EXPRESIONES Y OPERADORES

Una expresión es un conjunto de operandos y operadores cuyo resultado puede ser evaluado. Las expresiones constituyen la base de cualquier aplicación y se construyen agrupando variables, expresiones o datos que son afectados por cualquiera de los operadores válidos en un lenguaje. Las expresiones en JavaScript, como en la mayoría de lenguajes se finalizan con punto y coma (;).

JavaScript cuenta con un amplio conjunto de operadores: aritméticos, de asignación, de comparación, lógicos, literales, bitwise y operadores especiales. Dependiendo del número de operandos requeridos, las operaciones (y los operadores) se pueden clasificar en binarias y unarias. Las primeras requieren de la intervención de dos operandos; la suma, la resta, la multiplicación y la división son ejemplos de operaciones binarias. La expresión !X es un ejemplo de operación lógica unaria que necesita de un solo operando para ser ejecutada.

Operadores aritméticos

Además de los operadores aritméticos básicos: suma (+), resta (-), división (/) y multiplicación (*), existe el operador módulo (%) que devuelve el residuo de una división no entera. Por ejemplo, el resultado de evaluar 5%3; será 2. Que corresponde al residuo; 5 divido 3 es igual a 1 con un residuo de 2.

Es importante recordar que el operador + utilizado con variables literales actúa como operador de concatenación. Si por ejemplo, dentro de un script existen las siguientes expresiones:

```
var nombre = "Andrés";
var apellido1 = " Gaitán";
var apellido2 = " Guzmán";
todo = nombre + apellido1;
todo += apellido2;
```

En la cuarta línea del anterior ejemplo la variable **todo** tendrá como valor "Andrés Gaitán", y en la quinta "Andrés Gaitán Guzmán".

Operadores de asignación

La operación de asignación consiste en dar a la variable situada a la izquierda del operador el valor de la variable o de la expresión ubicada a la derecha del mismo. El operador de asignación por excelencia es el signo igual "=":

```
X = 5;
Y = X + 3;
```

En la primera expresión X toma el valor de 5. En el segundo ejemplo a Y se le asigna el resultado de evaluar la expresión X+3, en este caso 8.

En ocasiones es necesario afectar a una variable incrementado su valor en una cantidad determinada, o dividiendo el mismo por una cantidad dada. Considere el caso de las siguientes expresiones:

```
X = X + 3;
Y = Y - A;
Z = Z * 5;
W = W / 4;
T = T % 8;
```

Existen en JavaScript al igual que en C y otros lenguajes, operadores de asignación especiales que permiten escribir este tipo de expresiones de la siguiente manera:

```
X += 3;   // X es igual a X + 3
Y -= A;   // Y toma el valor de Y - A
Z *= 5;   // Similar a  Z = Z * 5
W /= 4;   //W es igual al valor de W dividido en 4
T %= 8;   // T toma el valor de T % 8
todo  +=  apellido2  //  la  cadena  todo  es  igual  a  todo
concatenada con apellido2
```

Operadores de incremento decremento

Es muy común cuando se hacen programas, encontrarse con la necesidad de escribir líneas de código como:

```
a = a + 1;
b = b - 1;
```

En la primera línea el valor de a es incrementado en 1, mientras que en la segunda línea la variable b es decrementada en 1. Existen dos operadores unarios, conocidos como operador incremento (++) y decremento (--), que permiten realizar las mismas operaciones:

```
var a = 6;
var b = 0;
a++;
b--;
```

Los valores de a y b finalmente serán 7 y –1 respectivamente.

En el ejemplo anterior los operadores (++ y --), se colocaron a la derecha del operando, lo que se conoce como operador posfijo. Podrían haberse colocado a la izquierda de las variables, lo que se conoce como operador prefijo (++a, --b) y el resultado obtenido para ese ejemplo sería el mismo.

Dado que los operadores de decremento e incremento tienen una prelación más alta que los operadores aritméticos y de asignación, existen diferencias cuando los primeros intervienen en expresiones como las del siguiente ejemplo:

```
var a = 6;
var b = 6;
var c = 0;
var d = 0;
c = ++a;
d = b++;
```

Si se visualizan los valores finales de cada variable, estos serán:

```
a = 7, b = 7, c = 7 d = 6
```

En la quinta línea de código, primero se incrementa en 1 el valor de a quedando este en 7 y luego se asigna este valor a la variable c que queda también en 7. En la sexta línea, el script asigna a d el valor de b (6) antes de incrementar a b en 1.

Operadores de comparación

Los operadores de comparación, llamados por algunos autores operadores de relación, son del tipo binario y se utilizan para comparar valores o expresiones dadas, el resultado de la comparación puede ser verdadero o falso. En JavaScript existen seis operadores de relación:

Operador	Significado
==	Igual a
!=	Diferente de
<	Menor que
>	Mayor que
<=	Menor o igual que
>=	Mayor o igual que

La evaluación de expresiones que contengan operadores de relación devuelven siempre un valor booleano que puede ser true, si el resultado de la evaluación es cierto (ejemplo 3>2), ó false, si el resultado de la evaluación es falso (ejemplo 3<=2). A diferencia de C y otros lenguajes de programación que requieren de funciones especiales, los operadores de relación de JavaScript permiten comparar cadenas de caracteres con base en el orden lexicográfico estándar. Por ejemplo:

```
var cadena1 = "c";
var cadena2 = "f";
x = (cadena1 < cadena2);
```

En este caso X valdrá true, dado que "c" ocupa un lugar inferior en el orden alfabético respecto de "f".

Aunque pueden utilizarse en expresiones como las del ejemplo anterior, los operadores de relación se usan más a menudo como argumentos dentro de sentencias de selección if, o expresiones que controlan las estructuras for, las cuales se verán más adelante. Es posible también, mezclar en una expresión operadores aritméticos y de relación. Por ejemplo,

```
Resultado = 3 + 5 < 8/11; //Resultado toma false como valor
```

Operadores lógicos

Los operadores lógicos se utilizan para agrupar o combinar los resultados de la evaluación de operadores de relación. Al igual que estos últimos, los operadores lógicos son del tipo binario (excepto el operador !, que es unario) y devuelven true ó false, dependiendo del valor de cada uno de los operandos, veamos:

El operador && (AND), devuelve true (verdadero) si la evaluación de ambos operandos es verdadera ó false (falso) cuando la evaluación de ambos o alguno de los operandos es false. En la siguiente línea de código, la variable Resultado recibirá como valor true:

```
Resultado = (3+5<9 && 7==14/2);
```

El operador || (OR), devuelve true cuando la evaluación de ambos o uno de los operandos es true, y, false cuando la evaluación de los dos operandos es false. En la siguiente línea, la variable Resultado valdrá false:

```
Resultado = (3+5>9 || 7!=14/2);
```

El operador ! (NOT), devuelve true, si el valor del operando es false y false si el valor del operando es true. La variable Resultado en el siguiente ejemplo vale true:

```
Resultado = !(3+5>9);
```

Operadores especiales

Dentro de los operadores especiales de JavaScript vale la pena destacar el operador condicional (?), que corresponde a un if - else simplificado, que también existe en C, JAVA y otros lenguajes. Su función consiste en devolver o ejecutar uno de dos valores o expresiones posibles, dependiendo de si una condición que lo controla es evaluada como true o false:

```
(condición) ? valor1[expresión1] : valor2[expresión2];
```

Si condición (que puede ser cualquier expresión válida), es evaluada como true, entonces el operador devuelve valor1 o ejecuta expresión1, si por el contrario condición es evaluada como false, entonces el operador devolverá valor2. En el siguiente ejemplo, la variable tipo tendrá como valor "festivo":

```
dias=["Lunes", "Martes", "Miércoles", "Jueves", "Viernes","Sábado",
     "Domingo"];

tipo = (dias[6] == "Domingo") ? "festivo" : "laboral";
```

Precedencia de los operadores

Las operaciones en una expresión que contiene más de un operador, se realizan en orden jerárquico dependiendo de la precedencia de los operadores.

Por ejemplo, en una expresión que involucre una suma y una división (3 + 4/5), primero se efectuará la división, por tener esta una precedencia más alta y luego la suma.

La siguiente tabla, muestra el orden de precedencia de los operadores de JavaScript de mayor a menor. No se incluyen algunos operadores que existen en JavaScript por considerar que no son del alcance de este módulo. Cuando varios operadores se encuentran en la misma línea, tendrán una precedencia más alta aquellos ubicados a la izquierda en la celda.

Tipo de operador	Operador
Llamada, miembro	(), [], .
Negación, incremento	!, -, ++, --
Multiplicación, división	*, /, %
Suma, resta	+, -
Comparación	<, <=, >, >=
Igualdad	==, !=
AND lógico	&&
OR Lógico	||
Condicional	? :
Asignación	=, +=, -=, *=, /=, %=