Curso de Javascript

Unidad Didáctica 04: Operadores

http://cursosdedesarrollo.com/

Ayuntamiento de Vitoria-Gasteiz
Vitoria-Gasteizko Udala

Índice de contenidos

- Introducción
- Asignación
- Aritméticos
    - Lógicos
    - Relacionales
    - typeof
    - instanceof
    - Conclusiones

http://cursosdedesarrollo.com/

Introducción

Los operadores permiten manipular el valor de las variables, realizar operaciones matemáticas con sus valores y comparar diferentes variables

http://cursosdedesarrollo.com/

Introducción

De esta forma, los operadores permiten a los programas realizar cálculos complejos y tomar decisiones lógicas en función de comparaciones y otros tipos de condiciones.

http://cursosdedesarrollo.com/

Asignación

El operador de asignación es el más utilizado y el más sencillo

Este operador se utiliza para guardar un valor específico en una variable

Asignación

El símbolo utilizado es = (no confundir con el operador == que se verá más adelante):

```
var numero1 = 3;
```

http://cursosdedesarrollo.com/

Asignación

A la izquierda del operador, siempre debe indicarse el nombre de una variable

A la derecha del operador, se pueden indicar variables, valores, condiciones lógicas, etc…

Asignación

```
var numero1 = 3;

var numero2 = 4;

/* Error, la asignación siempre se realiza a una variable,
por lo que en la izquierda no se puede indicar un número */

5 = numero1;

// Ahora, la variable numero1 vale 5

numero1 = 5;

// Ahora, la variable numero1 vale 4

numero1 = numero2;
```

http://cursosdedesarrollo.com/

Aritméticos

JavaScript permite realizar manipulaciones matemáticas sobre el valor de las variables numéricas

Los operadores definidos son: suma (+), resta (-), multiplicación (*), división (/) y módulo (%)

http://cursosdedesarrollo.com/

Aritméticos

var numero1 = 10;

var numero2 = 5;

resultado = numero1 / numero2; // resultado = 2

resultado = 3 + numero1; // resultado = 13

resultado = numero2 – 4; // resultado = 1

resultado = numero1 * numero 2; // resultado = 50

resultado = numero1 % numero2; // resultado = 0

numero1 = 9;

numero2 = 5;

resultado = numero1 % numero2; // resultado = 4

http://cursosdedesarrollo.com/

Aritméticos

Los operadores matemáticos también se pueden combinar con el operador de asignación para abreviar su notación:

```
var numero1 = 5;

numero1 += 3;   // numero1 = numero1 + 3 = 8

numero1 -= 1;   // numero1 = numero1 - 1 = 4

numero1 *= 2;   // numero1 = numero1 * 2 = 10

numero1 /= 5;   // numero1 = numero1 / 5 = 1

numero1 %= 4;   // numero1 = numero1 % 4 = 1
```

http://cursosdedesarrollo.com/

Aritméticos

Existen dos operadores especiales que solamente son válidos para las variables numéricas y se utilizan para incrementar o decrementar en una unidad el valor de una variable.

Ejemplo:

```
var numero = 5;

++numero;

console.log(numero);  // numero = 6
```

http://cursosdedesarrollo.com/

Aritméticos

El operador de incremento se indica mediante el prefijo ++ en el nombre de la variable

El resultado es que el valor de esa variable se incrementa en una unidad. Por tanto, el anterior ejemplo es equivalente a:

```
var numero = 5;

numero = numero + 1;

console.log(numero);  // numero = 6
```

http://cursosdedesarrollo.com/

Aritméticos

De forma equivalente, el operador decremento (indicado como un prefijo -- en el nombre de la variable) se utiliza para decrementar el valor de la variable:

```
var numero = 5;

--numero;

console.log(numero);  // numero =
4console.log(numero);  // numero = 6
```

http://cursosdedesarrollo.com/

Aritméticos

El anterior ejemplo es equivalente a:

```
var numero = 5;

numero = numero - 1;

console.log(numero);  // numero = 4
```

http://cursosdedesarrollo.com/

Aritméticos

Los operadores de incremento y decremento no solamente se pueden indicar como prefijo del nombre de la variable, sino que también es posible utilizarlos como sufijo

En este caso, su comportamiento es similar pero muy diferente. En el siguiente ejemplo:

```
var numero = 5;

numero++;

console.log(numero);  // numero = 6
```


Aritméticos

El resultado de ejecutar el script anterior es el mismo que cuando se utiliza el operador ++numero, por lo que puede parecer que es equivalente indicar el operador ++ delante o detrás del identificador de la variable

Sin embargo, el siguiente ejemplo muestra sus diferencias

Aritméticos

```
var numero1 = 5;

var numero2 = 2;

numero3 = numero1++ + numero2;

// numero3 = 7, numero1 = 6

var numero1 = 5;

var numero2 = 2;

numero3 = ++numero1 + numero2;

// numero3 = 8, numero1 = 6
```

http://cursosdedesarrollo.com/

Aritméticos

Si el operador ++ se indica como prefijo del identificador de la variable, su valor se incrementa antes de realizar cualquier otra operación

Si el operador ++ se indica como sufijo del identificador de la variable, su valor se incrementa después de ejecutar la sentencia en la que aparece

Aritméticos

Por tanto, en la instrucción

numero3 = numero1++ + numero2;

el valor de numero1 se incrementa después de realizar la operación (primero se suma y numero3 vale 7, después se incrementa el valor de numero1 y vale 6)

Aritméticos

Sin embargo, en la instrucción

numero3 = ++numero1 + numero2;

, en primer lugar se incrementa el valor de numero1 y después se realiza la suma (primero se incrementa numero1 y vale 6, después se realiza la suma y numero3 vale 8)

http://cursosdedesarrollo.com/

Lógicos

Los operadores lógicos son imprescindibles para realizar aplicaciones complejas, ya que se utilizan para tomar decisiones sobre las instrucciones que debería ejecutar el programa en función de ciertas condiciones

http://cursosdedesarrollo.com/

Lógicos

El resultado de cualquier operación que utilice operadores lógicos siempre es un valor lógico o booleano

http://cursosdedesarrollo.com/

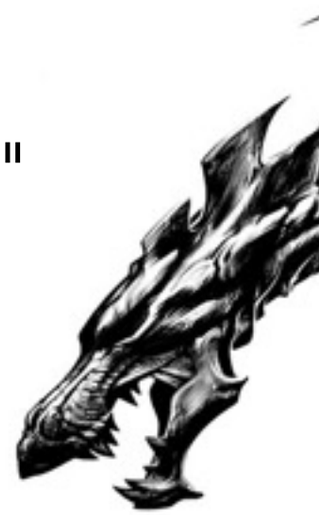
Lógicos

NEGACIÓN

Uno de los operadores lógicos más utilizados es el de la negación. Se utiliza para obtener el valor contrario al valor de la variable:

```
var visible = true;

console.log(!visible);  // Muestra "false" y no "true"
```



Lógicos

La negación lógica se obtiene prefijando el símbolo ! al identificador de la variable

El funcionamiento de este operador se resume en la siguiente tabla

Lógicos

variable	!variable
`true`	`false`
`false`	`true`

http://cursosdedesarrollo.com/

Lógicos

Si la variable original es de tipo booleano, es muy sencillo obtener su negación. Sin embargo, ¿qué sucede cuando la variable es un número o una cadena de texto?

Para obtener la negación en este tipo de variables, se realiza en primer lugar su conversión a un valor booleano:

- Si la variable contiene un número, se transforma en false si vale 0 y en true para cualquier otro número (positivo o negativo, decimal o entero).

- Si la variable contiene una cadena de texto, se transforma en false si la cadena es vacía ("") y en true en cualquier otro caso.

Lógicos

var cantidad = 0;

vacio = !cantidad; // vacio = true

cantidad = 2;

vacio = !cantidad; // vacio = false

var mensaje = "";

mensajeVacio = !mensaje; // mensajeVacio = true

mensaje = "Bienvenido";

mensajeVacio = !mensaje; // mensajeVacio = false

http://cursosdedesarrollo.com/

Lógicos

AND

La operación lógica AND obtiene su resultado combinando dos valores booleanos

El operador se indica mediante el símbolo && y su resultado solamente es true si los dos operandos son true

http://cursosdedesarrollo.com/

Lógicos

variable1	variable2	variable1 && variable2
`true`	`true`	`true`
`true`	`false`	`false`
`false`	`true`	`false`
`false`	`false`	`false`

http://cursosdedesarrollo.com/

Lógicos

var valor1 = true;

var valor2 = false;

resultado = valor1 && valor2; // resultado = false

valor1 = true;

valor2 = true;

resultado = valor1 && valor2; // resultado = true

Lógicos

OR

La operación lógica OR también combina dos valores booleanos

El operador se indica mediante el símbolo || y su resultado es true si alguno de los dos operandos es true

http://cursosdedesarrollo.com/

Lógicos

variable1	variable2	variable1 || variable2
`true`	`true`	`true`
`true`	`false`	`false`
`false`	`true`	`true`
`false`	`false`	`false`

http://cursosdedesarrollo.com/

Lógicos

var valor1 = true;

var valor2 = false;

resultado = valor1 || valor2; // resultado = true

valor1 = false;

valor2 = false;

resultado = valor1 || valor2; // resultado = false

http://cursosdedesarrollo.com/

Relacionales

Los operadores relacionales definidos por JavaScript son idénticos a los que definen las matemáticas: mayor que (>), menor que (<), mayor o igual (>=), menor o igual (<=), igual que (==) y distinto de (!=)

http://cursosdedesarrollo.com/

Relacionales

Los operadores que relacionan variables son imprescindibles para realizar cualquier aplicación compleja

El resultado de todos estos operadores siempre es un valor booleano

Relacionales

```
var numero1 = 3;

var numero2 = 5;

resultado = numero1 > numero2;  // resultado = false

resultado = numero1 < numero2;  // resultado = true
```

```
numero1 = 5;

numero2 = 5;

resultado = numero1 >= numero2; // resultado = true

resultado = numero1 <= numero2; // resultado = true

resultado = numero1 == numero2; // resultado = true

resultado = numero1 != numero2; // resultado = false
```

http://cursosdedesarrollo.com/

Relacionales

Se debe tener especial cuidado con el operador de igualdad (==), ya que es el origen de la mayoría de errores de programación, incluso para los usuarios que ya tienen cierta experiencia desarrollando scripts

http://cursosdedesarrollo.com/

Relacionales

El operador == se utiliza para comparar el valor de dos variables, por lo que es muy diferente del operador =, que se utiliza para asignar un valor a una variable

http://cursosdedesarrollo.com/

Relacionales

// El operador "=" asigna valores

var numero1 = 5;

resultado = numero1 = 3; // numero1 = 3 y resultado = 3

// El operador "==" compara variables

var numero1 = 5;

resultado = numero1 == 3; // numero1 = 5 y resultado = false

http://cursosdedesarrollo.com/

Relacionales

Los operadores relacionales también se pueden utilizar con variables de tipo cadena de texto:

```
var texto1 = "hola";

var texto2 = "hola";

var texto3 = "adios";

resultado = texto1 == texto3; // resultado = false

resultado = texto1 != texto2; // resultado = false

resultado = texto3 >= texto2; // resultado = false
```

http://cursosdedesarrollo.com/

Relacionales

Cuando se utilizan cadenas de texto, los operadores "mayor que" (>) y "menor que" (<) siguen un razonamiento no intuitivo: se compara letra a letra comenzando desde la izquierda hasta que se encuentre una diferencia entre las dos cadenas de texto

Relacionales

Para determinar si una letra es mayor o menor que otra, las mayúsculas se consideran menores que las minúsculas y las primeras letras del alfabeto son menores que las últimas (a es menor que b, b es menor que c, A es menor que a, etc.)

http://cursosdedesarrollo.com/

Typeof

El operador typeof se emplea para determinar el tipo de dato que almacena una variable

Su uso es muy sencillo, ya que sólo es necesario indicar el nombre de la variable cuyo tipo se quiere averiguar

http://cursosdedesarrollo.com/

Typeof

```
var myFunction = function() {

  console.log('hola');

};

    var myObject = {

        foo : 'bar'

    };

    var myArray = [ 'a', 'b', 'c' ];

    var myString = 'hola';

    var myNumber = 3;
```

```
typeof myFunction;    // devuelve
'function'

typeof myObject;      // devuelve
'object'

typeof myArray;       // devuelve
'object' -- tenga cuidado

typeof myString;      // devuelve 'string'

typeof myNumber;      // devuelve
'number'

typeof null;          // devuelve 'object' --
tenga cuidado
```

http://cursosdedesarrollo.com/

Typeof

```
if (myArray.push && myArray.slice && myArray.join) {

// probablemente sea un vector

// (este estilo es llamado, en inglés, "duck typing")

}

if (Object.prototype.toString.call(myArray) === '[object Array]') {

// definitivamente es un vector;

// esta es considerada la forma más robusta

// de determinar si un valor es un vector.

}
```

http://cursosdedesarrollo.com/

Typeof

Los posibles valores de retorno del operador son: undefined, boolean, number, string para cada uno de los tipos primitivos y object para los valores de referencia y también para los valores de tipo null

http://cursosdedesarrollo.com/

Typeof

El operador typeof no distingue entre las variables declaradas pero no inicializadas y las variables que ni siquiera han sido declaradas:

var variable1;

// devuelve "undefined", aunque la variable1 ha sido declarada

typeof variable1;

// devuelve "undefined", la variable2 no ha sido declarada

typeof variable2;

http://cursosdedesarrollo.com/

Instanceof

El operador typeof no es suficiente para trabajar con tipos de referencia, ya que devuelve el valor object para cualquier objeto independientemente de su tipo

Por este motivo, JavaScript define el operador instanceof para determinar la clase concreta de un objeto

Instanceof

```
var variable1 = new String("hola mundo");

typeof variable1;              // devuelve "object"

variable1 instanceof String;  // devuelve true
```

http://cursosdedesarrollo.com/

Instanceof

El operador instanceof sólo devuelve como valor true o false

De esta forma, instanceof no devuelve directamente la clase de la que ha instanciado la variable, sino que se debe comprobar cada posible tipo de clase individualmente

Conclusiones

Hemos visto los distintos tipos de operadores

http://cursosdedesarrollo.com/

Datos de Contacto

http://www.cursosdedesarrollo.com
info@cursosdedesarrollo.com

http://cursosdedesarrollo.com/

Licencia

David Vaquero Santiago