Chapitre 5

JavaScript

Introduction

Javascript – c'est quoi ?

- ✓ Javascript est un langage de programmation ressemble à C, C++ et Java.
- ✓ un langage pour augmenter l'interaction dans les pages HTML
- ✓ Javascript est un langage de script, dépendant de HTML
- ✓ Le code Javascript est **intégré** complètement dans le **code HTML**, et interprété par le navigateur côté client (client-side).
- ✓ Javascript permet d'interagir avec l'usager de la page Web grâce à des **gestionnaires d'événement** (clic de la souris, validation de formulaire.)
- ✓ Javascript est un langage interprété: l'ordinateur convertit à chaque exécution le code source en langage exécutable.
- ✓ Dans un langage de programmation compilé, le code source est compilé une seule fois afin de créer un fichier exécutable.
- ✓ Dans un langage interprété, le code source est toujours lisible (on peut vérifier ce qu'est le contenu du programme).

Qu'est que Javascript peut faire ?

- ✓ Javascript permet d'ajouter du code dynamique dans les pages HTML, par exemple la date du jour
- ✓ Javascript réagi aux événements, par exemple lorsqu'un usager clique sur un bouton d'un formulaire, les données peuvent être validées avant l'envoi au serveur web;
- ✓ Javascript peut détecter le navigateur de l'usager est ainsi adapter la page web selon les particularités de chaque navigateur

1. JavaScript

1.1 Historique

- ✓ Le nom JavaScript est une propriété de Netscape.
- ✓ L'implantation de Microsoft porte le nom JScript.
- ✓ Javascript a été standardisé par l'ECMA (European Computer Manufacturers Association). La version standardisée s'appelle ECMAScript.

1.2. Ou placer du code Javascript
On peut placer du code JavaScript dans une page HTML à des endroits différents.

a. Entre les balises <SCRIPT>....</SCRIPT>

Dans la section d'en-tête, dans le corps de la page ou dans les deux.

Scripts dans la section head et body

```
<head>
<script type="text/javascript">
....
</script>
</head>
<body>
...
<script type="text/javascript">
....
</script>
...
</body>
```

Dans ce cas :

- ✓ Dans le <HEAD> Le code est évalué au début du chargement de la page il n'est pas exécuté tout de suite.
- ✓ Dans le <body> il est immédiatement exécuté en même temps que le code HTML est interprété.
- ✓ Il est nécessaire d'inclure les déclarations de fonctions dans la section <HEAD>..</HEAD>.

b. Associé à une balise HTML qui gère un événement.

- ✓ Le code Javascript inséré sous forme d'un appel de fonction, affectée à un gestionnaire d'événement
- ✓ L'exécution du code est alors provoquée par appel d'une fonction Javascript
- ✓ exécution constitue une réponse à l'événement.

c. Dans un fichier externe

Le script dans un fichier externe doit avoir l'extension js et il peut être utilisé dans plusieurs pages HTML. Pour l'utiliser dans une page HTML, il faut mettre sont URL dans l'attribut src de l'élément script: <script src="xxx.js" />

Il faut mettre l'élément script où le code Javascript doit être exécuté dans la page HTML.

Le Javascript codes dans un fichier à insérer est le suivant:

<SCRIPT LANGUAGE=Javascript SRC="url/fichier.js"> </SCRIPT>

2. Structure d'un programme

- Javascript est sensible à la casse
- Les commentaires peuvent être spécifiés par // et par /* ... */

```
// Voici un commentaire d'une ligne

/*  et voici un commentaire

      Sur  plusieurs ligne */
```

- Les point-virgules ne sont pas obligatoires après une expression si elle est suivie d'un retour à la ligne (mais souvent on place des point-virgules après chaque expression).
- Les accolades ({ et }) permettent de former des blocs d'instruction.
- Chaque variable utilisée doit être déclarée
- un nom de variable doit commencer par une lettre (majuscule ou minuscule) ou un "_" un nom de variables peut comporter des lettres, des chiffres et les caractères _ et &, les espaces ne sont pas autorisés!

2.1 La déclaration de variables

La déclaration des variables peut se faire de deux façons:

- explicite, utilisez le du mot clé var

  - var chaine= " bonjour tout le monde " ;
  - var b=450 ;

- Implicite, utilisez =

```
chaine= "bonjour tout le monde " ;
```

2.2. Portée (visibilité) des variables

Une variable est déclarée **sans le mot clé var (varx)**, elle est accessible de partout dans le script donc on parle alors de **variable globale**

Une variable est déclare **de façon explicite** en javascript (var varx), sa portée dépend de l'endroit où elle est déclarée.

- au début du script, c'est-à-dire avant toute fonction sera globale
- dans une fonction aura une portée limitée à cette seule fonction.

3. Types de données

- nombres : entiers ou à virgules
- valeurs booléennes : true ou false
- chaînes de caractères (string): une suite de caractères
- objets
- tableaux

Caractères spéciaux

- \b : touche de suppression
- \f : formulaire plein
- \n : retour à la ligne
- \r : appui sur la touche ENTREE
- \t : tabulation
- \" : guillemets doubles
- \' : guillemets simples
- \\ : caractère antislash

3.1. Les chaînes de caractères : l'objet String

L'objet String est un objet qui contient un certain nombre de propriétés et de méthodes permettant la manipulation de chaînes de caractères.

Les propriétés de l'objet String

La propriété length : retourner la longueur d'une chaîne de caractères.

Les méthodes de l'objet String

Méthodes et leurs descriptions

charAt(chaîne, position**)** ou **chaîne.charAt(**position**)**

- Retourne le caractère situé à la position donnée en paramètre
  indexOf(sous-chaîne, position**)**
- Retourne la position d'une sous-chaîne dans une chaîne de caractère, en effectuant la recherche de gauche à droite, à partir de la position spécifiée en paramètre.

lastIndexOf(sous-chaîne, position**)**

- La méthode est similaire à indexOf(), à la différence que la recherche se fait de droite à gauche: Retourne la position d'une sous-chaîne dans une chaîne de caractère, en effectuant la recherche de droite à gauche, à partir de la position spécifiée en paramètre.

substring(position1, position2**)**

- La méthode retourne la sous-chaîne (lettre ou groupe de lettres) comprise entre les positions 1 et 2 données en paramètre.

toLowerCase() Convertit tous les caractères d'une chaîne en minuscule

toUpperCase() Convertit tous les caractères d'une chaîne en majuscule

toUpperCase() Convertit tous les caractères d'une chaîne en majuscule

4. Les Opérateurs

4.1. Opérateurs mathématiques

- x++ équivalent à x=x+1
- x-- équivalent à x=x-1
- x+=y équivalent a x=x+y
- x-=y équivalent à x=x-y
- x*=y équivalent a x=x*y
- x/=y équivalent à x=x/y
- x%y reste de la division entière de x par y

4.2. Opérateurs de comparaison

- x==y est VRAI si x est égal à y
- x!=y est VRAI si x est différent de y
- x<y est VRAI si x est plus petit que y
- x>y est VRAI si x est plus grand que y
- x<=y est VRAI si x est plus petit que y+1
- x>=y est VRAI si x est plus grand que y-1

4.3. Opérateurs logiques

- Une valeur logique (booléenne) est toujours soit VRAI soit FAUX
- X && y est VRAI si x et y sont VRAI. Si x est FAUX, la valeur de y n'est pas évaluée.
- X | | y test VRAI au moins un de x et y est VRAI. Si x est VRAI, la valeur de y n'est pas évaluée.
- !x correspond au NON logique

5. Structures de contrôle

- On des fois besoin de répéter une certaine opération plusieurs fois ou de seulement l'exécuter si une certaine condition est remplie.
- structures de contrôle disponibles:
- if ... else
- for(... ; ... ; ...)
- while(...)
- do ... while(...)

5.1. if

```
if (condition) {
                ….. ;
                }
else            {
                ….. ;
                }
```

- Si la condition est vraie, alors code 1 est exécuté.
- Si la condition est fausse, alors code 2 est exécuté.
- Le bloc else n'est pas obligatoire.

(condition) ? instruction si vrai : instruction si faux

5.2. while

```
while (condition) {
……. ;
}
```

- Tant que la condition est vraie, code est exécuté.
- Attention de ne pas créer de boucles infinies.
- Permet de répéter une action tant qu'une condition est remplie.

```
chiffre = 0 ;
somme = 0 ;
while ( chiffre < 5 ) {
chiffre = chiffre + 1;
somme = somme + chiffre;
}
```

5.3. do while

```
do {
```

….. ;

}

while (condition);

- Tant que la condition est vraie, code est executé.
- Similaire à la boucle while, mais la condition est vérifiée à la fin de la boucle.
- Le code est toujours executé au moins une fois.
- Attention à ne pas créer de boucles infinies.

somme = 0 ;

do {

chiffre = chiffre + 1 ;

somme = somme + chiffre;

} while (chiffre< 5)

5.4. for

- for (initialisation; test; incrementation){ }

somme = 0 ;

for (chiffre = 1 ; chiffre <= 5 ; chiffre = chiffre + 1) {

somme = somme + chiffre ;

}

6. Fonctions

On appelle fonction un sous-programme qui permet d'effectuer un ensemble d'instruction par simple appel de la fonction dans le **corps** du programme principal.

6.1. La déclaration d'une fonction

Avant d'être utilisée, une fonction doit être définie car pour l'appeler dans le corps du programme il faut que le navigateur la connaisse, c'est-à-dire qu'il connaisse son nom, ses arguments et les instructions qu'elle contient. La déclaration d'une fonction se fait grâce au mot clé function selon la syntaxe suivante:

function Nom_De_La_Fonction(argument1, argument2, ...) {liste d'instructions }

6.2. Appel de fonction

Pour exécuter une fonction, il suffit de faire appel à elle en écrivant son nom suivie d'une parenthèse ouverte (éventuellement des arguments) puis d'une parenthèse fermée:

Nom_De_La_Fonction();

Déclarez généralement les fonctions dans des balises SCRIPT situées dans l'en-tête de la page, c'est-à-dire entre les balises <SCRIPT> et </SCRIPT>)

Grâce au gestionnaire d'évènement onLoad (à placer dans la balise BODY) il est possible d'exécuter une fonction au chargement de la page, comme par exemple l'initialisation des variables pour votre script, et/ou le test du navigateur pour savoir si celui-ci est apte à faire fonctionner le script.

function carre(nombre) {

resultat = nombre * nombre ;

return(resultat) }

Le mot-clé this

Lorsque vous faîtes appel à une fonction à partir d'un objet, par exemple un formulaire, le mot clé this fait référence à l'objet en cours.

6.3. Les paramètres d'une fonction

Tous les arguments de type nombre et booléen sont transmis par valeur, i.e. la fonction obtient la valeur des arguments, mais ne peut pas les modifier. Les arguments de type objet sont transmis par référence.

Les fonctions mathématiques

- Math.abs(x) Math.acos(x) Math.asin(x)
- Math.atan(x) Math.atan(y,x) Math.cos(x)
- Math.exp(x) Math.log(x) Math.tan(x)
- Math.max(x,y) Math.min(x,y) Math.sqrt(x)
- Math.pow(x,y) Math.random() Math.sin(x)

Constantes mathématiques

- Math.E
- Math.PI

7. Tableau

Les tableaux permettent de stocker facilement des données et de pouvoir y accéder par la suite. Une variable ne peut contenir qu'une seule et unique valeur. Un tableau peut en contenir infiniment. JavaScript étant orienté objet, les tableaux ne font pas exception : ce sont des objets avec leurs propriétés et leurs méthodes. Pour créer un objet, on utilise la syntaxe suivante :

var indentificateur = new constructeur(argument1, argument2, ...);

L'**identificateur** doit être un identificateur valide, qui doit suivre les même règles que pour les noms de variables.

Le **constructeur** doit être une fonction constructeur valide (nous étudierons plus en détails par la suite). Ici, le constructeur est **Array** (avec un A majuscule).

Viennent ensuite les **arguments**. Il peut y en avoir ou pas. Donc, pour créer un tableau nous écrirons :

Exemple :

```
var a = new Array();              // cree un tableau vide

var b = new Array(5);             // cree un tableau de taille 5

var cours = new Array("web", "info1", "info8");

var c = cours[0];                 // c = "web"
```

var livres = new Array();// ne pas oublier les parenthèses vides

En peut ensuite accéder aux éléments de ce tableau en spécifiant l'index de l'élement conserné entre crochets ([et]). Ansi :

```
var livres = new Array();
livres[0] = "Math";
livres[1] = "Info";
livres[2] = "Bio";
document.write("1<sup>er</sup> livre : " + livres[0] +"<br>") // Math
document.write("2<sup>eme</sup> livre : " + livres[1] "<br>");
            // Info                  // Info
document.write("3<sup>eme</sup> livre : " + livres[2] "<br>");
            // Bio
```

Une seconde possibilité de créer un tableau est d'utiliser la manière littérale. On passe en arguments, au moment de la création du tableau, les différents élements. L'exemple suivant est donc équivalent au précedent :

```
var livres = new Array("Math", "Info", "Bio");
```

Ou on peut aussi écrire sous une autre forme, sans utiliser Array, avec des crochets :

```
var livres = ["Math", "Info", "Bio"];
```

Une dernière possibilité de créer un tableau est de passer en argument la longueur de celui-ci.

Exemple :

```
var MonTableau = new Array(10);              // 10 élements

document.write(MonTableau.length);                    // Sortie -> 10
```

Propriétés

Comme tout objet, Array à également des propriétés. En fait, Array n'en a qu'une seule :

length

Renvoi le nombre d'élements contenus dans la table. Par exemple :

```
var livres = new Array("Math", "Info", "Bio");
document.write(livres.length);// Sortie -> 3
```

Cette propriété est utile pour parcourir un tableau à l'aide d'une boucle :

```
var livres = new Array("Math", "Info", "Bio");
for(var i=0; i < livres.length; i++)
document.write("Livre N°" + i + " : " + livres[i] + "<br>");
```

Méthodes

concat()

Permet de faire de deux tableaux un seul. La table à ajouter doit être passée en argument :

```
var MesLivres = new Array("Math", "Info", "Bio");
var TesLivres = new Array("Voyage ", "Le Maroc ");
MesLivres.concat(TesLivres);      // On ajoute le tableau TesLivres
```

join()

Créer une chaîne de caractères avec les éléments du tableau. Les éléments sont séparés avec la chaîne de caractères passée en argument qui reste optionnelle.

```
var livres = new Array("Math", "Info", "Bio");
var Texte = livres.join("<br>");
document.write(Texte);
```

reverse()

Inverse l'ordre des élements du tableau. Le 1er devient le dernier, le 2ème l'avant dernier, etc...

```
var livres = new Array("Math", "Info", "Bio");
livres.reverse();
for(var i=0; i < livres.length; i++)
document.write(livres[i] + "<br>");
```

slice()

Renvoi un tableau contenant une partie du tableau. Le 1er argument correspond à l'index de départ. Le 2ème argument est optionnel. Il correspond à l'index de fin. Si il est omis, tous les index jusqu'à la fin du tableau seront renvoyés.

```
var liste = new Array("un", "deux", "trois", "quatre", "cinq");
var maliste = liste.slice(1, 4);
for(var i=0; i < maliste.length; i++)
document.write(maliste[i] + "<br>");
```

sort()

Permet de trier un tableau selon les valeurs des éléments de celui ci. Si aucun argument n'est transmis, le tableau est trié par ordre alphabétique. L'argument possible est une fonction de tri. Cette fonction doit recevoir 2 arguments et livrer en retour une valeur négative si le 1er argument doit être placé avant le 2ème, ou positive pour l'inverse.

```
var liste = new Array("un", "deux", "trois", "quatre", "cinq");
liste.sort();
for(var i=0; i < liste.length; i++)
document.write(liste[i] + "<br>");
```

Tableaux associatifs

La différence entre un tableau associatif et un tableau classique, c'est qu'un tableau associatif n'utilise pas la méthode numérique pour indexer les différents éléments, mais des noms.

Dans un tableau classique, on accéde par exemple ainsi à un élement :

```
var MonTableau = new Array();
```

```
MonTableau[0] = "Gold";// On accéde à l'élement 0

document.write(MonTableau[0]);// Sortie -> Gold
```

Avec un tableau associatif, on y accédera ainsi :

```
var MonTableau = new Array();
MonTableau["nom"] = "Gold";// On accède à l'élément "nom"
document.write(MonTableau["nom"]);      // Sortie -> Gold
```

C'est tableau sont dans certains cas plus commodes a utiliser. Pour parcourir un tableau de ce type à l'aide d'une boucle, il faut utiliser la boucle for...in :

8. L'objet Dates

- L'objet Date permet d'utiliser des dates et heures dans des programmes Javascript.

```
var aujourdhui = new Date() //la date du jour

var date1 = new Date(2001,02,30,8,45,0) // 30 mars 2001
```

8.1 Création d'un objet Date

La syntaxe pour créer un objet-date peut être une des suivantes:

variable = **new Date**(liste de paramètres)

- Date() :
- var aujourdhui = new Date() //la date du jour
- Date("month dd, yyyy hour:min:sec") : "December 25, 1995 13:30:00"
- Date(yy,mm,dd,hh,mm,ss) : 95, 11, 25 , 13, 30 , 00
- Date(yy,mm,dd): 95, 11, 25
- Date(milliseconds) : 1587243

8.2. Les méthodes de l'objet Date

Les méthodes de l'objet Date fournissent un moyen simple d'accéder à un seul élément, ou bien de le modifier. Leur syntaxe est la suivante:

Objet_Date.Methode()

Extraire un élément de la date

- ✓ getDate() : Permet de récupérer la valeur du jour du mois(entre 1 et 31)
- ✓ getDay() : Permet de récupérer la valeur du jour de la semaine pour la date spécifiée 0: dimanche 1: lundi ...
- ✓ getHour() : L'objet retourné est un entier (entre 0 et 23) qui correspond à l'heure
- ✓ getMinutes() : retourné est un entier (entre 0 et 59) qui correspond aux min
- ✓ getMonth() : Permet de récupérer le numéro du mois (entre 0 et 11)

- ✓ getTime() : Permet de récupérer le nombre de secondes depuis le 1er janvier 1970 permettent de récupérer une valeur:

9. Les Objets du Navigateur

Lorsque vous ouvrez une page Web, le navigateur crée des objets prédéfinis correspondant à la page Web, à l'état du navigateur, et peuvent donner beaucoup d'informations qui vous seront utiles.

Les objets de base du navigateur sont les suivants:

- navigator: qui contient des informations sur le navigateur de celui qui visite la page
- window: c'est l'objet où s'affiche la page, il contient donc des propriétés concernant la fenêtre elle-même mais aussi tous les objets-enfants contenus dans celle-ci
- location: contient des informations relatives à l'adresse de la page à l'écran
- history: c'est l'historique, c'est-à-dire la liste de liens qui ont été visités précédemment
- document: il contient les propriétés sur le contenu du document (couleur d'arrière plan, titre, ...)

L'objet Window

L'objet window est l'objet par excellence dans Javascript, car il est le parent de chaque objet qui compose la page web, il contient donc:

- l'objet document: la page en elle-même
- l'objet location: le lieu de stockage de la page
- l'objet history: les pages visitées précédemment
- l'objet frames: les cadres (division de la fenêtre en sous-fenêtres)

9.1. Les méthodes de l'objet window

L'objet window possède des méthodes relatives à l'ouverture et à la fermeture des fenêtres.

9.2. Les méthodes alert(), confirm() et prompt()

Les méthodes alert(), confirm() et prompt() sont des méthodes qui font apparaître une boîte de dialogue, elles sont expliquées en détail dans le chapitre Boîte de dialogue.

10. Les événements

Les évènements sont des actions de l'utilisateur, qui vont pouvoir donner lieu à une interactivité exemples : le clic de souris.

Grâce au Javascript il est possible d'associer des fonctions, des méthodes à des événements tels que le passage de la souris au-dessus d'une zone …

Ce sont les gestionnaires d'événements qui permettent d'associer une action à un événement.

La syntaxe d'un gestionnaire d'événement est la suivante:

onEvenement="Action_Javascript_ou_Fonction();"

Liste des événements et leurs descriptions :

Click (onClick) :

- ✓ Se produit lorsque l'utilisateur clique sur l'élément associé à l'événement

Load (onLoad) :

- ✓ Se produit lorsque le navigateur de l'utilisateur charge la page en cours

Unload (onUnload)

- ✓ Se produit lorsque le navigateur de l'utilisateur quitte la page en cours

MouseOver (onMouseOver)

- ✓ Se produit lorsque l'utilisateur positionne le curseur de la souris au-dessus d'un élément

MouseOut (onMouseOut)

- ✓ Se produit lorsque le curseur de la souris quitte un élément

Focus (onFocus)

- ✓ Se produit lorsque l'utilisateur donne le focus à un élément, c'est-à-dire que cet élément est sélectionné comme étant l'élément actif

Blur (onBlur)

- ✓ Se produit lorsque l'élément perd le focus, c'est-à-dire que l'utilisateur clique hors de cet élément, celui-ci n'est alors plus sélectionné comme étant l'élément actif

Change (onChange)

- ✓ Se produit lorsque l'utilisateur modifie le contenu d'un champ de données

Select (onSelect)

- ✓ Se produit lorsque l'utilisateur sélectionne un texte (ou une partie d'un texte) dans un champ de type "text" ou "textarea"

Submit (onSubmit)

- ✓ Se produit lorsque l'utilisateur clique sur le bouton de soumission d'un formulaire (le bouton qui permet d'envoyer le formulaire)

10.1. Objets auxquels on peut associer des événements

Chaque événement ne peut pas être associé à n'importe quel objet.

Objet Evénements associables

Lien hypertexte :

- ✓ onClick, onMouseOver, onMouseOut

Page du navigateur :

- ✓ onLoad, onUnload

Bouton, Case à cocher, Bouton radio, Bouton Reset :

- ✓ onClick

Liste de sélection d'un formulaire :

- ✓ onBlur, onChange, onFocus

Bouton Submit :

- ✓ onSubmit

Champ de texte et zone de texte :

- ✓ **onBlur, onChange, onFocus, onSelect**