



# ENTREGA-C-BLOQUE-B



PABLO HORCAJADA GONZALEZ

Contenido

BITWISE ..... 2

    ¿Qué son?..... 2

    ¿Cuántos son? ..... 2

    ¿Para qué se usan?..... 2

Anexo ..... 3

    & ..... 3

    | ..... 3

    ~ ..... 3

    ^ ..... 3

    >> ..... 4

    <<..... 4

    >>>..... 4

# BITWISE

## ¿Qué son?

Los operadores a nivel de bit o “bitwise operators” son operadores que actúan sobre números enteros, pero usando su representación binaria. (Foto 1)

## ¿Cuántos son?

Hay un total de 7 operadores.

## ¿Para qué se usan?

Los operadores “bitwise” son usadas para evaluar la presentación binaria de los valores de entrada. Por cada bit de la representación binaria, se realiza una operación binaria.

$\&$  → realiza la operación “AND”, pero por cada bit existente en la representación binaria de los dos números que le introducimos. Es decir, recorre ambos números en su forma binaria elemento a elemento, y hace una operación “and” con cada uno de ellos. Corresponden a las combinaciones  $0-0=0$ ,  $0-1=0$ ,  $1-0=0$ ,  $1-1=1$ . (Foto 2)

$|$  → realiza la operación “OR”, devolviendo 1 donde uno o ambos operadores son 1. corresponden a las combinaciones  $0-0=0$ ,  $0-1=1$ ,  $1-0=1$ ,  $1-1=1$ . (Foto 3)

$\sim$  → realiza a operación “NOT” sobre cada bit del número que el introducimos, es decir, invierte el valor de cada bit, poniendo los 0 a 1 y los 1 a 0. Corresponden a las combinaciones (Foto 4)

$\wedge$  → realiza la función de “XOR” con cada bit de las dos variables que se le proporciona. Lo que hace es devolver “true” o 1 cuando hay al menos un valor en “true”, pero no los dos. Corresponden a las combinaciones  $0-0=0$ ,  $0-1=1$ ,  $1-0=1$ ,  $1-1=0$ . (Foto 5)

$\gg$  → desplaza todos los bits “n” unidades a la derecha. Se le pasa dos parámetros, donde el primero es el numero que se desplaza y el segundo es el número de posiciones. (Foto 6)

$\ll$  → es análogo de “ $\gg$ ”, con la diferencia de que el desplazamiento es a la derecha. (Foto 7)

$\ggg$  → desplaza a la derecha un número especificado de bits, descartando los bits desplazados y sustituyéndolos por 0. (Foto 8)

Anexo

Foto 1

```
/*# Sistema binario
11011
1-> En realidad vale 1*16      = 16
  1-> En realidad vale 1*8      = 8
    0-> En realidad vale 1*4    = 0
      1-> En realidad vale 1*2  = 2
        1-> En realidad vale 1*1= 1
          +-----
Sumando todo                    27*/
```

&

Foto 2

```
Sumando todo
let a = 00000110; //6
let b = 00001010; //10
console.log(a & b);
cript>
```

|

Foto 3

```
Sumando todo
let a = 00000110; //6
let b = 00001010; //10
console.log(a | b);
```

~

Foto 4

```
Sumando todo
let a = 00000110; //6
let b = 00001010; //10
console.log(~a);
console.log(~b);
cript>
```

^

Foto 5

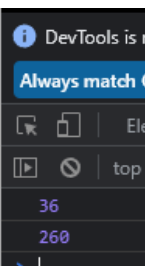
```
Sumando todo
let a = 00000110; //6
let b = 00001010; //10
console.log(a ^ b);
```

>>

Foto 6

```
Sumando todo
let a = 00000110; //6
let b = 00001010; //10
console.log(a >> 1);
console.log(b >> 1);

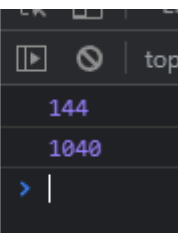
script>
```



<<

Foto 7

```
Sumando todo
let a = 00000110; //6
let b = 00001010; //10
console.log(a << 1);
console.log(b << 1);
```



>>>

Foto 8

```
Sumando todo
let a = 00000110; //6
let b = 00001010; //10
console.log(a >>> 1);
console.log(b >>> 1);
```

