Les bases du JavaScript

David Roche, lycée G Fichet Bonneville (Haute-Savoie)

Notes de l'auteur :

- Ce document est à l'origine destiné aux élèves ayant choisi de suivre la spécialité "Informatique et Sciences du Numérique" en terminale scientifique.
- Ce document n'est pas une "bible" du JavaScript, il a uniquement pour but de permettre aux élèves une première approche (en autonomie) de la programmation en JavaScript. Ce document ne prétend pas se substituer au travail en classe avec l'enseignant (il y a peu d'exemples et pas d'exercices; ils seront donnés en classe)
- Un deuxième document "JavaScript pour aller plus loin...." devrait voir le jour prochainement, il abordera des sujets tels que : l'objet en JavaScript, le DOM, le DHTML,..... Il sera hors programme ISN, mais permettra aux élèves qui le désirent "d'aller plus loin".
- Il est indispensable d'avoir quelques notions de HTML pour pouvoir suivre ce document dans de bonnes conditions (voir "HTML et CSS : les bases" (même auteur))
- Les généralités sur la programmation, les variables et les fonctions ont déjà été abordées dans le document "Introduction à la programmation" (même auteur).
- Pour toutes remarques et commentaires, n'hésitez pas à me contacter : informatiqueaulycee@gmail.com

David Roche

Introduction

Javascript est un langage de programmation de script très utilisé dans le développement des sites web. Il s'exécute (presque) exclusivement dans les navigateurs web.

Ce langage a été créé en 1995 pour le compte de la société Nescape. Malgré son nom, il n'a presque rien à voir avec le langage Java développée par Sun.

En ce qui nous concerne, nous n'allons pas (pour l'instant) nous intéresser au développement web avec JavaScript, mais apprendre les bases de la programmation (variable, fonction, condition, boucle). Malgré tout, puisque JavaScript s'exécute dans un navigateur web (pour nous Firefox) nous allons devoir écrire quelques lignes de HTML.

Premier programme

Nous allons créer une page de code HTML (enfin plutôt XHTML)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">
<head>
    <title>Programmation JavaScript</title>
</head>
<body>
</body>
</html>
```

Nous allons ajouter une ligne de code qui va "demander" à la page web d'exécuter du code JavaScript :

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">
<head>
    <title>Programmation JavaScript</title>
     <script type="text/javascript" src="monProgramme.js"></script>
</head>
<body>
</body>
</html>
```

Nous avons donc ajouté la balise <script>, cette balise accepte 2 attributs "type", qui aura toujours pour valeur "text/javascript" et "src" qui correspond au chemin du fichier JavaScript (extension .js) qui doit être exécuté. Dans notre exemple, notre fichier "JavaScript" sera dans le même dossier que notre fichier "HTML" et se nommera "monProgramme.js". Nous n'aurons plus à modifier ce fichier (sauf si vous décidez de modifier le nom du fichier JavaScript).

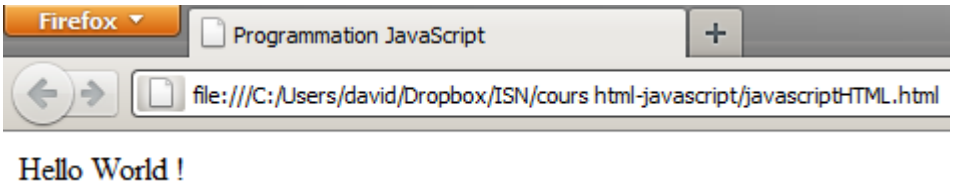
Nous allons maintenant créer notre fichier "monProgramme.js" dans notepad++ :
Fichier -> Nouveau, dans le menu Langage, choisir la lettre J puis JavaScript. Il vous reste à "Enregistrer sous..." (nom du fichier : monProgramme).

Passons à l'écriture du code (dans le fichier monProgramme.js) :

```
document.write ("Hello World !");
```

Enregistrez votre fichier (Ctrl+S) et ouvrez le fichier HTML que nous avons créé à l'aide d'un

navigateur (un double clic sur le fichier devrait suffire).
Voici le résultat dans Firefox :



À ce stade, vous devez juste avoir compris que le code document.write vous permet d'afficher la chaîne de caractère contenue entre les guillemets (dans notre exemple : Hello World !). Il est aussi important de noter qu'en JavaScript, une ligne de code doit se terminer par un point virgule.

Pour votre information, il est aussi possible d'inclure le code JavaScript directement dans le code HTML :

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">
<head>
    <title>Programmation JavaScript</title>
       <script type="text/javascript">
       document.write ("Hello World !");
       </script>
</head>
<body>

</body>
</html>
```

Nous n'utiliserons pas cette méthode, le mélange code HTML et code JavaScript dans un même fichier risque de compliquer inutilement les choses, surtout pour un débutant. Par la suite, je ne vous reparlerai plus du fichier HTML qui restera identique, nous nous concentrerons uniquement sur notre fichier JavaScript.

Les variables en JavaScript

Vous devez tout d'abord déclarer votre variable en utilisant le mot clé var.

```
var maVariable ;
```

Puis vous pouvez attribuer une valeur à votre variable

```
maVariable=4;
```

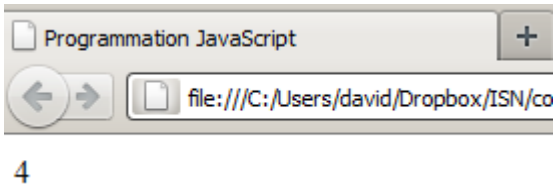
Ces 2 actions peuvent être couplées :

```
var maVariable=4;
```

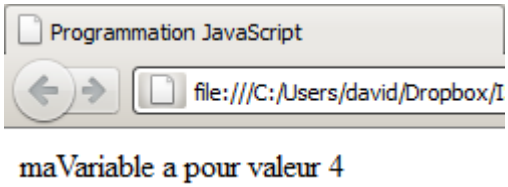
Voici un premier exemple :

```
var maVariable=4;
document.write (maVariable);
```

Résultat



Il est tout à fait possible de faire aussi ceci



avec le code suivant :

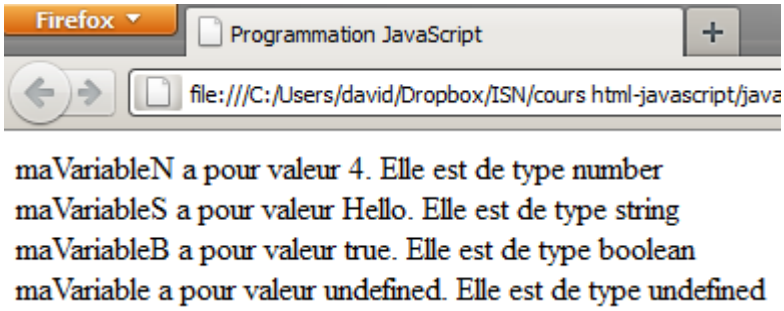
```
var maVariable=4;
document.write ("maVariable a pour valeur ", maVariable);
```

Remarquez juste la virgule entre la chaîne de caractères (entre guillemets) et la variable (sans guillemets). Je pense que vous avez tous compris qu'au moment de l'exécution du code le mot maVariable est remplacé par la valeur de maVariable (ici 4).
JavaScript est un langage faiblement typé (voir "Introduction à la programmation" pour plus de détails). Il n'est donc pas nécessaire de préciser le type de la variable. Attention, cela ne veut pas dire que votre variable n'a pas de type, juste que le programmeur n'a pas besoin de le préciser.
En JavaScript les types possibles sont : string (chaîne de caractères), boolean et number (number est divisé en 2 : les entiers (type integer) et les nombres à virgule (type float). Attention pour le type float vous devez utiliser le point à la place de la virgule, par exemple le nombre pi ne s'écrit pas 3,14 mais 3.14 (convention anglo-saxonne)).
La fonction typeof renvoie le type de la variable qui a été passé comme argument (pour la notion de fonction, revoir le document "Introduction à la programmation").

```
var maVariableN=4;
document.write ("maVariableN a pour valeur ", maVariableN, ". Elle est de type ",
typeof(maVariableN), "</br>");
var maVariableS="Hello";
document.write ("maVariableS a pour valeur ", maVariableS, ". Elle est de type ",
typeof(maVariableS), "</br>");
var maVariableB=true;
document.write ("maVariableB a pour valeur ", maVariableB, ". Elle est de type ",
typeof(maVariableB), "</br>");
var maVariable;
document.write ("maVariable a pour valeur ", maVariable, ". Elle est de type ",
typeof(maVariable), "</br>");
```

Attention chaque ligne de code se trouve ci-dessus sur 2 lignes.
Vous avez dû noter que pour une variable de type string, la valeur est entre guillemets.

Résultat



Deux choses à bien noter dans cet exemple :

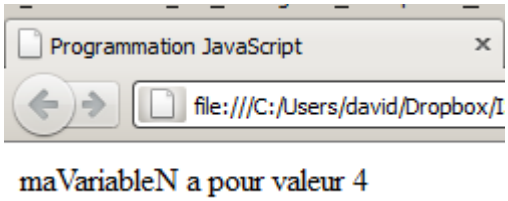
- Vous avez sans doute reconnu la balise </br> ("retour à ligne") du HTML. Sans trop entrer dans les détails document.write vous permet d'écrire du code HTML, il est donc logique d'utiliser la balise </br> pour effectuer un retour à la ligne (toute autre balise est aussi utilisable, essayez avec une balise par exemple). Enlevez les </br> du code pour vous convaincre de leur utilité. Enfin, attention, tout comme le texte, les balises HTML doivent être entre guillemets. La variable et la fonction typeof ne sont pas entre guillemets.
- Une variable quand elle a été déclaré, mais qu'aucune valeur ne lui a été attribué, a pour valeur undefined et est de type undefined.

Que se passe-t-il quand on utilise une variable qui n'a même pas été déclaré ?

```
var maVariableN=4;
document.write ("maVariableN a pour valeur ", maVariableN);
document.write ("maVariable a pour valeur ", maVariable);
var maVariableS="Hello";
document.write ("maVariableS a pour valeur ", maVariableS);
```

1ère ligne nous déclarons la variable maVariableN et nous lui attribuons la valeur (numérique) 4.
2ème ligne, nous utilisons la variable maVariableN
3ème ligne nous utilisons une variable maVariable qui n'a pas été déclaré
4ème ligne nous déclarons la variable maVariableS et nous lui attribuons la valeur (chaîne) "Hello"
5ème ligne nous utilisons maVariableS.

Voici le résultat

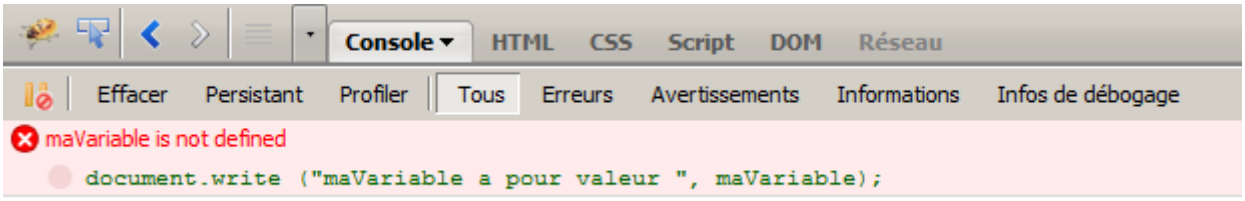


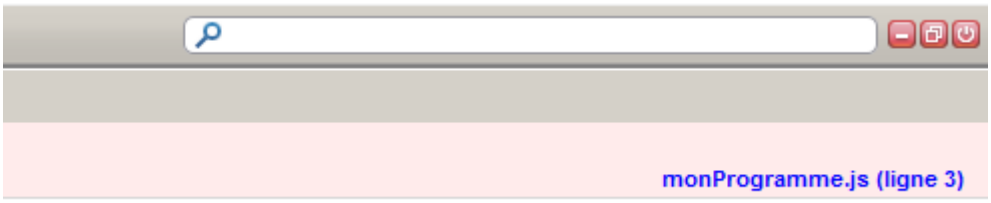
Le programme s'est arrêté à la 3ème ligne. Utiliser une variable non déclarée est une erreur qui provoque l'arrêt du programme.
Problème, rien ne nous signale cette erreur dans le navigateur (c'est assez logique, à la base un navigateur n'est pas un outil de développement !).
Nous allons installer une extension pour Firefox (firebug) qui nous nous donnera des informations sur les éventuelles erreurs (on parle de bug). Une recherche sur internet devrait vous permettre de trouver la procédure à suivre pour installer ce genre d'extension.
Une fois installé et Firefox redémarré, vous devriez trouver une icône supplémentaire (un insecte, bug = insecte en anglais) :

Un clic sur l'insecte ouvre firebug

Vous pouvez constater que nous avons ici notre page, enfin, pas tout à fait. Comme déjà dit au-dessus, notre script JavaScript modifie le code HTML (plus exactement, il modifie le DOM (Document Object Modele), mais nous verrons cela dans le deuxième document "JavaScript pour aller plus loin...."). À chaque exécution notre script crée une nouvelle page HTML, c'est cette page qui s'affiche (et que nous voyons ici). Bref, tout cela est assez compliqué, nous aurons l'occasion d'en reparler.

Revenons à firebug est au débogage de notre code JavaScript :
Cliquez sur Console (si la console est désactivée, activez-là), et rechargez votre page.





Voilà, toutes les informations sont là :

l'erreur : "maVariable is not defined"
la ligne qui pose problème
le nom du fichier et le numéro de ligne (monProgramme.js (ligne 3)

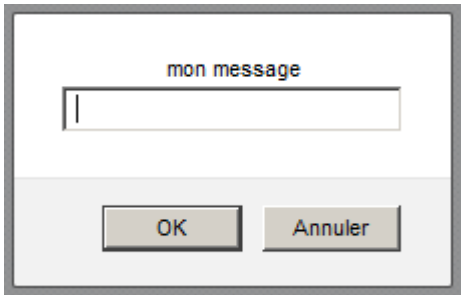
Firebug est un outil formidable que nous aurons l'occasion d'utiliser à de très nombreuses reprises.

Nous allons maintenant étudier la méthode (une méthode ressemble beaucoup à une fonction, nous verrons cela dans le deuxième document, pour l'instant, dans votre tête, vous devez avoir méthode = fonction) qui permet à l'utilisateur de rentrer des valeurs au clavier : la méthode prompt( ).

Nous aurons une structure de la forme :

var maVariable = prompt (message)

En réponse à la méthode prompt, le navigateur affichera une fenêtre avec : un bouton OK, un bouton Annuler, un message et une zone de saisie



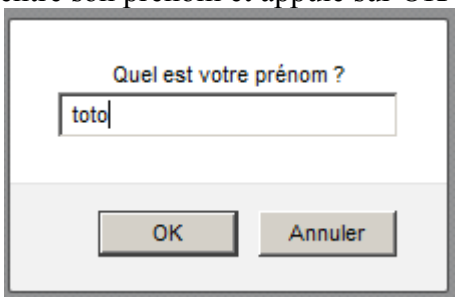
L'utilisateur va alors saisir (au clavier) du texte dans la zone de saisie. La validation avec le bouton OK permettra d'attribuer le texte entré par l'utilisateur à la variable maVariable. Au cas où l'utilisateur ne rentrerai rien ou qu'il appuierai sur Annuler on aura alors maVariable = null (pas de valeur).

Voici un exemple

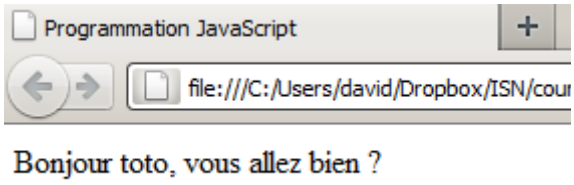
```
var prenom=prompt("Quel est votre prénom ?");
document.write ("Bonjour ", prenom, ", vous allez bien ?");
```

résultat

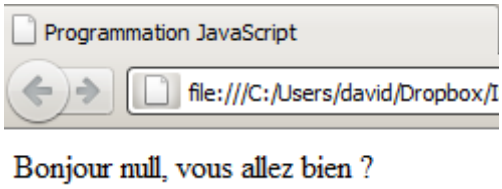
La fenêtre s'affiche, l'utilisateur entre son prénom et appuie sur OK



puis le message s'affiche sur la page



Et si l'utilisateur n'entre rien (ou appuie sur le bouton Annuler)

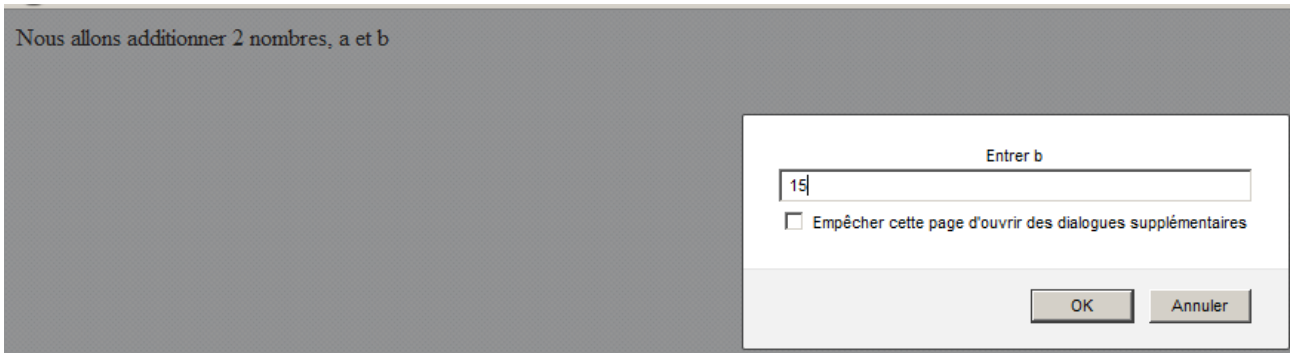


Pas très sympathique, non ?
Pour éviter ce genre de chose, nous verrons un peu plus loin l'utilisation du couple if/else (les conditions)

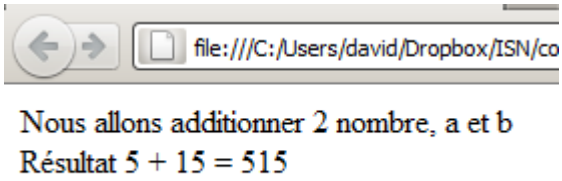
Un deuxième exemple : une machine à additionner

```
document.write ("Nous allons additionner 2 nombres, a et b </br>");
var a=prompt("Entrer a ");
var b=prompt("Entrer b ");
var resultat=a+b;
document.write ("Résultat ",a," + ",b," = ",resultat);
```

Voyons le résultat :



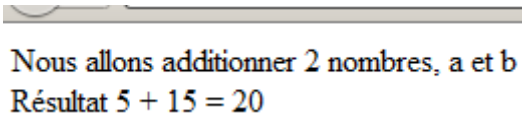
et voici l'étrange résultat



Pour essayer de comprendre ce qui se passe, faites un "typeof" sur les variables a et b.
Les variables a et b sont toutes deux de type string. Or, nous avons vu (voir le document "Introduction à la programmation") que si nous avons affaire à des chaînes de caractère le signe + est le signe de concaténation (mise bout à bout de 2 chaînes de caractère). Si nous mettons bout à bout 5 et 15, nous obtenons bien 515.
Pour que notre programme fonctionne, il faut "transformer" notre chaîne (ariable de type string) en nombre (variable de type number). Nous allons faire du transtypage (convertir un type en un autre).
Pour se faire, utilisons la méthode parseInt :

```
document.write ("Nous allons additionner 2 nombres, a et b </br>");
var as=prompt("Entrer a ");
var bs=prompt("Entrer b ");
var a=parseInt(as);
var b=parseInt(bs);
var resultat=a+b;
document.write ("Résultat ",a," + ",b," = ",resultat);
```

as et bs sont de type string, a et b sont de type nombre, le résultat est maintenant correct



Les tableaux

Les tableaux vont vous permettre de stocker plusieurs valeurs (chaîne, nombre) dans une structure unique.

Pour déclarer un tableau, la syntaxe est un peu particulière :

var monTableau = new Array( );

Le mot clé new est utilisé pour créer des objets (au sens informatique du terme, nous aurons l'occasion de revenir sur cette notion d'objet). Un tableau est donc un objet.

Ensuite, pour remplir le tableau il faut procéder comme suit :

monTableau [indice de position] = maValeur

monTableau [0] = "pomme"
monTableau [1] = "orange"
.....
ou alors avec des nombres :

monTableau [0] = 15
monTableau [1] = 20
.....

L'indice de position commence toujours à zéro.

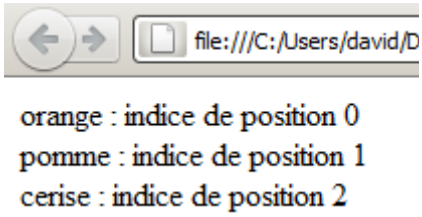
Il est aussi possible de déclarer et de remplir le tableau en même temps :

monTableau = new Array ("pomme", "orange", "cerise")
pomme aura l'indice 0, orange aura l'indice 1.....

Un exemple

```
var mesFruits = new Array ("orange","pomme","cerise");
document.write (mesFruits[0], " : indice de position 0 </br>");
document.write (mesFruits[1], " : indice de position 1</br>");
document.write (mesFruits[2], " : indice de position 2</br>");
```

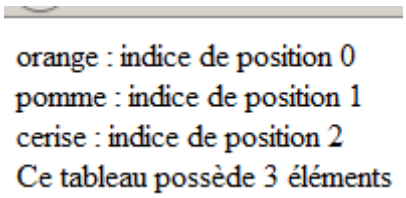
résultat



Il est possible de connaître la taille de votre tableau avec la méthode length :

```
var mesFruits = new Array ("orange","pomme","cerise");
document.write (mesFruits[0], " : indice de position 0 </br>");
document.write (mesFruits[1], " : indice de position 1</br>");
document.write (mesFruits[2], " : indice de position 2</br>");
document.write ("Ce tableau possède ", mesFruits.length, " éléments");
```

résultat

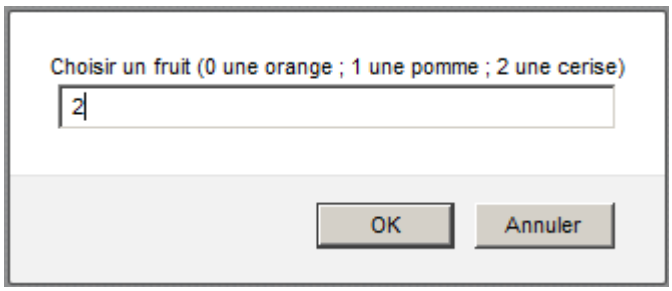


La méthode typeof est aussi utilisable avec un tableau.

Il est possible de remplacer l'indice de position par une variable

```
var mesFruits = new Array ("orange","pomme","cerise");
var i=prompt("Choisir un fruit (0 une orange ; 1 une pomme ; 2 une cerise)");
document.write ("Votre choix : une ", mesFruits[i]);
```

résultat



Votre choix : une cerise

Vous ne trouvez pas cela étrange que cela fonctionne ?
Comme nous l'avons déjà vu, prompt renvoie une chaîne de caractères, donc ici, i est de type string !
Or l'indice de position doit être de type number et pas de type string !
Avec mesFruits[i] nous devrions avoir une erreur.
Eh non, pas d'erreur, car JavaScript est capable de faire du "transtypage automatique" :
Etudions le "raisonnement" de JavaScript : "i est de type string, mais dans mesFruits[i] il doit être de type number, de mon propre chef j'applique donc un "parseInt" à i". Voilà pourquoi nous n'avons pas d'erreur !

Pour terminer sur les tableaux, nous verrons un peu plus loin que l'instruction for...in nous permettra de parcourir un tableau quasi automatiquement.

Les fonctions en JavaScript

Pour définir une fonction en JavaScript il faut utiliser le mot clé function. Nous allons avoir une structure de la forme :

function maFonction (paramètre1, paramètre2,...) {
....
....
....

return y ;
}

Prenons un exemple déjà traité dans le document "Introduction à la programmation" :

```
function calcul (x){
	var y;
	y=3*x+2;
	return y;
}
document.write ("y=3x+2 avec x = 4 </br>");
document.write ("Résultat : y = ",calcul(4));
```

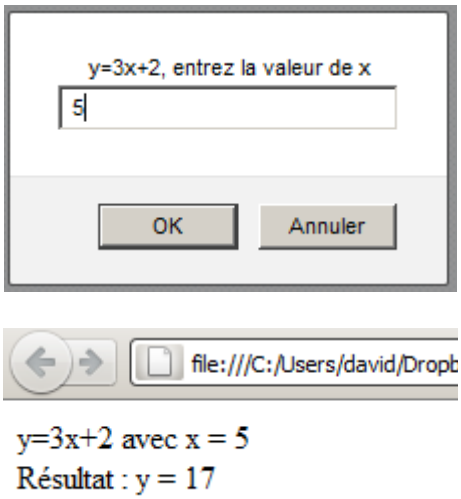
Remarquez l'indentation (décalage) du contenu de la fonction (entre l'accolade ouvrante et fermante). Cela a pour but de rendre le code plus lisible. Ce n'est pas obligatoire, mais fortement conseillé.

calcul(4) appelle la fonction calcul avec un paramètre (égal à 4). Dans le document.write de la dernière ligne, calcul(4) est "remplacé" par la valeur retournée (return y) par la fonction calcul.

Un exemple un peu plus évolué :

```
function calcul (x){
	var y;
	y=3*x+2;
	return y;
}
var valeur = prompt ("y=3x+2, entrez la valeur de x");
document.write ("y=3x+2 avec x = ", valeur, "</br>");
document.write ("Résultat : y = ",calcul(valeur));
```

résultat



Ici la valeur du paramètre de la fonction calcul est entrée par l'utilisateur.
Pour le reste, je vous laisse analyser le code (n'hésitez pas à poser des questions en cas de problème).

Comme déjà dit précédemment, une fonction peut très bien attendre plusieurs paramètres :

```
function calcul (x,b){
	var y;
	y=3*x+b;
	return y;
}
var valeur_1 = prompt ("y=3x+b, entrez la valeur de x");
var valeur_2st = prompt ("y=3x+b, entrez la valeur de b");
document.write ("y=3x+2 avec x = ", valeur_1, " et b = ",valeur_2st, "</br>");
var valeur_2=parseInt(valeur_2st);
document.write ("Résultat : y = ",calcul(valeur_1, valeur_2));
```

Attention pour cet exemple, nous avons bien un "transtypage automatique" pour valeur_1 (on ne peut pas multiplier une chaîne par 3 donc valeur_1 est forcément un nombre), mais pas pour valeur_2st (le programmeur voulait peut-être une concaténation et pas une addition ?)
Évidemment, le paramètre de la fonction peut-être de type string. Il faut aussi savoir que le return n'est pas obligatoire (une fonction peut ne rien renvoyer, juste "faire quelque chose") :

```
function afficheNom (nom){
	document.write("Bonjour", nom);
}
afficheNom("Toto");
```

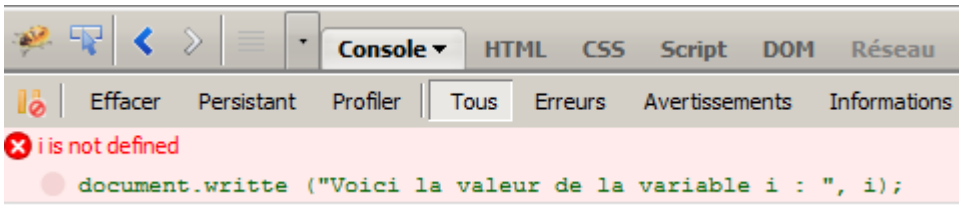
résultat

Un simple afficheNom("Toto"); vous permet d'appeler la fonction afficheNom, qui ne renvoie rien, mais qui affiche du texte.

Pour terminer cette partie sur les fonctions et le JavaScript, nous allons revenir sur une notion déjà abordée dans le document "Introduction à la programmation" : la différence entre les variables locales et les variables globales (on parle de la portée d'une variable).

Si vous essayez de faire fonctionner le programme suivant, vous aurez droit à une "belle" erreur, pourquoi ?

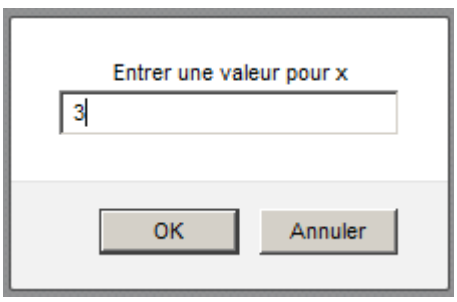
```
function maFonction (){
	var i=10;
}
document.write ("Voici la valeur de la variable i : ", i);
```



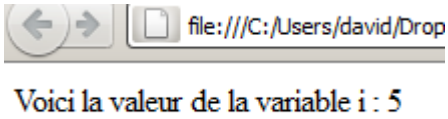
La variable i a été défini dans une fonction, elle n'existe pas en dehors de cette fonction, d'où l'erreur renvoyée par firebug.
Une variable définie en dehors d'une fonction est appelée variable globale, elle est accessible partout dans le programme.
Il faut faire très attention avec cette histoire de variable locale, vous pouvez vous retrouver devant des bugs difficilement détectables :

```
var i=5
function changerLaValeurDei (x){
	var i=5*x;
}
var a=prompt("Entrer une valeur pour x");
changerLaValeurDei(a)
document.write ("Voici la nouvelle valeur de la variable i : ", i);
```

résultat :



"Normalement" le résultat devrait être 15, eh bien non :



Je vous laisse chercher pour quelle raison cela ne fonctionne pas.

Les commentaires en JavaScript

Nos programmes commencent à être de plus en plus complexes.
Pour qu'ils restent malgré tout le plus clair possible, nous allons systématiquement ajouter des commentaires. Ces commentaires auront principalement 2 buts :

- aider une tierce personne à comprendre votre code
- assurer une maintenance efficace de votre code (il est parfois difficile de comprendre son propre code des mois après son écriture).

Un programme non commenté devient très rapidement illisible, même pour un programmeur expérimenté.
En JavaScript, il existe 2 façons d'écrire des commentaires :

- si vous écrivez votre commentaire sur une seule ligne, ce commentaire doit être précédé d'un double slash //
- si vous écrivez votre commentaire sur plusieurs lignes, il doit commencer par la combinaison /* et se terminer par la combinaison */

```
//++++++++++++++++++++++début du code++++++++++++++++++++++++++

/*Ce programme vous permet de calculer le carré d'un nombre
si l'utilisateur n'entre pas un nombre, la réponse sera NaN (Not a Number) */


// carre est la fonction qui permet de calculer le carré
function carre (x) {
      y=x*x;
      return y;
}
//demande à l'utilisateur d'entrer un nombre (transtypage "automatique")
var nombre=prompt("Entrer un nombre");
//affichage du résultat
document.write("Résultat ", carre(nombre));

//++++++++++++++++++++++++fin du code++++++++++++++++++++++++++
```

N. B. J'ai conscience que ce code n'apporte rien de plus, il a uniquement pour but de vous donner un exemple d'utilisation des commentaires.
Il faut aussi que vous sachiez que la qualité du commentaire entrera en compte lors des évaluations.

Les tests conditionnels : if /else

Vous avez déjà eu l'occasion d'aborder les tests conditionnels dans le cours de mathématiques.

Revenons sur cette structure :

si (condition) vrai fait {suite instructions 1}
sinon (sous entendu la condition est fausse) fait {suite instructions 2}

Soit var a = 5 et var b = 8

Imaginons la condition suivante : a > b, cette condition est-elle vraie ou fausse ?

a n'est pas supérieure à b donc la condition est fausse {suite instruction 1} ne sera donc pas pris en compte, en revanche {suite instruction 2} sera exécutée.
Autre exemple :

soit var a = 5

imaginons la condition a == 6 (je reviendrai un peu plus bas sur le double signe égal)

La question à se poser est : la variable a est-elle égale à 6 ? La réponse est non, la condition renvoie faux {suite instruction 1} ne sera pas pris en compte {suite instruction 2} sera exécutée.

Il faut savoir qu'en informatique le signe égal n'a pas la même signification qu'en mathématiques.

Le "simple égal" signifie "range cette valeur dans" :
var a = 5 signifie "range la valeur 5 dans la variable a" (certains langages utilisent une flèche à la place du "simple égal", ce qui je pense, est plutôt une bonne idée)

Le "double égal" se rapproche plus du signe égal des mathématiques :
Dans notre exemple, a == 5 est vraie (true) et a==9 est fausse (false)

Prenons un exemple en JavaScript :

```
var a=5
if (a==5) {
	document.write("a est égale à 5");
}
else {
	document.write("a n'est pas égale à 5");
}
```

résultat

a est égale à 5

autre exempe

```
var a=5
if (a==6) {
	document.write("a est égale à 5");
}
else {
	document.write("a n'est pas égale à 5");
}
```

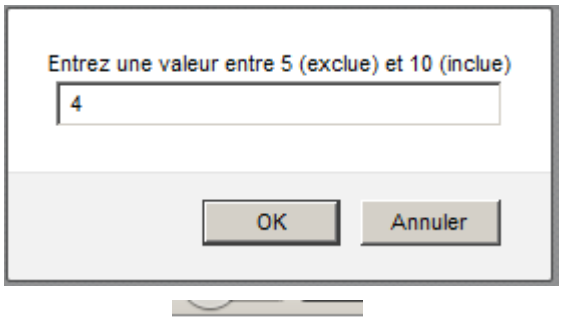
résultat

a n'est pas égale à 5

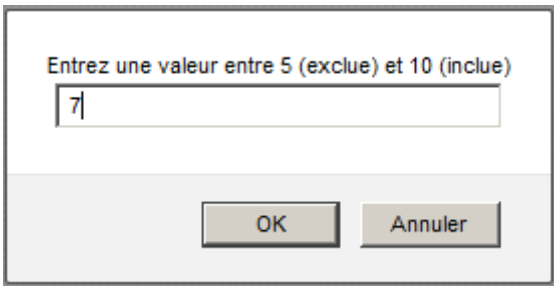
Il existe, en plus de == (égalité) et > (strictement supérieur), d'autres opérateurs :
< strictement inférieur
>= supérieur ou égal
<= inférieur ou égal
!= différent de
Il est aussi possible de faire des combinaisons de tests avec le ET logique (&&)
ou le OU logique (||).

```
var valeur=prompt("Entrez un chiffre entre 5 (exclue) et 10 (inclue)")
if ((valeur>5) && (valeur<=10)){
	document.write("Bravo, vous avez réussi !");
}
else {
	document.write("Raté");
}
```

résultat



Raté



Bravo, vous avez réussi !

Les boucles while

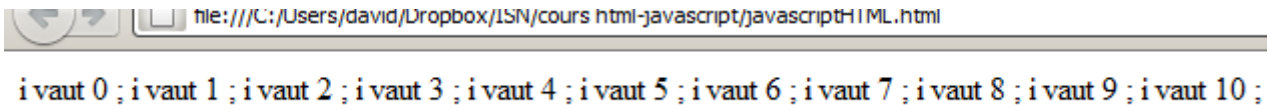
La notion de boucle est fondamentale en informatique. Une boucle permet d'exécuter plusieurs fois des instructions qui ne sont présentes qu'une seule fois dans le code.
La structure de la boucle while est la suivante :

while (condition) {
.....
.....instructions boucle.....
}

Tant que la condition reste vraie, les instructions à l'intérieur du bloc (entre les 2 accolades) seront exécutées.

```
var i=0;
while (i<=10) {
	document.write("i vaut ",i," ; ");
	i=i+1;
}
```

résultat



L'exemple ne devrait vous poser aucun problème.
J'attire votre attention sur la ligne i=i+1 :
Nous avons ici aussi un exemple de la non-équivalence du signe égal en mathématiques et en informatique. En mathématiques, écrire i=i+1 n'a aucun sens (cela reviendrait à dire par exemple que 5 = 6 !).
Ici i=i+1 veut dire : "Prends la valeur de i, ajoute un à cette valeur puis attribue cette nouvelle valeur à la variable i", on parle d'incrémentation.
Il est tellement courant d'utiliser i=i+1 que les informaticiens ont inventé "un raccourci" : i++

Voici un exemple un peu plus complet (table de multiplication) :

```
function table (a){
	var i=1;
	document.write("TABLE DES ",a, "</br>");
	while (i<=10){
		var resultat=a*i;
		document.write(a," x ",i," = ",resultat,"</br>");
		i++;
	}
}
var numS=prompt("Entrez un chiffre entre 1 et 10");
var num=parseInt(numS)
while ((num<1) || (num>10) || (isNaN(num))){
	numS=prompt("Entrez un chiffre entre 1 et 10");
	num=parseInt(numS);
}
table(num);
```

Petite nouveauté dans cet exemple, la fonction isNaN(num).

Si la variable num n'est pas de type nombre, cette fonction renvoie vraie (true). Si la variable num est de type nombre, la fonction isNaN renvoie alors faux (false).
Vous pouvez aussi constater que notre code débute par l'écriture d'une fonction. Si vous avez toute une série de fonctions à écrire, il est préférable de toutes les écrire en début de code.
Pour le reste, je vous laisse étudier cet exemple, n'hésitez pas à poser des questions si nécessaire.

La boucle do while

C'est un peu la "cousine" de la boucle while, voici sa structure :
do {
.....
.....instruction boucle.....
} while (condition)

Ici l'instruction est forcément exécutée au moins une fois (ce qui n'est pas le cas pour la boucle while)
voici le même exemple que précédemment, mais avec une boucle do while :

```
function table (a){
	var i=1;
	document.write("TABLE DES ",a, "</br>");
	while (i<=10){
		var resultat=a*i;
		document.write(a," x ",i," = ",resultat,"</br>");
		i++;
	}
}
do {
var numS=prompt("Entrez un chiffre entre 1 et 10");
var num=parseInt(numS);
} while ((num<1) || (num>10) || (isNaN(num)))
table(num);
```

Vous voyez que dans ce cas précis, l'utilisation de la boucle do while simplifie un peu le code.

La boucles for

Nous avons toujours ici affaire à une boucle, mais les conditions de répétitions sont un peu différentes :

for (début ; condition de répétition ; incrémentation) {
.....
.....instruction boucle.....
}

Reprenons notre exemple des tables de multiplication avec une boucle for :

```
function table (a){
	document.write("TABLE DES ",a, "</br>");
	for (i=1;i<=10;i++){
		var resultat=a*i;
		document.write(a," x ",i," = ",resultat,"</br>");
	}
}
do {
var numS=prompt("Entrez un chiffre entre 1 et 10");
var num=parseInt(numS);
} while ((num<1) || (num>10) || (isNaN(num)))
table(num);
```

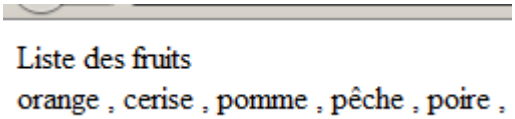
La boucle commence avec i = 1, elle va s'exécuter tant que i<=10 reste vraie, i est augmenté d'une unité à chaque boucle.

La boucle for in

for in va nous permettre de parcourir un tableau de façon "quasi automatique", voici un exemple :

```
var fruit=new Array("orange", "cerise", "pomme", "pêche", "poire");
document.write("Liste des fruits </br>");
for (var i in fruit){
	document.write(fruit[i], " , ");
}
```

résultat



Voici un dernier exemple que je vous laisse étudier seul :

```
var fruit=new Array();
var i=0;
do {
	fruit[i]=prompt("Entrez un nom de fruit (pour terminer taper fin)");
	i++
} while (fruit[i-1]!="fin");
document.write("Voici votre liste de fruits </br>");
for (var j=0 ; j<(fruit.length-1) ; j++){
	if (j==(fruit.length-2)){
		document.write(fruit[j]);
	}
	else {
		document.write(fruit[j], " , ");
	}
}
```

Quelques pistes de réflexion :
Pourquoi le `fruit[i-1]` ?
Pourquoi le `fruit.length-1` ?
Pourquoi le `fruit.length-2` ?
Bon courage et une fois de plus, n'hésitez pas à me poser des questions.