

Cours de programmation

La déclaration de variables :

Introduction :

Comme la plupart des langages, JavaScript dispose de variables permettant de stocker et de manipuler des données. Ainsi, une variable pourra référencer aussi bien des nombres, des chaînes de caractères, des expressions booléennes que des objets.

Un nom de variable est composé de lettres non accentuées et non espacées qui peuvent être séparées par des chiffres. Comme beaucoup de langages, deux variables de même nom mais de casse différente (majuscule ou minuscule) sont considérées comme différentes.

Pour déclarer une variable, on utilise l'instruction `var`. Le type de variable n'est pas précisée. En fait, il se précise automatiquement avec le contenu de la variable. Il est possible de préciser le contenu de la variable lors de sa création.

En JavaScript, on manipule cinq types de variables. Nous allons donc étudier les différents types d'entités. Pour l'instant, nous allons nous intéresser aux trois plus utiles : les chaînes de caractères, les nombres et les expressions booléennes.

Les chaînes de caractères :

Les chaînes de caractères sont définies en JavaScript comme dans la plupart des langages. Pour définir une chaîne de caractères, on doit placer la chaîne en question entre guillemets ou entre apostrophes pour délimiter la chaîne. En pratique, on préfère utiliser les apostrophes.

• Exemple :

```
var chaine1='Voici mon texte';  
var chaine2="Voici un autre texte";
```

Les esprits malins vont se demander comment insérer des guillemets ou des apostrophes dans une chaîne de caractères. Eh bien, c'est comme en C, il faut utiliser le caractère d'échappement `"\"`. Ce dernier permet aussi d'insérer un retour en arrière (`\b`), d'insérer un saut de page (`\f`), d'insérer un saut de ligne (`\n`), d'insérer un retour chariot (`\r`) ou encore d'insérer un caractère de tabulation (`\t`).

• Exemple :

```
var chaine1='Voici\n mon texte';  
var chaine2="Voici un \"autre\" texte";
```

Enfin, l'opérateur de concaténation des chaînes de caractères est la caractères `+`. Bien sûr, il peut être utiliser avec un nom de variable.

• Exemple :

```
var chaine1='Voici mon texte';  
var chaine2="Voici un autre texte";  
var chaine3=chaine1+chaine2;
```

Les nombres :

Bien sûr, vous pouvez créer des variables contenant une valeur numérique. Le nombre est exprimé par défaut dans la base décimale. Mais il peut très bien être exprimé dans une autre base. Si le nombre est précédé du chiffre "0", il est dans la base octale. Si il est précédé de "0x", il est dans la base hexadécimale.

Pour écrire un nombre à virgule, on utilise le caractère ".".

Exemple :

```
var num1=045;  
var num2=0x45;  
var num3=45;  
var num4=4.5;
```

Les expressions booléennes :

Une expression booléenne correspond en fait à la base binaire. En effet, il n'y a que deux possibilités soit l'expression est vraie, soit elle est fausse. La valeur vraie peut être écrite en toute lettre "true" ou à l'aide du chiffre "1". De même, la valeur fausse peut être écrite en toute lettre "false" ou à l'aide du chiffre "0".

Les opérateurs :

Comme le C et le C++, JavaScript dispose d'une multitude d'opérateurs permettant de travailler avec des expressions numériques, booléennes, alphanumériques et même binaires. Il existe des expressions dites conditionnelles qui correspondent à deux valeurs différentes selon qu'une condition est vérifiée ou non au moment de l'évaluation de l'expression. La syntaxe d'une telle expression est issue du C.

Exemple :

```
var statut=(age >= 18) ? 'adulte':'mineur';
```

Opérateurs arithmétiques :

Ces opérateurs s'appliquent à des opérandes numériques. En plus des opérateurs classiques (addition, soustraction...), JavaScript connaît l'opérateur "%" qui l'opérateur modulo. Il retourne le reste de la division entière du premier opérande par le deuxième.

Il existe aussi l'opérateur d'incrément "++". Placer devant l'opérande, il lui ajoute un et retourne la valeur ainsi obtenue. Placer derrière l'opérande, il retourne la valeur de l'opérande puis ajoute un à cet opérande.

Il existe de même l'opérateur de décrémentation "--". Il s'utilise exactement de la même manière que le précédent.

Opérateurs booléens :

JavaScript connaît les opérateurs booléens classiques.

Il existe le ET logique "&&". Si les deux opérandes sont vraies, il retourne vrai (true). Sinon, il retourne faux (false).

Il existe le OU logique "||". Si les deux opérandes sont fausses, il retourne faux (false). Sinon, il retourne vrai (true).

Enfin, il existe l'opérateur de négation "!". La négation de true est false et vice-versa.

Opérateurs binaires :

Les opérateurs binaires permettent de travailler sur la représentation binaire d'un entier. Les entiers sont représentés sur 32 bits.

Les opérateurs binaires réalisent des opérations bit à bit. Il existe pour cela l'opérateur ET "&", OU "|" et Ou exclusif "^".

Parmi les opérateurs binaires, il existe les opérateurs de décalages "<<" et ">>". Le premier est un l'opérateur de décalage à gauche, le second est l'opérateur de décalage à droite.

L'opérateur ">>>" est un opérateur de décalage à droite qui ne tient pas compte du bit de signe.

Opérateurs d'affectation :

Une expression peut permettre d'affecter une valeur à une variable. Pour cela, elle doit utiliser un opérateur d'affectation.

Exemple :

<code>x += y</code>	correspond à <code>x = x + y</code>
<code>x -= y</code>	correspond à <code>x = x - y</code>
<code>x *= y</code>	correspond à <code>x = x * y</code>
<code>x /= y</code>	correspond à <code>x = x / y</code>
<code>x %= y</code>	correspond à <code>x = x % y</code>
<code>x <<= y</code>	correspond à <code>x = x << y</code>
<code>x >>= y</code>	correspond à <code>x = x >> y</code>
<code>x >>>= y</code>	correspond à <code>x = x >>> y</code>
<code>x &= y</code>	correspond à <code>x = x & y</code>
<code>x ^= y</code>	correspond à <code>x = x ^ y</code>
<code>x = y</code>	correspond à <code>x = x y</code>

Opérateurs de comparaison :

Les opérateurs de comparaison permettent de créer des expressions booléennes. Ils servent à comparer deux valeurs. Si la comparaison est vérifiée, alors l'expression vaut true. Sinon, elle vaut false. Ces opérateurs peuvent aussi bien être utilisés pour comparer des valeurs numériques que pour comparer des chaînes de caractères.

"==" retourne true si les deux opérandes sont identiques.

"!=" retourne true si les deux opérandes sont différents.

"<=" retourne true si le premier opérande est inférieur ou égal au second.

">=" retourne true si le premier opérande est supérieur ou égal au second.

"<" retourne true si le premier opérande est inférieur au second.

">" retourne true si le premier opérande est supérieur au second.



Copyright © 2000–2001 ProgWeb Tous droits réservés
Dernière mise à jour : 22/10/2001

