Le Langage JavaScript

C. BENSARI

Plan

- Introduction
- JavaScript: un langage client
- Que fait-on avec le langage JavaScript ?
- Découverte du langage JavaScript

Introduction

- JavaScript est un langage de scripts, simple et rapide dans son apprentissage
- Permet aux développeurs débutants de réaliser leurs premières pages attrayantes et fonctionnelles
- JavaScript est un langage non typé (comme PHP):

```
var k;  // déclaration explicite
var i = 1; // déclaration explicite avec initialisation
valeur = i; // déclaration implicite
```

- JavaScript est un langage orienté Objet
- JavaScript n'a rien à voir avec Java

JavaScript: un langage du coté client

- JavaScript est un langage interprété par le navigateur
- Il est exécuté chez l'utilisateur lorsque la page web est affichée
- Le plus grand **objet** de l'hiérarchie de JavaScript est window qui contient entre autre l'objet document qui contient tous les éléments de la page web (titres, paragraphes, formulaires, …)
- Le JavaScript est parfois difficilement compatible entre les différents navigateurs
- Le JavaScript n'est pas sécurisé (ne jamais faire confiance à une donnée provenant du client)

Coder en JavaScript

- Un code JavaScript peut être inséré n'importe où dans une page HTML en utilisant la balise <script></script>

```
//CODE html/PHP
<script>
        // Code Javascript
</script>
//CODE html/PHP
```

- Il peut être aussi externe au fichier HTML, dans ce cas il faut préciser le chemin du fichier après la balise:

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

- JavaScript peut être invoqué de deux manières dans une page HTML:
  - Synchrone: exécution ligne par ligne. La ligne 2 n'est pas exécuté tant que la première n'a pas terminée
  - Asynchrone: exécution sans blocage. Utilisation de technologies asynchrones (Ajax)

Opérations et expressions

- Les variables

```
var a, b, c;
var i = 0;
```

- Les commentaires

```
// Ceci est un commentaire d une ligne
/* Ceci est un commentaire avec
plusieurs
Lignes */
```

- Conditions
  - **if** (a == b){ ….} **else** { … }
  - a != b **?** …. **:** …. ;
- Opérations pour les conditions: ==, ===, !=, =<, =>, &&, | |, &, |, !==

Opérations et expressions

- Les boucles

```
for (i=0; i<5; i++) { ..... }

while (condition) { ..... }

do { ..... } while (condition)
```

- Les tableaux :

```
var tableau = [];
tableau[3] = 25;
```

- Définition des fonctions:

```
function maFonction(){
        .....
        [return ..]
}
```

Les versions de Javascript

- La version utilisée jusqu'à présent dans ce cours est la version ECMAScript 5 ou ES5 (2009-2015)
- ECMAScript 6 ou ES6 (2015)
- ECMAScript 7 ou ES7 (2016)
- ECMAScript 8 ou ES8 (2017)
- ECMAScript 9 ou ES9 (2018)
- ECMAScript 10 ou ES10(2019)
- ESNext en cours de dévelopemment

Exercices :

Exercice 1 :

Ecrire un programme JavaScript permettant d'afficher la valeur absolue d'un nombre

Exercice 2 :

Ecrire un programme JS permettant de comparer deux nombres entiers

Exercice 3 :

Ecrire un programme JS permettant de savoir si un nombre est paire ou impaire. Utilisez l'opérateur % pour calculer le reste de la division

Exercice 4 :

Ecrire un programme JS qui permet de calculer le factoriel d'un nombre

Exercice 5 :

Ecrire un programme JS (organigramme) permettant d'afficher toutes les tables de multiplication de 1 à 10

Exercice 6 :

Ecrire un programme JS qui permet de calculer le factoriel d'un nombre

Remarques :

- Utilisez console.log() pour l'affichage de sortie dans la console web
- En JS, la concaténation se fait avec l'opérateur « + »

Objets internes: window

- JavaScript possède plusieurs objets qui peuvent être utilisés. Ces objets possèdent des méthodes et des attributs accessibles en plaçant un point (au lieu du -> de php) entre le nom de l'objet et la propriété
- L'objet window est créé lors de l'ouverture du navigateur
- Il contient les propriétés et méthodes de gestion de la fenêtre
- Il n'est pas nécessaire de préciser explicitement l'objet window pour accéder à ces propriétés:

```
window.alert(`HELLO`);      ⇔      alert(`HELLO`);
window.document             ⇔      document
```

Objets internes: document

- Exemple d'utilisation de l'objet « *document* » et sa méthode «*getElementById*» qui permet de récupérer un élément HTML depuis la page selon l'id :

```
var monObjet = document.getElementById(`elementId`) ;
// monObjet représente un élément Html qui a un id=”elementId”
 monObjet.style.display = `none`;

// est équivalente à
document.getElementById(` elementId`).style.display = `none`;
```

Objets internes: document

- L'objet document regroupe toutes les méthodes de gestion de la page (sélection d'éléments d'une page par id, par classe, par nom, .. et bien d'autres choses)
- Ces éléments possèdent eux même des propriétés et des méthodes pour modifier l'élément lui-même :

```
<h1 id=`titre1`> Titre principal </h1>
<script>
//monElement contient toutes les propriétés de « titre1 »
var monElement = document.getElementById(`titre1`);
//pour modifier la taille de la police de l'élément « titre1 »
monElement.style.fontSize = "12px";
//pour modifier la police
monElement.style.fontFamily = "Arial";
/*pour modifier le contenu du paragraphe (balises div, span, p et body
uniquement).*/
monElement.innerHTML = "Salut tout le monde !!";
</script>
```

L'attribut innerHTML

- **innerHTML** est une instruction qui permet de modifier le contenu d'une balise ou d'insérer un objet dans la page

```
//Ciblage du paragraphe
var monParagraphe = document.getElementById("idPg");
//Modification de son contenu
monParagraphe.innerHTML = "<img src=”imageInseree.gif” /> Mon nouveau texte";
```

Représentation de la page

- DOM: Document Object Model, est un modèle standardisé par le W3C. Le modèle propose le document sous la forme d'un arbre. Toutes les balises HTML sont donc des nœuds de l'arbre et les textes sont des feuilles
- JavaScript propose des fonctions permettant de cibler des éléments Html. Ces fonctions présentes pour chaque élément Html. Parmi ces fonctions :

Propriétés	Commentaires
childNodes	nœuds enfants
firstChild	premier nœud enfant
lastChild	dernier nœud enfant
nextSibling	prochain nœud d'un type (nœud de même niveau)
parentNode	nœud parent
previousSibling	nœud précédent d'un type (nœud de même niveau)
nodeName	nom du nœud
nodeValue	valeur / contenu du nœud
nodeType	type du nœud

Exemple d'arbres DOM

```
<!DOCTYPE HTML >
<html>
   <head>
      <title>DOM</title>
   </head>

   <body>
      <div id="idP1">
            <h4>DIV 1</h4>
            texte aléatoire 1
            <span> span</span>
            texte aléatoire 2
            <a href="#">ici</a>
      </div>

      <div id="idP2">
            <h4>DIV 2</h4>
            texte aléatoire 3
      </div>
   </body>
</html>
```

	BODY	DIV (idP1)	DIV (idP2)
childNodes	DIV ; #text ; DIV ;	H4 ; #text ; SPAN ; #text ; A ;	H4; #text (5)
firstChild	DIV	H4	H4
lastChild	DIV	A	#text
nextSibling	inexistant	#text	inexistant
parentNode	HTML	BODY	BODY
previousSibling	#text	inexistant	#text

Création d'éléments

- La fonction « createElement » de l'objet document permet de créer des balises html (titre, lien, input, formulaire, ..) mais ne l'insert pas dans la page
- La fonction prends en argument le nom de l'élément html à créer :

```
document.createElement("div");
```

- Exemples

```
//Création d'un input text
var elt = document.createElement("input");
elt.type = "text";

// Création d'un élément text
document.createTextNode("Texte du nœud");
```

Utilisation des nœuds

- Insertion d'une image dans un autre élément d'une page:

Objectif: Dans un paragraphe existant, insérer une image suivi d'un texte.

```
//Ciblage du paragraphe
var e = document.getElementById("idPg");
//Création de l'image
var i = document.createElement("img");
//Source de l'image
i.src = "imageInseree.gif";
//Modifiaction du texte (noeud #text)
e.firstChild.nodeValue ="mon nouveau texte";
//Ajout de l'image avant le texte déjà ajouté
e.insertBefore(i, e.firstChild);
```

Création d'un élément select et l'ajouter à un formulaire

```
//création de l'élément select
var elSelect = document.createElement("select");
//nombre d'éléments séléctionnables
elSelect.size = "1" ;
//Tableau contenant les options de la liste
var elOptions = new Array(
        new Option("Votre nationnalité", "", true, false),
        new Option("France", "FR", false, false),
        new Option("Belgique", "BL", false, false)
                        );
submit_elt = document.getElementById(`id-btn-submit`);
document.getElementById(`id-form`).insertBefore(elSelect, submit_elt);
//Ajout dans le select des options
for (i=0;i<elOptions.length;i++) {
        elSelect.options.add(elOptions[i]);
}
```

Exercices

Exercice 1 :

Créer le DOM suivant en utilisant JavaScript uniquement :

```
<div id="divTP1">
    Le <strong>World Wide Web Consortium</strong>, abrégé par le sigle
    <strong>W3C</strong>, est un
    <a href="http://fr.wikipedia.org/wiki/Organisme_de_normalisation"
       title="Organisme de normalisation"> organisme de standardisation</a>
       à but non-lucratif chargé de promouvoir la compatibilité des technologies du
    <a href="http://fr.wikipedia.org/wiki/World_Wide_Web" title="World Wide Web">
        World Wide Web</a>.
</div>
```

Exercices :

Exercice 2 :

Créer le DOM suivant avec JavaScript :

```
<div id="divTP2">
    <form enctype="multipart/form-data" method="post" action="upload.php">
        <fieldset>
            <legend>Uploader une image</legend>

            <div style="text-align: center">
                <label for="inputUpload">Image à uploader :</label>
                <input type="file" name="inputUpload" id="inputUpload" />
                <br /><br />
                <input type="submit" value="Envoyer" />
            </div>
        </fieldset>
    </form>
</div>
```

Les événements

- Les événements commencent généralement par le mot 'on': onclick, onload, onmouseover, ..
- Ils peuvent être insérés comme des attributs dans les balises HTML :

```
<body ... onload="alert('page chargée')"...>…<body/>
```

code recommandé:

page.html

```
<body onload="afficherAlerte('page chargée')"...>…<body/>
```

script.js

```
function afficherAlerte(message){
        alert(message);
  }
```

Autres manières pour déclarer un événement

- Souvent, il est plus intéressant de déclarer un événement et faire passer à la méthode déclenchée un paramètre fournissant des informations sur l'élément cible (élément ayant subit l'événement)
- L'exemple suivant permet d'utiliser la variable "e" qui fait référence à l'objet événement (event).

```
var elements = document.getElementsByClassName('red-link')
for (var i = 0; i < elements.length; i++) {
  var element = elements[i]
  element.addEventListener('click', function (e) {
    e.preventDefault()
    alert("Un lien qui ne mène nul part !")
  })
  element.style.color = "#FF0000"
}
```

- Grâce à l'attribut **target** de "e", il est possible de récupérer l'élément ayant subi le clic

Autres sélecteur Javascript

- Dans les récentes versions des navigateurs, il est possible d'utiliser le sélecteur suivant (compatible IE 10 et plus):

```
document.querySelector('#menu').classList.add('active')
```

- Pour réussir à faire cette opération dans les versions 9 et moins de IE, il faut utiliser le code suivant :

```
var cls = 'active'
var elt = document.getElementById('#menu')
if (elt.className.length == 0){
  elt.className = cls;
} else if (elt.className.indexOf(cls) < 0) {
  elt.className += ' ' + cls;
}
```

Exercices

Exercice 1 :

Créez 4 éléments input de type checkbox et deux boutons, le premier permettant de cocher toutes les cases et le deuxième pour décocher toutes les cases

Exercice 2 :

Créer une page avec différents éléments de type inline (span, a, strong, img, ..) séparés par des
. Ces éléments doivent se trouver dans un div. Utiliser un bouton qui permettra lors du clic de supprimer tous les éléments

Exercice 3 :

Ecrire un tableau Html avec des cellules (td) éditables (transformées en input) au clic. Une fois la saisie terminée l'input disparait et la valeur et maintenue.