Tipos de datos en PHP

PHP es un lenguaje débilmente tipado o no tipado

PHP es un lenguaje débilmente tipado o no tipado, como prefieras llamarlo.

Esto significa que no hay que declarar el tipo de las variables.

PHP no requiere (ni soporta) la definición explicita de tipos en la declaración de variables.

El tipo de la variable se determina por el contexto en el cual se emplea la variable. Es decir, si se asigna un valor string a una variable $var, entonces $var se convierte en un string.

Si después asignamos un valor integer a la misma variable $var, ésta se convierte en integer.

Tipos de datos en PHP

Los principales tipos de datos (considerados tipos de datos escalares) en PHP son:

boolean: almacenan valores verdadero o falso (true / false).
integer: números enteros.
float: números con decimales, usando el punto como separador decimal.
string: cadenas de texto.

PHP también incluye los siguientes tipos de datos compuestos (los trataremos en profundidad más adelante):

array: usados para almacenar varios valores.
object: almacena objetos en PHP.

Incluye también los siguientes tipos de datos especiales:

resource: hace referencia a recursos tales como archivos abiertos, conexiones establecidas con bases de datos, etc.
null: indica que el valor está vacío (no contiene nada).

```php
<?php
// Tamaño máximo y mínimo de int y float en PHP
// PHP proporciona una gran cantidad de constantes
// definidas por el núcleo de PHP y siempre están disponibles.

print "<br>";
print "  <strong>Números enteros en este servidor:</strong><br><br>";
print "  Valor mínimo: " . PHP_INT_MIN . "<br><br>";
print "  Valor máximo: " . PHP_INT_MAX . "<br><br>";
print "  En este servidor los enteros ocupan " . PHP_INT_SIZE . " bytes. <br><br>";
print "<br><br>";

print "  <strong>Números decimales en este servidor:</strong><br><br>";
print "  Valor máximo: " . PHP_FLOAT_MAX . "<br><br>";
print "  Valor mínimo: " . PHP_FLOAT_MIN . "<br><br>";
print "  En este servidor los float ocupan " . PHP_INT_SIZE . " bytes. <br>";
print "<br>";

$entero =
5555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555
5555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555
5555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555
555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555;
//Desbordamiento de una variable, retorna INF
print "<br>Desbordamiento de un variable de tipo entero: ". $entero;

$cadena =
"555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555
5555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555
5555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555
5555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555";
print "<br><br><br>El mismo contenido, pero tipo string: <br><br><br>" . $cadena;

print "<br><br><br>Nota tomada del manual de php: Un string puede llegar a alcanzar hasta 2 GB de tamaño (2147483647 bytes
máximo).";
?>
```

Directivas del núcleo de php configurada en el archivo php.ini

Esta lista incluye las directivas del núcleo en el archivo php.ini que se pueden establecer para la configuración de PHP.

Directivas:

memory_limit 512M

Establece el máximo de memoria en bytes que un script puede consumir. Ayuda a prevenir que scripts mal programados consuman toda la memoria disponible en el servidor.

max_execution_time 120

Este valor establece el tiempo máximo en segundos que se permite ejecutar antes de que el analizador termine. Esto ayuda a prevenir que scripts mal escritos bloqueen el servidor. El valor por defecto es 30 en producción.

php.ini: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
; Maximum amount of time each script may spend parsing request data. It's a good
; idea to limit this time on productions servers in order to eliminate unexpectedly
; long running scripts.
; Note: This directive is hardcoded to -1 for the CLI SAPI
; Default Value: -1 (Unlimited)
; Development Value: 60 (60 seconds)
; Production Value: 60 (60 seconds)
; https://php.net/max-input-time
max_input_time=60

; Maximum input variable nesting level
; https://php.net/max-input-nesting-level
;max_input_nesting_level = 64

; How many GET/POST/COOKIE input variables may be accepted
;max_input_vars = 1000

; Maximum amount of memory a script may consume
; https://php.net/memory-limit
memory_limit=512M
```

phpinfo()

Muestra información sobre la configuración de PHP

Como cada sistema se instala diferente phpinfo() se usa comúnmente para revisar opciones de configuración y variables predefinidas disponibles en un sistema dado

Directive	Local Value	Master Value
highlight.keyword	#007700	#007700
highlight.string	#DD0000	#DD0000
html_errors	On	On
ignore_repeated_errors	Off	Off
ignore_repeated_source	Off	Off
ignore_user_abort	Off	Off
implicit_flush	Off	Off
include_path	C:\xampp\php\PEAR	C:\xampp\php\PEAR
input_encoding	*no value*	*no value*
internal_encoding	*no value*	*no value*
log_errors	On	On
mail.add_x_header	Off	Off
mail.force_extra_parameters	*no value*	*no value*
mail.log	*no value*	*no value*
max_execution_time	120	120
max_file_uploads	20	20
max_input_nesting_level	64	64
max_input_time	60	60
max_input_vars	1000	1000
memory_limit	512M	512M
open_basedir	*no value*	*no value*
output_buffering	4096	4096

Conversión de tipo de datos

A diferencia de los lenguajes fuertemente tipados, los no tipados permiten combinar datos de diferente tipo, comparar cadenas de caracteres con números enteros o booleanos, por ejemplo, o sumar y restar datos de tipos distintos.

Cambiar una variable de tipo puede hacerse de dos formas:

- Automática o implícita.
- Manual o forzada.

Conversión de tipo de datos

A partir de PHP 8.1, se emite un aviso de desaprobación cuando un valor float se convierte implícitamente en un valor int, y se pierde el valor fraccionario en el proceso.

Este aviso de desaprobación no se emite cuando un valor float es convertido explícitamente a un entero.

```
Deprecated: Implicit conversion from float 1.5 to int loses precision
```

Un dato numérico válido en PHP

- Un signo opcional

- Seguido de uno o más dígitos

- Opcionalmente puede contener un punto decimal seguido de un exponente opcional.

- El exponente es una 'e' o 'E' seguida de uno o más dígitos.

Conversión de datos en PHP

Los integer pueden especificarse mediante notación decimal (base 10), hexadecimal (base 16), octal (base 8) o binaria (base 2), opcionalmente precedidos por un signo (- o +).

-Para utilizar la notación octal, se antepone al número un 0 (cero).
-Para utilizar la notación hexadecimal, se antepone al número 0x.
-Para utilizar la notación binaria, se antepone al número 0b.

Ejemplo: Literales de números enteros

```
<?php

$a = 1234; // número decimal
$a = -123; // un número negativo
$a = 0123; // número octal (equivale a 83 decimal)
$a = 0x1A; // número hexadecimal (equivale a 26 decimal)
$a = 0b11111111; // número binario (equivale al 255 decimal)

?>
```

Conversión de string a números

Cuando un string es evaluado en un contexto numérico, el valor resultante y el tipo se determina como se explica a continuación:

El valor es dado por la parte inicial del string, es decir, la regla para convertir un string en número es utilizar los primeros caracteres que formen un número.

Si el string empieza con un dato numérico válido, éste será el valor empleado de la siguiente forma:

-Si este número es entero, se convierte a entero.

-Si contiene notación científica se convierte a flotante.

-Si el string no contiene ninguno de los caracteres '.', 'e', o 'E', y el valor numérico está entre los límites del tipo integer (tal como está definido por PHP_INT_MAX), el string será evaluado como un integer.

-En todos los demás casos será evaluado como un float.

Conversión de string a números

Si el string NO comienza con un dato numérico válido y emitirá un warning, pero continua la ejecución del script

El valor será 0 (cero).

```php
<?php
$foo = 1 + "10.5";                  // $foo es float (11.5)
$foo = 1 + "-1.3e3";                // $foo es float (-1299)
$foo = 1 + "bob-1.3e3";             // $foo es integer (1)
$foo = 1 + "bob3";                  // $foo es integer (1)
$foo = 1 + "10 pequeños cerdos";    // $foo es integer (11)
$foo = 4 + "10.2 pequeños cerditos"; // $foo es float (14.2)
$foo = "10.0 cerdos " + 1;          // $foo es float (11)
$foo = "10.0 cerdos " + 1.0;        // $foo es float (11)
?>
```

Conversión implícita o automática de datos en PHP

La conversión automática significa que PHP buscará el tipo apropiado para el resultado de una operación dependiendo del tipo de los operandos y del operador.
Si los dos operandos son del mismo tipo (ej.: dos enteros), el resultado será de ese tipo (entero), el resto se comporta de acuerdo con la siguiente tabla:

Operadores		Resultados
int	float	float (operadores matemáticos)
int	string	int (operadores matemáticos)
int	string	int (operador concatenación)
float	string	float (operadores matemáticos)

Conversión implícita o automática de datos en PHP

Operadores		Resultados
int	float	float (operadores matemáticos)
int	string	int (operadores matemáticos)
int	string	int (operador concatenación)
float	string	float (operadores matemáticos)

```php
<?php
// Concatenación de enteros y cadenas
$var1 = 5;
$var2 = " edad";
$resultado = $var1 . $var2;
echo $resultado;
var_dump($resultado);
?>
```

```
5 edad
string '5 edad' (length=6)
```

Conversión manual o forzada (typecasting) de datos en PHP

Otra forma de convertir tipos de datos indicando el tipo de dato hacia el que deseamos convertir:

-(int) / (integer): forzado a integer.

-(bool) / (boolean): forzado a boolean.

-(float) / (double) / (real): forzado a float.

-(string): forzado a string.

-(array): forzado a array.

-(object): forzado a object.

-(unset): forzado a null (PHP 5). //Obsoleta en php 7.2 e inútil

Conversión manual o forzada (typecasting) de datos en PHP

También podemos usar la función de PHP settype() indicando como segundo parámetro el tipo de dato al que deseamos convertir (devuelve true si la conversión se ha realizado correctamente, o false en caso contrario):

```
settype(mixed &$var, string $type): bool
```

Conversión manual o forzada (typecasting) de datos en PHP

```php
<?php

    $var1 = "valor5";
    $var2 = "4valor";
    $var3 = "867";
    $var4 = "867.56";
    $var5 = "09";
    $var6 = "867.56";
    $var7 = true;
    $var8 = 567;
    $var9 = 55.89;

    // Realizar la conversión:

    $var1 = (int) $var1;
    $var2 = (int) $var2;
    $var3 = (int) $var3;
    $var4 = (int) $var4;
    $var5 = (int) $var5;
    $var6 = (float) $var6;
    $var7 = (string) $var7;
    $var8 = (string) $var8;
    $var9 = (string) $var9;
```

$var1 es ahora integer y su valor es [0]
$var2 es ahora integer y su valor es [4]
$var3 es ahora integer y su valor es [867]
$var4 es ahora integer y su valor es [867]
$var5 es ahora integer y su valor es [9]
$var6 es ahora float y su valor es [867.56]
$var7 es ahora string y su valor es '1'.
$var8 es ahora string y su valor es '567'
$var9 es ahora string y su valor es '55.89'

Conversión manual o forzada (typecasting) de datos en PHP

```php
<?php
// Realizar la conversión:
$var1 = "valor5";
$var2 = "4valor";
$var3 = "867";
settype($var1, "integer");
settype($var2, "integer");
settype($var3, "integer");

var_dump($var1, $var2, $var3);
?>
```

```
int 0

int 4

int 867
```

```php
<?php
    $var = "true";
    settype($var, 'bool');
    var_dump($var); // true

    $var = "false";
    settype($var, 'bool');
    var_dump($var); // true

    $var = "";
    settype($var, 'bool');
    var_dump($var); //false
?>
```

Conversión manual o forzada (typecasting) de datos en PHP

Precaución:

El comportamiento de la conversión de integer a otros tipos es indefinido.

No confíe en ningún comportamiento observado, ya que puede cambiar sin previo aviso.

Conversión manual o forzada (typecasting) de datos en PHP

Función

intval()

Para convertir un valor a entero también tenemos la función intval()

intval(mixed $var, int $base = 10): int

- El parámetro base no tiene ningún efecto a menos que el parámetro var sea una cadena.

- Devuelve el valor integer de una variable, con la base indicada para la conversión (por defecto es base 10)

Nota:

Si base es 0, la base usada estará determinada por el formato de var:

- si el string incluye el prefijo "0x" (o "0X"), la base se tomará como 16 (hex); si no,
- si el string comienza con "0", la base se tomará como 8 (octal);
- si no, la base se toma como 10 (decimal).

Conversión manual o forzada (typecasting) de datos en PHP

```
<?php
var_dump( intval(42));                                  → int 42
var_dump (intval(4.2));                                 → int 4
var_dump (intval('42'));                                → int 42
var_dump (intval('+42'));                               → int 42
var_dump (intval('-42'));                               → int -42
var_dump (intval(042));                                 → int 34 //transforma base 8 a base 10
var_dump (intval('042'));                               → int 42
var_dump (intval(1e10));                                → int 10000000000
var_dump (intval('1e10'));                              → int 10000000000
var_dump (intval(0x1A));                                → int 26
var_dump (intval('0x1A'));                              → int 0
var_dump (intval('0x1A',16));                           → int 26
var_dump (intval(420000000000000000000000000000000000000000));  → int 0  //desbordamiento
var_dump (intval(42,8));                                → int 42 //el parámetro base 8 no se aplica porque no es una cadena

var_dump (intval('042',8));                             → int 34 //el parámetro base 8 se aplica porque dato pasado a la función es una cadena
?>
```

var_dump()

Muestra información sobre una variable

Descripción

- var_dump (mixed $expression [, mixed $...]) : void

- Esta función muestra en el navegador información estructurada sobre una o más expresiones incluyendo su tipo y valor.
- Las matrices y los objetos son explorados recursivamente con valores sangrados para mostrar su estructura.

Eliminar una variable en PHP

- Si deseamos eliminar una variable en PHP (y con ello liberar memoria en el servidor) usaremos la función unset()

```
unset ( mixed $var [, mixed $... ] ) : void
```

- Las variables se eliminan automáticamente cuando se termina de procesar un script en PHP (el archivo .php, no el código fuente contenido entre las etiquetas <?php ... ?>), liberándose la memoria que ocupaban.

- Si se necesita pasar valores entre diferentes páginas en PHP podrás utilizar variables de sesión, las cuales explicaremos más adelante.

función gettype()

- Para obtener información sobre el tipo de dato que contiene una variable disponemos de la función gettype(), pero nos devuelve una cadena que es más difícil de utilizar en las estructuras de control.

```php
<?php
    $a = "123";
    $b = 123;
    echo gettype($a) . "<br>";  //string
    echo gettype($b);             //integer
?>
```

Averiguar tipo de dato una variable

Se recomienda usar en su lugar las siguientes funciones devuelven 'true' o 'false':

is_string(): el valor es de tipo cadena.

is_int(): el valor es un número entero. Si el número entero está como cadena de texto (entre comillas) no se considera como un número.

is_float(): el valor es un número con decimales. Si el número decimal está como cadena de texto (entre comillas) no se considera como un número.

is_numeric(): el valor es un número o una cadena numérica. Al contrario de lo que sucede con is_int() e is_float(), si el número se encuentra como cadena de texto (entre comillas) sí se considera que es numérico.

Averiguar tipo de dato de una variable

Importante

Nota: (del manual de PHP)
Los formularios HTML no pasan enteros, reales o booleanos; solo pasan cadenas. Para saber si una cadena es numérica, se puede usar is_numeric().

Por ello es recomendable usar is_numeric() para validar un formulario (ya que los datos de este llegan a PHP como cadena de texto).

Averiguar tipo de dato de una variable

Tipo Booleanos

Este es el tipo más simple. Un boolean expresa un valor que indica verdad. Puede ser TRUE (verdadero) o FALSE (falso).

Sintaxis

Para especificar un literal de tipo boolean se emplean las constantes TRUE o FALSE. Ambas no son susceptibles a mayúsculas y minúsculas.

```
<?php
    $foo = True; // asigna el valor TRUE a $foo
    $valor= FALSE;
?>
```

Averiguar tipo de dato de una variable

Los siguientes valores se consideran FALSE en php:

El boolean FALSE
El integer 0 (cero)
El float 0.0 (cero)
El valor string vacío, y el string "0"
Un array con cero elementos
El tipo especial NULL (incluidas variables no establecidas)

Cualquier otro valor se considera como TRUE

Averiguar tipo de dato de una variable

is_bool()
Comprueba si una variable es de tipo booleano

Descripción
is_bool (mixed $var) : bool

Valores devueltos

Devuelve TRUE si var es un boolean, FALSE de lo contrario.

Averiguar tipo de dato de una variable

```php
<?php
$a = false;
$b = 1;

// Ya que $a es un booleano, devolverá true
if (is_bool($a) === true) {
    echo "Sí, este es un booleano";
}

// Ya que $b no es un booleano, devolverá false
if (is_bool($b) === false) {
    echo "No, este no es un booleano";
}
?>
```

```php
<?php
$a = false;
$b = 0;
$c = true;
$d = 1;

echo "<br> el valor es  " . (is_bool($a));
echo "<br> el valor es  " . (is_bool($b));
echo "<br> el valor es  " . (is_bool($c));
echo "<br> el valor es  " . (is_bool($d));
if ($a == $b) {
    echo "<br> Los valores 0 y false son iguales utilizando el operador  ==";
} else {
    echo "<br>Los valores 0 y false no son iguales utilizando el operador  ==";
}

if ($a === $b) {
    echo "<br>Los valores 0 y false son iguales  el operador ===";
} else
    echo "<br>Los valores 0 y false NO son iguales el operador  ===";
?>
```

```
el valor es 1
el valor es
el valor es 1
el valor es
Los valores 0 y false son iguales utilizando el operador ==
Los valores 0 y false NO son iguales el operador ===
```

el valor es 1

el valor es

el valor es 1

el valor es

El echo no imprime el valor false y el valor true se muestra 1

Los valores 0 y false son iguales utilizando el operador ==

Los valores 0 y false NO son iguales utilizando el operador ===

Averiguar tipo de dato de una variable

Tipo dato :NULO

El valor especial NULL representa una variable sin valor.

Una variable es considerada null:

-si se le ha asignado la constante NULL.

-si no se le ha asignado un valor todavía.

-si se ha destruido con unset().

No hay más que un valor de tipo null, y es la constante NULL insensible a mayúsculas/minúsculas.

Averiguar tipo de dato de una variable

```php
<?php
$var = NULL;

$a;

$z=23;

unset($z);

var_dump ($var);
var_dump ($a);
var_dump ($z);
?>
```

Averiguar tipo de dato de una variable

C:\wamp64\www\respuesta.php:144:null

(!) Notice: Undefined variable: **a** in C:\wamp64\www\respuesta.php on line 145
Call Stack

#	Time	MemoryFunction	Location
1	0.0008	402328 {main}( )	...\respuesta.php:0

C:\wamp64\www\respuesta.php:145:null

(!) Notice: Undefined variable: **z** in C:\wamp64\www\respuesta.php on line 146
Call Stack

#	Time	MemoryFunction	Location
1	0.0008	402328 {main}( )	...\respuesta.php:0

C:\wamp64\www\respuesta.php:146:null

Averiguar el tipo de dato de una variable

is_null ()

Comprueba si una variable dada es NULL

Descripción
is_null (mixed $var) : bool

Valores devueltos

Devuelve TRUE si var es null, FALSE de lo contrario.

Averiguar tipo de dato de una variable

```php
<?php
    $var1;
    $var2 = null;
    $var3 = "null";

     var_dump ($var1);  //muestra NULL

     var_dump ($var2); // muestra NULL

     var_dump ($var3); // string 'null' (length=4)
?>
```

Averiguar tipo de dato de una variable

empty()

Otra función de PHP que nos resultará útil es empty(), que devolverá true si la variable está vacía.

empty (mixed $var) : bool

Una variable se considera vacía si no existe o si su valor es igual a FALSE.

empty() no genera una advertencia si la variable no existe.

Devuelve FALSE si var existe y tiene un valor no vacío, distinto de cero. De otro modo devuelve TRUE.

Averiguar tipo de dato de una variable

empty()

Son expresiones consideradas como vacías:

- "" (una cadena vacía)
- 0 (0 como un entero)
- 0.0 (0 como un real)
- "0" (0 como una cadena)
- NULL
- FALSE
- array() (un array vacío)
- $var; (una variable declarada, pero sin un valor)

Averiguar tipo de dato de una variable

Ejemplo empty()

```
<?php
    $var1;
    $var2 = null;
    $var3 = "null";
    $var4 = "";
    $var5 = 0;
    $var6 = "0";
    $var7 = true;
    $var8 = false;

        var_dump (empty($var1));  → boolean true //no genera advertencia
        var_dump (empty($var2));  → boolean true
        var_dump (empty($var3));  → boolean false
        var_dump (empty($var4));  → boolean true
        var_dump (empty($var5));  → boolean true
        var_dump (empty($var6));  → boolean true
        var_dump (empty($var7));  → boolean false
        var_dump (empty($var8));  → boolean true
?>
```

Averiguar tipo de dato de una variable

Ejemplo empty()

```
<?php
    $var1;
    $var2 = null;
    $var3 = "null";
    $var4 = "";
    $var5 = 0;
    $var6 = "0";
    $var7 = true;
    $var8 = false;

        var_dump (empty($var1));  → boolean true //no genera advertencia
        var_dump (empty($var2));  → boolean true
        var_dump (empty($var3));  → boolean false
        var_dump (empty($var4));  → boolean true
        var_dump (empty($var5));  → boolean true
        var_dump (empty($var6));  → boolean true
        var_dump (empty($var7));  → boolean false
        var_dump (empty($var8));  → boolean true
?>
```

Averiguar tipo de dato de una variable

isset ()

Determina si una variable está definida y no es NULL

Descripción

```
isset ( mixed $var [, mixed $... ] ) : bool
```

Si una variable ha sido eliminada con unset(), ésta ya no estará definida.

isset() devolverá FALSE si una variable que ha sido definida como NULL.

Averiguar tipo de dato de una variable

Si se le pasan varios parámetros, entonces isset() devolverá TRUE únicamente si todos los parámetros están definidos.

La evaluación se realiza de izquierda a derecha y se detiene tan pronto como se encuentre una variable no definida.

Devuelve TRUE si $var existe y tiene un valor distinto de NULL, devuelve FALSE de lo contrario.

Averiguar tipo de dato de una variable

```
<?php

$var1 = '';//cadena vacía
$var2 = "prueba";
$var3 = "0";
$var4 = NULL;
$var5;

var_dump(isset($var1)); ——→ boolean true
var_dump(isset($var2)); ——→ boolean true
var_dump(isset($var3)); ——→ boolean true
var_dump(isset($var4)); ——→ boolean false
var_dump(isset($var5)); ——→ boolean false
var_dump(isset($var6)); ——→ boolean false // $var6 no aparece en el script

unset($var2);
var_dump(isset($var2)); ——→ boolean false

?>
```

Averiguar tipo de dato de una variable

Hacer un simple if ($x) mientras que $x no esté definido, generará un error de nivel E_NOTICE.

En lugar de esto, considere el uso de empty() o isset(), y/o inicializa las variables.

¿Cuándo usar isset(), empty() o is_null()?

1. Hay que usar isset() cuando necesitemos averiguar si una variable está definida en el script que estamos ejecutando, es decir, si existe.

2. La función empty() hay que usarla cuando queramos averiguar si una variable está vacía o no, es decir, si tiene contenido o no.

3. La función is_null() se usa cuando queremos saber si una variable es igual a NULL.

Comparaciones de $x con funciones PHP

Expresión	gettype()	empty()	is_null()	isset()	boolean : *if($x)*
$x = "";	string	**TRUE**	**FALSE**	**TRUE**	**FALSE**
$x = null;	NULL	**TRUE**	**TRUE**	**FALSE**	**FALSE**
var $x;	NULL	**TRUE**	**TRUE**	**FALSE**	**FALSE**
$x no está definido	NULL	**TRUE**	**TRUE**	**FALSE**	**FALSE**
$x = array();	array	**TRUE**	**FALSE**	**TRUE**	**FALSE**
$x = array('a', 'b');	array	**FALSE**	**FALSE**	**TRUE**	**TRUE**
$x = false;	boolean	**TRUE**	**FALSE**	**TRUE**	**FALSE**
$x = true;	boolean	**FALSE**	**FALSE**	**TRUE**	**TRUE**
$x = 1;	integer	**FALSE**	**FALSE**	**TRUE**	**TRUE**
$x = 42;	integer	**FALSE**	**FALSE**	**TRUE**	**TRUE**
$x = 0;	integer	**TRUE**	**FALSE**	**TRUE**	**FALSE**
$x = -1;	integer	**FALSE**	**FALSE**	**TRUE**	**TRUE**
$x = "1";	string	**FALSE**	**FALSE**	**TRUE**	**TRUE**
$x = "0";	string	**TRUE**	**FALSE**	**TRUE**	**FALSE**
$x = "-1";	string	**FALSE**	**FALSE**	**TRUE**	**TRUE**
$x = "php";	string	**FALSE**	**FALSE**	**TRUE**	**TRUE**
$x = "true";	string	**FALSE**	**FALSE**	**TRUE**	**TRUE**
$x = "false";	string	**FALSE**	**FALSE**	**TRUE**	**TRUE**

Comparaciones flexibles con operador ==												
	TRUE	FALSE	1	0	-1	"1"	"0"	"-1"	NULL	array()	"php"	" "
TRUE	TRUE	FALSE	TRUE	FALSE	TRUE	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE
FALSE	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	TRUE	TRUE	FALSE	TRUE
1	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
0	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	TRUE	FALSE	TRUE	TRUE
-1	TRUE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE
"1"	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
"0"	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE
"-1"	TRUE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE
NULL	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	FALSE	TRUE
array()	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	FALSE	FALSE
"php"	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE
" "	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE

Comparaciones estrictas con el operador ===												
	TRUE	FALSE	1	0	-1	"1"	"0"	"-1"	NULL	array()	"php"	""
TRUE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
1	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
0	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
-1	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
"1"	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
"0"	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE
"-1"	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE
NULL	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE
array()	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE
"php"	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE
""	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE

Averiguar tipo de dato de una variable

Otras funciones de PHP que devuelven información sobre los tipos de datos son:

is_array(): la variable es un array.

is_resource(): la variable contiene una referencia a un recurso (un archivo abierto, conexión con una base de datos, etc.)

is_object(): la variable es un Objeto.

method_exists(): un Objeto contiene un método con el nombre especificado.

Más adelante aprenderemos a usar arrays, recursos y objetos.

Operador ternario ? :

La expresión (expr1) ? (expr2) : (expr3) evalúa la expr2 si expr1 es TRUE y evalúa expr3 si expr1 es FALSE.

Es un if-else con sintaxis abreviada

```
if ($A == $B) {
  $res = "La variable A es igual a la B";
} else {
  $res = "La variable A es distinta a la B";
}
```

```
$res = ($A == $B) ? "A es igual a B" : "A no es igual a B"
```

```
echo ($A > $B) ? "A es mayor que B" : "B es menor o igual que A";
```

Operador ternario ? :

A partir de PHP 5.3, es posible dejar de lado la parte media del operador ternario.

La expresión expr1 ?: expr3 retorna expr1 si expr1 se evalúa como TRUE y expr3 si en cualquier otro caso.

Operador ternario ? :

```
<?php
$a=6;
$b=7;
$c=0;
var_dump( $a<$b?:"mayor");          → boolean true
var_dump( $a>$b?:"mayor");          → string 'mayor' (length=5)
var_dump ($b?:"estoy en false");    → int 7
var_dump ($c?:"estoy en false");    → string 'estoy en false' (length=14)

echo $resultado=$a<$b?:"mayor";     → 1
echo "<br>";
echo $resultado=$a>$b?:"mayor";     → mayor
echo "<br>";
echo $b?:"estoy en false";          → 7
echo "<br>";
echo $c?:"estoy en false";          → estoy en false

?>
```

Operador fusión a null ??

Lo que hace este poderoso operador es devolver el primer valor que no sea nulo.

```php
<?php
    if(isset($_GET["limite"])){
      $limite = $_GET["limite"];
    }else{
      $limite = 10;
    }
?>
```

Si queremos ser un poco más elegantes, usamos el operador ternario y quedaría rescrito así:

```php
<?php
    $limite = isset($_GET["limite"]) ? $_GET["limite"] : 10;
?>
```

Pero el código es más claro en la siguiente instrucción:

```php
<?php
    $limite = $_GET["limite"] ?? 10;
?>
```

Operador fusión a null ??

```php
<?php
    if(isset($_GET["limite"])){

        $limite = $_GET["limite"];
    }else if(isset($_POST["limite"])){

        $limite = $_POST["limite"];
    }else{
        $limite = 10;
    }
?>
```

Operador fusión a null ??

Con el operador ternario (se recomienda no anidar el operador ternario por falta de claridad)

```php
<?php
    $limite = isset($_GET["limite"]) ? $_GET["limite"] : isset($_POST["limite"]) ? $_POST["limite"]
    : 10;
?>
```

Finalmente, con el operador fusión a null, queda de la siguiente manera:

```php
<?php
    $limite = $_GET["limite"] ?? $_POST["limite"] ?? 10;
?>
```

Esta línea de código intenta obtener el valor de la variable limite desde la URL (mediante $_GET) o desde el formulario (mediante $_POST), y si no encuentra ningún valor, asigna un valor predeterminado de 10 a la variable $limite.

Operador fusión a null ??

Este operador viene a remplazar los if, isset

Se puede evaluar una lista infinita de operandos. Ejemplo:

```php
<?php
    $limite = null ?? null ?? null ?? 10 ?? 20 ?? 30;
?>
```

En este caso, límite es 10, ya que es el primer valor que no es nulo y ahí deja de evaluar.

Operador fusión a null ??

El operador de fusión a null ofrece varias ventajas:

- Evita errores de tipo "Notice: Undefined variable" cuando se intenta acceder a una variable que no existe.
- Permite asignar un valor predeterminado a una variable si no se ha definido previamente.
- Simplifica el código y lo hace más legible.