A

JavaScript

Présentation

- Créé par Netscape (nom original : LiveScript)
- Langage interprété :
  - Interpréteur présent dans le navigateur
  - Pas de compilation vers du ByteCode
  - Code source présent dans la page
- Langage orienté objet
  - Pas d'héritage
- Syntaxe proche de Java

Evolution

- Concurrent Microsoft : jscript
  - Développement parallèle
  - Quelques incompatibilités
- Tentative de fusion : ECMAScript (DHTML)
  - ECMA = European Computer Manufacturers Association
- Possibilité d'utilisation de JavaScript côté serveur :
  - Entreprise Server (Netscape)
  - Internet Information Server (Microsoft)
  - Intra Builder (Inprise)

Balise SCRIPT

- Nouvelle balise HTML
- Indique le langage du script

```
<script language="javascript">
...
</script>
```

- Navigateurs ne supportant pas JavaScript (lynx par ex) :
  - La balise est ignorée

  $\Rightarrow$ le code apparaît dans la page
  - Solution : le mettre entre commentaires HTML

Balise SCRIPT

```
 <script language="javascript"><!--
.....
//--></script>
```

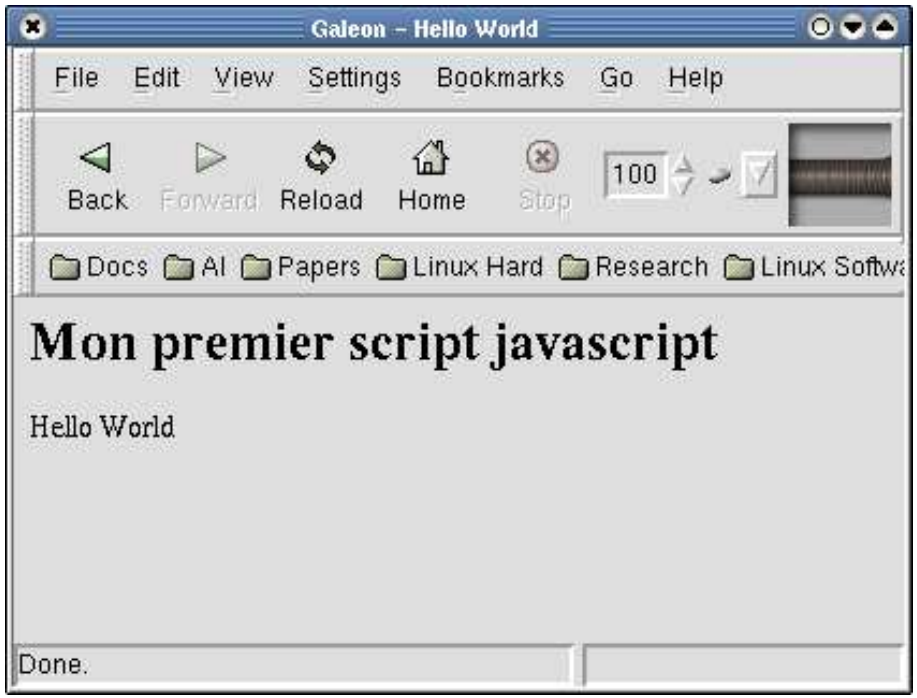
- Prévenir l'utilisateur : balise `NOSCRIPT`
- Exemple :

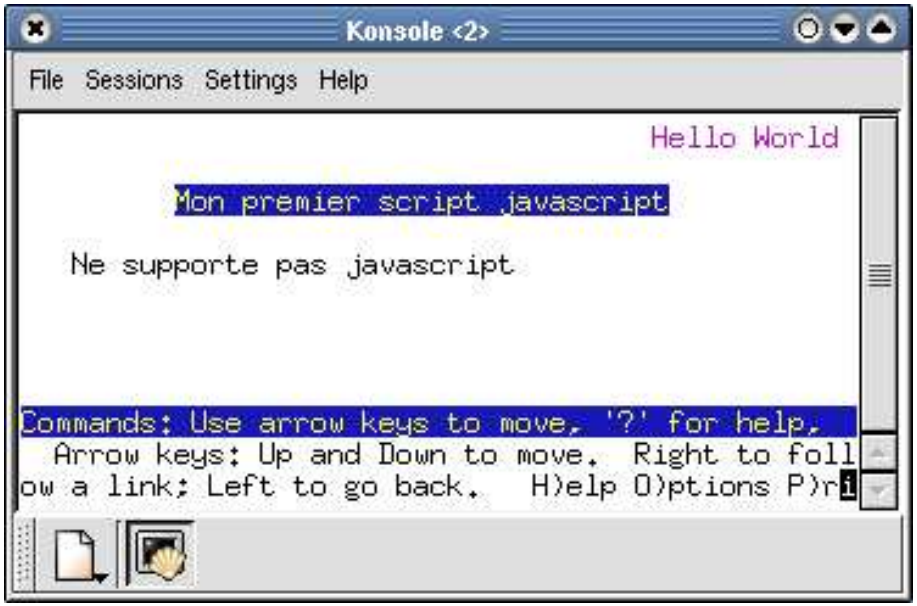
```
<noscript>
<h3>Votre navigateur ne supporte pas JavaScript</h3>
</noscript>
```

Hello World

```
<html>
  <body>
  <H1> Mon premier script javascript</H1>
  <script language="javascript"><!--
document.write("Hello World") ;
//--></script>
 <noscript>
<h3>Votre navigateur ne supporte pas JavaScript</h3>
 </noscript>
  </body>
</html>
```

Résultats


Galeon - Hello World
File Edit View Settings Bookmarks Go Help
Back Forward Reload Home Stop
100
Docs AI Papers Linux Hard Research Linux Softw
Mon premier script javascript
Hello World
Done.


Konsole <2>
File Sessions Settings Help
Hello World
Mon premier script javascript
Ne supporte pas javascript
Commands: Use arrow keys to move, '?' for help,
Arrow keys: Up and Down to move. Right to foll
ow a link; Left to go back. H)elp O)ptions P)ri

Variables et fonctions

- Variables :
  - Types prédéfinis : chaînes, nombres, booléens (true, false)
  - Pas de typage fort
  - Déclaration : `var x ;`
- Fonctions :

```
function incremente(x)
{
  var y = x+1 ;
  return y ;
}
```

Boucles, conditions, . . .

- Similaire au C
- Exemples :

```
var i = 0 ;
var x = 0 ;
while (i<10) i++;
for (i=0; i<30; i++) x += i ;
if (x == 12) document.write('oui') else document.write('non') ;
switch (i)
{
    case 1 : ...; break ;
    case 2 : ...; break ;
    default : ....
}
...
```

Tableaux

- Déclaration :
  - `var tab = new Array() ;`
  - `var tab = new Array(taille) ;`
  - `var tab = [e1, e2, e3] ;`
- Taille :
  - `tab.length ;`
- Accès aux éléments
  - `tab[i]`
- Nombreuses méthodes prédéfinies :
  - Appel : `tab.methode(vars);`

Tableaux (2)

`join(sep)`	transforme le tableau en chaîne de caractères
`pop()`	dépile et retourne le dernier élément du tableau
`push(val,..)`	ajoute des éléments en fin de tableau
`shift()`	supprime et retourne le premier élément du tableau
`unshift(val,..)`	ajoute des éléments en tête du tableau
`sort()`	trie le tableau
`reverse()`	inverse l'ordre des éléments du tableau.
`toString()`	transforme le tableau en chaîne (séparateur =',')

Exemple

```
var tab = ['1', '2', '3'] ;
tab.reverse() ; // tab vaut ['3', '2', '1']
tab.push('0') ; // tab vaut ['3', '2', '1', '0']
var chaine = tab.toString() ; // chaine vaut "3,2,1,0"
var chaine2 = tab.join(':') ; // chaine2 vaut "3:2:1:0"
```

Chaînes de caractères

- Classe prédéfinie `String`.
  - `var chaine = new String('essai') ;`
  - `var chaine = 'essai' ;`
- nombre de caractères :
  - `chaine.length ;`
- Nombreuses méthodes prédéfinies.
  - Appel : `chaine.method(args);`

Chaînes de caractères (2)

`charAt(i)`	retourne le $ieme$ caractère de la chaîne
`+`	concaténation de chaînes (ch = ch1 + ch2)
`indexOf(ch,i)`	index de la première occurrence de ch à partir de i (optionnel)
`lastIndexOf(ch, i)`	index de la dernière occurrence de ch à partir de i (optionnel)
`split(ch)`	transforme la chaîne en tableau
`match(exp)`	recherche les sous-chaînes correspondant à l'expression régulière `exp`
`search(exp)`	index de la première correspondance entre la chaîne et `exp`
`replace(exp, ch)`	remplace les occurrences de `exp` par `ch`
`substr(d,l)`	sous-chaîne de longueur l commençant en d
`substring(d,f)`	sous-chaîne entre d et f
`toLowerCase()`	passe la chaîne en minuscule
`toUpperCase()`	passe la chaîne en majuscule

Expressions régulières

- Mise en oeuvre :
  - `var reg = /exp/options ;`
  - option : `i` pour ignorer les majuscules, `g` pour rechercher toutes les correspondances.
  - exp : syntaxe identique à Perl.

Classe de caractères pour les expressions régulières

- Définition de classe en utilisant l'opérateur [ ]
- Négation de classe ou de caractères [ˆ...]

Exemples de classe de caractères

```
[aeiou]          # Classe des voyelles

[0-9]            # Classe des chiffres
                 # (équivaut à [0123456789] et \d)

[0-9a-zA-Z]      # Chiffre ou lettre

[\~\@,;\^_]      # Classe de caractères spéciaux

[^0-9]           # Caractères hors chiffres
                 # (équivaut à [^\d] ou \D)
```

Caractères spéciaux ou classes de caractères

`\a`	Alarme
`\n`	Nouvelle ligne
`\r`	Retour chariot
`\t`	Tabulation
`\f`	Saut de page
`\e`	Escape
`\d`	Caractère numérique (équivaut à `[0-9]`)
`\D`	Caractère non numérique (équivaut à `[^\d]`)
`\w`	Caractère de mot (alphanumérique) (équivaut à `[0-9a-zA-Z]`)
`\W`	Caractère de non mot (équivaut à `[^\w]`)
`\s`	Espace (équivaut à `[ \t\n\r\f]`)
`\S`	Non espace (équivaut à `[^\s]`)

Méta–caractères

`^`	Début de ligne
`$`	Fin de ligne
`\`	Supprime l'interprétation des méta–caractères
`.`	Tous les caractères sauf `\n`
`|`	Alternative. Exemple : `(0|1|2|3|4|5|6|7|8|9)`
`()`	Regroupement. Exemple : `(Frs)|f|(francs)`
`[]`	Spécification d'une classe de caractères. Exemple : `[a-A]`

Quantifieurs

`c{n}`	n instances du caractère c
`c{n,}`	au moins n instances du caractère c
`c{n,m}`	au moins n et au plus `m` instances du caractère c
`c+`	une ou plusieurs instances du caractère c (équivaut à `c{1,}`
`c*`	zéro ou plusieurs instances du caractère c (équivaut à `c{0,}`
`c?`	zéro ou une instance du caractère c (équivaut à `c{0,1}`

Exemples

Mot	`[a-zA-Z]*`
Mot de 2 lettres	`[a-zA-Z]{2}`
Nombre entier	`[0-9]+`
Nombre réel	`([0-9]+\.[0-9]*)|(\.[0-9]+)`
Variable C/C++	`[_a-zA-Z]+[_a-zA-Z0-9]*`

Utilisation

- Expression régulière = objet.
- Méthodes :
  - `reg.test(chaine)` renvoie *true* si la chaîne contient l'expression régulière.
  - `reg.exec(chaine)` recherche l'expression régulière dans la chaîne.

Exemple

```
var chaine = "hello world" ;
var reg = /\w+/ ;
var reg1 = /\w+/g ;

var res = reg.test(chaine) ; // res == true

var mot ;
mot = reg.exec(chaine) ; // mot == hello
mot = reg.exec(chaine) ; // mot == hello

mot = reg1.exec(chaine) ; // mot == hello
mot = reg1.exec(chaine) ; // mot == world
mot = reg1.exec(chaine) ; // mot == null
mot = reg1.exec(chaine) ; // mot == hello
```

Utilisation (suite)

- Utilisation dans des méthodes de la classe String :
  - `replace`, `search`, `match`
- Exemple :

```
var ch  = "00 12 34".replace(/\d{2}/, "xx") ;
var res = "00 12 34".match(/\d{2}/g) ;
// ch  == "xx 12 34"
// res == [00,12,34]
```

JavaScript

- Principe :
  - Décharger le serveur de traitements effectués côté client.

  $\Rightarrow$ Pouvoir modifier localement la page `html` envoyée par le serveur.

  $\Rightarrow$ Pouvoir réagir aux événements utilisateurs
- Deux aspects :
  - Gérer les évènements.
  - Représentation en JavaScript de la structure et du contenu du document.

Evénements

- Associé aux balises HTML
- Gestion :
  - Ajout du paramètre onEvent="..." à la balise.
- Exemple :

```
<html>
  <head>
    <title>Hello World</title>
  </head>
  <body>
   <form>
    <input value="Click" onClick="alert('You did it !')">
   </form>
  </body>
</html>
```

Evénements (2)

nom	événement	s'applique à
onAbort	interruption de chargement	image
onBlur	perte du focus	champs de saisie, window
onFocus	attribution du focus	champs de saisie, window
onClick	clic sur un objet	button, checkbox, radio, reset, submit, lien
onChange	changement de champ de saisie	fileupload, password, select, text, textarea
onDblClick	double clic	lien, image, bouton
onKeyDown	touche enfoncée	document, image, lien, text
onKeyPress	l'utilisateur a appuyé sur une touche	document, image, lien, text
onKeyUp	touche relâchée	document, image, lien, text
onLoad	chargement	image, window
onMouseDown	un bouton de la souris est enfoncé	document, lien, image, button
onMouseUp	le bouton de la souris est relâché	document, lien, image, button
onMouseMove	déplacement de la souris	document, lien, image, button
onMouseOut	la souris vient de quitter une zone	lien, image, layer
onMouseOver	la souris entre sur une zone	lien, image, layer
onReset	annulation des données saisies	formulaire
onSubmit	soumission du formulaire	formulaire
onSelect	sélection d'un champ de texte	champs de saisie

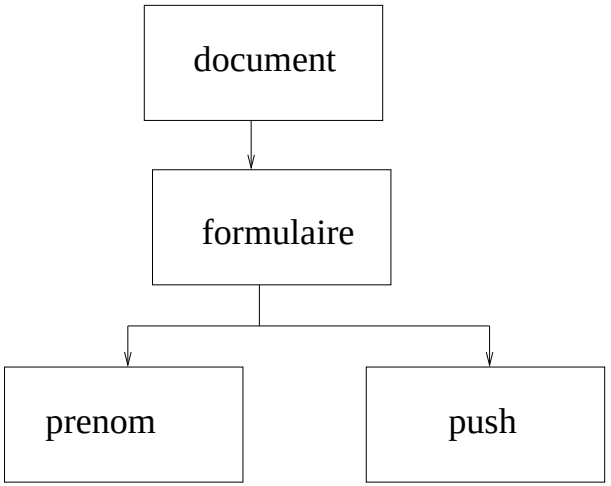
Document HTML

- Document HTML = arbre
- Exemple :

```
<html>

<body>
<form name="formulaire">
<input type="text" name="prenom"
       value="">
<input type="button" name="push">
</form>
</body>

</html>
```



JavaScript

- Représentation sous forme d'objets.
- Composition (structure du document).
- Propriétés (contenu du document).
- Sommet : `document`
- Exemple précédent :

```
document.formulaire.prenom.value = "patrick" ;
var chaine = document.formulaire.prenom.value ;
```

- Autre accès :

```
<input type="text" id="nom" name="nom">
document.getElementById('nom') ;

document.getElementsByTagName("tagname")[n] ;
```

navigator

- Quelques propriétés :

`appName`	nom du navigateur
`appVersion`	numéro de version
`language`	langue supportée
`mimeTypes`	tableau des type MIMES reconnus
`platform`	système d'exploitation
`plugins`	liste des plugins installés

screen

- Quelques propriétés :

`width`	largeur en pixel de l'écran
`height`	hauteur en pixel de l'écran
`availHeight`	hauteur de l'écran moins la barre de menu ou de tâche
`availWidth`	largeur de l'écran moins la barre de tâche
`colorDepth`	nombre de couleurs

window

- Quelques propriétés :

`closed`	indique si la fenêtre est fermée.
`defaultStatus`	message par défaut à afficher dans la barre de status.
`document`	page HTML associée à la fenêtre.
`history`	historique de navigation associé à la fenêtre.
`location`	URL de la zone.
`screen`	référence vers l'objet `screen`.
`status`	message de status .

window (2)

- Quelques méthodes :

`open(url,name,feature)`	feature = liste d'options séparées par des virgules : – `width, height` – `screenX, screenY` – `status, toolbar, menubar` – `resizable`

- Exemple :

```
var win = open("http://www-ufrima", "",
               "menubar=yes,width=800,height=600") ;
```

window (3)

- Quelques méthodes :

`close()`	ferme la fenêtre
`alert(message)`	ouverture d'une fenêtre d'alerte avec un boutton `ok`
`confirm(message)`	ouverture d'une fenêtre de confirmation (boutton `cancel` et `ok`). Retourne un booléen.
`prompt(message, default)`	ouverture d'une fenêtre de requête (message + zone de saisie de texte)
`setTimeout(func,msec,arg1,..,argn)`	appel périodique d'une fonction. Retourne un identificateur
`clearInterval(id)`	supprime l'appel périodique d'une fonction

history

- Quelques propriétés :

`current`	URL courante
`length`	nombre d'entrée dans l'historique
`next`	URL suivante
`previous`	URL précédente

- Quelques méthodes :

`back()`	charge l'URL précédente
`next()`	charge l'URL suivante
`go(d)`	se déplace de d éléments
`go(location)`	location représente une partie de l'URL recherchée

location

- Quelques propriétés :

`hostname`	nom du serveur
`href`	url courante

- Quelques méthodes :

`reload(force)`	force (optionnel) permet de passer outre le cache
`replace(url)`	remplace l'URL courante par la nouvelle (y compris dans l'historique

document

- Quelques propriétés :

`linkColor`	couleur des liens (”0000FF” ou ”blue”)
`bgColor`	couleur de fond
`fgColor`	couleur du texte
`title`	contenu du tag `TITLE`
`URL`	URL du document
`vlinkColor`	couleur des liens visités

- Quelques méthodes :

`open(mime, replace)`	génération dynamique du contenu. Les paramètres sont optionnels (`replace` concerne l’historique)
`close()`	fermeture du document et affichage du contenu
`write()`	envoie de la chaîne au document

Autres

- `Textarea.value`
- `Radio.checked`
- `Select.selectedIndex`
- `Select.length` (nombre de champs sélectionnés)

Exemple

```
<html>
  <head>
    <title></title>
    <meta content="">
    <style></style>
  </head>
  <body>
  <h2>Exemple d'aide sur un lien</h2>
  <p>
  <a href="http://localhost"
      onMouseOver="window.status='Page d\'acceuil du serveur'; return true ;"
 cliquez </a> ici.
  </body>
</html>
```

Exemple (2)

```
<html>

  <body>
  .....
  <br>
  <input type="button" value="back" onClick="history.back()">
  </body>

</html>
```

Exemple (3)

```
<html>
<head>
  <title>Formulaire</title>
</head>
<body>
  <h1 align="center">Exemple de confirmation</h1>
  <p>
  <form action="cgi-bin/formulaire" method="GET"
        onSubmit="return confirm('Vous êtes sûr ?')">
  .....
    <input type="submit" value="Valider">
  </form>
</body>
</html>
```

InnerHTML

Javascript + XML

B

Ajax

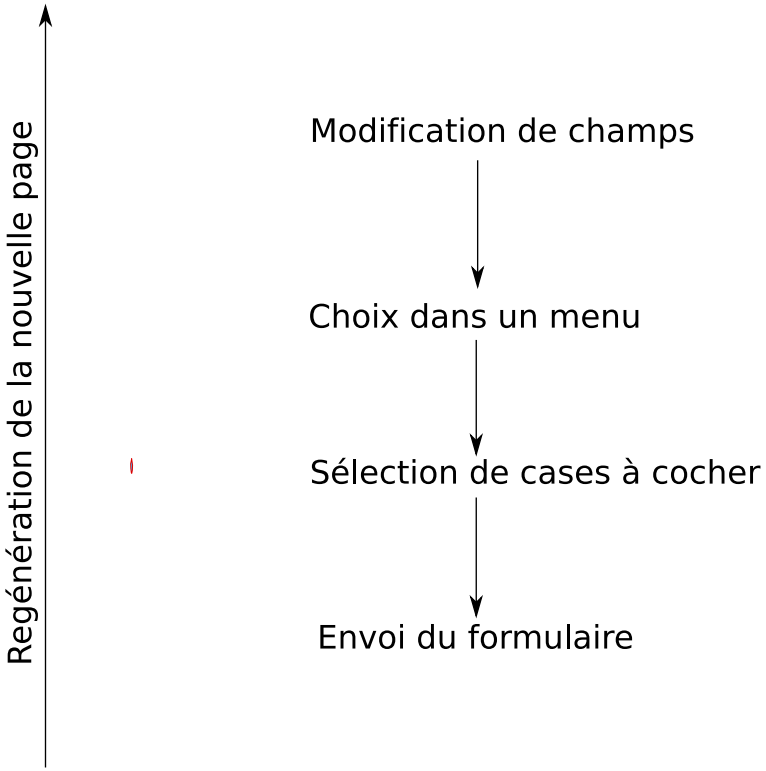
Concepts (banalités ....)

- Deux grandes catégories d'applications interactives :
  - Applications "de bureau"
  - Applications web
- Applications de bureau :
  - Installation préalable sur la machine
  - Grande réactivité
- Applications web :
  - Mise en place immédiate au travers un navigateur
  - Réactivité faible :
    - Mise à jour = rechargement de la page

    $\Rightarrow$ l'application est indisponible pendant 1 à plusieurs secondes
  - Application distribuée.

Interaction Web

- Interaction classique :



Objectif

- Se rapprocher du comportement d'une application de bureau
- Permettre le rafraîchissement partiel d'une page
  - Dès l'interaction avec un widget
  - Sans bloquer le reste de la page :
    - Prise en compte du délai d'accès au serveur
    - Asynchrone

Ajax

- Asynchronous Javascript And Xml
- Nouvelle utilisation d'anciennes technologies :
  - Javascript
  - DOM (modification du document html)
  - XML
  - XMLHttpRequest()
- Utilisé la première fois par Microsoft (Outlook Web Access)
- Popularisé par Google
  - Google Suggest, Google Mail, Google Maps
- Exemple d'utilisation :
  - auto-complétion, listes déroulantes liées, . . .

Principe

- Réaction à un événement utilisateur.
- Déclenchement en javascript d'une requête vers le serveur.
- Réception *asynchrone* de la réponse.
- Modification de la page en cours.

XMLHttpRequest

- Envoi d'une requête au serveur
- Introduit par Microsoft (1997, IE5)
- Repris par les autres navigateurs :
  - Firefox, Mozilla, Opera, Konqueror, Safari . . .

XMLHttpRequest

- 3 façons différentes de l'instancier
- Microsoft :
  - Objet ActiveX
  - Pas bloqués par la politique de sécurité
  - 2 versions différentes
- Les autres

XMLHttpRequest

```
var req ;

function getXmlRequest()
{
  if (window.XMLHttpRequest)
       req = new XMLHttpRequest() ;
  else if (window.ActiveXObject)
  {
      try {
                    req = new ActiveXObject("Msxml2.XMLHTTP") ;
           } catch (e) {
                   req = new ActiveXObject("Microsoft.XMLHTTP") ;
           }
  }
  else
     alert ("Votre navigateur ne supporte pas Ajax") ;
}
```

Mise en place de la requête

- Requête asynchrone

$\Rightarrow$ Mise en place d'une callback pour la réception de la réponse.

- Utilisation d'une fonction javascript :

```
req.onreadystatechange = maFonction ;
```

- Utilisation d'une fonction anonyme :

```
req.onreadystatechange = function()
{
...
}
```

Envoi de la requête

- `open("méthode", "url", "type")`
  - **methode** : *GET* ou *POST*
  - **url** : URL de la requête
  - **type** : booléen. True = asynchrone, False = synchrone
- Méthode POST :
  - Encodage des paramètres en `x-www-form-urlencoded`

```
req.setRequestHeader("Content-Type", "x-www-form-urlencoded")
```

  - Spécification des paramètres et envoi de la requête :

```
send("a=3&b=4")
```

- Méthode GET :
  - Spécification des paramètres : sur l'URL
  - Envoi de la requête : `req.send(null)`

Réception de la réponse

- Tester l'état de la réponse :
  - `req.readyState`
  - 0 : non initialisé
  - 1 : Ouverture (juste après `open`)
  - 2 : Envoyé (juste après `send`)
  - 3 : En cours
  - 4 : Réponse arrivée
- Tester le code de la réponse HTTP :
  - 200 : réponse ok
  - 404, 500, ... : erreur sur la requête
  - Inutile de traiter les codes de redirection :
    - Prise en charge direct par le navigateur

Réception de la réponse

- Réponse texte :
  - `req.responseText`
- Réponse XML :
  - `req.responseXML`

Côté serveur

- Positionner le type mime correspondant à la réponse (envoi d' XML) :
  - `Content-Type: application/xml`
- Penser à ne pas mettre la réponse en cache :
  - `Cache-Control: no-cache`

Exemple simple : côté client

```
<SCRIPT language="javascript">
var req ;
function getXmlRequest()
{
  if (window.XMLHttpRequest)
       req = new XMLHttpRequest() ;
  else if (window.ActiveXObject)
  {
      try {
                    req = new ActiveXObject("Msxml2.XMLHTTP") ;
            } catch (e) {
                     req = new ActiveXObject("Microsoft.XMLHTTP") ;
            }
  }
  else
     alert ("rate !") ;
}

function newMessage()
{
   if (req.readyState == 4 && req.status == 200)
   {
         document.getElementById("message").value = req.responseText ;
   }
}
```

Côté client

```
function makeRequest()
{
    getXmlRequest() ;
    req.onreadystatechange = newMessage ;
    req.open("GET", "go.php", true) ;
    req.send(null) ;
}
</SCRIPT>
<h3>Première page Ajax (sans X!)</h3>
<br>
<br>
<input value="Appuyez" type="button" onClick="makeRequest()">
<br>
<br>
<input type="text" id="message">
```

Côté serveur

```
<?php

echo "Bonjour : " . date("h:m:s");

?>
```

Exemple 2 : utilisation de l'attribut inner

- Objectif : menus liés.
- Choix d'un premier menu détermine le contenu d'un second menu
- Principe :
  - Détection du changement de sélection du premier menu : `onChange="..."`
  - Envoi d'une requête
  - Réception du code HTML du nouveau menu

Menus

```
<form method="GET" action="go">

    <select name="marque">
       ...
    </select>

   <div id="modele" style="display:inline">
   <select name="modele" id="modeleId">
       <option><-- Choisissez une marque</option>
   </select>
   </div>
```

Mise à jour du menu

```
if (req.readyState == 4 && req.status == 200)
 {
     document.getElementById("modele").innerHTML=req.responseText ;
}
```

- Réponse asynchrone
  - Le chargement du nouveau menu peut ne pas être instantanné
  - Mais l'utilisateur peut toujours interagir

  $\Rightarrow$ Au lancement de la requête, invalider le menu qui va être changé
  - Le revalider lors de la mise en place du nouveau code.

- innerHTML :
  - Solution la plus simple :
    - retourner la liste des *options*
    - Affecter cette liste à l'*innerHTML* du select
  - Incompatible avec certains navigateurs (cf IE)

  $\Rightarrow$ Utilisation d'un div

Exemple avec Struts

- Utilsation d'un contrôleur pour récupérer la liste des entrées du menu
- Forward vers une JSP chargée de générer le texte :

```
<%@page contentType="text/plain"%>
<%@page pageEncoding="UTF-8"%>
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>


<select name="..." id=".....">
<c:forEach .... >
    <option><c:out value="${..}"/></option>
</c:forEach>
</select>
```

Frameworks

- Existence de framework pour la prise en charge d'Ajax :
  - Javascript pur :
    - *rico*, *protoype* : `http://openrico.org`
    - Attention à la compatibilité des navigateurs
  - Tags JSP utilisant Ajax
    - ajax anywhere : `http://ajaxanywhere.sourceforge.net/`
    - WebParts (AjaxTags) : `http://javawebparts.sourceforge.net/`
    - AjaxTags : `http://ajaxtags.sourceforge.net/`