JavaScript

1) Objectif

- Se familiariser avec le langage Javascript et acquérir une pratique minimale.
- Être capable d'intégrer des scripts clients dans un site ou une page en respectant les bonnes pratiques.
- Acquérir les capacités d'auto-formation nécessaires pour suivre les évolutions à venir.

2) Définition

- JavaScript est un **langage de programmation de scripts** principalement utilisé dans les pages web interactives côté client.
- C'est un **langage orienté objet** inspiré de nombreux langages dont Java mais il reste très différent de celui-ci.
- Le langage est maintenant une implémentation de la norme ECMA-262 (standard ECMAScript).

Le langage a été créé en 1995 par Brendan Eich, membre de la fondation *Mozilla*, pour le compte de *Netscape Communications Corporation*.

3) Gérer des scripts

Les scripts Javascript :

- sont de simples **fichiers "texte"** (extension conseillée **.js**) à créer avec un **éditeur de texte**.
- sont **intégrés au sein des pages web**.
- sont exécutés **côté client par le navigateur web**.

Remarque : les scripts en Javascript sont "débogables" avec l'extension *Firebug* pour *Mozilla Firefox* ou tout simplement avec la "Console d'erreurs" de ce navigateur.

4) Utilisation

Généralement, JavaScript sert :

- à contrôler les données saisies dans des formulaires HTML
- à interagir avec le document HTML via l'interface DOM (*Document Object Model*) fournie par le navigateur (on parle alors parfois de HTML dynamique ou DHTML)
- à modifier le contenu des pages web par programmation avec la méthode Ajax (*Asynchronous Javascript And XML*)

Remarque : javascript est aussi utilisé pour réaliser des services dynamiques, parfois futiles, strictement cosmétiques ou à des fins ergonomiques.

5) Exemple 1 : Insertion de code javascript dans une page WEB

Le code javascript est généralement inséré entre les balises <HEAD> et </HEAD>.

```
<HTML>
   <HEAD>
      <TITLE>Exemple 1</TITLE>
      <SCRIPT TYPE="text/javascript">
      // Du code Javascript
      alert("Hello world !"); // affiche une boîte de dialogue modale
      </SCRIPT>
   </HEAD>
   <BODY>
   </BODY>
</HTML>
```

6) Exemple2 : Ecrire dans une page WEB

L'élément NOSCRIPT permet de fournir un contenu de remplacement pour les navigateurs qui ne peuvent exécuter un script.

```
<HTML>
   <HEAD>
      <TITLE>Exemple 2</TITLE>
      <SCRIPT TYPE="text/javascript">
      <!--
      document.write("<P>Votre navigateur accepte le Javascript.</P>");
      //-->
      </SCRIPT>
      <NOSCRIPT><P>Votre navigateur n'accepte pas le Javascript.</P></NOSCRIPT>
   </HEAD>
   <BODY>
   </BODY>
</HTML>
```

Votre navigateur accepte le Javascript.

7) Exemple3 : Intégrer un script javascript dans une page.

Le programmeur préfèrera souvent rassembler son code javascript dans des fichiers de scripts qu'il intègrera alors dans les pages web de son site de la manière suivante :

```
<HTML>
   <HEAD>
      <TITLE>Exemple 3</TITLE>
      <SCRIPT TYPE="text/javascript" SRC="monscript.js"></SCRIPT>
   </HEAD>
   <BODY>
   </BODY>
</HTML>
```

8) Classe et Objet

Une **classe** est un type qui est la **description d'un ensemble** de :

- **propriétés** (les données) et de
- **méthodes** (les fonctions).

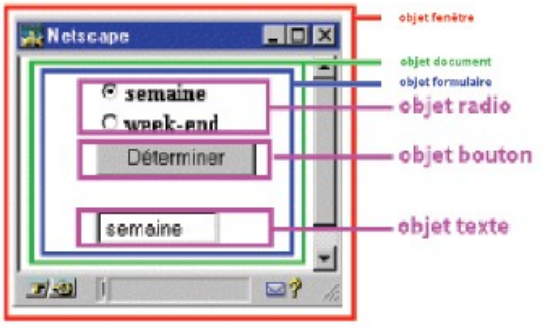
Un **objet** est une **instance de classe** (c'est à dire une variable de type classe). On écrira :

- `o.p` pour accéder à une propriété `p` d'un objet `o`
- `o.m()` pour appeler une méthode `m` d'un objet `o`

9) DOM(1)

Lorsqu'un document HTML est chargé dans un navigateur, celui-ci fournit une interface **DOM** (*Document Object Model*) pour accéder aux objets le composant :

- Ces objets sont classés de manière hiérarchique (notion d'arbre).
- L'objet le plus haut dans la hiérarchie est l'objet de la classe `window` (fenêtre).
- Dans cette fenêtre, il y a un document HTML : c'est l'objet `document`. Donc, L'objet `window` contient l'objet `document`.
- et ainsi de suite ...



10) DOM(2)

Dans ce document HTML, on pourrait donc accéder aux objets de la manière suivante :

```
// Accès à une propriété de l'objet button
window.document.form.button.value = "Déterminer";

// Accès à une méthode de l'objet button
window.document.form.button.focus();

// Accès à une méthode de l'objet window
window.alert("Hello world");
```

Remarque : l'objet window est souvent facultatif. On ne doit pas forcément préciser son nom pour utiliser ses méthodes ou ses propriétés.

11) DOM(3)

On recommande d'utiliser la méthode `getElementById()` pour accéder aux objets par leur identifiant (attribut `ID` de l'élément HTML) :

```
var bouton = document.getElementById('id_button');

bouton.click(); // pour simuler un clic de souris sur ce bouton
```

12) Programmation événementielle

Les IHM (Interface Homme-Machine) sont généralement basées sur la **programmation événementielle** qui permet la gestion d'événements.

- Un événement est généralement associé à une action de l'utilisateur : appui sur une touche, clic ou déplacement de la souris, ...
- En HTML, il y a très peu d'événements qui sont gérés par défaut : clic sur un lien ou sur un bouton de formulaire.

Le javascript va permettre de gérer et contrôler ces événements (`EVENT`) par des gestionnaires d'événements (`EVENT HANDLER`).

13) Les événements

Quelques événements classiques gérés en Javascript :

- L'utilisateur clique sur un bouton, un lien ou tout autre élément : `Click`
- La page est chargée par le navigateur : `Load`
- L'utilisateur quitte la page : `Unload`
- L'utilisateur place le pointeur de la souris sur un élément : `MouseOver`
- Le pointeur de la souris quitte un lien ou tout autre élément : `MouseOut`
- Un élément de formulaire a le focus (devient la zone d'entrée active) : `Focus`
- Un élément de formulaire perd le focus : `Blur`
- La valeur d'un champ de formulaire est modifiée : `Change`
- L'utilisateur sélectionne un champ dans un élément de formulaire : `Select`
- L'utilisateur clique sur le bouton submit pour envoyer un formulaire : `Submit`
- L'utilisateur appuie sur une touche : `Keydown`

Autres événements : `Abort`, `Error`, `Move`, `Resize`, `KeyPress`, `KeyUp`, `DblClick`, `MouseDown`, `MouseUp`, `MouseMove`, `Reset`,

14) Gestionnaire d'événements(1)

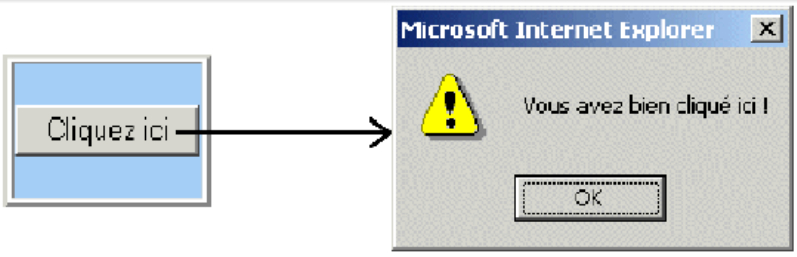
Pour gérer un évènement en JavaScript, il faut installer un **gestionnaire d'événement** :

- Un gestionnaire d'événement sera l'action déclenchée automatiquement lorsque l'évènement associé se produit.
- La syntaxe courante est la suivante : `onEvenement=fonction()` où Evenement est le nom de l'évènement géré.

15) Gestionnaire d'événements(2)

Exemple : gestion de l'évènement *click* d'un bouton

```
<FORM>
   <INPUT TYPE="button" VALUE="Cliquez ici" onClick="alert('Vous avez bien
        cliqué ici !')">
</FORM>
```



16) Les variables

Pas de déclaration préalable des types des variables :

```
// déclaration donc Local (uniquement dans le script ou la fonction) :
var vloc = 0;
// pas déclarée donc Global (en tout point du document) :
vglob = 0;

nbr1 = 10; // un nombre entier
var nbr2 = 3.141; // un nombre réel
str1 = "L'étoile"; // une chaîne de caractère
var str2 = 'brille'; // une chaîne de caractère
var str3 = str1 + " " + str2 ; // une chaîne de caractère concaténée avec +

// Remarque : Il existe aussi le type booléen (true ou false)
```

Remarque : lorsque l'on fait référence à une variable, celle-ci est d'abord cherchée dans la fonction courante (portée locale). Si elle n'y est pas, elle est cherchée dans le script (portée globale).

17) Les fonctions

Le programmeur écrira souvent des fonctions en JavaScript qu'il rassemblera le plus souvent dans des scripts d'extension `.js`.

Exemple : définition et appel de fonction

```
// Définition :
function mafonction(param1, ..., paramN)
{
   // code JavaScript
   // ...
   return variable_ou_valeur ;
}

// Appel :
var res = mafonction(var1, val2, varN);

// Remarque : la passage des paramètres est réalisé par valeur
```

18) Variables et fonctions(1)

```
var entier = 10;
var reel = 3.14;
chaine = "salut";

function test()
{
   entier = 5;
   var reel = 1.789;
   document.write("Dans la fonction test() : <BR>");
   document.write(entier + " is a " + typeof(entier) + "<BR>");
   document.write(reel + " is a " + typeof(reel) + "<BR>");
   document.write(chaine + " is a " + typeof(chaine) + "<BR>");
   document.write("<BR>");
}

test();

document.write("Dans le script : <BR>");
document.write(entier + " is a " + typeof(entier) + "<BR>");
document.write(reel + " is a " + typeof(reel) + "<BR>");
document.write(chaine + " is a " + typeof(chaine) + "<BR>");
```

19) Variables et fonctions(2)

Dans la fonction test() :
5 is a number
1.789 is a number
salut is a string

Dans le script :
5 is a number
3.14 is a number
salut is a string