Javascript - Les variables

Septembre 2015

Le concept de variable

Une variable est un objet repéré par son nom, pouvant contenir
des données, qui pourront être modifiées lors de l'exécution du programme.

En Javascript, les noms de variables peuvent être aussi long que l'on désire, mais
doivent répondre à certains critères :

- 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!)
- Les noms de variables ne peuvent pas être les noms suivants (qui sont des noms réservés) :
  - abstract
  - boolean break byte
  - case catch char class const continue
  - debugger default delete do double
  - else export extends
  - false final finally float for function
  - goto
  - if, implements, import, in, infinity, instanceof, int, interface
  - label, long
  - native, new, null
  - package, private, protected, public
  - return
  - short, static, super, switch, synchronized
  - this, throw, throws, transient, true, try, typeof
  - var, void, volatile
  - while, with
  - sont également considérés comme mots réservés le nom des objets Javascript

Nom de variable correct	Nom de variable incorrect	Raison
Variable	Nom de Variable	comporte des espaces
Nom_De_Variable	123Nom_De_Variable	commence par un chiffre
nom_de_variable	toto@mailcity.com	caractère spécial @

nom_de_variable_123	Nom-de-variable	signe - interdit
_707	transient	nom réservé

Les noms de variables sont sensibles à la casse (le Javascript fait la différence entre un nom en majuscule et un nom en minuscules), il faut donc veiller à utiliser des noms comportant la même casse!

La déclaration de variables

Le Javascript étant très souple au niveau de l'écriture (à double-tranchant car il laisse passer des erreurs...), la déclaration des variables peut se faire de deux façons :

- soit de façon explicite, en faisant précéder la variable du mot clé *var* qui permet d'indiquer de façon rigoureuse qu'il s'agit d'une variable :

```
var chaine= "bonjour"
```

- soit de façon implicite, en laissant le navigateur déterminer qu'il s'agit d'une déclaration de variable.

Pour déclarer implicitement une variable, il suffit d'écrire le nom de la variable suivie du caractère = et de la valeur à affecter :

```
chaine= "bonjour"
```

Même si une déclaration implicite est tout à fait reconnue par le navigateur, il est plus rigoureux de déclarer les variables de façon explicite avec le mot *var*.
Voici un exemple dans lequel deux variables sont déclarées :

```
<script type="text/javascript">
<!--
var MaVariable;
var MaVariable2 = 3;
MaVariable = 2;
document.write(MaVariable*MaVariable2);
// -->
</script>
```

Portée (visibilité) des variables

Selon l'endroit où une variable est déclarée, celle-ci pourra être accessible (visible) de partout dans le script ou bien uniquement dans une portion confinée du code, on parle de « **portée** » d'une variable.

Lorsqu'une variable est déclarée sans le mot clé *var*, c'est-à-dire **de façon implicite**, elle est accessible de partout dans le script (n'importe quelle fonction du script peut faire appel à cette variable). On parle alors de **variable globale**

La portée d'une variable déclarée **de façon explicite** (précédée du mot-clé *var*), dépend de l'endroit où elle est déclarée.

- Une variable déclarée au début du script, avant toute fonction, sera globale. Elle peut être utilisée n'importe où dans le script .
- Une variable déclarée explicitement dans une fonction aura une portée limitée à cette seule fonction,

c'est-à-dire qu'elle est inutilisable ailleurs. On parle alors de « **variable locale** ».
Voici deux exemples permettant de l'illustrer :

```
<script type="text/javascript">
<!--
var a = 12;
var b = 4;

function MultipliePar2(b) {
  var a = b * 2;
  return a;
}
document.write("Le double de ",b," est ",MultipliePar2(b));
document.write("La valeur de a est ",a);
// -->
</script>
```

Dans l'exemple ci-dessus, la variable *a* est déclarée explicitement en début de script,
ainsi que dans la fonction. Voici ce qu'affiche ce script :

```
Le double de 4 est 8
La valeur de a est 12
```

Voici un autre exemple dans lequel *a* est déclarée implicitement dans la fonction :

```
<script type="text/javascript">
<!--
var a = 12;
var b = 4;

function MultipliePar2(b) {
  a = b * 2;
  return a;
}
document.write("Le double de ",b," est ",MultipliePar2(b));
document.write("La valeur de a est ",a);
// -->
</script>
```

Voici ce qu'affiche ce script :

```
Le double de 4 est 8
La valeur de a est 8
```

Ces exemples montrent la nécessité de déclarer systématiquement des nouvelles variables avec le mot clé var

Les types de données dans les variables

En Javascript il n'est pas nécessaire de déclarer le type des variables,
contrairement à des langages évolués tels que le langage C ou le Java pour lesquels il faut préciser s'il s'agit d'entier (*int*), de nombre à virgule flottante (*float*) ou de caractères (*char*).

En fait, le Javascript n'autorise la manipulation que de 4 types de données :

- des **nombres**: entiers ou à virgules
- des **chaînes de caractères** (string): une suite de caractères

- des **booléens**: des variables permettant de vérifier une condition.

Les bolléens peuvent prendre deux états : *

- *true* : si le résultat est vrai ;
- *false* : lors d'un résultat faux.

- des **variables de type *null***: un mot caractèristique pour indiquer que la variable ne contient aucune donnée.

Nombre entier

Un nombre entier est un nombre sans virgule, qui peut être exprimé dans différentes bases :

- Base décimale: L'entier est représenté par une suite de chiffre unitaires (de 0 à 9) ne devant pas commencer par le chiffre 0
- Base hexadécimale: L'entier est représenté par une suite d'unités (de 0 à 9 ou de A à F (ou a à f)) devant commencer par *0x* ou *0X*
- Base octale: L'entier est représenté par une suite d'unités (incluant uniquement des chiffres de 0 à 7) devant commencer par *0*

  Les angles dans Javascript sont toujours exprimés en radians (lorsque l'on utilise par exemple les méthodes trigonomériques de l'objet Math.

Nombre à virgule (*float*)

Un nombre à virgule flottante est un nombre à virgule, il peut toutefois être représenté de différentes façons :

- un entier décimal: 895
- un nombre comportant un point (et non une virgule): 845.32
- une fraction: 27/11
- un nombre exponentiel, c'est-à-dire un nombre (éventuellement à virgule) suivi de la lettre *e* (ou *E*), puis d'un entier correspondant à la puissance de 10 (signé ou non, c'est-à-dire précédé d'un + ou d'un -)

```
var a = 2.75e-2;
var b = 35.8E+10;
var c = .25e-2;
```

Chaîne de caractères (*string*)

Une chaîne de caractère est, comme son nom l'indique, une suite de caractères.
Les chaînes de caractères sont traitées en détail dans l'article consacré à ce sujet. Voici deux façons dont une variable peut être déclaré pour être interprétée comme une chaîne de caractère :

```
var a = "Bonjour";
var b = 'Au revoir !';
```

Booléens (*booleans*)

Un booléen est une variable spéciale servant à évaluer une condition, il peut donc posséder deux valeurs :

- *Vrai* (en anglais *True*) : représenté par la valeur 1
- *Faux* (en anglais *False*) : représenté par la valeur 0

Conversions de types

Même si Javascript gère de façon transparente les changements de type des variables, il est parfois nécessaire de forcer la conversion du type. Ainsi Javascript fournit deux fonctions natives permettant de convertir le type des variables passées en paramètre :

- parseInt() permet de convertir une variable en nombre
- parseFloat() permet de convertir une variable en nombre décimal

parseInt()

La fonction *parseInt()* permet de convertir une variable passée en paramètre (soit en tant que chaîne de caractère, soit en tant que nombre dans la base précisée en second paramètre) et le convertit en nombre entier (en base décimale).La syntaxe de la fonction *parseInt()* est la suivante :

```
parseInt(chaine[, base]);
```

Pour que la fonction *parseInt()* retourne un entier, la chaine passée en paramètre doit commencer par des caractère valides c'est-à-dire les chiffres [0-9] ou le préfixe hexadécimal *0x*, et/ou les caractères *+, -, E* et *e*. Dans le cas contraire la fonction *parseInt()* retournera la valeur *NaN* (*Not a Number*).

> *Dans les navigateurs supportant une version de Javascript antérieure à la version 1.1, le chiffre 0 sera renvoyé.*

Si les caractères suivants ne sont pas valides, ils seront ignorés par la fonction *parseInt()*.
Si la chaîne passée en paramètre représente un nombre possédant une partie littérale, celle-ci sera tronquée.

Le paramètre *base* est un entier facultatif permettant de préciser la base devant être utilisée pour interpréter la chaîne. Il vaut 10 par défaut . Si le paramètre *base* n'est pas précisé (ou s'il est fixé à la valeur 10), la base utilisée sera la base décimale; la base sera 16 si la chaîne commence par *0x*, elle sera 8 si la chaîne commence par *0*.

Pour illustrer l'intérêt de la fonction *parseInt()* rien de tel qu'un exemple.
Soient les variables *a* et *b* :

```
var a = "123";
var b = "456";
```

Selon que l'on utilise la fonction *parseInt()* ou non, l'utilisation de l'opérateur *+* avec ces deux variables donnera un résultat différent :

```
document.write(a+b,"<BR>"); // Affiche 123456
document.write(parseInt(a)+parseInt(b),"<BR>"); // Affiche 579
```

Le tableau suivant donne des exemples d'utilisation de la fonction *ParseInt()* :

Exemple	Résultat
parseInt("128.34");	128
parseInt("12.3E-6");	12
parseInt("12E+6");	12
parseInt("Bonjour");	NaN
parseInt("24Bonjour38");	24
parseInt("Bonjour3824");	NaN
parseInt("AF8BEF");	NaN
parseInt("0284");	284
parseInt("0284",8);	2
parseInt("AF8BEF",16);	11504623
parseInt("AB882F",16);	11241519
parseInt("0xAB882F");	11241519
parseInt("0xAB882F",16);	11241519
parseInt("00100110");	100110
parseInt("00100110",2);	38
parseInt("00100110",8);	32840
parseInt("00100110",10);	100110
parseInt("00100110",16);	1048848

parseFloat()

La fonction *parseFloat()* est une fonction du noyau Javascript permettant de convertir une variable passée en paramètre
en nombre en virgule flottante (nombre avec une partie décimale). La syntaxe de la fonction *parseFloat()* est la suivante :

```
parseFloat(chaine);
```

Pour que la fonction *parseFloat()* retourne un flottant, la chaine passée en paramètre
doit commencer par des caractère valides c'est-à-dire les chiffres [0-9] et/ou les caractères *+, -, E* et *e*. Dans le cas contraire la fonction *parseFloat()* retournera la valeur *NaN* (*Not a Number*).

Dans les navigateurs supportant une version de Javascript antérieure à la version 1.1, le chiffre 0 sera renvoyé.

Si les caractères suivants ne sont pas valides, ils seront ignorés par la fonction *parseFloat()*.
Si la chaîne passée en paramètre représente un nombre possédant une partie littérale, celle-ci sera tronquée.

Le tableau suivant donne des exemples d'utilisation de la fonction *parseFloat()* :

Exemple	Résultat
parseFloat("128.34");	<script language="Javascript"> </script>
parseFloat("128,34");	<script language="Javascript"> </script>
parseFloat("12.3E-6");	<script language="Javascript"> </script>
parseFloat("Bonjour");	<script language="Javascript"> </script>
parseFloat("24.568Bonjour38");	<script language="Javascript"> </script>
parseFloat("Bonjour38.24");	<script language="Javascript"> </script>
parseFloat("AF8BEF");	<script language="Javascript"> </script>
parseFloat("0284");	<script language="Javascript"> </script>
parseFloat("0xAB882F");	<script language="Javascript"> </script>

‹ Précédent

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10

Suivant ›

Réalisé sous la direction de Jean-François PILLOU,
fondateur de CommentCaMarche.net.