AJAX

Le web 2.0

E. Viennet

IUT de Villetaneuse
Université Paris 13

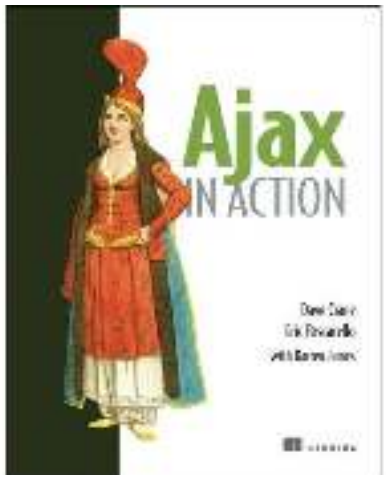
DUT Informatique - S3 2009

Plan

Le langage JavaScript

Bibliographie

JavaScript, the definitive guide. David Flanagan, O'Re



Ajax in Action. David Crane et al., Manning.

Lien utiles

- Outils et bibliothèques
  - FireBug http ://www.getfirebug.com/ (extension Firefox)
  - Bibliothèques :
    - ★ jQuery `http://jquery.com`
    - ★ Prototype `http://prototype.conio.net`
    - ★ ...
- Documentation en ligne
  - Mozilla JavaScript Reference
    `http://developer.mozilla.org/en/docs/JavaScript`
  - Mozilla DOM reference
    `http://developer.mozilla.org/en/docs/DOM`
  - (docs en français sur `http://developer.mozilla.org/fr/docs`)

Généralités

JavaScript est un langage interprété, normalement embarqué dans un navigateur web. Ce n'est pas du Java interprété (mais la syntaxe ressemble un peu).

- Normalisation : ECMA (`http://www.ecma-international.org`)
- Langage utilisé au départ pour vérifier des formulaires HTML, s'est enrichi de manière désordonnée (mais non libre).
- Autres environnement d'exécution : Flash (ActionScript, Acrobat Reader, PhotoShop, ...)

JavaScript est supporté par la plupart des navigateurs. Firefox possède de nombreux bons outils de développement, sous forme d'extensions.

JavaScript & page (X)HTML

Code inclus dans la page

```
<script langage="JavaScript">
... code ...
</script>
```

Code dans un fichier séparé

```
<script language="JavaScript"
        type="text/javascript"
        src="URL">
</script>
```

Le code JavaScript est chargé à l'URL indiquée.

Exemple (DOM + JavaScript)

```
<script langage="JavaScript">
function double() {
    var Form = document.getElementById("MonFormulaire");
    var valeur = Form.valeur.value;
    var r = document.getElementById("resultat");
    r.innerHTML = valeur + valeur;
}
</script>

<body>

<form id="MonFormulaire">
<input type="text" id="valeur" onkeyup="double();"/>

<p>Double=<span id="resultat"></span></p>
</form>
</body>
```

Quelques conseil avant de commencer...

Le code JavaScript peut être pénible à mettre au point.

- utilisez un débugueur ;
- soignez la présentation (indentation du code) ;
- ne jamais utiliser de variables globales (utiliser le mot clé `var`).

Types de base

- Nombres
- Chaînes de caractères
- Booléens
- Fonctions
- Objets
- Array
- Null, undefined

Nombres

Tous les nombres sont des flottants (pas d'entiers), en 64 bits (IEEE 754).

- Opérateurs habituels : +, -, /, *, %
- Math :

  ```
  Math.PI, Math.E, ...
  Math.abs(x), Math.cos(x), Math.sqrt()
  Math.ceil(x), Math.floor(x), Math.round(x)
  Math.random()
  ```

- Conversions : `parseInt("123")`, `parseFloat("3.14")`
- `NaN`, `Infinity`

  ```
  1/0 == Infinity
  NaN + 1 == NaN  (fonction: isNaN(x))
  ```

Chaînes de caractères

Séquences de caractères.

```
"Bonjour"
```

Les chaînes sont des objets :

```
n = "Bonjour".length
```

Les caractères sont des chaînes de longueur 1 :

```
c = "bruit".charAt(0)
```

Nombreuses méthodes utiles (voir doc)

```
"Feu rouge".replace("rouge", "vert")
"Bobo".toUpperCase()
```

Null, undefined et booléens

Deux types spéciaux ayant une seule valeur :

- `null` : variable sans valeur (comme en SQL)
- `undefined` : variable non initialisée (ou non existante)

Booléens : `true`, `false`
Opérateurs logiques `&&`, `||`, `!`

Variables

Déclaration d'une variable :
Utiliser `var` :

```
var x;
var pi = 22/7; // approximatif
```

Toujours utiliser `var`, sinon la variable est globale, et c'est le début de la fin...
Si la variable n'est pas initialisée, elle est égale à `undefined`.

Opérateurs

- `+, -, *, /, %`
- `+=, -=, *=, /=`
- `a++, b--, --a, ++a`
- `"Salut " + "Toto" == "Salut Toto"`
- `"3" + 45 == 48`

Strutures de contrôles

- Comme en C ou Java...
  (`if, while, do { ... } while (); switch`)
- Boucle for : `for (var i =0; i < 10; i++) { ... }`
- Exceptions : try/catch/finally

Objets

Deux notations équivalentes :

- `var obj = new Object();`
- `var obj = {}`

Accès aux membres (“propriétés”) :

- `obj.nom = "Toto";`
- `obj["name"] = "Toto";`
  (notez que "name" peut être une variable)

Notation :

```
var obj = {
   nom : "Toto";
   prenom : "Marcel";
   age : 22;
}
```

Tableaux (array)

Les tableaux sont juste une classe spéciale d'objets. Les clés sont des nombres (indices).

Les tableaux sont donc hétérogènes :

```
var A = new Array();
A[0] = "Fromage";
A[1] = 3.12;
A[2] = new Object();
// ici A.length == 3
```

Autre notation :

```
var A = [ "Fromage", 3.12,  new Object() ]
```

Ajout en fin :

```
A[A.length] = 12;
```

Fonctions

Très simple, arguments non typés statiquement :

```
function ajouter( x, y ) {
  var somme = x + y;
  return somme;
}
```

- Si pas de valeur retournée, la fonction retourne `undefined` ;
- si un argument manque, il est aussi `undefined`.

JavaScript et document HTML

Comment utiliser JavaScript pour interagir avec le navigateur ?
Le plus primitif (mais parfois utile) : `alert("Salut !");`
Pas de saisie clavier, pas d'entrées sortie (sécurité !)

On veut que le programme JS puisse :

Interagir avec la page web

- récupérer des valeurs (champs de formulaires) ;
- modifier la page

⇝ interaction avec le DOM : *Modèle Objet de Document*

Communiquer avec le serveur web !

⇝ XMLHTTPRequest

Exercices

1. Ecrire une page web qui lors de son chargement affiche un message "Bienvenue" (fenêtre popup) lors de son chargement ;.
2. Modifier le script pour qu'il affiche successivement les phrases contenues dans un tableau. Par exemple si

```
var Messages = [ "Bienvenue", "sur ce site",
                 "ennuyeux, non ?" ];
```

   la page affichera successivement 3 fenêtres popups.
3. Reprendre l'exemple (transparent 8) et modifier le pour qu'il affiche la phrase entrée à l'envers (si on saisi “ABC”, on verra “CBA” en dessous.

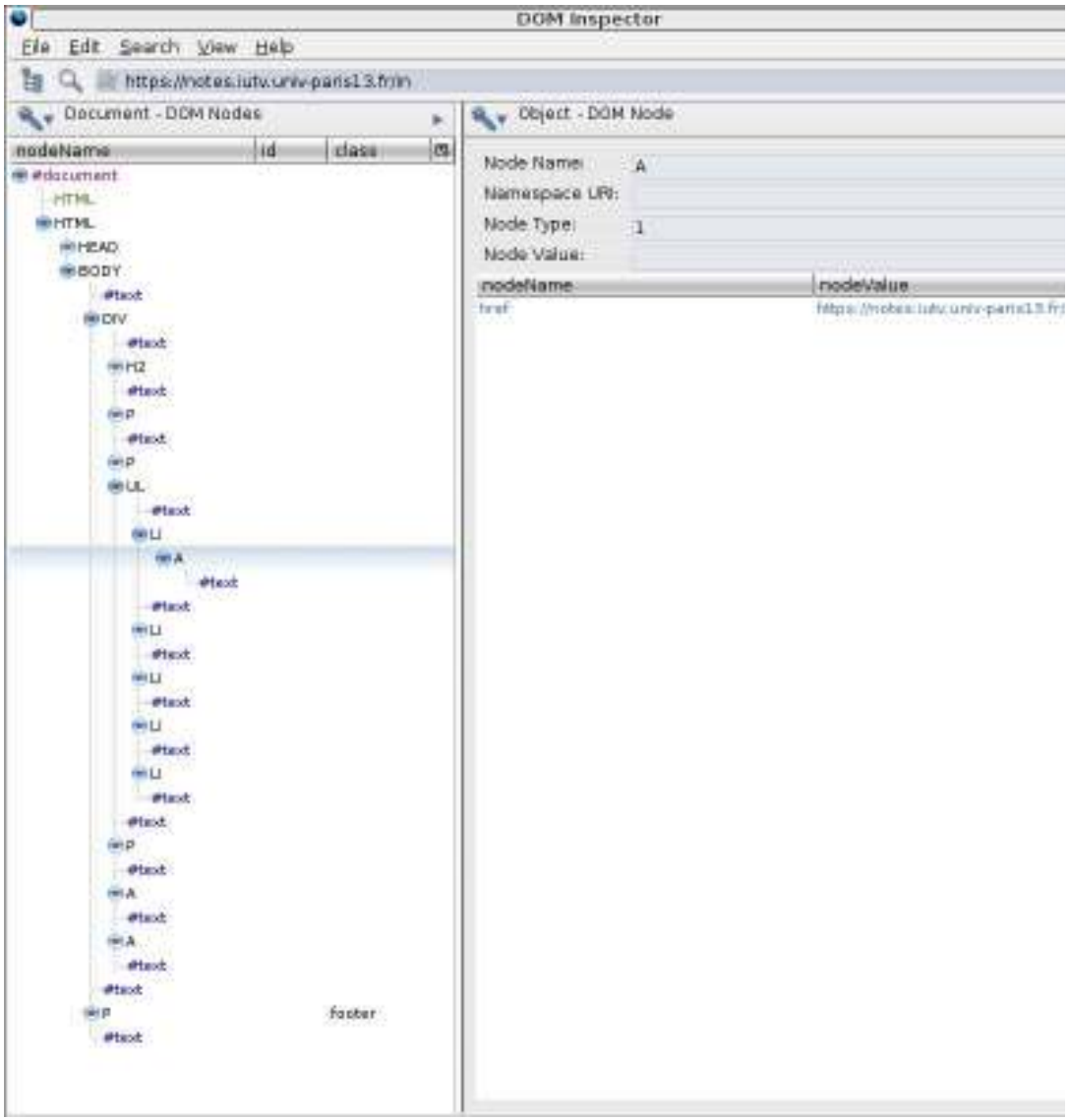
Le Modèle Objet de Document (DOM)

Le DOM est une interface permettant aux programmes de manipuler les documents. Le document (XHTML) est vu comme un arbre. Le DOM ne dépend pas d'un langage particulier (mais nous utiliserons JavaScript). Le DOM est défini par le W3C.

- `http://www.w3.org/DOM`
- `http://developer.mozilla.org/en/docs/Gecko_DOM_Reference`

Le DOM défini la struture d'un document (arbre, éléments) et une API pour y accéder.

Inspecteur DOM



API JavaScript / DOM

Quelques méthodes importantes...

Recherche d'un nœud du DOM

- `document.getElementById( id )`
- `parentNode`, `childNodes`
- `getElementsByTagName( name )`

Création de nœuds

- `document.createElement( type )`
- `document.createTextNode( text )`
- `element.appendChild( element )`
- raccourci commode : `element.innerHTML`

Style CSS

$\rightsquigarrow$ propriétés `className` et `style`

Exemple : DOM (helloDOM.html)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http
<html>
<head>
<title> Essai DOM </title>
<link rel="stylesheet" type="text/css" href="helloDOM.css" />
<script type="text/javascript" src="helloDOM.js" />
</head>

<body>

<p id="bonjour">Bonjour</p>

<div id="vide"></div>

</body>
</html>
```

Exemple : DOM (helloDOM.css)

```
/* Elements "statiques" */
.statique {
  color: blue;
  font-family: arial;
  font-weight: normal;
}

/* Elements ajoutes */
.dynamique {
  color: red;
  font-family: helvetica;
  font-weight: bold;
}
```

Exemple : DOM (helloDOM.js, part. 1)

```
window.onload = function() {
  var bonjour = document.getElementById('bonjour');
  bonjour.className = 'statique';

  var vide = document.getElementById('vide');
  ajouterNoeud( vide, "Etudiant de l'IUT" );
  ajouterNoeud( vide, "de Villetaneuse" );

  // change les styles
  var child = vide.childNodes;
  for (var i=0; i < child.length; i++) {
    child[i].className = "dynamique";
  }

  // on peut aussi agir directment sur le style CSS:
  vide.style.border = 'solid green 2px';
  vide.style.width = "200px";
```

Exemple : DOM (helloDOM.js, part. 2)

```
// Ajoute un noeud (texte)
function ajouterNoeud(el, text) {
  var childElement = document.createElement("div");
  el.appendChild(childElement);
  var txtNode = document.createTextNode(text);
  childElement.appendChild(txtNode);
}
```

Récupération du texte “dans” un élément

(et tous ses descendants)

```
function getInnerText(el) {
  var str = "";
  var cs = el.childNodes;
  var l = cs.length;
  for (var i = 0; i < l; i++) {
    switch (cs[i].nodeType) {
    case 1: //ELEMENT_NODE
      str += getInnerText(cs[i]);
      break;
    case 3: //TEXT_NODE
      str += cs[i].nodeValue;
      break;
    }
  }
  return str;
}
```

Exercices

1. Écrire une page XHTML avec un menu “rouge, vert, bleu, blanc” qui permette de changer la couleur de fond de la page.

2. Écrire une page avec des paragraphes de classe “texte”. En bas de la page, indiquer le nombre de total de caractères dans les paragraphes de classe “texte”. Cette valeur doit être calculée par une fonction JavaScript après le chargement de la page.

3. Écrire une page XHTML contenant un tableau. Chaque colonne a un titre (élément th). Lorsqu'on clique sur un titre, trier les lignes du tableau selon les valeurs de cette colonne.
(NB : on pourra regarder, voire utiliser, le code disponible : `http://www.kryogenix.org/code/browser/sorttable`).

Communication asynchrone client/serveur

Le code JavaScript peut communiquer avec le serveur web via le protocole HTTP.
Par sécurité, seul le serveur d'origine peut être contacté ($\rightsquigarrow$ URLs relatives).

Requêtes principales :

- HTTP GET
- HTTP POST

Plusieurs techniques existent (utilisation de frame, iframe), mais la plus pratique (et moderne) et d'utiliser l'objet XMLHTTPRequest.

XMLHTTPRequest existe dans IE, Firefox, Mozilla et Safari, mais diffère légèrement d'un navigateur à l'autre...

Création d'une instance de XMLHTTPRequest

Le code suivant fonctionne sur les navigateurs Mozilla (Firefox) et Safari (appel à `XMLHttpRequest()`, ainsi que sur Internet Explorer (composant ActiveX).

```
function getXMLHTTPRequest() {
  var xRequest = null;
  if (window.XMLHttpRequest) {
    // Mozilla / Safari
    xRequest= new XMLHttpRequest();
  } else if (typeof ActiveXObject != "undefined"){
    xRequest= new ActiveXObject("Microsoft.XMLHTTP"); // IE
    // note: on peut raffiner pour utiliser
    // d'autres versions d'IE
  }
  return xRequest;
}
```

Envoi d'une requête

```
function envoiRequete() {
  req = getXMLHTTPRequest();
  if (req) {
    req.onreadystatechange = traiteReponse;
    req.open( "GET", "/cgi-bin/getinfos.cgi", true);
    req.setRequestHeader(
      "Content-Type", "application/x-www-form-url-encoded");
    req.send();
  } else {
    alert("envoyerRequete: pas de XMLHTTP !");
  }
}
```

L'attribut `onreadystatechange` spécifie une fonction (callback) appelée automatiquement lorsque la requête change d'état (traitement *asynchrone*).

Codes d'états

Définissons quelques constantes :

```
var READY_STATE_UNINITIALIZED=0;
var READY_STATE_LOADING=1;
var READY_STATE_LOADED=2;
var READY_STATE_INTERACTIVE=3;
var READY_STATE_COMPLETE=4;
```

Exemple de traitement de la réponse

```
function traiteReponse() {
  var ready = req.readyState;
  //alert("reponse, ready=" + ready);
  if (ready==READY_STATE_COMPLETE) {
    var vide = document.getElementById('vide');
    data = req.responseText;
    var status = req.status;
    if (status==200) {
      // insere resultat dans document
      vide.innerHTML = data;
    } else {
      vide.innerHTML = "erreur serveur, code "
                       + status;
    }
  }
}
```

Script CGI (en bash)

Le script suivant renvoie juste une chaîne de caractères au client...

```
#!/bin/bash


echo 'Content-type: text/plain'
echo

echo 'voici une reponse'
```

- En général on utilise côté serveur du PHP, du Python ou du Java...
- Si les données à renvoyer sont structurées, on utilise en général un format XML.

Page XHTML pour l'exemple précédent

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title> Essai XMLHTTPRequest </title>
<script type="text/javascript" src="helloXMLHttp.js" />
</head>

<body>
<p id="bonjour">Bonjour</p>

<div id="vide"></div>

<p style="color: red" onclick="envoiRequete()">
charger !
</p>
</body>
```

Exercices

1. Modifier le script CGI en introduisant un délai dans le traitement (sleep). Modifier le code de `traiteReponse` pour afficher les autres états (par exemple “informations en cours de chargement...”.
2. Soit un fichier texte sur le serveur, au format suivant :

   ```
   nom    ville    pays    age
   ```

   - Écrire deux scripts CGI (en bash), l'un renvoyant simplement la liste des noms, l'autre le nom, la ville, le pays et l'age correspondant à un nom passé en argument (utiliser `QUERY_STRING`, cut et grep).
   - Lors du chargement de la page, créer un menu qui permette de sélectionner un nom parmi la liste (la liste sera demandée au serveur).
   - Lorsque l'utilisateur choisi un nom dans le menu, afficher les informations correspondantes à cet utilisateur.

Conclusion (provisoire)

Nous n'avons fait qu'effleurer les bases d'AJAX.
En particulier, il faudrait vous familiariser avec l'utilisation du XML.
Dans les applications réelles, on utilise en général un toolkit existant.
Citons en vrac :

- Prototype http ://prototype.conio.net
- OpenRico http ://openrico.org
- Google Web Toolkit http ://code.google.com/webtoolkit/
- JSON-RPC http ://json-rpc.org/
- Scriptaculous http ://script.aculo.us
- http ://www.potix.com/
- Mochikit http ://mochikit.com
- etc etc ... sujet très actif en ce moment ...