Qué son los operadores

Los operadores son símbolos que se utilizan para trabajar con las variables, tal como se trabaja en la aritmética elemental, los signos más y menos son operadores. A continuación, se listan los operadores usados en JavaScript con un ejemplo correspondiente.

Operadores aritméticos

Ahora que se declara una variable y se asignar un valor, podemos comenzar la sección sobre los operadores aritméticos. Se verá más adelante que hay varios tipos de operadores, pero por ahora nos centraremos exclusivamente en los operadores aritméticos. Estos son la base para todos los cálculos y son cinco.

Operador	Símbolo
suma	+
sustracción	-
multiplicación	*
división	/
módulo	%

El último operador, módulo, es simplemente el resto de una división. Por ejemplo, si se divide 5 entre 2 se tiene resto 1, que es el módulo.

Algunos cálculos sencillos

Programar el cálculo es casi tan fácil como en una calculadora, por ejemplo:

Código: JavaScript

```
var resultado = 3 + 2;
alert (resultado) / / Muestra « 5 »
```

Así que puedes hacer cálculos con dos números, es bueno, pero con dos variables que contienen números en sí es más útil:

Código: JavaScript

```
var número1=3, número2 = 2,
resultado; resultado = numero1 *
numero2;
alert (resultado) / / Muestra: « 6 »
```

Podemos ir aún más lejos al escribirlo como cálculos con operadores múltiples y variables:

Código: JavaScript

```
var divisor = 3, resultado1, resultado2,
resultado3; resultado1 = (16 + 8) / 2 - 2; / /
10
resultado2 = resultado1 /
divisor; resultado3 =
resultado1 % divisor;
alerta (resultado2) / / Resultado de la división: 3,33
alerta (resultado3) / / Resto de la división: 1
```

Notarás que usamos paréntesis para el cálculo de la variable resultado1. Se utilizan como en matemáticas: el navegador que primero calcula 16 + 8 y divide el resultado por 2.

Introducción a la concatenación y conversión de tipos

Algunos operadores se han omitido previamente. Toma el operador +, además de las sumas, se puede hacer lo que se conoce como concatenación entre cadenas.

Concatenación

La concatenación es añadir una cadena al final de otra, como en este

ejemplo: Código: JavaScript

```
var hola= 'Hola', nombre = 'tu',
resultado; resultado = hola +
nombre;
alert (resultado) / / muestra: « Holatu »
```

En este ejemplo se muestra la frase "Holatu". Te darás cuenta de que no hay espacio entre las dos palabras, de hecho, es la concatenación de lo que se definió en las variables. Si quieres un espacio, debes agregar un espacio en una variable, como var hola = 'Hola ';

Si se recuerda el ejemplo previo de adición, que se expresaba +=, se puede actuar de forma análoga con cadenas de caracteres:

Código: JavaScript

```
var text = Hola";
text + = ' tu';
alert (text) / / Muestra « Hola tu ».
```

Condicionales

En el capítulo anterior se aprendió cómo crear y modificar variables. Todavía se siente un tanto limitados nuestros códigos. En este capítulo, descubrirás las condiciones de todo tipo y especialmente darse cuenta de que las posibilidades del código serán mucho más abiertas porque las condicionales afectarán directamente cómo el código va a reaccionar a ciertos criterios.

En la mayoría de las condicionales, también se podrá profundizar mediante un tipo famoso de variable: boolean.

La base de cualquier condición: booleana

En este apartado, vamos a discutir las condicionales, pero para eso primero tenemos que volver a un tipo de variable que hemos mencionado en el capítulo anterior: booleanas.

¿Para qué van a servir? Para obtener un resultado como verdadero (*true*) o falso (*false*) cuando se verifica una condición. Para aquellos que se preguntan el significado , una condición es una especie de "*test*" para comprobar que una variable contiene un cierto valor. Por supuesto, las comparaciones no se limitan sólo a las variables, pero por el momento lo vamos a considerar más que suficiente para comenzar.

En primer lugar, ¿cuáles son las condiciones establecidas? Valores a ensayar dos tipos de operadores: uno lógico y el otro de comparación.

Los operadores de comparación

Como su nombre indica, estos operadores pueden realizar comparaciones entre diferentes valores entre ellos. En total, hay muchos, ocho, aquí están:

Operador	Significado
=	Igual a
!=	Diferente a
===	Contenido y tipo igual a
!==	Contenido o tipo diferente de
>	Mayor que
>=	Mayor o igual que
<	Menor que
<=	Menor o igual que

No vamos a hacer un ejemplo para cada uno de ellos, pero por lo menos te mostraremos cómo utilizarlos para que puedas probar el otro:

Código: JavaScript

```
var numero1 = 2, numero2 = 2, numero3 = 4, resultado;

resultado = numero1 == numero2 / / En lugar de un único valor,
se han escrito dos con el operador de comparación entre ellos
alert (resultado) / / Muestra "true", si la condición se verifica
porque las dos variables contienen el mismo valor

resultado = numero1 == numero3;
alert (resultado) / / Muestra "false", la condición no es verificada
porque 2 es distinto de 4

resultado = numero1 < numero3;
alert (resultado) / / Muestra "true", la condición es verdadera
porque 2 es inferior a 4
```

Como se puede ver, el concepto no es muy complicado, simplemente escribe dos valores con el operador de comparación deseado entre los dos y devuelve un booleano. Si esto es true (verdadero), entonces la condición está verificada, si es false (falsa), entonces no.

De estos ocho operadores, dos de ellos pueden ser difíciles de entender para un principiante: se trata === y ! ==. de modo que se indica cómo trabajan con algunos ejemplos:

Código: JavaScript

```
var numero = 4, text = '4 ', resultado;
resultado = numero == texto;
alert (resultado) / / Muestra "verdadero", mientras que "número" es
un número y el "texto" de una cadena de caracteres
resultado = numero === texto;
alert (resultado) / / Muestra "false" porque este operador también
compara tipos de variables en adición a sus valores
```

Los condiciones "normales" hacen conversiones de tipos para verificar las igualdades de modo que si se quiere diferenciar el número 4 en una cadena de caracteres que contiene el número 4 entonces tienes que utilizar la igualdad triple ===.

Para todos los operadores de comparación, tienes todas las herramientas que se necesitan para hacer experimentos.

Operadores lógicos

¿Por qué estos operadores se denominan como "lógicos"? Puesto que funcionan con el mismo principio como una tabla electrónica de verdad. Antes de describir su funcionamiento, lo primero que debes hacer es una lista, son tres:

Operador	Tipo de lógica	Utilización
&&	Y	valor1 && valor2
||	O	valor1 || valor2
!	NO	!valor

- Operador Y (AND)

Este operador satisface la condición true cuando todos los valores que se pasan al mismo son verdaderos. Si uno de ellos es false, entonces la condición no será verificada. Por ejemplo:

Código: JavaScript

```
var resultado = true && true;
alert (resultado) / / Muestra "verdadero"

resultado = true && false;
alert (resultado) / / Muestra: "false"

resultado = false && false;
alert (resultado) / / Muestra: "false"
```

- Operador O (OR)

Este operador es más "flexible" porque devuelve true (verdadero) si uno de los valores que contiene es verdadero, no importan otros valores. Por ejemplo:

Código: JavaScript

```
var resultado = true | | verdadero;
alert (resultado) / / Muestra "verdadero"

resultado = true | | false;
alert (resultado) / / Muestra "verdadero"

resultado = false | | false;
alert (resultado) / / Muestra: "false"
```

- El operador NO (NOT)

Este operador se diferencia de los otros dos porque necesita sólo un valor. Se le llama "NO", porque su función es la de invertir el valor que se le pasa y se convierte verdadero en falso y viceversa. Por ejemplo:

Código: JavaScript

```
var resultado = false;
resultado =! resultado / / se almacena en "resultado" el
inverso    de "resultado" ello, es posible
alert (resultado) / / Muestra "true" porque queríamos el opuesto de
"false"

resultado = !resultado;
alert (resultado) / / Muestra "false", ya que se invirtió de nuevo,
como resultado, "se cambia de "true" a "false"
```

- Combinación de operadores

Estamos casi al final de la sección sobre valores booleanos, no te preocupes, será más fácil el resto del capítulo. Sin embargo, antes de continuar, debes asegurarte de que entiendes que todos los operadores que acabas de descubrir puedes combinarlos.

En primer lugar un breve resumen (leer atentamente): los operadores de comparación toma dos valores de entrada y devuelven un booleano, mientras que los operadores lógicos aceptar la entrada del múltiples booleanos y devuelven un booleano. Habrás entendido que entonces puedes acoplar los valores de salida de los operadores de comparación con los valores de entrada de los operadores lógicos. Por ejemplo:

Código: JavaScript

```
var condicion1, resultado, condicion2;
condicion1 = 2> 8 / /
false condicion2 = 8> 2
/ / true
resultado = condicion1 &&
condicion2; alert (resultado) / /
Muestra "false"
```

Por supuesto, es posible acortar el código si se combinan en una sola línea, todas las condiciones:

Código: JavaScript

```
var resultado = 2> 8 && 8> 2;
alert (resultado) / / Muestra
"false"
```