



Open
TO
Inspiration



Elementos Básicos

PHP – Elementos Básicos

DAW – Desarrollo Web de Entorno Servido

Fernando Díaz-Alonso Dorado



Consideraciones

Script PHP

- Todos los scripts de PHP comienzan con “<?php” y terminan con “?>”. En algunos IDE se puede omitir la terminación ?>
- Cada línea de comando debe terminarse con un punto y coma → “;”
- Existen dos formas básica de mostrar datos por la pantalla:
 - echo
 - print

```
<?php  
    echo "Hola";  
    print "mundo";  
?>
```



Hola mundo

Consideraciones II

Comentarios

- Por norma general los comentarios no se ejecutaran y no se tienen en cuenta **hasta PHP 8**.
 - A partir de PHP 8, los comentarios pueden usarse para programar mediante los Attributes. <https://www.php.net/manual/es/language.attributes.php>
- Comentarios podemos hacer comentarios de línea → // o #
- Comentarios de párrafo: → inicio → /* fin → */
- PHPDoc: → sirve para crear comentarios de desarrollo y son parecidos a los de párrafo, pero se inician → /**

```
<?php
//Esto se considera un comentario de línea --> echo "Podemos ver que esto NO se muestra";
# Comentario de línea --> echo "Podemos ver que esto NO se muestra";
/*
 * Comentario de párrafo.
 * echo "Podemos ver que esto NO se muestra";
 */
/**
 * Esto se considera un PHPDoc que nos suele ayudar durante el desarrollo.
 * echo "Podemos ver que esto NO se muestra";
 */
echo "Podemos ver que esto si se muestra";
?>
```



Podemos ver que esto sí se muestra

<https://www.php.net/manual/es/language.basic-syntax.comments.php>

Variables

Identificación

- Una variable en PHP **comienza siempre por el símbolo de dólar “\$”**, seguido después del nombre que tenga o se quiera dar a la variable.
- El nombre es **sensible** a las mayúsculas y minúsculas.
 - \$hora, \$hOra y \$Hora son variables diferentes.
- El nombre de la variable **siempre** debe comenzar por una letra o carácter de subrayado, underscore “_”. Por tanto, no puede empezar por un número.
- El nombre de la variable **solamente** puede contener caracteres alfanuméricos y guiones bajos. (A-z, 0-9 y _)
- **Existen variables especiales cómo es el caso de \$this, \$self, \$_POST, etc...**

Variables II

Asignación o definición

Para asignar valor a una variable en PHP, usamos el signo “=” por defecto

```
<?php

//Variable inicializada con un valor numérico
$variable1 = 1;
//Variable inicializada con una cadena de texto (string)
$variable2 = "hola";
//Variable inicializada con un valor numérico en formato decimal
$variable3 = 15.3;
//Variable inicializada con un valor booleano.
$variable4 = true;
//Variable inicializada con el valor de otra variable.
$variable5 = $variable4;

?>
```

Variables III

Otros tipos de asignaciones

Existen otras formas de asignar valores a las variables.

- Asignación directa → Mostrada anteriormente → “=”
- Referenciada → Referenciada a otra variable → “&”

```
<?php
$foo = 'Bob';           // Asigna el valor 'Bob' a $foo
$bar = &$foo;           // Referenciar $foo vía $bar.
$bar = "Mi nombre es $bar"; // Modifica $bar...
echo $bar;
echo $foo;              // $foo también se modifica.
?>
```



Preview of referencia de variables.php ×
Mi nombre es BobMi nombre es Bob

- Variable variables → Aquellas variables que toma dinámicamente → “\$\$”

```
<?php
$a = "Hola";
$b = " mundo";
//Esto es lo mismo que llamar
//a la variable $Hola
$$a = " mundo";

//Lanzamos $a más $b
echo $a . $b;
//Lanzamos $a más $$a
echo $a . $$a;
//Lanzamos $b más $$a
echo $b . $$a;
```

Ojo las variables variables, no pueden usarse con Arrays superglobales. Las variables especiales tampoco pueden ser referenciadas dinámicamente.

Preview of variables variables.php ×
Hola mundoHola mundo mundo mundo

Variables IV

Tipeado de variables

En PHP no es necesario asignar un tipo de dato a cada variable, por defecto PHP se encarga del correcto tipeado de las variables. **SÍ SE DEBE ASIGNAR EL TIPO A LAS VARIABLES, SÍ SE QUIERE GARANTIZAR EL CORRECTO USO DEL TIPO EN DETERMINADAS OCASIONES Y VARIABLES.**

```
<?php
$salto_de_linea = "<br>";
(int) $variable_entera = 5;
(boolean) $variable_booleana = false;
(float) $variable_decimal = 5.3;

//Ejemplo de tipeado de una variable
echo $variable_decimal;
echo $salto_de_linea;
echo (int)$variable_decimal;
echo $salto_de_linea;
echo (float)$variable_decimal;
echo $salto_de_linea;
echo (boolean)$variable_booleana;
```



Preview of type.php ✕

```
5.3
5
5.3
1
```

Constantes

Definición

En PHP una constante debe tener un nombre que siempre debe comenzar por una letra o carácter de subrayado, underscore “_”. Por tanto, no puede empezar por un número.

- En este caso no se usa ningún símbolo.
- Se usa la función *define(nombre, valor)* o *const nombre = valor*
- **A partir de PHP 8** el nombre es sensible a las mayúsculas y minúsculas.
- Se recomienda **no usar nombres reservados**, porque para obtener su valor deberá usarse la función *constant()*

Constantes II

Ejemplos

```
<?php

// Simple scalar value
2 usages
const CONSTANT = 'Hello World';

echo CONSTANT;

// Scalar expression
1 usage
const ANOTHER_CONST = CONSTANT . ' Goodbye World';
echo ANOTHER_CONST;

1 usage
const ANIMALS = array('dog', 'cat', 'bird');
echo ANIMALS[1]; // outputs "cat"

// Constant arrays
define('ANIMALS_ARRAY', array(
    'dog',
    'cat',
    'bird'
));
echo ANIMALS_ARRAY[1]; // outputs "cat"

?>
```



Hello World



Hello World; Goodbye World



catcat

Operadores

Precedencia

Debemos de tener muy claro que operadores son más prioritarios o tienen mayor precedencia que otros. Porque podemos obtener valores no deseados si obviarnos esta precedencia.

*1 + 5 * 3 nos devuelve 16 y no 18*

Porque el operador “*” tiene mayor precedencia que el “+”.

Recomendamos revisar la documentación oficial de php:

<https://www.php.net/manual/es/language.operators.precedence.php>

Operadores II

Operador de asignación

Es el operador más básico **“=”**, se encarga de **asignar** un valor a la expresión de la izquierda con la expresión de la derecha. Es decir “Se define como...”

- **IMPORTANTE**, no significa “igual a”.
- Se puede combinar con otros operadores, se verá más adelante.

```
<?php

//Variable inicializada con un valor numérico
$variable1 = 1;
//Variable inicializada con una cadena de texto (string)
$variable2 = "hola";
//Variable inicializada con un valor numérico en formato decimal
$variable3 = 15.3;
//Variable inicializada con un valor booleano.
$variable4 = true;
//Variable inicializada con el valor de otra variable.
$variable5 = $variable4;

?>
```

Operadores III

Operadores aritméticos o matemáticos

Procesan acciones básicas sobre datos, en este caso operaciones aritméticas o matemáticas. Igual que en matemáticas.

- Suma → “+”
- Resta → “-”
- Multiplicación → “*”
- División entera → “/”
- Resto → “%”
- Potencia → “**”

```
<?php
$x = 5;
$y = 3;

echo $x + $y; //5+3 = 8
echo $x - $y; //5-3 = 2
echo $x * $y; //5 multiplicado 3 = 15
echo $x / $y; //5 dividido 3 = 1,66666
echo $x % $y; //5 dividido 3 -> resto 2
echo $x ** $y; // 5 elevado a 3 = 125
?>
```



```
8
2
15
1.666666666666667
2
125
```

Operadores IV

Operadores aritméticos o matemáticos II

Operadores que simplifican cálculos al actuar sobre una misma variable.

- Suma → “+=”
- Resta → “-=”
- Multiplicación → “*=”
- División entera → “/=”
- Resto → “%=”
- Potencia → “**=”
- Identidad → “+ \$variable”
- Negación → “- \$variable”

```
<?php
$x = 5;

echo $x += 3; //5+3 = 8
echo $x -= 3; //8-3 = 5
echo $x *= 3; //5 multiplicado 3 = 15
echo $x /= 3; //15 dividido 3 = 5
echo $x %= 3; //5 dividido 3 -> resto 2
echo $x **= 3; //2 elevado a 3 = 8
echo +$x; //Devuelve el valor en formato número
echo -$x; //Devuelve el valor número negado.
?>
```



8
5
15
5
2
8
8
-8

Operadores V

Operadores incremento o decremento

Procesan acciones básicas sobre datos, en este caso operaciones de incrementar o decrementar.

- ++
- --

La posición del operador es importante, porque puede implicar la acción antes de la realización de otra o no.

```
<?php
$x = 3;
//$resultado vale 3 al igual que $x
echo $resultado = $x;
echo $x;

//$resultado vale 4 al igual que $x
echo $resultado = ++$x;
echo $x;

//$resultado vale 4 y $x que 5
echo $resultado = $x++;
echo $x;

//$resultado vale 4 al igual que $x
echo $resultado = --$x;
echo $x;

//$resultado vale 4 y $x que 3
echo $resultado = $x--;
echo $x;
```



3
3
4
4
4
5
4
4
4
3

Operadores VI

Operadores de cadenas de texto

Procesan acciones básicas sobre datos, en este caso operaciones sobre cadenas de texto (string) o caracteres.

- " " → Asignación de carácter o cadena de texto.
- . → Concatenar cadenas de texto o carácter.
- .= → Concatenar el contenido de la cadena de texto o carácter con lo asignado y lo guarda en la cadena de texto o carácter otra vez.

```
<?php
$cadena = 'Hola';
$caracter = 'a';

//Nos devuelve -> aa
echo $resultado = $caracter . $caracter;
//Nos devuelve -> Hola cómo va?
echo $resultado = $cadena . 'cómo va?';
//Nos devuelve -> Hola cómo va? Bien, gracias.
echo $resultado .= 'Bien, gracias.';
```



```
aa
Hola cómo va?
Hola cómo va? Bien, gracias.
```

Operadores VII

Operadores lógicos

Se utilizan para procesar valores lógicos de verdadero o falso o sus valores equivalentes.

Este tipo de operadores analizan datos de tipo bool o booleanos y sirven para comparar o procesar los estados de los datos en determinados momentos.

- and → Operación AND (y) → true si ambos son true.
- (&&) → Operación AND (y)
- or → Operación OR (o inclusivo) → true si cualquiera es true.
- (||) → Operación OR (o inclusivo)
- xor → Operación Xor (o exclusivo) → true si alguno es true, pero no ambos a la vez
- ! → Operador not (no) → true si es falso

Operadores VIII

Operadores lógicos II

```
<?php

$a = true;
$b = false;
$c = 1; //Se considera true
$d = 0; //Se considera false

//Muestra 1 que es equivalente a true
echo ($a and $a);
//No muestra nada porque es false y es equivalente a 0
echo ($a && $b);
//Nos muestra 1 porque tenemos un valor 1
echo ($c or $d);
//No muestra nada porque ambos son false
echo ($d || $b);
//Nos muestra 1 porque uno es 1 y el otro no.
echo $a xor $d;
//No muestra nada porque ambos son 1
echo $a xor $c;
//Nos muestra 1 porque es la negación del false
echo !$b;
```



1
1
1
1

Operadores IX

Operadores de comparación

Se utilizan para comparar dos valores.

Es interesante tener en cuenta el resultado de la comparación de los tipos en la tabla del siguiente link:

<https://www.php.net/manual/es/types.comparisons.php>

- `==` → igual → true si ambos son iguales. Después de evaluar los tipos.
- `===` → idéntico → true si ambos son iguales y del mismo tipo.
- `!= o <>` → Diferente → true si ambos no son iguales. Después de evaluar los tipos.
- `!==` → No idéntico → true si ambos no son iguales o si no son del mismo tipo

Operadores X

Operadores de comparación II

- < → menor que → true si es estrictamente menor.
- <= → menor o igual que → true si es menor o igual.
- > → mayor que → true si es estrictamente mayor.
- >= → mayor o igual que → true si es mayor o igual.

DISPONIBLES A PARTIR DE PHP 7

- <=> → spaceship → -1 si el de la izquierda es menor, 0 si son iguales, y 1 si el de la izquierda es mayor.
- ?? → fusión de null → El primer el primer operador sino es nulo, si no devuelve el segundo.

Operadores XI

Operadores de comparación III

```
<?php

//igual que
if (true == false){
    echo "verdadero<br>";
}else{
    echo "falso<br>";
}

//mayor que
if(25 > 26){
    echo "verdadero<br>";
}else{
    echo "falso<br>";
}

//menor o igual que
if(2<= 3){
    echo "verdadero<br>";
}else{
    echo "falso<br>";
}

//Nave espacial Spaceship
echo 25 <=> 6;

//fusión de null
$x = null;
$y = 25;
echo $x??$y;
```



```
falso
falso
verdadero
1
25
```

Operadores XII

Operadores de arrays

Se utilizan para comparar dos arrays.

- + → unión → realiza la unión de ambos arrays
- == → igualdad → true si ambos tienen las mismas parejas de clave/valor.
- === → identidad → true si ambos tienen las mismas parejas de clave/valor en el mismo orden.
- != o <> → Desigualdad → true si ambos no son iguales.
- !== → No identidad → true si ambos no son idénticos.

```
<?php
$a = array("a"=>"manzana", "b"=>"pera");
$b = array("a"=>"plantano", "b"=>"fresa", "c"=>"sandia");
//La unión coge los datos del array de la derecha y añade los de la izquierda
//ignorando los de la izquierda si son iguales a los de la derecha
var_dump($c = $a+$b);
```



```
array(3) { ["a"]=> string(7) "manzana" ["b"]=> string(4) "pera" ["c"]=> string(6) "sandia" }
```

Operadores XIII

Operador de tipo

Se puede comprobar si una variable esta instanciada a una determinada clase.

- instanceof → devuelve verdadero si la variable esta instanciada

```
<?php

2 usages
class MyClass{

}

1 usage
class OtraClase
{

}

$a = new MyClass();

var_dump( value: $a instanceof MyClass);
var_dump( value: $a instanceof OtraClase);
?>
```



```
bool(true)
bool(false)
```

Operadores XIV

Operador de ejecución

En PHP también se puede ejecutar las comillas invertidas (` `) de forma que ejecutarán en la Shell como si se tratara de un comando. Equivale a usar la función `shell_exec()`

```
<?php
$host = 'www.google.com';
echo `ping -n 3 {$host}`;
?>
```



Haciendo ping a www.google.com [142.250.178.164] con 32 bytes de datos: Respuesta desde 142.250.178.164: bytes=32 tiempo=9ms TTL=116 Respuesta desde 142.250.178.164: bytes=32 tiempo=8ms TTL=116 Respuesta desde 142.250.178.164: bytes=32 tiempo=8ms TTL=116 Estadísticas de ping para 142.250.178.164: Paquetes: enviados = 3, recibidos = 3, perdidos = 0 (0% perdidos), Tiempos aproximados de ida y vuelta en milisegundos: Mínimo = 8ms, Máximo = 9ms, Media = 8ms

Referencias

Webgrafía

Todos los ejemplos se han sacado de la documentación oficial de PHP

- <https://www.php.net/>

Bibliografía

Existen muchos libros de PHP, pero se recomienda:

Lopez, A. (2023). *Learning PHP 7*. Packt Publishing.