INF4375 - Paradigmes des échanges Internet

Javascript client-side

Jacques Berger

Objectifs

Introduire les scripts côté client

Prérequis

HTML

CSS

Javascript

Langage de scriptage

Interprété par le fureteur

Typage faible et dynamique

Langage sans license

Javascript

Originalement conçu par Netscape

Standardisé par ECMA International

Maintenant supporté par tous

Standards

Le standard : ECMA-262
Le nom réel : ECMAScript
Version courante : ECMAScript Edition 5

Nom des implémentations :

- Mozilla : Javascript
- Microsoft : JScript
- Adobe : ActionScript

Utilité

Outil de programmation pour les développeurs web

Permet de modifier une page dynamiquement sans passer par le serveur

HTML

On inclut le Javascript dans le document HTML avec la balise script

```
<script type="text/javascript">
    /* Code JS */
</script>
```

HTML

Fichier Javascript externe
Extension habituelle : .js

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

Syntaxe

Style C

Le ; est optionnel à la fin des instructions
    Bonne pratique : mettre le ;

Si le ; est omis, la fin de ligne termine l'instruction

Délimiteur de bloc : {}

Syntaxe

Sensible à la casse

Commentaires :

```
//
/* … */
```

Variables

On déclare les variables avec le mot-clé var

```
var element = document.getElementById("test");
var index = 0;
```

Variables

La déclaration de la variable (avec var) est optionnelle

```
result = tableLength + index;
```

Par contre, toute variable qui n'est pas déclarée explicitement devient une variable globale

Opérateurs

Les opérateurs arithmétiques, logiques et d'affectation sont les mêmes qu'avec Java

Arithmétique : + - * / % ++ --
Logique : && || ! < > <= >=
Affectation : = += -= /=

Opérateurs

Vérifie l'égalité des valeurs : ==

Vérifie l'égalité des valeurs et des types : ===

Et l'inverse :
Des valeurs : !=
Des valeurs et des types : !==

Chaînes

Délimiteurs de chaînes : "chaine" ou 'chaine'

Concaténation de chaînes : +

```
var entier = 5;
var chaine = "3 pommes";
var resultat = entier + chaine;

resultat vaut "53 pommes"
```

Conditions

Structure conditionnelle de base
if, else if, else

```
if (mark < 60) {
    tdElement.style.backgroundColor = "#FF0000";
} else if (mark > 90) {
    tdElement.style.backgroundColor = "#00FF00";
} else {
    tdElement.style.backgroundColor = "#FFFFFF";
}
```

Conditions

Le switch, pour les choix multiples (identique à Java)

```
switch (entier) {
    case 0:
        …
        break;
    case 1:
        …
        break;
    case 2:
        …
        break;
    default:
        …
}
```

Boucles

Comme Java :

- for
- while
- do/while
- break
- continue

Fonctions

Déclaration

```
function nomDeLaFonction(var1, var2, var3) {
…
}
```

Exemple :

```
function changeBGColor(colorName) {
    var bodyElement = document.getElementsByTagName("body")[0];
    bodyElement.style.backgroundColor = colorName;
}
```

Fonctions

Une fonction peut également retourner une valeur avec le mot-clé return

```
function numberOfTables() {
    return document.getElementsByTagName("table").length;
}
```

Document

La variable document permet d'accéder ou de modifier le contenu de la page

Cette variable implémente un parser DOM

DOM

Document Object Model

Permet de lire et modifier les documents HTML et XML

DOM

Représentation du document sous forme d'arborescence

Navigable de façon aléatoire car le document est entièrement chargé en mémoire

DOM

Méthodes du document

getElementById : trouver un élément HTML selon son attribut id

Ne retourne qu'un seul élément

```
var spacer = document.getElementById("spacer");
```

DOM

Méthodes du document

getElementsByName : trouve tous les éléments ayant l'attribut name donné

Retourne une liste d'éléments

```
var textFieldList = document.getElementsByName("textfield");
```

DOM

Méthodes du document

getElementsByTagName : trouve les éléments selon leur nom de balise

Retourne une liste d'éléments

```
var tdList = document.getElementsByTagName("td");
```

DOM

Propriété d'élément

innerHTML : permet de consulter ou de modifier le contenu d'un élément HTML

```
document.getElementById("result").innerHTML = "Insignifiant";
```

DOM

Méthodes d'élément

blur() : enlève le focus de l'élément
focus() : donne le focus à l'élément

Utile pour les champs de formulaire

```
document.getElementById("prenom").focus();
```

DOM

Méthodes d'élément

getElementsByTagName : même chose que pour le document mais dans le sous-arbre de l'élément courant

DOM

Méthodes d'élément

getAttribute
hasChildNodes
appendChild
removeAttribute
removeChild
setAttribute

Et plusieurs autres...

CSS

Toutes les propriétés CSS sont accessibles à partir de Javascript sur un élément HTML

```
tdElement.style.backgroundColor = "#FF0000";
document.getElementById("extra").style.visibility = "hidden";
```

Événements

Les éléments HTML supportent certains événements d'interface utilisateur

Ces événements sont représentés sous forme d'attributs dans l'élément en question

La valeur de l'attribut est un script Javascript

Événements

En règle générale, on place dans la valeur de l'attribut d'événement un appel de fonction Javascript

```
<td onmouseover="changeBGColor('red');"
    onmouseout="resetBGColor();">
   Rouge
</td>
```

Événements

onblur : Appelé lorsque l'élément perd le focus

onfocus : Appelé lorsque l'élément obtient le focus

Événements

onclick : Appelé lorsqu'on clique dans l'élément

ondblclick : Appelé lorsqu'on double-clique dans l'élément

Événements

onmousedown : Appelé lorsqu'un bouton de souris est enfoncé sur l'élément

onmouseup : Appelé lorsqu'un bouton de souris est relâché sur l'élément

onmousemove : Appelé lorsque la souris bouge sur l'élément

Événements

onmouseover : Appelé lorsque la souris est mise au-dessus de l'élément

onmouseout : Appelé lorsque la souris quitte l'élément

Événements

onkeydown : Appelé lorsqu'une touche du clavier est enfoncée sur l'élément

onkeyup : Appelé lorsqu'une touche du clavier est relâchée sur l'élément

onkeypress : Appelé lorsqu'une touche du clavier est enfoncée et relâchée sur l'élément

Événements

onchange : Appelé lorsqu'on change la valeur de l'élément (input)

onselect : Appelé lorsque l'élément est sélectionné

Exceptions

Vous pouvez lancer des exceptions avec throw et les attraper avec try/catch

```
function activateControl() {
    …
    if (value == null) {
        throw "E03";
    }
    …
}
```

Exceptions

```
try {
    activateControl();
} catch (errorcode) {
    if (errorcode == "E03") {
        …
    }
    …
}
```

Fonction anonyme

Une fonction sans nom qu'on code directement au niveau de l'appelant

```
xmlHttp.onreadystatechange = function() {
    var container = document.getElementById("page");
    container.innerHTML = xmlHttp.responseText;
}
```

Programmation fonctionnelle

Javascript permet plusieurs aspects de la programmation fonctionnelle

Les fonctions sont des données

On peut les affecter à des variables et les passer en paramètre à des fonctions

Programmation fonctionnelle

Lorsqu'on donne une fonction en paramètre à une autre fonction dans l'attente que la fonction appelée fasse un appel à la fonction en paramètre, il s'agit d'un callback

Ce concept est largement utilisé en Javascript

HTTP

Le concept de fichier n'existe pas dans le fureteur, si on veut utiliser un fichier qui est placé sur le serveur, on doit faire une requête HTTP

XMLHttpRequest

On utilise l'objet XMLHttpRequest lorsqu'on veut coder une requête HTTP

On utilise souvent l'acronyme xhr pour désigner l'objet XMLHttpRequest

XMLHttpRequest

Les requêtes peuvent être synchrones ou asynchrones

Synchrone : On attend le résultat de la requête avant de continuer l'exécution

Asynchrone : On continue l'exécution du script et l'objet appelle un callback lorsque la réponse est arrivée (paradigme Ajax)

XMLHttpRequest

Un appel synchrone

```
var xhr = new XMLHttpRequest();
xhr.open("GET", "data.json", false);
xhr.send();

Le résultat est accessible via xhr.responseText
```

XMLHttpRequest

Si le résultat de la requête HTTP est un document XML, on utilise responseXML au lieu de responseText

La variable responseXML contient un DOM XML prêt à être utilisé

JSON

Pour transformer un document JSON en objet Javascript :

JSON.parse(...)

Pour transformer un objet Javascript en document JSON :

JSON.stringify(...)

Liens

ECMA-262

http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf

Documentation ECMAScript

http://www.ecmascript.org/docs.php

Liens

DOM

http://www.w3.org/DOM/

Référence DOM pour Javascript

http://www.w3schools.com/jsref/default.asp