En pocas palabras, la **precedencia de operadores** es cuando el lenguaje nos dice qué operación pasará primero y qué operación pasará después de acuerdo al operador. Por ejemplo:

<?php

$operacion = 1 + 5 \* 3;

¿Primero suma y luego multiplica? O ¿Primero multiplica y luego suma?

La respuesta es: primero multiplica y luego suma, por lo que nuestra variable $resultado valdrá 16. Esto es debido a que la multiplicación tiene mayor precedencia que la suma.

**La asociatividad**

Cuando los operadores tienen la misma precedencia, su asociatividad elige si se resuelve la instrucción a partir de la derecha o desde la izquierda.

<?php

// Tiene asociatividad de izquierda:

**echo** 1 - 2 - 3;

Observamos que se usa un mismo operador, por tanto, el nivel de precedencia va a ser igual. La asociatividad de “-” es de izquierda, en consecuencia, primero resolverá la resta 1 - 2 y luego restará 3.

<?php

**echo** 1 - 2 - 3;

/\*

Veamos como serían los estados de la operación ayudándonos del uso de paréntesis:

echo 1 - 2 - 3;

echo (1 - 2) - 3;

echo -1 - 3;

echo (-1 - 3);

echo -4

\*/

Ahora un ejemplo con asociatividad de derecha:

<?php

$c = 10;

$a = $b = $c;

**echo** $a; # 10

**echo** “\n”; # Salto de línea

/\*

Primero la variable $b se iguala al valor de la variable $c y luego resultado se iguala a la variable $a:

$a = $b = $c

$a = ($b = $c)

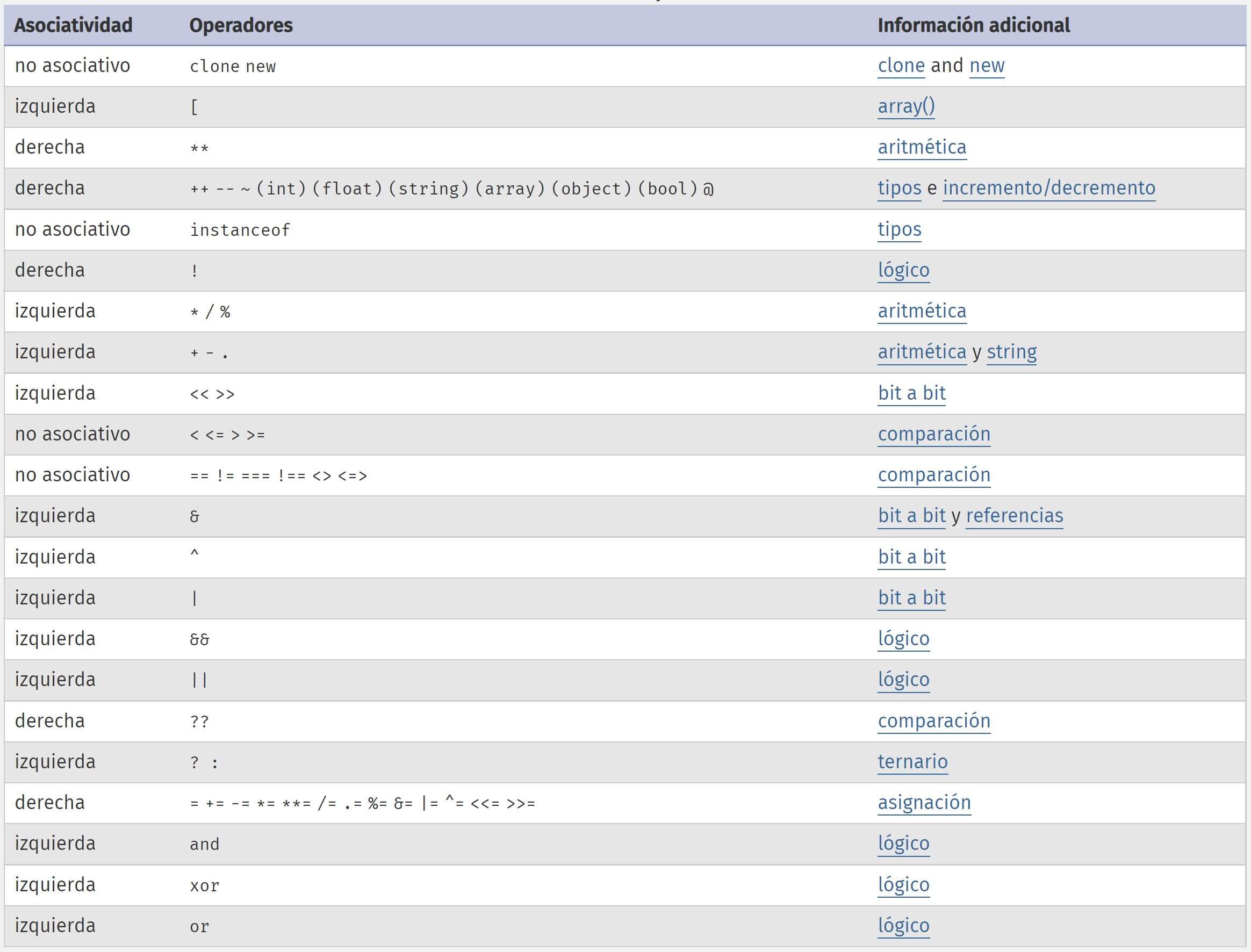
$a = (10)

$a = 10

Finalmente, se imprime en pantalla el número 10, pues este es el valor de la variable $a.

\*/

**Tabla de precedencia y asociatividad en PHP**

Observemos mediante una [tabla](https://www.php.net/manual/es/language.operators.precedence.php) la precedencia (los de más alta estarán al inicio) y asociatividad de los operadores en PHP:  


**Problemas con la precedencia**

Hay ocasiones en la que la precedencia y asociatividad de los operadores nos puede jugar en contra provocando resultados no deseados. Por ejemplo, si en una evaluación AND una de las afirmaciones es falsa, el resultado será falso (false):

<?php

$michis\_4\_patas = **true**;

$michis\_programan\_con\_PHP = **false**;

$resultado = $michis\_4\_patas **and** $michis\_programan\_con\_PHP;

var\_dump( $resultado ); # Imprimimos en pantalla el valor de la variable $resultado

**echo** “\n”; # Salto de línea

# Mensaje en consola:

# bool(true)

Sin embargo, si ejecutamos este código nos aparecerá en pantalla el valor de true. Esto es debido a que la asociatividad del operador and es de izquierda y hará que primero la variable $resultado se iguale al valor de $michis\_4\_patas el cual es true, por ello cuando queramos imprimirlo en consola nos dirá que el valor de $resultado es true:

<?php

$michis\_4\_patas = **true**;

$michis\_programan\_con\_PHP = **false**;

$resultado = $michis\_4\_patas **and** $michis\_programan\_con\_PHP;

/\*

Usemos unos paréntesis para entender que fue lo que pasó:

($resultado = $michis\_4\_patas) and $michis\_programan\_con\_PHP;

A la variable $resultado se le asignará el valor de la variable $michis\_4\_patas:

$resultado = true;

\*/

var\_dump( $resultado ); # Se imprimirá true en pantalla

**echo** “\n”; # Salto de línea

**Forzando la evaluación**

Una solución para los problemas que podemos tener con la precedencia y su asociatividad, es el uso de paréntesis en las operaciones que no queremos que se vean afectadas. Arreglemos el ejemplo anterior:

<?php

$michis\_4\_patas = **true**;

$michis\_programan\_con\_PHP = **false**;

$resultado = ($michis\_4\_patas **and** $michis\_programan\_con\_PHP);

var\_dump( $resultado ); # Imprimimos en pantalla el valor de la variable $resultado

**echo** “\n”; # Salto de línea

# Mensaje en consola:

# bool(false)

Los paréntesis forzarán a realizar la evaluación que este dentro. Por ello es que la variable $resultado ya no es igualado al valor de $michis\_4\_patas directamente, sino que tomará el resultado de la evaluación AND entre $michis\_4\_patas y $michis\_programan\_con\_PHP y como uno de estas variables es falsa, el valor final será false.

*Contribución creada por: Martín Álvarez*