Tema 3 - Fundamentos de PHP

Índice de contenidos

Tipos de datos en PHP

• Booleanos
• Números enteros
• Números de coma float
• Cadenas de caracteres
• Matrices o arrays
• NULL

- PHP es muy flexible en la manejo de los tipos de datos:
- No obliga a definir qué tipo de datos contendrán las variables
- Una variable pueden contener datos de diferentes tipo a lo largo del script
- Permite combinar datos de distintos tipos en las operaciones

¿Se pueden sumar diferentes tipos?
No, unas reglas convierten los operandos a un tipo común.

Reglas automáticas de conversión de tipos de datos:

– En **operaciones lógicas**, los datos NULL, 0, '0' y '' se consideran FALSE. Cualquier otro dato se considera TRUE (incluida la cadena 'FALSE').

– En **operaciones aritméticas** las cadenas se intentan leer como números y si no se puede se convierten en 0, TRUE se convierte en 1, y FALSE se convierte en 0.

– En **operaciones de comparación**, si un operando es un número, el otro también se convertirá en un número. Sólo si ambos operandos son cadenas se comparan como cadenas de caracteres.

– En **operaciones de cadenas de caracteres**, NULL y FALSE se convierten en '', y TRUE se convierte en '1'.

Operaciones aritméticas con enteros:

si resultado está en el rango de los enteros, devuelve un entero.

si resultado sale del rango de los enteros, devuelve un float

Operaciones aritméticas de enteros con float:

el resultado es float

Operaciones aritméticas con cadenas

Sí cadena comienza con dígitos, los extrae y opera con ello como entero

Sí cadena no comienza con dígitos, toma el valor 0 (cero)

Operaciones aritméticas con booleanos:

TRUE lo toma como 1 y FALSE como 0 (cero)

Operaciones de concatenación devuelven tipo string

Los números (enteros y reales) se convierten en cadenas

Valor booleano TRUE se convierte en cadena ‘1’ y FALSE en cadena vacía

Operaciones de comparación:

Números: El 0 lo toma como FALSE, otros valores los toma como TRUE

Cadenas: La cadena vacía (sin contenido) la toma como FALSE,

el resto como TRUE

Casting

El forzado explícito de tipos (Casting) establece el tipo para un dato sin tener en cuenta su contenido.

Tipos de casting:

(boolean)

(integer)

(float)

(real)

(double)

(string)

(array)

(unset) convierte al tipo NULL

Ejemplos de casting

```
<?php
// Casting a boolean
$numero = 42;
$booleano = (bool) $numero;
var_dump($booleano); // bool(true)

// Casting a integer
$cadena = "123";
$entero = (int) $cadena;
var_dump($entero); // int(123)

// Casting a float, real, and double
$numero = 3.14159;
$float = (float) $numero;
$real = (real) $numero; // 'real' es un alias para 'float'
$doble = (double) $numero;
var_dump($float); // float(3.14159)
var_dump($real); // float(3.14159)
var_dump($doble); // float(3.14159)

// Casting a string
$entero = 42;
$cadena = (string) $entero;
var_dump($cadena); // string(2) "42"
```

```
// Casting a array
$objeto = (object) ["clave" => "valor"];
$miarray = (array) $objeto;
var_dump($miarray); // array(1) { ["clave"]=> string(5) "valor" }

// Casting a unset
/* (Nota: Unset no es un tipo de dato, este ejemplo muestra cómo destruir una variable) */
$variable = "Hola";
unset($variable); // Destruir la variable
var_dump($variable); // NULL (variable no definida)
```

Cadenas de caracteres

Comillas simples

Se usa comillas simples o dobles para delimitar su valor

```
<?php $nombre= ' Maria ' ; ?>
```

Comillas dobles

Usando comillas dobles, sustituye nombres de variables por su valor valor

```
<?php $Nombre= " user : $nombre " ; ?>
```

Caracteres especiales

Cuando el valor contiene caracteres especiales deben ir precedidos de \

```
<?php $Nombre =" user : \" $nombre\" " ; ?>
```

Concatenación

El . (punto) es el operador de concatenación de cadenas

```
<?php
$mensaje = 'Confirmada ' . $nombre . " su peticion de " ;
```

```
$mensaje = $mensaje . $a + $b . 'unidades ' ;
?>
```

Sistema HereDoc

```
<?php
$cad = <<<FOO
<h3 style='color:$color'>Los datos del script son:</h3>
<table border='1'><tr><td>nombre: $name </td>
<td> repetidor: $rep </td>
<td>notas de $ev : $v1 , $v2, $v3</td><tr></table>
FOO;
echo $cad;
?>
```

Tipos de datos

gettype(var): string

La función **gettype** devuelve una string con el tipo del dato recibido como argumento

```
<?php
$variable1 = 42;
$variable2 = "Hola, mundo!";
$variable3 = 3.14;
$variable4 = true;
$variable5 = ["hola", "adiós"];

echo gettype($variable1); // Imprime "integer"
echo gettype($variable2); // Imprime "string"
echo gettype($variable3); // Imprime "double" (para números de punto flotante)
```

echo gettype($variable4); // Imprime "boolean"
echo gettype($variable5); // Imprime "array"

settype(mixed &var, string): bool

Cambia el tipo de dato de una variable.

```php
<?php
$foo = "5bar"; // string
$bar = true; // boolean

settype($foo, "integer"); // $foo es ahora 5 (integer)
settype($bar, "string"); // $bar es ahora "1" (string)
?>
```

Valores posibles del tipo de dato:

- "boolean" o "bool"
- "integer" o "int"
- "float" o "double"
- "string"
- "array"
- "object"
- "null"

Ejemplos de tipos de datos

<?php
// Asignación mixta con enteros y cadenas
$numero = 10;
$cadena = "20";
$resultado = $numero + $cadena;

```
echo "Resultado (entero + cadena): $resultado\n";
 // Resultado (entero + cadena): 30

// Asignación mixta con float y enteros
$decimal = 5.5;
$entero = 2;
$resultado = $decimal * $entero;
echo "Resultado (decimal * entero): $resultado\n";
// Resultado (decimal * entero): 11

// Asignación mixta con booleanos y enteros
$booleano = true;
$entero = 3;
$resultado = $booleano + $entero;
echo "Resultado (booleano + entero): $resultado\n";
// Resultado (booleano + entero): 4

// Asignación mixta con números float y cadenas que contienen números
$decimal = 3.14159;
$cadena = "2.5";
$resultado = $decimal / $cadena;
echo "Resultado (float / cadena): $resultado\n";
// Resultado (float / cadena): 1.256636

// Asignación mixta con números float y booleanos
$mifloat = 2.5;
$booleano = false;
$resultado = $mifloat + $booleano;
echo "Resultado (float + booleano): $resultado\n";
// Resultado (float + booleano): 2.5



$nombre = "Juan";
```

```
$edad = 30;

$texto = <<<EOD
¡Hola, soy $nombre!
Tengo $edad años de edad.
Este es un ejemplo de HEREDOC en PHP.
EOD;

echo $texto;
/* Muestra por pantalla
¡Hola, soy Juan!
Tengo 30 años de edad.
Este es un ejemplo de HEREDOC en PHP.
*/
```

Tipos de salida por pantalla

echo y print

Ya explicados anteriormente

printf

PRINTF (cadena de formato, variable1 [, variable2 …])

Permite presentar varios valores o variables con distintos formatos cadena de formato "%[relleno][alineación][longitud][precisión][tipo]"

```
<?php
printf("%010d" , 32) ; // Ancho de 10 y rellena a izq con 0 -------- 0000000032
printf("%'*10d", 32) ; // rellena a izq con * ---------------------- ********32
printf("%'*-10d", 32) ; // rellena a derecha con * ------------------- 32********
```

```
printf("%'*10.5d", 32.25) ; // con enteros no se aplica precisión  ********32
printf("%'*10.5f", 32.25) ; // con tipos float si se aplica ----------     **32.25000
printf("%'*10b", 17) ; // dato en binario ------------------------------- *****10001
printf("%0-10s", 32) ; // dato tipo cadena y relleno a dcha con 0  3200000000
printf("%'*10o", 17) ; // dato en octal ---------------------------------- ********21
printf("%'*10x", 170) ; // dato en hexadecimal ---------------------- ********aa
printf("%'*10X", 170) ; // dato en hexadecimal ---------------------- ********AA

printf("%.3e", 1234.56789);  // Imprime "1.23e+3"

printf("%.3e", 0.00012345);  // Imprime "1.234e-4"

printf("%.3E", 987654321);  // Imprime "9.877E+8"

?>
```

1. **relleno**

- (espacio): Rellena el resultado con espacios. Este es el valor predeterminado.
  Ejemplo: printf ("%10d", 123); //Imprime   123 (7 espacios en blanco delante)
- 0: Rellena los números solo en el lado izquierdo con ceros.
  Ejemplo: printf ("%010d", 123); // Imprime 0000000123

- '(char): Rellena el resultado con el carácter (char).
  Ejemplo: printf ("%'A10d", 123); // Imprime AAAAAAA123

2. **alineación**

   +: alineación a la izquierda (valor por defecto)
   -: alineación a la derecha

3. **longitud**

   Tamaño mínimo de la salida generada

4. **precisión**
    - Para los especificadores e, E, f y F: este es el número de dígitos que se imprimirán después del punto decimal (por defecto, esto es 6).

```
$number = 3.14159265;
printf("%.2f\n", $number);  // Imprime "3.14"
```

    - Para el especificador s: actúa como un punto de corte, estableciendo un límite máximo de caracteres para la cadena.

```
$text = "Este es un ejemplo de una cadena larga";
printf("%.10s", $text);  // Imprime "Este es un"
printf("%.10s…", $text);  // Imprime "Este es un…"
```

5. **tipo**

Especificador	Descripción
%	Un carácter de porcentaje literal. No se requiere ningún argumento.
b	El argumento se trata como un número entero y se presenta como un número binario.
c	El argumento se trata como un número entero y se presenta como el carácter con ese ASCII.
d	El argumento se trata como un número entero y se presenta como un número decimal (con signo).
e	El argumento se trata como notación científica (por ejemplo, 1,2e+2).
E	Como el e pero usa letras mayúsculas (por ejemplo, 1.2E+2).

f	El argumento se trata como float y se presenta como un número de punto decimal (afectado por la configuración regional).
F	El argumento se trata como float y se presenta como un número de punto decimal (sin reconocimiento regional).
o	El argumento se trata como un número entero y se presenta como un número octal.
s	El argumento se trata y presenta como una cadena.
u	El argumento se trata como un número entero y se presenta como un número decimal sin signo.
x	El argumento se trata como un número entero y se presenta como un número hexadecimal (con letras minúsculas).
X	El argumento se trata como un número entero y se presenta como un número hexadecimal (con letras mayúsculas).

La función printf() permite presentar varios valores o variables con distintos formatos utilizando cadenas de formatos específicas para cada valor a presentar .
printf("%'*15.2f Euros=%'*18.0f Pesetas " , 1475.875, 1475.875*166.386);

devuelve como resultado:
********1475.88 Euros=**********245565 Pesetas

var_dump

La función var_dump sirve para obtener información sobre un dato

```
$ej1 = 42;
var_dump($ej1);
```

```
// Imprime int(42)

$ej2 = ["hola", "adiós"];
var_dump($ej2);
// Imprime array(2) { [0]=> string(4) "hola" [1]=> string(6) "adiós" }
```