Javascript: les opérateurs

Septembre 2015

Qu'est-ce qu'un opérateur'

Les opérateurs sont des symboles qui permettent de manipuler des variables, c'est-à-dire effectuer des opérations, les évaluer, ...
On distingue plusieurs types d'opérateurs :

Les opérateurs de calcul

Les opérateurs de calcul permettent de modifier mathématiquement la valeur d'une variable

Opérateur	Dénomination	Effet	Exemple	Résultat (avec x valant 7)
+	opérateur d'addition	Ajoute deux valeurs	x+3	10
-	opérateur de soustraction	Soustrait deux valeurs	x-3	4
*	opérateur de multiplication	Multiplie deux valeurs	x*3	21
/	plus: opérateur de division	Divise deux valeurs	x/3	2.3333333

=	opérateur d'affectation	Affecte une valeur à une variable	x=3	Met la valeur 3 dans la variable x
%	opérateur modulo	Retourne le reste de la division entière de l'opérande de gauche par celle de droite	x % 2	1

Les opérateurs d'affectation

Ces opérateurs permettent de simplifier des opérations telles que *ajouter une valeur dans une variable et stocker le résultat dans la variable*. Une telle opérations s'écrirait habituellement de la façon suivante par exemple: *x=x+2*
Avec les opérateurs d'assignation il est possible d'écrire cette opération sous la forme suivante: *x+=2* Ainsi, si la valeur de x était 7 avant opération, elle sera de 9 après...

Les opérateurs de ce type sont les suivants :

Opérateur	Effet
+=	ajoute l'opérande de gauche par l'opérande de droite et stocke le résultat dans l'opérande de gauche.
-=	soustrait l'opérande de droite à l'opérande de gauche et stocke le résultat dans l'opérande de gauche.
***=**	multiplie l'opérande de gauche par l'opérande de droite et stocke le résultat dans l'opérande de gauche.
/=	divise l'opérande de gauche par l'opérande de droite et stocke le résultat dans l'opérande de gauche.
%=	calcule le reste de la division entière de l'opérande de gauche par l'opérande de droite et stocke le résultat dans l'opérande de gauche.

Les opérateurs d'incrémentation

Ce type d'opérateur permet de facilement augmenter ou diminuer d'une unité une variable. Ces opérateurs sont très utiles pour des structures telles que des boucles, qui ont besoin d'un compteur (variable qui augmente de un en un).

Un opérateur de type *x++* permet de remplacer des notations lourdes telles que *x=x+1* ou bien *x+=1*

Opérateur	Dénomination	Effet	Syntaxe	Résultat (avec x valant 7)
++	Incrémentation	Augmente d'une unité la variable	x++	8
--	Décrémentation	Diminue d'une unité la variable	x--	6

Les opérateurs de comparaison

Opérateur	Dénomination	Effet	Exemple	Résultat (avec x valant 7)
== **A ne pas confondre avec le signe d'affectation (=)!!**	opérateur d'égalité	Compare deux valeurs et vérifie leur égalité	x==3	Retourne *True* si X est égal à 3, sinon *False*
===	opérateur d'identité	Vérifie l'identité de valeur **et de type** de deux valeurs	a===b	Retourne *True* si a est égal à b et est de même type, sinon *False*
!=	opérateur de différence	Vérifie qu'une variable est différente d'une valeur	x!=3	Retourne 1 si X est différent de 3, sinon 0
!==	opérateur de non identité	Vérifie la non identité de valeur **et de type** de deux valeurs, c'est-à-dire si les deux valeurs n'ont pas la même valeur ou bien sont de types différents.	a!==b	Retourne *True* si a est différent de b ou bien est de type différent, sinon *False*
<	opérateur d'infériorité stricte	Vérifie qu'une variable est strictement inférieure à une valeur	x<3	Retourne *True* si X est inférieur à 3, sinon *False*
<=	opérateur d'infériorité	Vérifie qu'une variable est inférieure ou égale à une valeur	x<=3	Retourne *True* si X est inférieur ou égale à 3, sinon *False*
>	opérateur de supériorité stricte	Vérifie qu'une variable est strictement supérieure à une valeur	x>3	Retourne *True* si X est supérieur à 3, sinon *False*

>=	opérateur de supériorité	Vérifie qu'une variable est supérieure ou égale à une valeur	x>=3	Retourne *True* si X est supérieur ou égal à 3, sinon *False*

Les opérateurs logiques (booléens)

Ce type d'opérateur permet de vérifier si plusieurs conditions sont vraies :

Opérateur	Dénomination	Effet	Syntaxe
||	OU logique	Vérifie qu'une des conditions est réalisée	((expression1)|| (expression2))
&&	ET logique	Vérifie que toutes les conditions sont réalisées	((expression1)&& (expression2))
!	NON logique	Inverse l'état d'une variable booléenne (retourne la valeur 1 si la variable vaut 0, 0 si elle vaut 1)	(!condition)

(Les opérateurs bit-à-bit)

Si vous ne comprenez pas ces opérateurs cela n'est pas important, vous n'en aurez probablement pas l'utilité. Pour ceux qui voudraient comprendre, rendez- vous aux chapitres suivants :

- compréhension du binaire
- représentation des données
- Instructions arithmétiques et logiques en assembleur

Ce type d'opérateur traite ses opérandes comme des données binaires, plutçt que des données décimales, hexadécimales ou octales. Ces opérateurs traitent ces données selon leur représentation binaire mais retournent des valeurs numériques standards dans leur format d'origine.

Les opérateurs suivants effectuent des opérations bit-à-bit, c'est-à-dire avec des bits de même poids.

Opérateur	Dénomination	Effet	Syntaxe	Résultat (avec x valant 7)
&	ET bit-à-bit	Retourne 1 si les deux bits de même poids sont à 1	9 & 12 (1001 & 1100)	8 (1000)
|	OU bit-à-bit	Retourne 1 si l'un ou l'autre des deux bits de même poids est à 1 (ou les deux)	9 | 12 (1001 | 1100)	13 (1101)

^	OU bit-à-bit exclusif	Retourne 1 si l'un des deux bits de même poids est à 1 (mais pas les deux)	9 ^ 12 (1001 ^ 1100)	5 (0101)

(Les opérateurs de rotation de bit)

Si vous ne comprenez pas ces opérateurs cela n'est pas important, vous n'en aurez probablement pas l'utilité. Pour ceux qui voudraient comprendre, rendez- vous aux chapitres suivants :

- compréhension du binaire
- représentation des données
- Instructions arithmétiques et logiques en assembleur

Ce type d'opérateur traite ses opérandes comme des données binaires d'une longueur de 32 bits, plutçt que des données décimales, hexadécimales ou octales. Ces opérateurs traitent ces données selon leur représentation binaire mais retournent des valeurs numériques standards dans leur format d'origine.

Les opérateurs suivants effectuent des rotation sur les bits, c'est-à-dire qu'il décale chacun des bits d'un nombre de bits vers la gauche ou vers la droite. La première opérande désigne la donnée sur laquelle on va faire le décalage, la seconde désigne le nombre de bits duquel elle va être décalée.

Opérateur	**Dénomination**	**Effet**	**Syntaxe**	**Résultat (avec x valant 7)**
<<	Rotation à gauche	Décale les bits vers la gauche (multiplie par 2 à chaque décalage). Les zéros qui sortent à gauche sont perdus, tandis que des zéros sont insérés à droite	6 << 1 (0110 << 1)	12 (1100)
>>	Rotation à droite avec conservation du signe	Décale les bits vers la droite (divise par 2 à chaque décalage). Les zéros qui sortent à droite sont perdus, tandis que le bit non-nul de poids plus fort est recopié à gauche	6 >> 1 (0110 >> 1)	3 (0011)
>>>	Rotation à droite avec remplissage de zéros	Décale les bits vers la droite (divise par 2 à chaque décalage). Les zéros qui sortent à droite sont perdus, tandis que des zéros sont insérés à gauche	6 >>> 1 (0110 >>> 1)	3 (0011)

Opérateurs de manipulation de chaînes de caractères

L'opérateur '*+*' lorsqu'il est utilisé avec des chaînes de caractères permet de les concaténer, c'est-à-dire de joindre bout-à-bout les deux chaînes de caractères :

Ainsi *var='a'+'b'* est équivalent à *var='ab'*.

```
var1='a'
var=var1+'b' -> var='ab'
```

Les priorités

Lorsque l'on associe plusieurs opérateurs, il faut que le navigateur sache dans quel ordre les traiter, voici donc dans **l'ordre décroissant** les priorités de tous les opérateurs :

Priorité des opérateurs												
++++++++++++	()	[]										
++++++++++	--	++	!	~	-							
++++++++++	*	/	%									
+++++++++	+	-										
++++++++	<	<=	>=	>								
+++++++	==	!=										
++++++	^											
+++++	|											
++++	&&	||										
+++	'	:										
++	=	+=	-=	*=	/=	%=	<<=	>>=	>>>=	&=	^=	|=
+	,											

‹ Précédent

- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11

Suivant ›

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