Operadores en JavaScript

Trátase de símbolos e identificadores que representan tanto a forma en que os datos se modifican como a maneira en que unha combinación de expresións é evaluada.

JavaScript soporta operadores binarios e unarios:

- Os operadores binarios requiren que existan dous operadores na expresión.
- Os operadores unarios só precisan un operando.

Sumario

Operadores de asignación

O operador de asignación é un dos mais empregados. A súa función básica é asignar un valor a unha variable, deste xeito gárdase o valor en memoria.

Operador de asignación e aritméticos	
Versión longa	**Versión curta**
x = x + y	x += y
x = x - y	x -= y
x = x * y	x *= y
x = x / y	x /= y
x = x % y	x %= y

Operador de asignación e de bit	
Versión longa	**Versión curta**
x = x << y	x <<= y
x = x >> y	x >>= y
x = x >>> y	x >>>= y
x = x & y	x &= y
x = x ^ y	x ^= y
x = x ! y	x != y

Operadores aritméticos

- Utilízanse para traballar con números. Os operadores mais básicos deste grupo inclúen o signos: (**+**), (**-**), (*****) e (/).
- Outro operador interesante é o **operador módulo** que se representa polo símbolo de porcentaxe (**%**). Con este operador calculamos o resto enteiro da división do primeiro operando có segundo.

Vexamos un exemplo no que se realizan operacións aritméticas básicas:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Operadores</title>
</head>
<body>
<h1 align="center">Operadores en JavaScript</h1>

<script language="javascript" >

</script>

</body>
</html>
```

- Outra operación moi típica é incrementar, ou decrementar, o valor dunha variable e despois reasignar ese valor á propia variable. Vexamos unha táboa onde se mostran estes operadores:

Operador de incremento e decremento	
x++	Asíngnase e logo se incrementa
x--	Asígnase e logo se decrementa
++x	Increméntase e logo se asigna
--x	Decreméntase e logo se asigna

- Finalmente, contamos có operador unario "negación" (-) que se emprega para cambiar un valor de positivo a negativo e viceversa. É unario porque opera cun só operando.

Operadores de comparación

Os operadores de comparación empréganse para comparar expresións. As expresións que empregan operadores de comparación normalmente realizan preguntas sobre os dous valores contidos nos operandos. A resposta a dita pregunta pode ser *true* ou *false*.

A lista completa de operadores de comparación móstrase na seguinte táboa:

Operadores de comparación	
Versión longa	**Versión curta**
==	Operador igual-que. Devolve *true* se ambos operadores son iguais.
!=	Operador non-igual. Devolve *true* se ambos operadores son distintos.
>	Operador maior-que. Devolve *true* se o operador da esquerda é maior que o da dereita.
>=	Operador maior-ou-igual-que. Devolve *true* se o operador da esquerda é maior ou igual que o da dereita.
<	Operador menor-que. Devolve *true* se o operador da esquerda é menor que o da dereita.

Operadores de comparación	
<=	Operador menor-ou-igual-que. Devolve *true* se o operador da esquerda é menor ou igual que o da dereita.

Operadores de cadea

O conxunto de operadores de cadea dispoñibles para JavaScript inclúe todos os operadores de comparación e o **operador de concatenación (+)**. Empregando o operador de concatenación pódese facilmente unir cadeas para formar cadeas de maior tamaño.

No seguinte código vese un exemplo de manexo de cadeas de texto:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Operadores</title>
</head>
<body>
<h2 align="center">Operadores de cadea en JavaScript</h2>

<script language="javascript" >

</script>

</body>
</html>
```

Operadores condicionais

JavaScript utiliza dous operadores: *(?)* e *(:)*, para formar expresións condicionais. Estes operadores realizan la misma operación que as sentencias ***if***. A sintaxe destes operadores é:

```
<expresión>?<sentencia>:<sentencia>;
```

Se a "expresión" é *true* devólvese a sentencia que se atopa antes dos dous puntos **(:)** e se o valor é *false* o operador condicional devolve a sentencia que se atopa despois dos dous puntos.

Deste xeito a expresión **(x>y)?"o maior é x":"o maior é y"** devolvería a cadea "o maior é x" no caso en que x fora maior que y ou a cadea "o maior é y" no caso contrario.

Operadores lóxicos

Os operadores lóxicos (tamén chamados operadores booleanos) utilízanse conxuntamente con expresións que devolven valores lóxicos. A sintaxe destes operadores é a seguinte:

Operadores lóxicos	
Operador	**Significado**
exp1 && exp2	O operador **"y"** devolve *true* se ambas expresións son *true*
exp1 ¡¡ exp2	O operador ***ou*** devolve *true* se algunha das expresións é *true*
!exp1	O operador ***not*** devolve *true* se a expresión é *false* e *false* se a expresión é *true*

Operadores de bits

JavaScript emprega 32 bits para almacenar en memoria os valores enteiros.

JavaScript permítenos acceder á representación binaria dos enteiros ao través dos operadores de bit. A seguinte táboa nos mostra os operadores:

Operadores de bits

Operador	Significado
~op	O operador "complemento a un" devolve o complemento a un do operando
op1 && op2	O operador de bit's "e" realiza a operación AND
op1 ¡¡ op2	O operador de bit's "ou" realiza a operación OR
op1 ^ op2	O operador de bit's "ou-exclusivo" realiza a operación OR-Exclusiva
op1 << op2	O operador de "desprazamento á esquerda" enche o operando1 con tantos 0's pola dereita como indique o operando2.
op1 >> op2	O operador de "desprazamento á dereita con propagación do signo" enche o operando1 con tantos 1's ou 0's pola esquerda como indique o operando2. O valor de recheo o indica o bit mais á esquerda do operando1.
op1 >>> op2	O operador de "desprazamento á dereita con recheo de 0's". Enche o operando1 con tantos 0's pola esquerda como indique o operando2.

O operador *typeof*

Prececencia de operadores