INF2005 – Programmation web

Javascript – partie 1

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 licence

Javascript

Originalement conçu par Netscape

Standardisé par ECMA International

Maintenant supporté partout

Standards

Le standard original : ECMA-262
Le nom réel : ECMAScript

Version courante : ECMAScript 2018

Utilité

Outil de programmation pour les développeurs web

Permet de modifier une page dynamiquement sans passer par le serveur

HTML

Tout comme CSS, le Javascript peut être inclut directement dans un document HTML ou il peut être dans un fichier séparé

HTML

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

```
<script>
  /* Code JS */
</script>
```

HTML

Fichier Javascript externe
Extension habituelle : .js

```
<script src="fichier.js"></script>
```

HTML

La position de la balise script va avoir un impact sur l'exécution du script

Note : la balise script peut être mise n'importe où dans le document HTML

HTML

Si la balise est dans le body :
Le script est exécuté au chargement de la page

Si la balise est dans le head :
Le script ne sera exécuté que lorsque le code Javascript est invoqué

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 différents mot-clés : var, let, const

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

Variables

Lorsque déclarée avec var, la variable aura une portée de fonction

```
var index = 0;
```

Variables

Lorsque déclarée avec let, la variable aura une portée de bloc d‘exécution

```
let index = 0;
```

Variables

Lorsque déclarée avec const, la variable aura une portée de bloc d‘exécution et sera une constante

```
const index = 0;
```

Variables

Le mot-clé est optionnelle

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

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

Variables

C'est une bonne pratique de toujours déclarer ses variables

C'est également une bonne pratique de limiter autant que possible la portée des variables

Variables

Javascript possède 3 portées de variable :

- Globale : disponible dans toute la page
- Fonction : disponible dans toute la fonction
- Bloc : disponible dans un bloc d‘exécution

Variables

Typage faible

Typage dynamique

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

For

Itérations avec compteur

```
for (var i = 0; i < tableLength; i++) {
  …
}
```

Boucles

While

Itérations tant qu'une condition est vraie

```
while (state) {
  …
  state = false;
  …
}
```

Boucles

do...while

Comme le while mais avec une première itération assurée

```
do {
  …
} while (nombre < 9);
```

Boucles

Même comportement qu'en Java :

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 peuvent être représentés sous forme d'attributs dans l'élément en question

La valeur de l'attribut est un script Javascript

Événements

Avant, on plaçait 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

Événements

Au lieu d‘utiliser l‘attribut HTML, on peut utiliser Javascript pour définir un événement sur un élément

```
document.getElementById("#bouton").onclick = function(e) {
  // code de l‘événement
}
```

Événements

Par contre, on ne peut placer ainsi qu‘une seule fonction par événement sur un élément HTML

Événements

Pour ajouter plusieurs fonctions, on peut utiliser addEventListener

```
document.getElementById("#bouton").addEventListener("click",
fonction);
```

Fenêtres

Pour ouvrir une nouvelle fenêtre, on utilise l'objet window

window n'est pas un objet standard mais il est largement supporté

Fenêtres

Pour ouvrir la fenêtre : window.open

Prend en paramètre l'URL du document à ouvrir dans la nouvelle fenêtre

```
window.open("pour-ouvrir.html");
```

Fenêtres

La méthode open permet quelques paramètres optionnels

```
window.open("pour-ouvrir.html", "blackwindow",
            "width=450,height=250");
```

Fenêtres

Pour fermer la fenêtre : méthode close

```
self.close();
```

Fenêtres

La méthode open retourne une référence qu'on peut conserver pour manipuler la fenêtre ensuite

```
var blackWindow = window.open("pour-ouvrir.html");
blackWindow.close();
```

Boîtes de dialogue

Message d'avertissement : alert

Bouton : OK

```
alert("Ce site contient des images choquantes.");
```

Boîtes de dialogue

Confirmation : confirm

Boutons : OK, Annuler

Retourne true si l'utilisateur appuie sur OK

```
var result = confirm("Ce site contient du contenu sensible,
voulez-vous continuer?");
if (result == true) {
  …
}
```

Boîtes de dialogue

Demander une réponse à l'utilisateur : prompt

Boutons : OK, Annuler

Retourne le contenu du textbox si l'utilisateur appuie sur OK, sinon retourne null

Boîtes de dialogue

```
var result = prompt("Entrez votre âge", "18");
if (result != null && result != "" && result >= 18) {
  …
}
```

Plus loin...

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

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

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

Javascript: The Definitive Guide, 6th edition
David Flanagan
O'Reilly Media

Javascript: The Good Parts
Douglas Crockford
O'Reilly Media