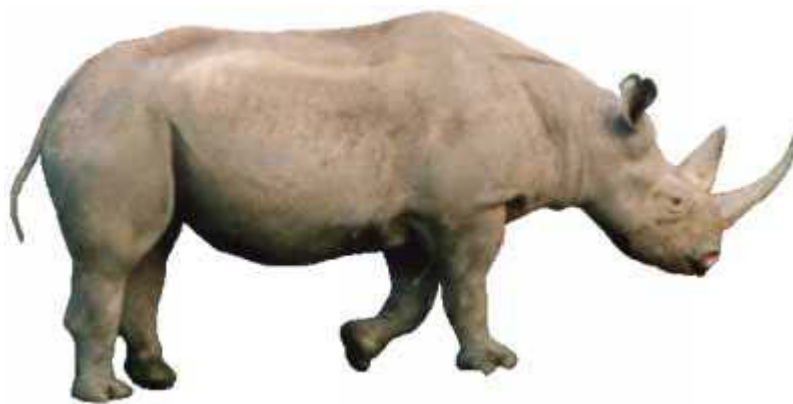


Charles COLSON

PROGRAMMATION JAVASCRIPT





PROGRAMMATION JAVASCRIPT

Charles COLSON

Sommaire	2
I. Introduction	3
II. Les bases	4
III. La notion d'objets	5
IV. Les variables	6
V. Les fonctions	8
VI. Les conditions	9
VII. Les boucles	11
VIII. Les tableaux	12
IX. Les événements	13
X. L'objet window	14
XI. L'objet date	17
XII. L'objet math	18



I. Introduction

Avant de commencer ce tutorial sur le Javascript, il faut bien maîtriser le HTML : il s'agit d'une extension qui est incluse dans le code. Grâce au Javascript vous allez pouvoir apporter des améliorations au langage HTML en permettant d'exécuter des commandes.

Pour faire des pages avec du Javascript vous devez avoir :

- Un éditeur de texte pour créer les pages (*bloc notes* par exemple), il faudra veiller à enregistrer les pages au format .htm et non .txt
- Un navigateur Internet pour afficher les pages créées (*Internet explorer* par exemple)

Afin de mieux comprendre chaque élément, il est recommandé de tester chaque exemple de ce cours.



II. Les bases

A. Composition

Le Javascript se place simplement dans le corps d'une page HTML :

```
<html>
<head>
<title></title>
</head>
<body>
<SCRIPT language="Javascript">
</SCRIPT>
</body>
</html>
```

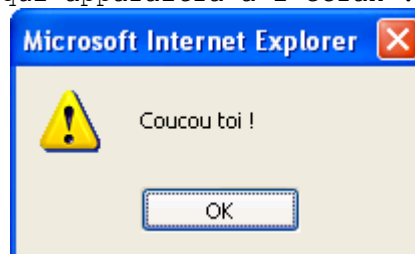
Ce qui apparaîtra à l'écran :

- `<script language="Javascript">` indique que l'on va travailler avec le langage Javascript.
- N'oublions pas de fermer l'inclusion de Javascript avec `</script>`

Complétons un peu le code :

```
<html>
<head>
<title></title>
</head>
<body>
<SCRIPT language="Javascript">
alert("Coucou toi !") ;
</SCRIPT>
</body>
</html>
```

Ce qui apparaîtra à l'écran :



- `alert` indique qu'il faut afficher une boîte de dialogue avec un message que l'on mettra entre parenthèses
- `"Coucou toi !"` on tape entre guillemets le texte que l'on souhaite voir apparaître
- On lui dit que c'est la fin de l'instruction grâce au `;` C'est facultatif mais beaucoup plus rigoureux.

Il faut savoir qu'il est aussi possible de mettre l'ensemble de son code Javascript dans un fichier qu'on enregistrera avec l'extension `.js` et auquel on fera appel à l'ouverture de la page HTML :

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

- On indique grâce à l'attribut `src` où se trouve le code Javascript que l'on souhaite insérer, ici il s'agit de `fichier.js`

B. Commentaires

Pour ajouter un commentaire (quelque chose qui ne sera pas vu par les visiteurs) on met ://

```
<SCRIPT language="Javascript">
// Voici une ligne de commentaire
</SCRIPT>
```

Ce qui apparaîtra à l'écran :

C. Anciens navigateurs

N'oublions pas les anciens navigateurs qui ne prennent pas en charge le Javascript :

```
<SCRIPT language="Javascript">
// code Javascript
</SCRIPT>
<NOSCRIPT>
Votre navigateur est trop vieux !
</NOSCRIPT>
```

Ce qui apparaîtra à l'écran :
(si le navigateur ne supporte pas Javascript) : Votre navigateur est trop vieux !

- On place entre les balises `noscript` le code HTML à afficher si jamais le navigateur ne prend pas en charge le Javascript



III. La notion d'objets

A. Se repérer

La programmation Javascript est orientée objet, c'est à dire qu'elle tourne autour d'une véritable arborescence (ou hiérarchie). Ainsi, pour appeler un élément on commencera généralement par l'objet le plus grand pour aller jusqu'à l'objet voulu. Pour que cela soit plus clair, voici un exemple :

1	2	3
<input type="text"/>	<input type="checkbox"/>	<input type="text" value="coucou"/>

Pour repérer le premier objet (un champ de formulaire dont le nom est « objet1 ») :

```
window.document.forms[0].objet1
```

- On commence par l'objet le plus grand : *window* (la fenêtre du navigateur)
- On poursuit avec l'objet juste un peu plus petit : *document* (la page html)
- On continue avec l'objet juste un peu plus petit : *forms[0]* (le formulaire)
- On termine en précisant le nom de l'objet : *objet1*

Sachez que pour une page avec des frames (donc englobant plusieurs pages) pour déterminer la fenêtre principale on indique *parent* et non *window*.

Pour repérer le deuxième objet (une case à cocher dont le nom est « objet2 ») :

```
window.document.forms[0].objet2
```

- Même principe que l'exemple ci dessus

Pour repérer le troisième objet (le texte d'un champ de formulaire dont le nom est « objet3 ») :

```
window.document.forms[0].objet3.value
```

- On commence par l'objet le plus grand et on va jusqu'au champ de formulaire appelé *objet3*
- On termine en précisant qu'on veut le texte entré en ajoutant *.value* (comme l'attribut en HTML)

Maintenant que nous savons comment fonctionne la notion d'objets, voici un exemple pour afficher du texte directement sur la page :

```
<SCRIPT language="Javascript">  
window.document.write("Coucou") ;  
</SCRIPT>
```

Ce qui apparaîtra à l'écran :
Coucou

- On commence par appeler l'élément avec lequel on veut travailler (il s'agit de la page) donc *window.document*
- On poursuit en indiquant que l'on souhaite écrire sur l'élément grâce à *write()*
- On inscrit dans les parenthèses ce que l'on souhaite voir s'afficher : *coucou*

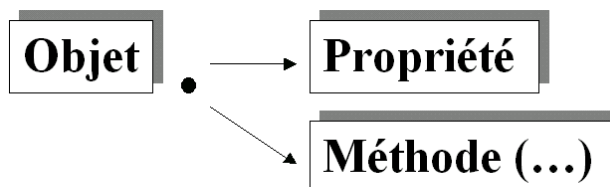
Sachez qu'il est aussi possible de mettre des balises HTML dans un *write()*.

B. Objets, propriétés, méthodes et classes

La notion d'objet fait appel à ces différents thèmes, il est impératif de bien connaître leur signification pour comprendre la suite de ce cours. En fait, il existe des **classes** qui représentent des familles d'**objets** ayant une structure et un comportement commun.

Pour un objet on peut appliquer soit une

propriété qui représente ses caractéristiques (pour reprendre un exemple précédent : *window.document.forms[0].objet3.value*) ou alors une **méthode** qui représente un outil s'appliquant aux objets d'une classe (pour reprendre un exemple précédent : *window.document.write("Coucou")*).





IV. Les variables

Une variable est une information qui est stockée temporairement. Il faut savoir qu'en Javascript une variable ne doit pas commencer par un nombre et qu'elle est sensible à la casse (majuscules et minuscules).

A. Les types

➤ Le texte :

```
<SCRIPT language="Javascript">  
var test = "Bonjour" ;  
</SCRIPT>
```

- *Var test* indique que l'on crée une variable qui s'appellera *test*. Le terme *var* est facultatif son utilisation est beaucoup plus propre.
- = signifie équivaut à
- "Bonjour" indique la valeur que va prendre la variable *test*, ici *Bonjour*. On place les textes entre guillemets
- On finit l'instruction par un ;

➤ Les nombres :

```
<SCRIPT language="Javascript">  
var nbtest = 12 ;  
</SCRIPT>
```

- 12 indique la valeur que va prendre la variable *nbtest*. On ne place pas les nombres entre guillemets

➤ Les booléens (sur le principe soit vrai soit faux) :

```
<SCRIPT language="Javascript">  
var gagne = true ;  
var perdu = false ;  
</SCRIPT>
```

- = *true* indique que la variable *gagne* est vrai, *true* ne se met pas entre guillemets
- = *false* indique que la variable *perdu* est faux, *false* ne se met pas entre guillemets

Pour afficher la valeur d'une variable il suffit d'ajouter la méthode *write* :

```
<SCRIPT language="Javascript">  
var test = "Bonjour" ;  
window.document.write(test) ;  
</SCRIPT>
```

Ce qui apparaîtra à l'écran :
Bonjour

B. Concaténation

La concaténation permet d'associer deux variables :

```
<SCRIPT language="Javascript">  
var prenom = "James" ;  
var nom = "Bond" ;  
var name = prenom + nom ;  
document.write ("Mon nom est " + name) ;  
</SCRIPT>
```

Ce qui apparaîtra à l'écran :
Mon nom est JamesBond

- *var name = prenom + nom* la variable *name* prendra la valeur de la variable *prenom* et (correspond au plus) la valeur de la variable *nom*. Soit pour valeur finale : *James Bond*
- *alert("Mon nom est "+ name)* pour passer d'une chaîne de caractères à une variable on utilise le +. On ne met pas de doubles quotes pour les variables (contrairement aux chaînes de caractères)

C. Quotes

```
<SCRIPT language="Javascript">
var prenom = 'James';
document.write ("prenom" );
</SCRIPT>
```

Ce qui apparaîtra à l'écran :
prenom

- La variable *prenom* entre doubles quotes ne permet pas d'afficher son contenu

En utilisant uniquement les simples quotes, il faudra mettre des antislash :

```
<SCRIPT language="Javascript">
document.write ('C\'est moi !') ;
</SCRIPT>
```

Ce qui apparaîtra à l'écran :
C'est moi !

En utilisant uniquement les doubles quotes, il faudra mettre des antislash :

```
<SCRIPT language="Javascript">
document.write("C'est du
\"Javascript\"") ;
</SCRIPT>
```

Ce qui apparaîtra à l'écran :
C'est du « Javascript »

D. Fonction types

```
<SCRIPT language="Javascript">
var coucou="test";
var type = typeof(coucou) ;
window.document.write(type);
</SCRIPT>
```

Ce qui apparaîtra à l'écran :
string

- La fonction *typeof()* permet de renvoyer le type de la variable entre parenthèses (ici *coucou*)

String = chaîne de caractères, *number* = nombre, *boolean* = booléen, *undefined* = autre.

E. Propriétés et méthodes

➤ Longueur & Nombre

```
<SCRIPT language="Javascript">
var coucou="test";
nb = coucou.length;
window.document.write(nb);
</SCRIPT>
```

Ce qui apparaîtra à l'écran :
4

- En ajoutant la propriété *.length* on peut connaître la longueur d'une variable. Ici *coucou* qui est égale à *test* contient 4 caractères

Il faut savoir que cette propriété peut s'appliquer sur de nombreux objets, par exemple dans un tableau elle renverra le nombre d'entrées.

➤ Sous chaîne

```
<SCRIPT language="Javascript">
var mot = "facile le js !";
var Resultat = mot.substring(1,5) ;
window.document.write(Resultat);
</SCRIPT>
```

Ce qui apparaîtra à l'écran :
acil

- La méthode *subtring()* permet d'extraire les caractères 1 à 5 de l'objet *mot*

➤ Minuscules & Majuscules

```
<SCRIPT language="Javascript">
var mot = "Facile LE Javascript";
var Resultat = mot.toLowerCase() ;
window.document.write(Resultat);
</SCRIPT>
```

Ce qui apparaîtra à l'écran :
facile le javascript

- La méthode *toLowerCase()* permet de convertir l'objet *mot* en minuscules, les autres caractères sont laissés tels quels

L'inverse de la méthode *toLowerCase()* (pour passer le tout en majuscules) est *toUpperCase*.



IV. Les fonctions

Une fonction est une série d'instructions qui retourne une valeur, vous pourrez tout automatiser.

Prenons un exemple d'une fonction pour afficher bonjour :

```
<SCRIPT language="Javascript">
```

```
function Salut(nom)
```

```
{  
window.document.write("Bienvenue" + nom +  
"!<br>");  
}
```

```
Salut("Philippe") ;
```

```
Salut("Abdel") ;
```

```
Salut("Franck") ;
```

```
</SCRIPT>
```

Ce qui apparaîtra à l'écran :

Bienvenue Philippe !

Bienvenue Abdel !

Bienvenue Franck !

- *function Salut* indique que vous créez une fonction qui s'appellera *Salut*
- (*nom*) est un paramètre, c'est en fait toutes les variables que la fonction va utiliser
- {...} le début et la fin des accolades indiquent le début et la fin de la fonction, entre les deux se trouve le contenu de la fonction, ce qu'on veut automatiser
- *Salut("Philippe") ;* on fait appel à la fonction en indiquant entre parenthèse ses paramètres (ici le nom)

Prenons un autre exemple pour effectuer le calcul du périmètre d'un rectangle :

```
<SCRIPT language="Javascript">
```

```
function perimetre_rectangle(longueur,  
largeur)
```

```
{  
var perimetre = (2 * longueur) + (2 *  
largeur) ;  
return perimetre ;  
}
```

```
window.document.write("Le périmètre du  
rectangle est : " +
```

```
perimetre_rectangle(2, 4.5)) ;
```

```
</SCRIPT>
```

Ce qui apparaîtra à l'écran :

Le périmètre du rectangle est 13

- *function perimetre_rectangle* indique que vous créez une fonction *perimetre_rectangle*
- (*longueur, largeur*) sont les paramètres, les variables que la fonction va utiliser
- {...} on met entre les accolades la formule du périmètre en utilisant les paramètres
- *return perimetre* est une instruction qui indique qu'il faut renvoyer le résultat de la fonction
- *perimetre_rectangle(2,4.5)* on lui demande d'écrire le résultat de la fonction quand le paramètre longueur est égal à 2 et le paramètre largeur est égal à 4.5. Vous remarquez que pour les nombres décimaux on met un point et non une virgule.



VI. Les conditions

Elles permettent de donner des ordres différents à Javascript en fonction du cas par exemple égal, supérieur, inférieur... Voici le tableau correspondant à ces différents cas :

Symbole	Signification
&&	Et
(alt gr + 6)	ou
==	Est égal à
>	Est supérieur à
<	Est inférieur à
>=	Est supérieur ou égal à
<=	Est inférieur ou égal à
!=	Est différent de

A. Forme classique

Prenons un exemple en langage courant :

SI tu es sage (1^{ère} possibilité)

ALORS tu auras des bonbons

SINON tu n'auras pas de bonbons (2^{ème} possibilité)

Voilà ce que cela donne en langage Javascript :

```
<SCRIPT language="Javascript">
var sage = "oui" ;
var notes = 12 ;
if (sage == "oui" && notes>=10)
{
window.document.write("C'est bien mon
petit.<br>") ;
var bonbons = "oui" ;
}
else
{
window.document.write("Ce n'est pas bien
mon petit.<br>") ;
var bonbons = "non" ;
}
window.document.write("La décision du
père Noël pour donner des bonbons : " +
bonbons) ;
</SCRIPT>
```

Ce qui apparaîtra à l'écran :

C'est bien mon petit

La décision du père Noël pour donner des bonbons : oui

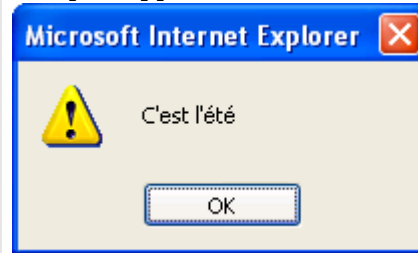
- `var sage = "oui"` On indique que la variable `sage` a une valeur de oui
- `var notes = 12` On indique que la variable `notes` a une valeur de 12
- `if (sage == "oui" && notes>=10)` Si la variable `sage` est égale à `oui` et que la variable `notes` est supérieure ou égale à 10 alors on fait ce qu'il y a entre accolades (c'est la condition)
- `Else {...}` Sinon (donc que `sage` n'est pas égale à `oui` ou que les notes sont inférieures à 10) alors on fait ce qu'il y a entre accolades
- On peut noter la présence d'un + dans le `write()` en fait cela permet d'insérer une chaîne de caractères qu'on place entre doubles quotes (ici *La décision du père Noël pour donner des bonbons :*) et une variable qu'on place sans doubles quotes (ici `bonbons`), on appelle ceci la concaténation.

B. Le Switch

Pour en finir avec les conditions voyons une autre forme conditionnelle :

```
<SCRIPT language="Javascript">
var saison = "été" ;
switch (saison)
{
case "printemps":
alert("C'est le printemps") ;
break
case "été":
alert("C'est l'été") ;
break
case "automne":
alert("C'est l'automne") ;
break
case "hiver":
alert("C'est l'hiver");
break
default:
alert("Saison inconnue") ;
}
</SCRIPT>
```

Ce qui apparaîtra à l'écran :



- *switch (saison){...}* fonction conditionnelle, on indique entre parenthèses la variable avec laquelle on travaille (ici *saison*).
- *case "printemps":* pour le cas *saison* égale à *printemps* on indique ce qu'il faut faire (ici on affiche *C'est le printemps* par l'intermédiaire d'un *alert*). Vous noterez qu'avec cette fonction on ne peut faire qu'égal à
- *case "été":* pour le cas *saison* égale à *été* on indique ce qu'il faut faire (ici on affiche *C'est l'été* par l'intermédiaire d'un *alert*) et ainsi de suite pour tous les cas...
- *Break* doit être placé à la fin de chaque cas, il permet de faire sortir Javascript du *switch*
- *Default* : cela indique le cas pour toute autre possibilité



VII. Les boucles

Une boucle permet de répéter plusieurs fois une instruction. Pour ne pas que les boucles se répètent à l'infini il faut indiquer une condition. Tant que la condition est remplie les instructions seront répétées, dès qu'elle n'est plus remplie, on sort de la boucle.

A. Le WHILE

Voici un exemple dans le langage courant :

ARGENT = 3 euro

TANT QUE tu n'as pas 5 euro

ALORS tu n'as pas de bonbons

ARGENT = ARGENT + 1 euro

Fin des instructions, on recommence tant que la condition n'est pas remplie, c'est une boucle !

Voici un autre exemple correspondant en Javascript :

```
<SCRIPT language="Javascript">
var compteur = 2 ;
while (compteur<5)
{
window.document.write("Ligne de
test<br>") ;
compteur=compteur+1 ;
}
</SCRIPT>
```

Ce qui apparaîtra à l'écran :
Ligne de test
Ligne de test
Ligne de test

- *var compteur = 2* on donne une valeur de 2 à la variable *compteur* (c'est l'initialisation)
- *while (compteur < 5) {...}* Tant que la variable *compteur* est inférieure à 5 (c'est la condition) il faut faire ce qu'il y a entre accolades
- *compteur=compteur+1* on ajoute 1 à la valeur de la variable *compteur* c'est ce qu'on appelle l'incréméntation. Ainsi la condition sera remplie 3 fois. ($2+1=3$, $3+1=4$, $4+1=5$)

B. Le FOR

Il existe une autre forme pour faire des boucles, le *for* :

```
<SCRIPT language="Javascript">
for (compteur = 1; compteur <= 5;
compteur++)
{
window.document.write("Ligne de test
n°"+compteur+"<br>") ;
}
</SCRIPT>
```

Ce qui apparaîtra à l'écran :
Ligne de test n°1
Ligne de test n°2
Ligne de test n°3
Ligne de test n°4
Ligne de test n°5

- *for (compteur = 1; compteur <= 5; compteur++){...}* trois éléments sont séparés par des points virgules :
 - *compteur = 1* initialisation du compteur à 1
 - *compteur <= 5* condition, tant qu'elle est remplie, elle sera re-exécutée
 - *compteur++* incréméntation qui ajoute à *compteur* la valeur 1 (équivalent à *compteur=compteur+1*)
- on place entre accolades ce qu'il faut répéter



VII. Les tableaux

Les tableaux sont générés à partir de la classe `array()`, ils vont permettre à une variable de prendre plusieurs valeurs par l'intermédiaire de cases. Il faudra, pour afficher la variable, préciser sa case. Voici un exemple :

```
<SCRIPT language="Javascript">
var tableauobjet = new Array() ;
tableauobjet[0]= "Chien" ;
tableauobjet[1]= "Chat" ;
tableauobjet[2]= "Poisson" ;
window.document.write("J'ai un " +
tableauobjet[1]) ;
</SCRIPT>
```

Ce qui apparaîtra à l'écran :
J'ai un Chat

- On commence par créer un objet `tableauobjet` en lui attribuant la classe `Array()` : `new Array()`
- On dit que l'objet `tableauobjet` à plusieurs cases (leur numéro se trouve entre crochets, par exemple `tableauobjet[0]` correspond à la case 0). On indique pour chaque case de l'objet `tableauobjet` sa valeur (la case 2 à pour valeur `Poisson` par exemple)
- Pour afficher la valeur d'une case, on procède comme avec une variable ordinaire, en précisant entre crochets la case que l'on souhaite voir apparaître (ici `tableauobjet[1]`)

On peut aussi changer le nom des cases :

```
<SCRIPT language="Javascript">
var tableauobjet = new Array() ;
tableauobjet["prénom"]="Léni" ;
tableauobjet["nom"]="CHON" ;
tableauobjet["adresse"]="3, rue Barbe" ;
tableauobjet["ville"]="Nancy" ;
window.document.write("Bonjour " +
tableauobjet["prénom"]) ;
</SCRIPT>
```

Ce qui apparaîtra à l'écran :
Bonjour Léni

- On commence par créer un objet `tableauobjet` en lui attribuant la classe `Array()` : `new Array()`
- On dit que l'objet `tableauobjet` à plusieurs cases (leur nom se trouve entre crochets, par exemple `tableauobjet["adresse"]` correspond à la case de nom `adresse`). Il s'agit donc case renommer donc on n'oublie pas de mettre des doubles quotes. On indique pour chaque case de l'objet `tableauobjet` sa valeur (la case `ville` à pour valeur `Nancy` par exemple)
- Pour afficher la valeur d'une case, on procède comme avec une variable ordinaire, en précisant entre crochets la case que l'on souhaite voir apparaître (ici `tableauobjet["prénom"]`)



V. Les événements

Un événement est une action de l'utilisateur qui va provoquer une interactivité. Voici une liste des différents événements.

Evénement	Description	Eléments
Click (onClick)	Se produit lorsque l'utilisateur clique sur l'élément associé à l'événement	Lien hypertexte, Bouton, Case à cocher, Bouton radio, Bouton Reset
Load (onLoad)	Se produit lorsque le navigateur de l'utilisateur charge la page en cours	Page du navigateur
Unload (onUnload)	Se produit lorsque le navigateur de l'utilisateur quitte la page en cours	Page du navigateur
MouseOver (onMouseOver)	Se produit lorsque l'utilisateur positionne le curseur de la souris au-dessus d'un élément	Lien hypertexte
MouseOut (onMouseOut)	Se produit lorsque le curseur de la souris quitte un élément	Lien hypertexte
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	Liste de sélection d'un formulaire, Champ de texte et zone de texte
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	Liste de sélection d'un formulaire, Champ de texte et zone de texte
Change (onChange)	Se produit lorsque l'utilisateur modifie le contenu d'un champ de données	Liste de sélection d'un formulaire, Champ de texte et zone de texte
Select (onSelect)	Se produit lorsque l'utilisateur sélectionne un texte (ou une partie d'un texte) dans un champ de type "text" ou "textarea"	Champ de texte et zone de texte
Submit (onSubmit)	Se produit lorsque l'utilisateur clique sur le bouton de soumission d'un formulaire	Bouton d'envoi de formulaire

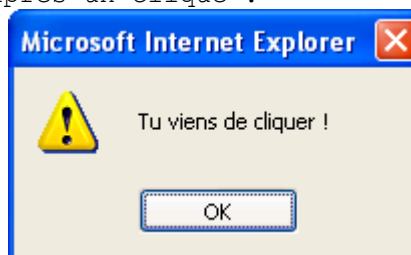
Voici un exemple :

```
<a href="#" onClick="alert('Tu viens de cliquer !') ;">Cliquez ici!</a>
```

Ce qui apparaîtra à l'écran :

[Cliquez ici](#)

Puis après un clique :



- On indique qu'en cas de clique sur le lien (événement) il y aura une action. Pas la peine de mettre la balise `<script>` car on travaille directement sur une balise, ici `<a>` (lien hypertexte). On place donc `onClick` comme attribut dans la balise `<a>`
- On ne souhaite pas que le lien redirige vers une page donc on indique `#` dans l'attribut `href`. On ajoute après le `onClick` l'action à exécuter entre doubles quotes, juste après un `=` (ici c'est un `alert('Tu viens de cliquer !')`)



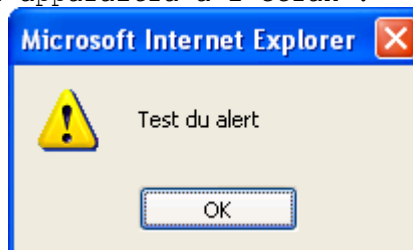
V. L'objet window

Voici des éléments qui permettent de manier les fenêtres du navigateur.

➤ Alert()

```
<SCRIPT language="Javascript">  
alert("Test du alert") ;  
</SCRIPT>
```

Ce qui apparaîtra à l'écran :

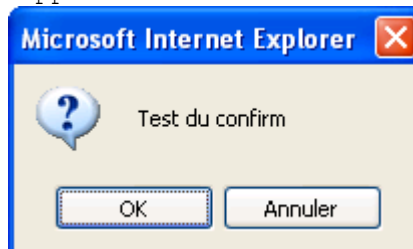


- L'utilisateur n'a que la possibilité de cliquer sur OK

➤ Confirm()

```
<SCRIPT language="Javascript">  
if confirm("Test du confirm")  
{  
alert("Tu as cliqué sur OK") ;  
}  
else  
{  
alert("Tu as cliqué sur annuler") ;  
}  
</SCRIPT>
```

Ce qui apparaîtra à l'écran :

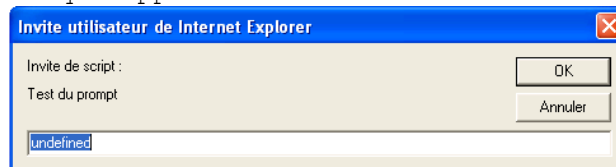


- La méthode renvoie la valeur de *true* quand l'utilisateur clique sur OK (passera donc dans le *if*) et *false* dans le cas contraire (*else*)

➤ Prompt() : invite de commande

```
<SCRIPT language="Javascript">  
var reponse = prompt("Test du  
prompt","Votre réponse") ;  
</SCRIPT>
```

Ce qui apparaîtra à l'écran :



- On indique entre doubles quotes le texte d'invite puis toujours entre doubles quotes, séparés par une virgule, le texte du champ de saisie
- La méthode renvoie la valeur saisie par l'utilisateur ou NULL s'il n'a rien saisi dans la variable *reponse*

➤ Open() : permet d'ouvrir une nouvelle fenêtre

```
<SCRIPT language="Javascript">  
window.open("http://www.site.com", "T  
itre", "resizable=no, width=100") ;  
</SCRIPT>
```

Ce qui apparaîtra à l'écran :



- On indique tout d'abord l'url de la fenêtre à ouvrir entre doubles quotes, ici *http://www.site.com*
- On précise après une virgule entre doubles quotes le nom de la fenêtre, ici *Titre*

- On précise enfin l'ensemble des options séparées par des virgules, ici *resizable*, *width*...
Voici les différentes options possibles :

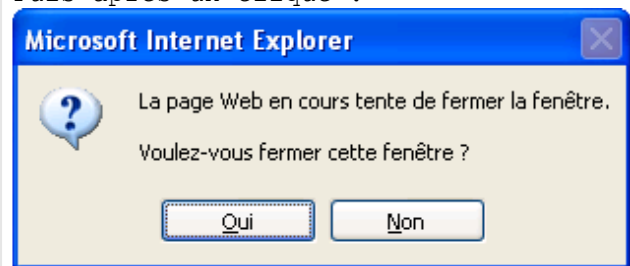
Option	Description
<i>directory = yes/no</i>	Affiche ou non les boutons de navigation
<i>location = yes/no</i>	Affiche ou non la barre d'adresse
<i>menubar = yes/no</i>	Affiche ou non la barre de menu (fichier, édition, ...)
<i>resizable = yes/no</i>	Définit si la taille de la fenêtre est modifiable ou non
<i>scrollbars = yes/no</i>	Affiche ou non les ascenseurs (barres de défilement)
<i>status = yes/no</i>	Affiche ou non la barre d'état
<i>toolbar = yes/no</i>	Affiche ou non la barre d'outils
<i>width = largeur (en pixels)</i>	Définit la largeur
<i>height = hauteur (en pixels)</i>	Définit la hauteur

➤ **Close()** : permet de fermer une fenêtre

```
<A href="#" onClick="window.close();" >Cliquez ici pour fermer la fenêtre</A>
```

Ce qui apparaîtra à l'écran :
[Cliquez ici pour fermer la fenêtre](#)

Puis après un clique :

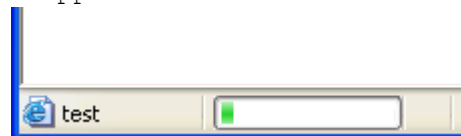


- On précise qu'à l'action de clique (*onClick*) il y aura un événement sur l'élément *window* (correspond à la page actuelle) de fermeture (*close()*)

➤ **Status** : message temporaire lors du chargement de la page

```
<SCRIPT language="Javascript">
window.status="test";
</SCRIPT>
```

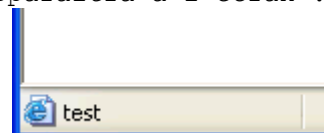
Ce qui apparaîtra à l'écran :



➤ **DefaultStatus** : message dans la barre de statut

```
<SCRIPT language="Javascript">
window.defaultStatus="test";
</SCRIPT>
```

Ce qui apparaîtra à l'écran :



➤ **History.go()** : aller sur une autre page de l'historique

```
<A href="#"
onClick="window.history.go(-1);" >Retour</A>
<br>
<A href="#"
onClick="window.history.go(+1);" >Suivant</A>
```

Ce qui apparaîtra à l'écran :
[Retour](#)
[Suivant](#)

- On précise qu'à l'action de clique (*onClick*) il y aura un événement sur l'élément *window* (correspond à la page actuelle) d'historique (*history()*) qui correspondra à la page précédente avec le paramètre *-1* et à la page suivante avec *+1*.

➤ **Location** : rediriger vers une autre page

```
<SCRIPT language="Javascript">  
window.location.href =  
"http://www.site.com";  
</SCRIPT>
```

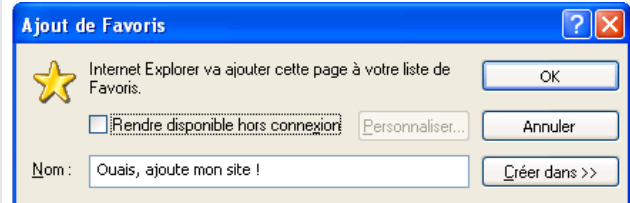
Ce qui apparaîtra à l'écran :
(la page <http://www.site.com>)

- L'objet *location* avec la propriété *href* permet de faire une redirection vers la page <http://www.site.com>

➤ **AddFavorite()** : mettre une page dans les favoris

```
<a href="#"  
onClick="window.external.AddFavorite  
( 'http://www.site.com', 'Ouais,  
ajoute mon site !' ) ;">Lien</a>
```

Ce qui apparaîtra à l'écran :

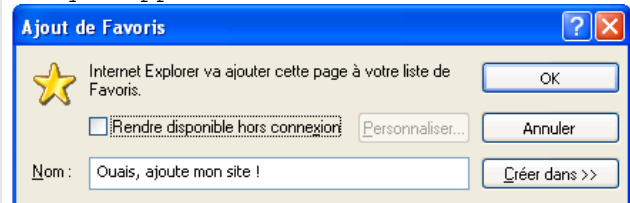


- L'objet *external* avec la méthode *AddFavorite* permet de demander à l'utilisateur de mettre le site <http://www.monsite.com> avec comme nom par défaut *Ouais, ajoute mon site !* en Favoris

➤ **AddFavorite()** : mettre une page dans les favoris

```
<a href="#"  
onClick="window.external.AddFavorite  
( 'http://www.site.com', 'Ouais,  
ajoute mon site !' ) ;">Lien</a>
```

Ce qui apparaîtra à l'écran :



- L'objet *external* avec la méthode *AddFavorite* permet de demander à l'utilisateur de mettre le site <http://www.monsite.com> avec comme nom par défaut *Ouais, ajoute mon site !* en Favoris



V. L'objet date

Voici une classe qui permet de se repérer dans le temps.

```
<SCRIPT language="Javascript">
var dateobjet = new Date() ;
window.document.write("Il est : " +
dateobjet.getHours()
+ ":" + dateobjet.getMinutes() + "
heures") ;</SCRIPT>
```

Ce qui apparaîtra à l'écran :
hh : mm

- On commence par créer un objet *dateobjet* en lui attribuant la classe *date()* : *new date()*
- On lui demande d'afficher l'heure avec la méthode *getHours()* de l'objet *dateobjet* qu'on a créé juste auparavant

Il est également possible de définir une date précise :

```
<SCRIPT language="Javascript">
var dateobjet = new Date(2000,0,1) ;
</SCRIPT>
```

Ce qui apparaîtra à l'écran :

- On crée un objet *dateobjet* en lui attribuant la classe *date()* en écrivant : *new date()*
- On indique entre parenthèses à la classe *date()* l'année, le mois et le jour. (ici le 1^{er} janvier 2000)

Les paramètres s'indiquent ainsi dans la classe *date()* : *year, month, day, hours, minutes, seconds* ou *year, month, day*.

Voici une liste de méthodes qui peuvent s'appliquer aux objets de la classe *Date* :

Méthode	Description
<i>getDate()</i>	Permet de récupérer la valeur du jour du mois (entier entre 1 et 31)
<i>getDay()</i>	Permet de récupérer la valeur du jour de la semaine pour la date spécifiée (entier qui correspond au jour de la semaine 0 pour dimanche, 1 pour lundi...)
<i>getHour()</i>	Permet de récupérer la valeur de l'heure (entier entre 0 et 23)
<i>getMinutes()</i>	Permet de récupérer la valeur des minutes (entier entre 0 et 59)
<i>getMonth()</i>	Permet de récupérer le numéro du mois (entier entre 0 et 11, 0 : janvier, 1 : février...)
<i>getYear()</i>	Permet de récupérer le nombre d'années depuis 1900.
<i>getTime()</i>	Permet de récupérer le nombre de secondes depuis le 1 ^{er} janvier 1970 (date de création d'Unix)



IV. L'objet math

➤ Minimum et Maximum

```
<SCRIPT language="Javascript">
var nombre =
Math.max(10,20,50,30,40) ;
window.document.write(nombre);
</SCRIPT>
```

Ce qui apparaîtra à l'écran :
50

- L'objet *Math* associé à la méthode *max()* retourne le plus grand des entiers donnés en paramètre

L'inverse de la méthode *max()* (pour retourner le plus petit des entiers) est *min()*.

➤ Racine carré

```
<SCRIPT language="Javascript">
var nombre = Math.sqrt(25) ;
window.document.write(nombre);
</SCRIPT>
```

Ce qui apparaîtra à l'écran :
5

- L'objet *Math* associé à la méthode *sqrt()* retourne la racine carrée de 25

➤ Puissance

```
<SCRIPT language="Javascript">
var nombre = Math.pow(4,2) ;
window.document.write(nombre);
</SCRIPT>
```

Ce qui apparaîtra à l'écran :
16

- L'objet *Math* associé à la méthode *pow()* retourne le nombre 4 à la puissance 2

➤ Arrondi

```
<SCRIPT language="Javascript">
var nombre = Math.round(4.041) ;
window.document.write(nombre);
</SCRIPT>
```

Ce qui apparaîtra à l'écran :
4

- L'objet *Math* associé à la méthode *round()* retourne le l'arrondi de 4,041

➤ Entier supérieur & Entier inférieur

```
<SCRIPT language="Javascript">
var nombre = Math.ceil(4.041) ;
window.document.write(nombre);
</SCRIPT>
```

Ce qui apparaîtra à l'écran :
5

- L'objet *Math* associé à la méthode *ceil()* retourne l'entier supérieur ou égal à la valeur 4,041

L'inverse de méthode *ceil()* (pour retourner l'entier inférieur) est *floor()*.

➤ Random

```
<SCRIPT language="Javascript">
var nombre =
Math.round(100*Math.random()) ;
window.document.write(nombre);
</SCRIPT>
```

Ce qui apparaîtra à l'écran :
53

- L'objet *Math* associé à la méthode *random()* et *round()* permet de générer un nombre entier compris entre 0 et 100