) { alert("parametre : " + p);}  
};  
  
alert(obj.attribut);  
obj.methode("test");
```

Les Objets

- Le contenu HTML est accessible sous forme d'objets :
 - ▣ Le navigateur : objet "navigator"
 - ▣ La fenêtre du navigateur : "window"
 - ▣ La page HTML : "document"
 - ▣ Un formulaire est aussi un objet
 - ▣ Un lien hypertexte dans une page HTML est un objet...
- Les objets peuvent réagir à des "Evénements"
- Les navigateurs ne supportent pas les mêmes objets

Les objets du navigateur



L'objet String

- Propriété :
 - length : retourne la longueur de la chaîne de caractères
- Méthodes :
 - anchor("lien") : formate la chaîne avec la balise <A>
 - bold() : formate la chaîne avec la balise
 - charAt() : renvoie le caractère se trouvant à une certaine position
 - charCodeAt() : renvoie le code du caractère se trouvant à une certaine position
 - concat() : permet de concaténer 2 chaînes de caractères
 - fromCharCode() : renvoie le caractère associé au code
 - indexOf() : trouve l'indice d'occurrence d'un caractère dans une chaîne
 - substr(), substring() : retourne une portion de la chaîne
- <http://www.toutjavascript.com/reference/index.php>

L'objet Array

□ Propriété :

- ▣ `length` : retourne le nombre d'éléments du tableau

□ Méthodes :

- ▣ `concat()` : permet de concaténer 2 tableaux
- ▣ `reverse()` : inverse le classement des éléments du tableau
- ▣ `slice()` : retourne une section du tableau
- ▣ `sort()` : permet le classement des éléments du tableau

L'objet Math

□ Propriétés :

- ▣ E, LN2, LN10, LOG2E, LOG10E, PI, SQRT2, SQRT1_2

□ Méthodes :

- ▣ `abs()`, `max()`, `min()`, `sqrt()`, `pow()`, `exp()`, ...
- ▣ `ceil()` : retourne le plus petit entier supérieur à un nombre
- ▣ `floor()` : retourne le plus grand entier inférieur à un nombre
- ▣ `round()` : arrondi un nombre à l'entier le plus proche
- ▣ `random()` : retourne un nombre aléatoire entre 0 et 1

L'objet Date

- Propriété : aucune

- Méthodes :
 - `getFullYear()`, `getYear()`, `getMonth()`, `getDay()`, `getDate()`, `getHours()`, `getMinutes()`, `getSeconds()`, `getMilliseconds()`
 - `getUTCFullYear()`, ... idem en temps universel
 - `setFullYear()`, `setYear()`, ... modification d'une date
 - `getTime()` : retourne le temps stocké dans l'objet Date
 - `getTimezoneOffset()` : retourne la différence entre l'heure du client et le temps universel
 - `toGMTString()`, `toLocaleString()`, `toUTCString()` : convertissent la date en chaîne de caractère selon la convention GMT, selon la convention locale ou en temps universel

www.Mcours.com
Site N°1 des Cours et Exercices Email: contact@mcours.com

JAVASCRIPT

MANIPULATION DE PAGE

Entrées/sorties

- 3 types de boîtes de messages peuvent être affichés en utilisant Javascript
 - Méthode alert()
 - sert à afficher à l'utilisateur des informations simples de type texte. Une fois que ce dernier a lu le message, il doit cliquer sur OK pour faire disparaître la boîte
 - Méthode confirm()
 - permet à l'utilisateur de choisir entre les boutons OK et Annuler.
 - Méthode prompt()
 - La méthode prompt() permet à l'utilisateur de taper son propre message en réponse à la question posée

- La méthode `document.write` permet d'écrire du code HTML dans la page WEB

Entrées/sorties

```
<html>
<head>
<title> une page simple </title>
</head>
<body>
  <script language='javascript'>
    alert('bonjour');
    document.write (
      prompt('quel est votre nom ?', 'Indiquer votre nom ici')
    );
    confirm('quel bouton allez-vous choisir ?');
  </script>
</body>
</html>
```

Les objets du navigateur

- L'objet le plus haut dans la hiérarchie est "window" qui correspond à la fenêtre même du navigateur.
- L'objet "document" fait référence au contenu de la fenêtre :
 - ▣ "document" = ensemble des éléments HTML de la page.
- On peut accéder ces éléments avec :
 - ▣ méthodes propres à l'objet document :
 - getElementById() trouve l'élément avec son identifiant (ID)
 - getElementByName
 - ▣ soit des collections d'objets qui regroupent sous forme de tableaux Javascript tous les éléments de type déterminé.

L'objet window

- Propriétés : (accessibles avec IE et N)
 - closed : indique que la fenêtre a été fermée
 - defaultStatus : indique le message par défaut dans la barre de status
 - document : retourne l'objet document de la fenêtre
 - frames : retourne la collection de cadres dans la fenêtre
 - history : retourne l'historique de la session de navigation
 - location : retourne l'adresse actuellement visitée
 - name : indique le nom de la fenêtre
 - navigator : retourne le navigateur utilisé
 - opener : retourne l'objet window qui a créé la fenêtre en cours
 - parent : retourne l'objet window immédiatement supérieur dans la hiérarchie
 - self : retourne l'objet window correspondant à la fenêtre en cours
 - status : indique le message affiché dans la barre de status
 - top : retourne l'objet window le plus haut dans la hiérarchie

L'objet window

- Méthodes :
 - ▣ blur() : enlève le focus de la fenêtre
 - ▣ close() : ferme la fenêtre
 - ▣ focus() : place le focus sur la fenêtre
 - ▣ moveBy() : déplace d'une distance
 - ▣ moveTo() : déplace la fenêtre vers un point spécifié
 - ▣ open() : ouvre une nouvelle fenêtre
 - ▣ print() : imprime le contenu de la fenêtre
 - ▣ resizeBy() : redimensionne d'un certain rapport
 - ▣ resizeTo() : redimensionne la fenêtre
 - ▣ setTimeout() : évalue une chaîne de caractère après un certain laps de temps

L'objet document

□ Propriétés :

- applets, forms, images, links : retourne les collection d'applets java, formulaires... présents dans le document
- cookie : permet de stocker un cookie
- domain : indique le nom de domaine du serveur ayant apporté le document
- referrer : indique l'adresse de la page précédente
- title : indique le titre du document

```
<html><head><title>Test</title></head>
<body>
<a href="http://www.yahoo.fr/">Yahoo</a>
<a href="http://www.google.fr/">Google</a>
<script type="text/javascript">
  alert(document.title);
  for(var i=0; i < document.links.length; ++i)
    alert(document.links[i]);
</script>
</body></html>
```

L'objet document

□ Méthodes :

- `close()` : ferme le document en écriture;
- `open()` : ouvre le document en écriture;
- `write()` : écrit dans le document;
- `writeln()` : écrit dans le document et effectue un retour à la ligne

L'objet navigator

□ Propriétés

- appName : application (Netscape, Internet Explorer)
- appVersion : numero de version.
- platform : système d'exploitation (Win32)
- plugins
- language
- mimeTypees
- JavaEnabled()

Manipulation des objets

- Pour adresser un objet, il faut préciser son « chemin d'accès » dans l'arborescence de la structure.
- Si le nom de la fenêtre est omis, le navigateur utilisera par défaut la fenêtre courante (attention aux frames)
- On peut omettre `window.document` (un seul objet "document")

```
<html>
<body onLoad="window.document.f1.zone.value='Bonjour';">
<form name="f1">
  <input name="zone" type="text">
</form>
</body>
</html>
```

Exemples

```
<html><head>
<script type="text/javascript">
function changeCouleur(color) {
    var ml = document.getElementById("maListe");
    ml.setAttribute("style","color:"+color);
}
</script>
</head><body>

<ul id="maListe">
    <li><a href="javascript: changeCouleur ('red');">Rouge</a></li>
    <li><a href="javascript: changeCouleur ('blue');">Bleu</a></li>
    <li><a href="javascript: changeCouleur ('black');">Noir</a></li>
</ul>
</body></html>
```

Pour aller plus loin

- Toutes les commandes sur :
 - ▣ <https://developer.mozilla.org/fr/DOM/element#Propriétés>
- Pour tester/débugger :
 - ▣ Firebug
 - ▣ Console d'erreur.
 - ▣ Utiliser des alertes.

JSON

JAVASCRIPT OBJECT

NOTATION

JSON ?

- Format d'échange de données.
- Objectifs :
 - Simple.
 - Extensible.
 - Ouvert.
 - Lisible par un humain.
- Similaire à la définition des objets Javascript.

JSON : JavaScript Object Notation

- Les types de base
 - ▣ Nombres entiers, réels ou à virgule flottante
 - ▣ Chaînes de caractères
 - ▣ Booléen true et false
 - ▣ Tableaux [..., ...] ou tableaux associatifs (objets) "clé":valeur: {..., ...}
 - ▣ null

```
{  
  "Nom": "Guillaume",  
  "Adresse": {"rue": "4 place Jussieu", "cp": "75004",  
    "ville": "Paris"},  
  "notes": [1, 2, 4, 8, 16, 32]  
}
```

JSON et Javascript

- JSON peut être utilisé directement :
 - ▣ Inclusion dans du HTML
 - `<script> var data = JSONdata; </script>`
 - ▣ Peut être converti en un objet Javascript
 - `responseData = eval('(' + responseText + ')');`

JSON ou XML ?

- JSON est très utilisé avec AJAX (le X de AJAX est pour XML)
- JSON :
 - ▣ {"nom": "Guillaume", "prenom": "Jean-Loup"}
- XML :
 - ▣ <?xml version='1.0' encoding='UTF-8'?>
 - ▣ <element>
 - ▣ <nom>Guillaume</nom>
 - ▣ <prenom>Jean-Loup</prenom>
 - ▣ </element>

JSON ou XML ?

- Evaluation en JSON :

- ▣ `var name = eval('(' + req.responseText + ')').nom.value;`

- Ou :

- ▣ `eval('(' + req.responseText + ')').xyz.abc.value;`

- ▣ Accès simplifié aux différent niveaux.

JSON ou XML ?

- Evaluation en XML :
 - ▣ `var root = req.responseXML;`
 - ▣ `var name = root.getElementsByTagName('nom');`

- Ou :
 - ▣ `root.getElementsByTagName('xyz')[0].firstChild`

- Un peu plus complexe

JSON ou XML

- Taille des données :
 - ▣ plus petite en JSON (pas de fermeture de tag)
 - ▣ XML se compresse mieux
- Vitesse :
 - ▣ XML se parse mieux
 - ▣ JSON s'évalue avec eval : peu efficace (pour l'instant)
- Choix :
 - ▣ JSON : structures de données
 - ▣ XML : structuration de documents
- A vous de faire votre choix !

JAVASCRIPT COOKIES

Les Cookies

- Chaîne de caractères pouvant être écrite sur le disque dur.
 - ▣ A un endroit défini.
 - ▣ Ne peut être lue que par le seul serveur qui l'a générée.

- Que faire avec un cookie
 - ▣ Transmettre des valeurs d'une page HTML à une autre.
 - Exemple : gestion d'un caddie sur un site de vente en ligne.
 - ▣ Personnaliser les pages présentées à l'utilisateur.

Les Cookies

- Limitations (théoriques, en pratique c'est faux) :
 - ▣ Limites en nombre : Pas plus de 20 cookies par serveur.
 - ▣ Limites en taille : un cookie ne peut excéder 4 Ko.
 - ▣ Limites du poste client : Un poste client ne peut stocker plus de 300 cookies en tout.

- Stockage - dépend du navigateur
 - ▣ IE :
 - C:\Users\admin\AppData\Local\Microsoft\Windows\Temporary Internet Files
 - ▣ Firefox :
 - ...

Les Cookies - structure

- `Nom=Contenu; expires=expdate; path=Chemin; domain=NomDeDomaine; secure`
 - ▣ `Nom=Contenu;`
 - En-tête du cookie : deux variables suivies d'un ";"
 - La variable `Nom` contient le nom à donner au cookie.
 - La variable `Contenu` contient le contenu du cookie
 - Exemple `mon_cookie="oui:visite"`

Les Cookies - structure

- Nom=Contenu; expires=expdate; path=Chemin; domain=NomDeDomaine; secure
 - Expires= expdate;
 - Le mot réservé expires suivi du signe "=" (égal).
 - Date à laquelle le cookie sera supprimé du disque dur.
 - Format :
 - Wdy, DD-Mon-YYYY HH:MM:SS GMT
 - A générer avec les fonctions de l'objet Javascript Date
 - Règle générale : indiquer un délai en nombre de jours (ou d'années) avant disparition du Cookie.

Les Cookies - structure

- `Nom=Contenu; expires=expdate; path=Chemin; domain=NomDeDomaine; secure`
 - `path=Chemin;`
 - chemin de la page qui a créé le cookie.
 - `domain=NomDeDomaine;`
 - le nom du domaine de la page.
 - `secure` (false par défaut)
 - "true" (HTTPS) ou "false" (HTTP) : protocole utilisé.
 - `path, domain et secure` sont facultatifs.

Les Cookies - écriture

- propriété de l'objet document :
 - `document.cookie = Nom + "=" + Contenu + ";"`
`expires=" + expdate.toGMTString();`

```
// creation des variables
var Nom = "MonCookie" ;
var Contenu = "contenu" ;
var expdate = new Date () ;

// modification de la date d'expiration (10 jours)
expdate.setTime (expdate.getTime() + ( 10 * 24 * 60 * 60 * 1000)) ;

// ecriture du cookie sur le disque
document.cookie = Nom + "=" + Contenu + ";" expires=" + expdate.toGMTString() ;
```

Les Cookies - lecture

- Accéder à la propriété cookie de l'objet document :
 - ▣ Document.cookie

```
var lesCookies ;  
lesCookies = document.cookie ;
```

Les Cookies - modification

- Modification d'un cookie
 - ▣ Modifier le contenu de la variable Contenu puis réécrire le cookie sur le disque dur du client

```
Contenu = "Le cookie a été modifié..." ;  
document.cookie = Nom + "=" + Contenu + "; expires=" +  
    expdate.toGMTString() ;
```

Les Cookies - suppression

- Modifier la date de péremption :
 - Mettre une date dans le passé.
 - Le navigateur se charge du reste

```
// on enlève une seconde a la date courante
expdate.setTime (expdate.getTime() - (1000)) ;

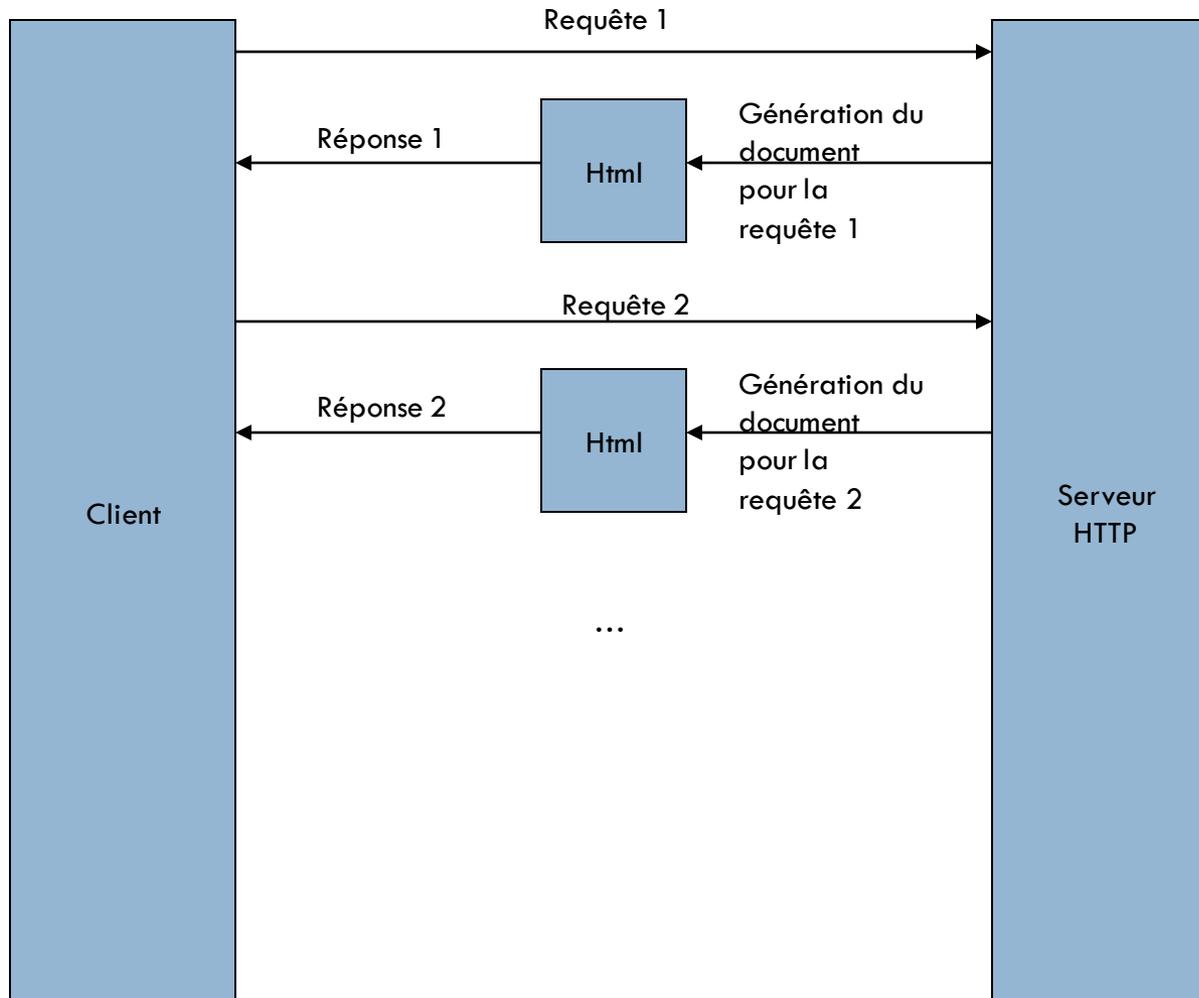
// écriture sur le disque
document.cookie = Nom + "=" + Contenu + "; expires=" +
    expdate.toGMTString() ;
```

AJAX

Application traditionnelle

- Application WEB traditionnelle :
 - ▣ Le client envoie une requete HTTP
 - ▣ Le serveur renvoie une page
- Consommation inutile de la bande passante :
 - ▣ Une grande partie du code HTML est commun aux différentes pages de l'application.
- Le chargement d'une nouvelle page à chaque requête n'est pas ergonomique

Application traditionnelle



AJAX

- Qu'est-ce qu'AJAX ?
 - ▣ Asynchronous Javascript and XML

- Pourquoi AJAX:
 - ▣ Javascript est très utilisé au niveau du client :
 - validation de formulaire, modifications de la page, ...
 - ▣ Tout ne peut pas être confié au client :
 - Manque de sécurité/confiance
 - Limitations

AJAX

- Principe de base :
 - ▣ Le client et le serveur dialoguent.

- Autant faire en sorte que les messages soient le plus petits possibles.
 - ▣ Le client n'a pas besoin de toute la base de données, juste de suffisamment de données pour le client.

- Le serveur et le client ont chacun un travail
 - ▣ L'application ne doit donc pas être prise en charge entièrement d'un côté ou de l'autre.

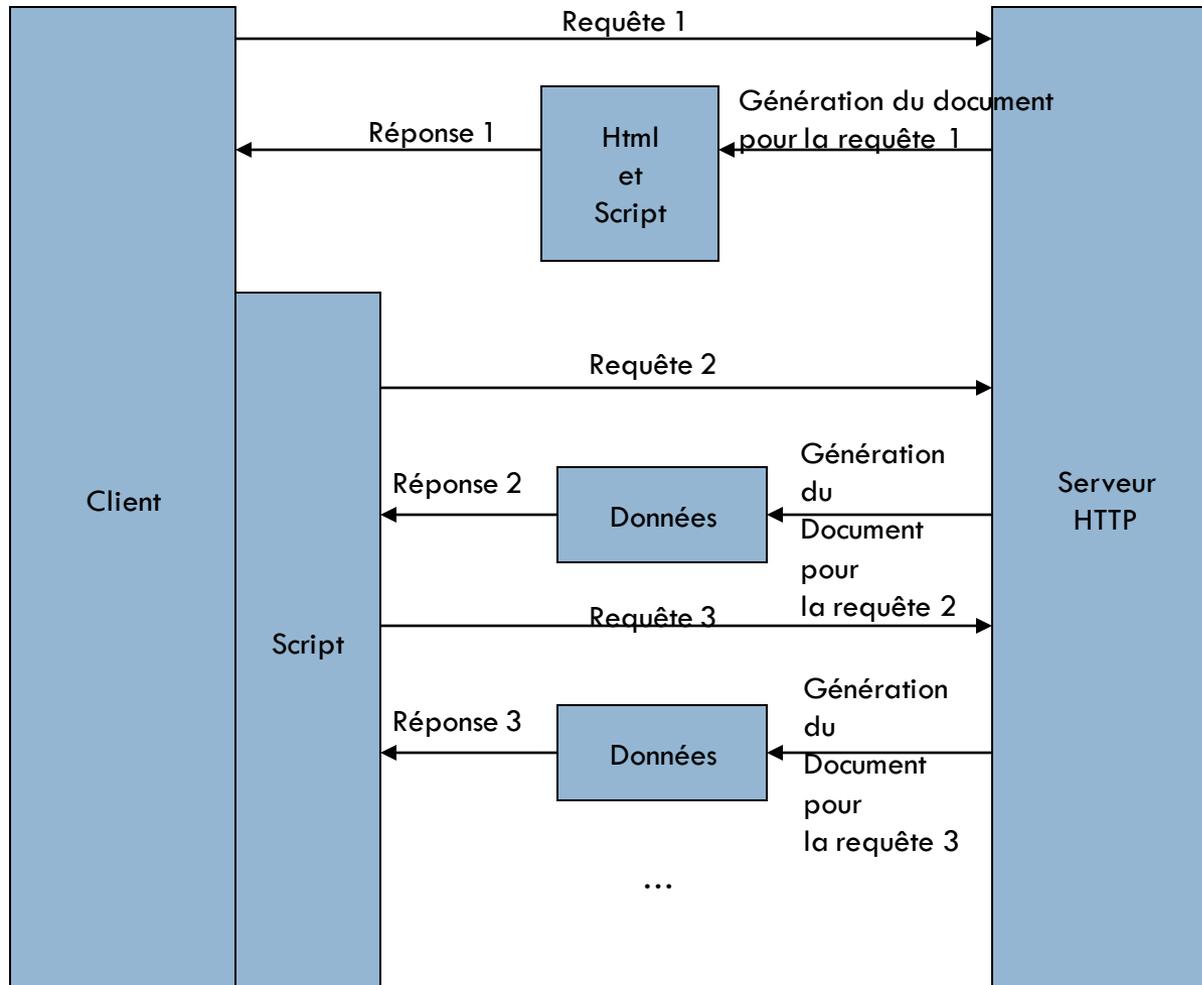
AJAX = un acronyme de plus

- C'est juste du Javascript classique.
- Principe de base :
 - ▣ L'application Javascript émet des requêtes vers le serveur avec un protocole donné.
 - ▣ Le serveur répond avec les informations demandées.
 - ▣ Tout se passe sans rechargement de la page.
 - Mode synchrone ou asynchrone pour le client.
 - ▣ Javascript traite les données reçues et modifie la page en conséquence.

Qui utilise Ajax

- Clients de messagerie : Gmail, Yahoo Mail, HotMail
- Google Maps
- Flickr, Picasa
- Deezer
- Youtube, Dailymotion
- Myspace, Facebook

AJAX



Comment ça marche

- Un exemple sans AJAX :
 - ▣ requête faite automatiquement par le navigateur
 - ▣ récupération d'une image à distance

```
<body onLoad="javascript:document.images[0].src =  
  'http://...'">
```

```
</a>
```

```
</body>
```

Comment ça marche

- `<input type="text" size="5" onBlur="miseAJour()">`
- 1. La fonction `miseAJour` crée un objet `XMLHttpRequest`
 - ▣ Existe sur tous les navigateurs "modernes".
- 2. Cet objet permet de faire une requête vers la page concernée
 - ▣ `http://www.test.com/page.php?valeur=12`
 - ▣ Conceptuellement identique à la recherche d'image.
- 3. On traite le résultat et on met à jour la page web

L'objet XMLHttpRequest

- AJAX se base sur XMLHttpRequest
 - ▣ Initialement développé par Microsoft, en tant qu'objet ActiveX, pour Internet Explorer 5
 - ▣ Puis repris et implémenté sous Mozilla 1 Safari 1.2, Konqueror 3.4 et Opera 8.
 - ▣ Pas supporté par certains vieux navigateurs.
 - ▣ Proposé en 2006 pour devenir une recommandation du W3C :
 - <http://www.w3.org/TR/XMLHttpRequest/>
 - Draft novembre 2009

L'objet XMLHttpRequest

- Problèmes :
 - ▣ Nécessite un navigateur compatible, autorisant le Javascript et XMLHttpRequest.
 - ▣ Nécessite plus de tests car il existe de grandes différences entre les navigateurs.
 - XMLHttpRequest n'est pas implémenté de la même manière selon les navigateurs (et les versions des navigateurs).

- Solution la plus simple :
 - ▣ Aller chercher le code générique avec google.

Création de l'objet XMLHttpRequest

- Pour Internet Explorer (avant IE7) :
 - ▣ `xhr = new ActiveXObject("Microsoft.XMLHTTP");`
- Ou
 - ▣ `xhr = new ActiveXObject("Msxml2.XMLHTTP");`
- Pour les autres navigateurs :
 - ▣ `xhr = new XMLHttpRequest();`

Création de l'objet XMLHttpRequest

```
function getXMLHttpRequest() {
  if (window.XMLHttpRequest) {
    return new XMLHttpRequest();
  } else {
    if (window.ActiveXObject) {
      try {
        return new ActiveXObject("Msxml2.XMLHTTP");
      } catch (e) {
        try {
          return new ActiveXObject("Microsoft.XMLHTTP");
        } catch (e) {
          return NULL;
        }
      }
    }
  }
}
```

Création (bis)

```
function getXMLHttpRequest() {
    if (window.XMLHttpRequest)
        return new XMLHttpRequest();
    if (window.ActiveXObject) {
        var names = [
            "Msxml2.XMLHTTP.6.0",
            "Msxml2.XMLHTTP.3.0",
            "Msxml2.XMLHTTP",
            "Microsoft.XMLHTTP" ];
        for(var i in names) {
            try {
                return new ActiveXObject(names[i]);
            } catch(e) {}
        }
    }
    window.alert("Votre navigateur ne prend pas en charge l'objet XMLHttpRequest.");
    return null;
}
```

Exemple simple

```
<html>
<body>
<script language="javascript">
function getXMLHttpRequest() {...}
function ajax() {
    var xhr=getXMLHttpRequest();
    xhr.open("GET", "test.html", false);
    xhr.send(null);
    alert(xhr.responseText);
}
</script>
<a href="javascript:ajax();">Cliquez-moi !</a>
</body>
</html>
```

Propriétés de l'objet

- `onreadystatechange` :
 - ▣ Fonction appelée à chaque changement d'état.

- `readyState` :
 - ▣ statut de l'objet :
 - 0 : non initialisé.
 - 1 : ouverture = méthode `open()` appelée avec succès.
 - 2 : envoyé = méthode `send()` appelée avec succès.
 - 3 : en train de recevoir = données en cours de transfert.
 - 4 : terminé = données chargées.

- `responseText` / `responseXML`
 - ▣ Réponse sous forme de chaîne de caractères / objet DOM.

Propriétés de l'objet

□ status :

- code numérique de réponse du serveur HTTP, à tester quand les données sont chargées (readyState=4)
 - 200 : OK.
 - 404 : page introuvable.
 - ...
- En local (sans serveur web), 0=OK

□ statusText :

- message accompagnant le code de réponse :
 - 404 : Not Found...

Méthodes de l'objet

- `open(method, url, async, user, password)`
 - ▣ Prépare une requête en indiquant la méthode, l'URL, le drapeau de synchronisation (et éventuellement le nom d'utilisateur et le mot de passe).

- `send (contenu)`
 - ▣ Effectue la requête, éventuellement en envoyant les données.

- `setRequestHeader("champ", "valeur")`
 - ▣ Assigne une valeur à un champ d'entête HTTP qui sera envoyé lors de la requête.

Méthodes de l'objet

- abort()
 - ▣ Abandonne la requête.
- getAllResponseHeaders()
 - ▣ Renvoie l'ensemble de l'entête de la réponse sous forme de chaîne de caractères.
- getResponseHeader("champEntete")
 - ▣ Renvoie la valeur d'un champ d'entête HTTP.

Synchrone ou asynchrone

- Requête synchrone :
 - ▣ Tout est bloqué en attendant la réponse.
 - Mauvais pour l'utilisateur.
 - ▣ Les réponses arrivent forcément dans l'ordre.
 - ▣ C'est l'approche classique.

- Requête asynchrone :
 - ▣ Le navigateur continue à répondre aux événements en attendant la réponse.
 - ▣ Attention à ne pas faire n'importe quoi.

Javascript Asynchrone

- Le choix entre synchrone et asynchrone se fait dans l'appel à XMLHttpRequest (méthode open) :
 - ▣ true pour asynchrone
 - ▣ false pour synchrone
- Dans le cas d'un appel asynchrone, le résultat est récupéré par une fonction :
 - ▣ `xhr.onreadystatechange = fonction() { ...};`
 - ▣ Cette fonction sera appelée à chaque changement d'état de notre objet.

Pour résumer

- Deux méthodes principales :
 - ▣ open : pour établir une connexion.
 - ▣ send : pour envoyer une requête au serveur.
- Récupération des données :
 - ▣ Champs `responseXml` ou `responseText`.
- Créer un nouvel objet `XmlHttpRequest`, pour chaque fichier à charger.

Un exemple

```
function ajax() {
  var xhr=getXMLHttpRequest();
  xhr.onreadystatechange = function() {
    if(xhr.readyState == 4) {
      if(xhr.status == 200)
        alert("Received : " + xhr.responseText);
      else
        alert("Error code : " + xhr.status);}}};
  xhr.open("GET", "test2.html", true);
  xhr.send(null);
}
```

<body>

Cliquez-moi !

</body>

HTTP GET ou POST

- GET pour récupérer des données
 - ▣ Ne devrait pas provoquer de mises à jour sur le serveur.
 - ▣ Les requêtes GET doivent pouvoir être bookmarkées ou mises en cache.
- POST pour envoyer des données
 - ▣ Pour tout ce qui ne correspond pas à un GET

AJAX : X = XML

- Le serveur renvoie des données XML
- La méthode `responseXML` de l'objet `XMLHttpRequest` renvoie un document XML à traiter
- La méthode javascript `getElementsByTagName(nom)` d'un objet (en l'occurrence XML) permet de récupérer les éléments par rapport à leur nom dans un élément.

```
var docXML= xhr.responseXML;
var items = docXML.getElementsByTagName("donnee");
for (i=0;i<items.length;i++) {
    alert (items.item(i).firstChild.data);
}
```

AJAX ou AJAJ : J = JSON

- ❑ XML pas évident à parser en Javascript (cf cours 1)
- ❑ On utilise plutôt JSON

```
{
  "menu": {
    "id": "file",
    "value": "File",
    "popup": {
      "menuitem": [
        {"value": "New", "onclick": "CreateNewDoc()"},
        {"value": "Open", "onclick": "OpenDoc()"},
        {"value": "Close", "onclick": "CloseDoc()"}]
    }
  }
}
```

Exemple d'utilisation de JSON

- Coté client :
 - ▣ JSON inclus dans JavaScript.
 - ▣ Le contenu est assigné à une variable et devient un objet.

```
// Création de la connexion :
var req = new XMLHttpRequest();
req.open("GET", "fichier.json", true);
req.onreadystatechange = function() {
    if (req.readyState == 4) {
        var doc = eval('(' + req.responseText + ')');
    }
}
req.send(null);
```

Exemple d'utilisation de JSON

- Coté serveur :
 - ▣ Parseurs pour générer du JSON à partir d'objets en Php ou JAVA
- L'échange de données :
 - ▣ Envoi avec AJAX

```
<?php
    $arr = array ('a'=>1, 'b'=>2, 'c'=>3, 'd'=>4, 'e'=>5);
    echo json_encode($arr);
?>
```

Affiche :

```
{"a":1,"b":2,"c":3,"d":4,"e":5}
```

Attention !

- Les requêtes AJAX asynchrones passent par Internet
 - ▣ Aucune garantie que les paquets arrivent dans l'ordre.
 - ▣ Aucune garantie qu'une requête soit terminée avant qu'une autre ne soit lancée :
 - Les délais peuvent varier énormément à cause de la charge du serveur et du réseau.

Inconvénients

- ❑ JavaScript doit être activé.
- ❑ Les données chargées de façon dynamique ne font pas partie de la page. Prise en compte par les moteurs de recherche pas claire.
- ❑ Asynchrone => affichage avec délai, peut poser problème à l'utilisateur.
- ❑ Le bouton « Page précédente » ne marche pas en général.

Conclusions sur Ajax

- Combinaison des langages standards du WEB (Javascript, DOM HTML, XML)
- Grâce à l'objet XMLHttpRequest
- WEB dynamique « coté client »
- Utilisé par tous les sites « WEB 2.0 »