

# **Índice de contenidos**

## Lenguaje PHP

Definición

Etiquetas PHP

Instrucciones PHP

Ejecución de un script en PHP

Variables

Ámbito de las variables

Variables superglobales

Constantes

Visualización

echo y print

print\_r

var\_dump()

Palabras reservadas

Operadores

Asignación

Aritméticos

De comparación

De incremento

Lógicos

Cadenas de texto

Tipos de datos

Estructuras de control

Condicionales:

if – elseif – else

switch

Bucles:

while

do-while

for

foreach ( para recorrer arrays)

foreach (\$nombre\_array as \$valor)

foreach (\$nombre\_array as \$clave=>\$valor)

Require e include

require y require\_once

include e include\_once

Funciones

# Lenguaje PHP

---

## Definición

---

PHP (Hipertext PreProcessor) es un lenguaje interpretado de alto nivel, embebido en documentos HTML y ejecutado en el lado servidor.

PHP es ampliamente utilizado, gratuito y permite construir páginas dinámicas de forma sencilla y rápida

## Etiquetas PHP

---

PHP se puede escribir dentro de un documento HTML, y como en cualquier otro lenguaje embebido, necesitamos delimitar el código PHP con etiquetas.

Estilo XML: `<?php ..... ?>` o bien `<? ..... ?>`



```
1 <!DOCTYPE html>
2 <html>
3   <head><title> Mi primer php </title></head>
4   <body>
5     <?php
6       echo "Mi primer php";
7     ?>
8   </body>
9 </html>
```

**Otros modos obsoletos desde PHP 7.0:**

Estilo Script: `<script language="php"> .....</script>`

Estilo ASP: `<% ..... %>`

## Instrucciones PHP

---

Una instrucción está delimitada por el símbolo punto y coma: “;”

Un fragmento de código PHP está compuesto por una o varias instrucciones.

Los comentarios a una instrucción se hacen al estilo de C++ o Java:

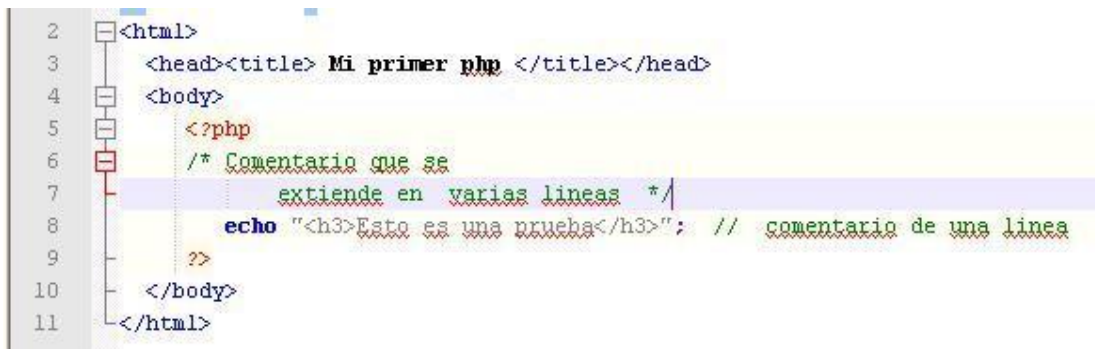
// para comentarios de una línea

/\*

para comentarios de un párrafo completo

podemos utilizar esta nomenclatura

\*/



```
2 <html>
3 <head><title> Mi primer php </title></head>
4 <body>
5 <?php
6 /* Comentario que se
7    extiende en varias lineas */
8    echo "<h3>Esto es una prueba</h3>"; // comentario de una linea
9    ??
10 </body>
11 </html>
```

Los comentarios de párrafo no deben anidarse

## Ejecución de un script en PHP

---

Pasos necesarios para ejecutar un archivo php:

1. Los archivos deben tener la extensión ".php"
2. El servidor Web (Apache) con el módulo para PHP debe estar iniciado
3. Los archivos deben desplegarse en el directorio raíz de archivos del sitio Web: "**C:/xampp/htdocs**". Este directorio puede cambiarse mediante la directiva DocumentRoot, que se encuentra en el archivo de configuración C:\xampp\apache\conf\httpd.conf

4. En el cliente web, es decir, en la barra de direcciones del navegador, debemos escribir la ruta al archivo mediante el protocolo **http**.

Ejemplos:

- Para pruebas locales <http://localhostcurso/prueba.php>
- Para pruebas en remoto se debe indicar la dirección IP o el nombre de dominio si es un dominio registrado, como <http://www.elsitio.es/cursoPHP/prueba.php>

Código de ejemplo:

```
<!DOCTYPE html>
<html>
<head>
    <title>Ejemplo PHP</title>
</head>
<body>
    <?php
        echo "Esto es una prueba. Hoy es " . date("d/m/Y");
    ?>
</body>
</html>
```

### **Actividad:**

Copia el ejemplo anterior en un archivo llamado prueba.php y guárdalo en el directorio C:\xampp\htdocs\tema2\

Arranca el servidor Apache y escribe: `http://localhost/tema2/prueba.php` en el navegador. Deberás obtener en pantalla el siguiente mensaje:

Esto es una prueba. Hoy es 11/11/1111

(11/11/1111 representa la fecha de hoy en formato día/mes/año)

## Variables

---

Los nombres de variable comienzan con el signo \$ y son sensibles a mayúsculas y minúsculas. El nombre de la variable debe continuar por una letra o guión bajo, seguido de cualquier cantidad de letras, números y guiones.

```
1 <?php
2 $nombre1 = "Luis"; // variable de tipo string
3 $nombre2 = "Maria"; // variable de tipo string
4 $nivel = 24; // variable de tipo integer
5 $profe = TRUE; // variable de tipo boolean
6 $_sueldo = 'poco'; // variable de tipo string
7
8 echo "$nombre1, $nombre2, $nivel, $profe, $_sueldo";
9 ?>
```

PHP es un lenguaje **débilmente** tipado, es decir, no es necesario definir el tipo antes de utilizar una variable. Las variables se declaran cuando se le asigna un valor.

Las variables también pueden declararse **por referencia** a otra variable. Consiste en establecer un puntero, usando el símbolo & (ampersand) al comienzo de la variable.

## Ámbito de las variables

---

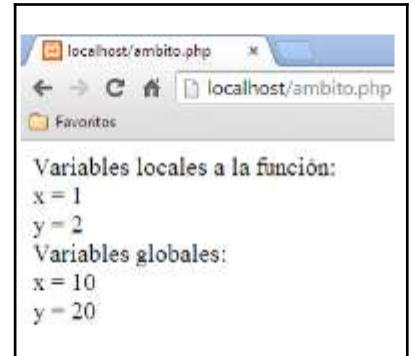
En PHP las variables pueden ser declaradas en cualquier lugar del código.

El ámbito de una variable es la parte del código donde puede ser usada, PHP tiene los siguientes ámbitos para las variables:

**Local:** Una variable declarada dentro de una función, tiene ámbito local y solo puede ser usada dentro de dicha función

**Global:** Una variable declarada fuera de toda función tiene un ámbito global y solo puede ser usada fuera de toda función.

```
1 <?php
2 $x=10; $y=20; // ámbito global
3
4 function ambito() {
5     $x=1; $y=2; // ámbito local a la función
6
7     echo "Variables locales a la función: <br>";
8     echo "x = $x <br>"; echo "y = $y <br>";
9 }
10 ambito();
11 echo "Variables globales: <br>";
12 echo "x = $x <br>"; echo "y = $y <br>";
13 ?>
```

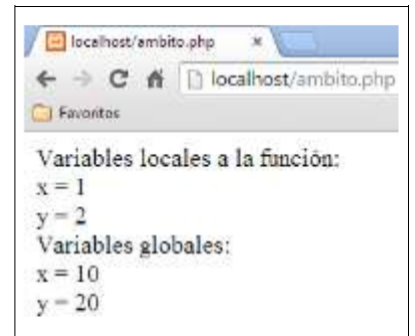


localhost/ambito.php

Variables locales a la función:  
x = 1  
y = 2  
Variables globales:  
x = 10  
y = 20

Una variable global puede ser usada dentro de una función, indicándolo previamente con el modificador "global". El siguiente ejemplo proporciona el mismo resultado que el código anterior:

```
1 <?php
2 $x=10; $y=20; // ámbito global
3
4 function ambito() {
5     $x=1; $y=2; // ámbito local a la función
6
7     echo "Variables locales a la función: <br>";
8     echo "x = $x <br>"; echo "y = $y <br>";
9     global $x,$y;
10    echo "Variables globales: <br>";
11    echo "x = $x <br>"; echo "y = $y <br>";
12 }
13 ambito();
14 ?>
```



localhost/ambito.php

Variables locales a la función:  
x = 1  
y = 2  
Variables globales:  
x = 10  
y = 20

## Variables superglobales

---

Se trata de un conjunto de variables predefinidas y accesibles desde cualquier ámbito (funciones, clases o archivos).

Los tipos de variables superglobales en PHP son:

**\$GLOBALS:** contiene todas las variables globales definidas en el script

**\$\_SERVER:** contiene las variables del servidor Web (cabeceras, rutas, etc.)

**\$\_REQUEST:** contiene los datos enviados en un formulario HTML

**\$\_POST:** contiene los datos enviados en un formulario HTML con  
method="post"

**\$\_GET :** contiene los datos enviados en un formulario HTML con method="get"

**\$\_FILES:** contiene variables proporcionadas por medio de ficheros

**\$\_ENV:** contiene las variables proporcionadas por el entorno

**\$\_COOKIE:** contiene las variables proporcionadas por cookies

**\$\_SESSION:** contiene las variables registradas en la sesión del script

PHP almacena todas las variables globales en un array llamado:

**\$GLOBALS[nombre\_variable].**

```
1 <?php
2 $x=10;    $y=20;    // variables globales
3
4 function ambito() {
5     $x=1;    $y=2;    // variables locales
6     echo "Variables locales a la función: <br>";
7     echo "x = $x <br>";    echo "y = $y <br>";
8     echo "Variables globales: <br>";
9     echo "x = $GLOBALS[x]    y = $GLOBALS[y] <br>";
10 }
11 ambito();
12 >?
13
```

El índice del array \$GLOBALS[], es el nombre de la variable global sin “\$”. El resultado del script es el mismo que en el ejemplo anterior con el modificador “global”.

## Constantes

---

Las constantes **no** van precedidas del símbolo \$

El nombre de la constante sigue las mismas reglas que las variables, es decir, deben comenzar por una letra o un guión bajo.

Las constantes pueden definirse mediante la función define(), cuya sintaxis simplificada es la siguiente: int **define** (string *nombre*, mixed *valor*)

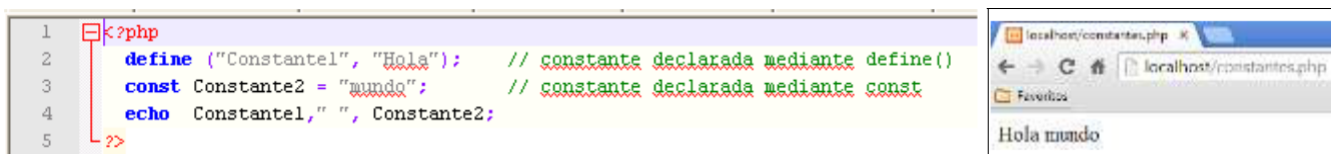
A partir de PHP 5.3.0 pueden definirse mediante la palabra reservada **const**

Las constantes pueden ser definidas y accedidas desde cualquier sitio

Las constantes no pueden ser eliminadas o redefinidas

Solo pueden contener valores escalares: **string**, **integer**, **float** y **boolean**

Para conocer el valor de una constante basta con utilizar su nombre



Para conocer el valor de una constante, cuyo nombre se conoce en tiempo de ejecución, también se puede utilizar la función constant(), de la siguiente manera:



Por ejemplo:

```
define("MAXSIZE", 100);

echo MAXSIZE;
echo constant("MAXSIZE"); // hace lo mismo que la línea anterior
```

## Visualización

---

### echo y print

Son construcciones del lenguaje, no funciones, por lo que se deben utilizar **sin paréntesis**, soportan etiquetas HTML y se aplican sobre cadenas de caracteres.

La sintaxis es:

```
void echo cadena1, cadena2, ... , cadenaN
int print cadena
```

Si la cadena tiene comillas simples se visualiza tal cual

Si la cadena tiene comillas dobles, las variables se expanden, es decir, son sustituidas por su valor (excepto si usamos el carácter “\”)

No se pueden usar arrays, funciones, ni objetos directamente dentro de una cadena, las alternativas son concatenar o utilizar llaves {}

```
1 <?php
2 $x=10;
3 $mensaje ="La variable";
4 $nombres = array ('Luis', 'Ana', 'Jorge');
5
6 echo "<u>Visualización con echo</u>: <br>";
7 echo "\$x = $x <br>";
8 echo "$mensaje", ' $x vale: ', "$x", "<br>";
9 echo "$nombres[0] $nombres[1] $nombres[2]";
10
```

localhost/manual/echo.php

Visualización con echo:  
Sx = 10  
La variable Sx vale: 10  
Luis Ana Jorge

```
1 <?php
2 $x=10;
3 $mensaje ="La variable";
4 $nombres =array ('Luis', 'Ana', 'Jorge');
5
6 print "<u>Visualización con print</u>: <br>";
7 print "\$x = $x <br>";
8 print "$mensaje", ' $x vale: ', "$x", "<br>";
9 print "$nombres[0]" . " " $nombres[1]" . " " $nombres[2]";
10
```

localhost/manual/print.php

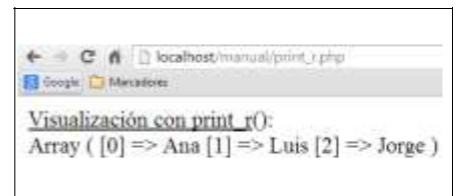
Visualización con print:  
Sx = 10  
La variable Sx vale: 10  
Luis Ana Jorge

## print\_r

Visualiza el contenido de una variable string, integer o float.

Si es un array, los valores se presentan de forma ordenada primero los índices o claves y después los elementos.

```
1 <?php
2     $nombres= array('Ana', 'Luis', 'Jorge');
3
4     echo "Visualización con print_r(): <br>";
5     print_r ($nombres);
6     ?>
```

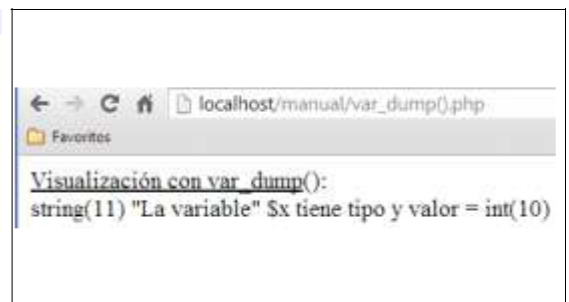


Visualización con print\_r():  
Array ( [0] => Ana [1] => Luis [2] => Jorge )

## var\_dump()

Es una función que visualiza el tipo y valor de una variable

```
1 <?php
2     $x=10;
3     $mensaje = 'La variable';
4
5     echo "<u>Visualización con var_dump</u>(): <br>";
6     var_dump ($mensaje);
7     echo '$x tiene tipo y valor = ';
8     var_dump ($x);
9
10     ?>
```



Visualización con var\_dump():  
string(11) "La variable" \$x tiene tipo y valor = int(10)

## Palabras reservadas

Las palabras reservadas o construcciones del lenguaje PHP, no deben confundirse con funciones y tienen las siguientes características:

1. No se pueden usar como constantes, nombres de clase, nombres de funciones o de métodos
2. Se pueden usar como nombres de variables, pero no se recomienda

3. Con las construcciones del lenguaje, en general, no se requiere el uso de paréntesis
4. Las funciones se simplifican hasta obtener construcciones del lenguaje

Listado de palabras reservadas:

`__halt_compiler()`, `abstract`, `and`, `array()`, `ask`, `break`, `callable`, `case`, `catch`, `class`, `clone`, `const`, `continue`, `declare`, `default`, `die()`, `do`, `echo`, `else`, `elseif`, `empty()`, `enddeclare`, `endfor`, `endforeach`, `endif`, `endswitch`, `endwhile`, `eval()`, `exit()`, `extends`, `final`, `finally`, `for`, `foreach`, `function`, `global`, `goto`, `if`, `implements`, `include`, `include_once`, `instanceof`, `insteadof`, `interface`, `isset()`, `list()`, `namespace`, `new`, `or`, `print`, `private`, `protected`, `public`, `require`, `require_once`, `return`, `static`, `switch`, `throw`, `trait`, `try`, `unset()`, `use`, `var`, `while`, `xor`, `yield`

Para más detalles: <http://www.php.net/manual/es/reserved.keywords.php>

## Operadores

---

Para ejemplos en vivo podemos visitar

[https://www.w3schools.com/php/php\\_operators.asp](https://www.w3schools.com/php/php_operators.asp)

### Asignación

Operador	Nombre	Resultado
<code>\$a = 7;</code>	Asignación	

### Aritméticos

<code>\$a + \$b</code>	Suma	
<code>\$a - \$b</code>	Resta	
<code>\$a * \$b</code>	Multiplicación	
<code>\$a / \$b</code>	División	

\$a % \$b	Módulo	Resto de la división entera
- \$a	Negación	El opuesto

## De comparación

\$a == \$b	Comparación	Cierto si el valor de los operandos es igual
\$a === \$b	Identidad	Cierto si el valor y el tipo de los operandos es igual
\$a != \$b	Distinto	Cierto si el valor de \$a es distinto al valor de \$b
\$a !== \$b	No identidad	Cierto si el valor o el tipo de \$a es distinto de \$b
\$a < \$b	Menor que	Cierto si el valor de \$a es menor que el valor de \$b
\$a <= \$b	Menor o igual	Cierto si el valor de \$a es menor o igual
\$a > \$b	Mayor que	Cierto si el valor de \$a es mayor que el valor de \$b
\$a >= \$b	Mayor o igual	Cierto si el valor de \$a es mayor o igual
Si se compara un entero con una cadena, la cadena es convertida a número.		

## De incremento

\$a++	Post-incremento	Devuelve \$a y después lo incrementa en 1
++\$a	Pre-incremento	Incrementa \$a en 1 y devuelve el nuevo valor

\$a--	Post-decremento	Devuelve \$a y después lo decrementa en 1
--\$a	Pre-decremento	Decrementa \$a en 1 y devuelve el nuevo valor

## Lógicos

\$a and \$b	Y	Cierto si \$a y \$b son ciertos
\$a or \$b	O	Cierto si \$a o \$b son ciertos
\$a xor \$b	O excluyente	Cierto si \$a o \$b son ciertos pero no ambos
!\$a	NO	Cierto si \$a es falso

## Cadenas de texto

El operador punto “.” sirve para concatenar cadenas de texto.

## Tipos de datos

---

PHP soporta 8 tipos de datos primitivos:

1. **Boolean** Para declarar un literal booleano se utilizan las palabras reservadas “true/false”.
2. **Integer** Un entero es un número sin decimales, no puede tener coma. Puede ser positivo o negativo y se puede expresar en sistema decimal, hexadecimal (0x) u octal (0).
3. **Float** Es un número real expresado con decimales o en notación exponencial.
4. **String** Es una cadena de caracteres (bytes) que se puede declarar con comillas simples, comillas dobles o mediante la sintaxis “heredoc”. Las comillas dobles expanden el contenido de una variable. Las cadenas se pueden concatenar con el operador punto.

5. **Array** Es una variable compuesta que almacena variables simples. PHP soporta arrays indexados y arrays asociativos.

Se pueden definir con la función **list()**, con el constructor **array()** o bien asignar el valor a cada elemento del array de forma explícita usando corchetes.

6. **Object** Un objeto es un tipo de datos que engloba variables y funciones a la vez.

7. **Resource** Es una variable especial que apunta a un recurso externo

8. **NULL** Se utiliza para indicar que una variable no tiene valor  
(No confundir con el carácter fin de cadena '\0' )

## Ejemplos

```
1 <?php
2 $logico1= true;  $logico2= false;
3 $entero = -10;
4 $real = 1.5e3;
5 $cadena = 'Mi cadena';
6 $frase1[0] = 'Hola'; $frase1[1] = 'mundo'; // por asignación
7 $frase2 = array ('Adiós', 'mundo', 'cruel'); //usando array()
8
9 echo "logico1 = $logico1, logico2 = $logico2"; echo "<br>";
10 echo "entero = $entero"; echo "<br>";
11 echo "real = $real"; echo "<br>";
12 echo "cadena = $cadena"; echo "<br>";
13 echo "frase1[0] y frase2[0] = $frase1[0] y $frase2[0]";
14 ?>
```

Favoritos

logico1 = 1, logico2 =  
entero = -10  
real = 1500  
cadena = Mi cadena  
frase1[0] y frase2[0] = Hola y Adiós

\$frase1 y \$frase2 son arrays indexados (tienen un índice entero).

Para acceder al elemento n-ésimo del array indexado \$v, se utiliza: \$v[n-1] (ya que empieza en la posición 0)

Un array asociativo es un array que en lugar de utilizar un índice de tipo entero, utiliza una clave de tipo cadena de caracteres.

Como en el ejemplo anterior, se pueden declarar mediante corchetes o con la función array().

Para visualizar con echo cada elemento del array es preciso concatenar las cadenas con el operador punto “.”

## Estructuras de control

---

### Condicionales:

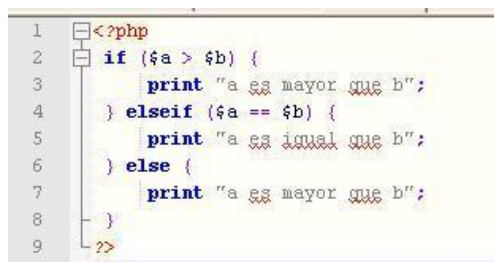
#### if – elseif – else

```
if ( expr ) { instr; instr; ... ; }
```

```
[ elseif (expr) { instr ; instr ; .. }
```

```
elseif (expr) { instr ; instr ; .. }
```

```
else { instr ; instr ; .. }]
```



```
1 <?php
2 if ($a > $b) {
3     print "a es mayor que b";
4 } elseif ($a == $b) {
5     print "a es igual que b";
6 } else {
7     print "a es mayor que b";
8 }
9 ?>
```

#### switch

```
switch ( expr ) {
```

```
case valor1: instr ; instr ; ..
```

```
[ case valor2: instr ; instr ; ..
```

```
default : instr ; instr ; .. ]
```

```
}
```

```

1 <?php
2 switch ($i) {
3     case 0:
4     case 1:
5     case 2:
6         print "¡ es menor que 3, pero no negativo";
7         break;
8     case 3:
9         print "¡ es 3";
10        break;
11    default: echo " ¡ es negativo o mayor de 3";
12 }
13 ?>

```

## Bucles:

### while

```

while ( expr ) {
    instr ; instr; ... ;
}

```

```

1 <?php
2 $i = 0;
3 while ($i<10) {
4     echo "contador = $i";
5     $i++;
6 }
7 ?>

```

### do-while

```

do
{
    instr ; instr; ...
} while (expr) ;

```

```

1 <?php
2 $i = 0;
3 do {
4     echo "contador = $i";
5     $i++;
6 } while ($i>3);
7 ?>

```

### for

```

for (expr1 ; expr2 ; expr3)
{ instr ; instr; ... ;}

```

```

1 <?php
2 /* mostrar n°s del 1 al 10 */
3 for ($i = 1; $i <= 10; $i++) {
4     print $i;
5 }
6 //
7 for ($i = 1; ;$i++) {
8     if ($i > 10) {
9         break;
10    }
11    print $i;
12 }
13 //
14 $i = 1;
15 for (;;) {
16     if ($i > 10) {
17         break;
18     }
19     print $i;
20     $i++;
21 }
22 ?>

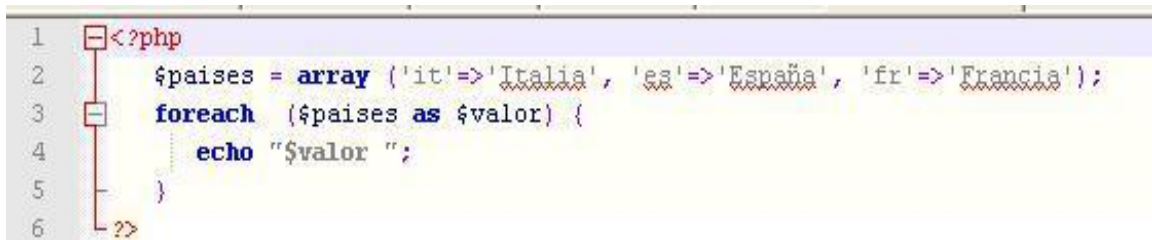
```



**foreach ( para recorrer arrays)**

**foreach (\$nombre\_array as \$valor)**

```
{  
    sentencias;  
}
```



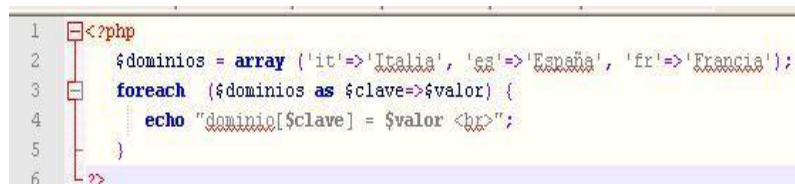
```
1 <?php  
2 $países = array ('it'=>'Italia', 'es'=>'España', 'fr'=>'Francia');  
3 foreach ($países as $valor) {  
4     echo "$valor ";  
5 }  
6 ?>
```

Recorre array indicado en \$nombre\_array. En cada iteración el elemento actual del array se guarda en \$valor y se incrementa el puntero interno del array. En el ejemplo anterior escribirá:

Italia España Francia

**foreach (\$nombre\_array as \$clave=>\$valor)**

```
{  
    sentencias;  
}
```



```
1 <?php  
2 $dominios = array ('it'=>'Italia', 'es'=>'España', 'fr'=>'Francia');  
3 foreach ($dominios as $clave=>$valor) {  
4     echo "dominio[$clave] = $valor <br>";  
5 }  
6 ?>
```

Recorre array indicado en \$nombre\_array. En cada iteración el elemento actual del array se guarda en \$valor , el índice en \$clave y se incrementa el puntero interno del array.

```
dominio[it] = Italia  
dominio[es] = España  
dominio[fr] = Francia
```

## Require e include

---

### require y require\_once

Sirve para insertar en nuestro documento y en la posición exacta donde está [require](#), el código contenido en un archivo externo, antes de ser ejecutado por el servidor.

En caso de no encontrar el archivo especificado, se produce un FATAL ERROR que interrumpe la ejecución. ( require\_once solo inserta la primera vez aparece)

### include e include\_once

Sirve para pegar en nuestro documento y en la posición exacta donde está [include](#), el código contenido en un archivo externo, antes de ser ejecutado por el servidor.

En caso de no encontrar el archivo especificado se envía un WARNING pero continúa la ejecución. (include\_once solo inserta la primera vez que aparece)

#### header.php

```
<?php
    echo '<nav>';
    echo '<ul>';
    echo '<li><a href="contacto.php">Contacto</a></li>';
    echo '<li><a href="nosotros.php">Sobre nosotros</a></li>';
    echo '</ul>';
    echo '</nav>';
?>
```

#### menu.html

```
<nav>
    <ul class="menu">
        <li><a href="#">Inicio</a></li>
        <li><a href="#">Productos</a></li>
        <li><a href="#">Servicios</a></li>
        <li><a href="#">Contacto</a></li>
        <li><a href="#">Blog</a></li>
    </ul>
</nav>
```

#### pagina.php

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
</head>
<body>
    <?php require 'header.php'; ?> <!-- Obligatoria la cabecera desde header.php -->
    <?php include 'menu.html'; ?> <!-- Incluye el menú desde menu.html -->
</body>
</html>
```

# Funciones

---

Para declarar funciones de usuario se usa la siguiente sintaxis:

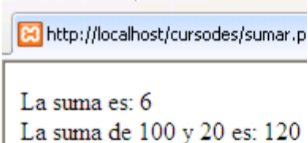
```
function nombre_funcion ($parametro1, ..., $parametroN) {  
    sentencias;  
}
```

- Las funciones no se ejecutan inmediatamente al cargar la página php en el servidor, sólo cuando se llaman.
- Las funciones pueden recibir varios valores mediante los parámetros.
- PHP soporta paso de parámetros por **valor**, por **referencia** y por **defecto**.
- Las funciones pueden retornar un valor mediante return:

```
return $valor;
```

## Ejemplo 1: Paso de parámetros por valor

```
1 <?php  
2 function sumar ($a1, $a2) {  
3     return $a1 + $a2;  
4 }  
5 $s = sumar (2, 4); // paso de parámetros por valor  
6 echo "La suma es: $s <br>";  
7 echo "La suma de 100 y 20 es: ", sumar(100,20);  
8 ?>
```



http://localhost/cursodes/sumar.p  
La suma es: 6  
La suma de 100 y 20 es: 120

## Ejemplo 2: Paso de parámetros por referencia (precediendo el argumento con & )

```

1 <?php
2 function acumular(&$a, $valor) { // param1 se pasa por referencia
3     $a = $a + $valor;
4 }
5 $acum = 1;
6 echo "\$acum = $acum <br>";
7 for ($i=1; $i<=4; $i++){
8     acumular($acum, 4);
9     echo "\$acum = $acum <br>";
10 }
11 ?>

```

http://localhost/cu

```

Sacum = 1
Sacum = 5
Sacum = 9
Sacum = 13

```

El paso de parámetros por referencia permite a una función cambiar el valor del parámetro. En el ejemplo el parámetro \$acum se modifica dentro de la función, al cambiar el valor de \$a.

### Ejemplo 3: Paso de parámetros por defecto

```

1 <?php
2 function soñar($a = 'ser rico') {
3     return "Quiero $a";
4 }
5 echo soñar('un Volvo V40 <br>');
6 echo soñar();
7 ?>

```

http://localhost/cursodes/defecto.

```

Quiero un Volvo V40
Quiero ser rico

```

Si la llamada a la función no tiene parámetros se usa el valor por defecto definido en la función.

### Ejemplo 4: Cantidad variable de parámetros

```

1 <?php
2 function escribe() {
3     for ($i=0; $i<func_num_args(); $i++) {
4         echo func_get_arg($i)."\n";
5     }
6     echo "<br>";
7 }
8 escribe('hola'); // 1 parámetro
9 escribe('adios', 'mundo', 'cruel'); // 3 pa
10 ?>

```

http://localhost/cursodes/func

```

hola
adios mundo cruel

```

las funciones PHP `func_num_args()`, `func_get_arg()` y `func_get_args()` son útiles en funciones con un número variable de parámetros.