

MANUAL DE INICIACIÓN A PHP

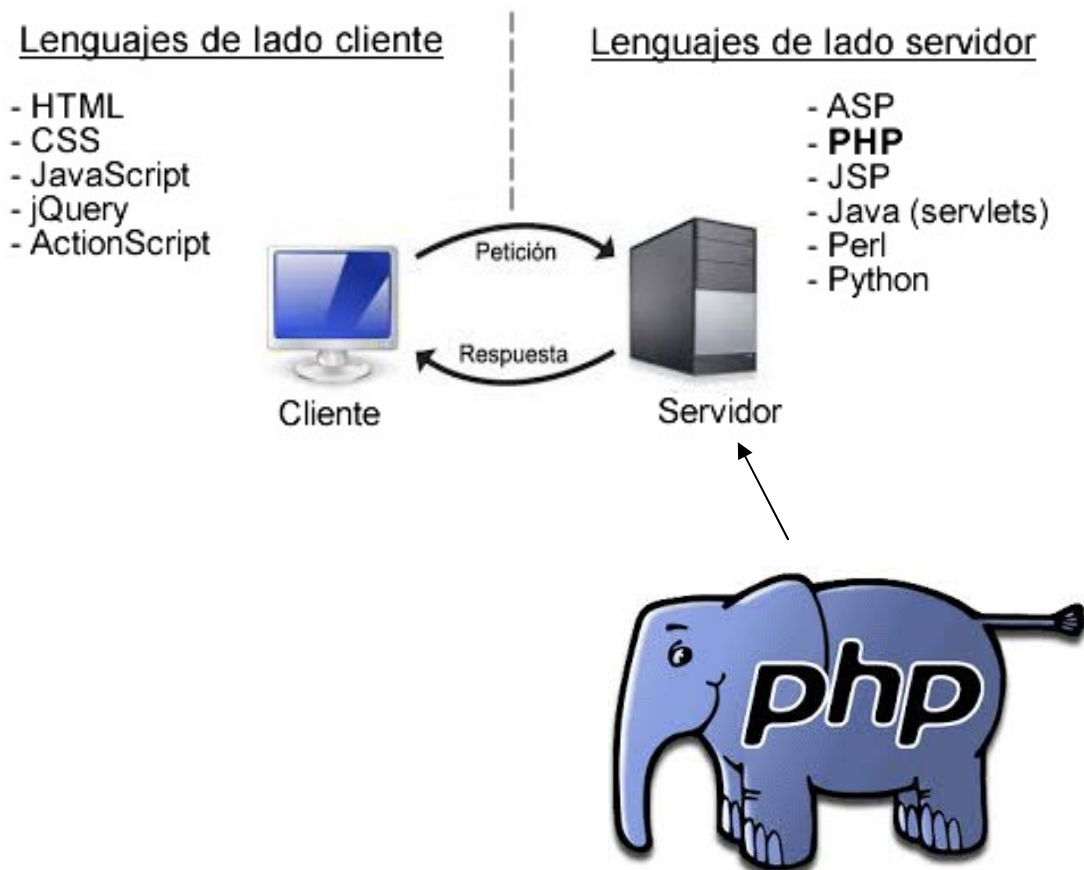
INDICE

1. DEFINICIÓN.....	1
2. ETIQUETAS PHP	2
3. INSTRUCCIONES PHP	3
4. EJECUCIÓN DE UN SCRIPT PHP.....	4
5. VARIABLES	5
6. ÁMBITO DE VARIABLES	6
7. VARIABLES SUPERGLOBALES.....	7
8. CONSTANTES	8
9. VISUALIZACIÓN.....	9
10. PALABRAS RESERVADAS.....	11
11. OPERADORES.....	12
12. TIPOS DE DATOS.....	13
13. ESTRUCTURAS DE CONTROL	15
14. FUNCIONES.....	18
15. MANEJO DE CADENAS.....	20
16. FUNCIONES PARA MANEJO DE FECHAS	23
ANEXO 1. TIPOS DE ERRORES.....	24

1. 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



2. 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 `<? ?>`
- Estilo Script: `<script language="php"></script>`
- Estilo ASP: `<% %>` (válido a partir de PHP 3.0.4)

```
<doctype html>
<html>
  <head><title> Mi primer php </title></head>
  <body>
    <?php
      echo "Mi primer php";
    ?>
  </body>
</html>
```

[primer.php](#)

3. 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 */

Los comentarios de párrafo no deben anidarse.

```
<?php
    echo "Esto es una prueba"; //Comentario de una línea
    /* echo “Esto no sale porque está comentado”; Este comentario abarca
    un párrafo completo */
?>
```

[prueba.php](#)

4. EJECUCIÓN DE UN SCRIPT PHP

Pasos necesarios para ejecutar un archivo php:

- ✓ Los archivos deben tener la extensión ".php"
- ✓ 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
- ✓ El servidor Web (Apache) con el módulo para PHP debe estar iniciado
- ✓ En el cliente web, es decir, en la barra de direcciones del navegador, debemos escribir la ruta al archivo mediante el protocolo **http**.

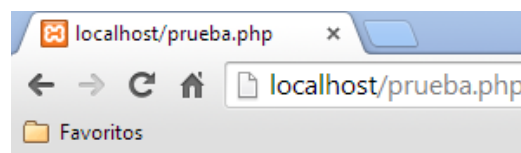
Ejemplos:

- Para pruebas locales <http://localhost/manual/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.juanmacr.es/manual/prueba.php>

Actividad:

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

Arranca el servidor Apache y escribe: <http://localhost/manual/prueba.php> en el navegador



5. 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.

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.

Por ejemplo:

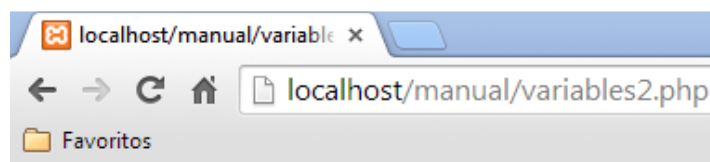
```
<?php
$nombre1 = "Juanma";      // variable de tipo string
$nombre2 = "Joserra";     // variable de tipo string
$nivel = 24;              // variable de tipo integer
$profe = TRUE;            // variable de tipo boolean
$_sueldo = 'poco';        // variable de tipo string
echo "$nombre1, $nombre2, $nivel, $profe, $_sueldo";
?>
```

[variables.php](#)

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

```
<?php
$var1 = 100;              // variable declarada por valor
$var2 = 100;              // variable declarada por valor
$var3 = &$var2;           // variable declarada por referencia a $var2
echo "Comienzo: $var1, $var2, $var3 <br>";
$var2 = 200;              // modifica el valor de $var2 y por referencia a $var3
echo "Asignar: var2 = $var2 <br>";
echo "Fin: $var1, $var2, $var3";
?>
```

[variables2.php](#)



```
Comienzo: 100, 100, 100
Asignar: var2 = 200
Fin: 100, 200, 200
```

6. ÁMBITO DE VARIABLES

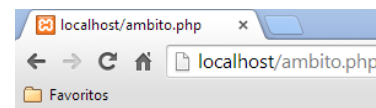
En PHP las variables pueden ser declaradas en cualquier lugar del código.

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

- **Local**
Una variables 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.

```
<?php
$x=10; $y=20;           // ámbito global

function ambito() {
    $x=1; $y=2;          // ámbito local a la función
    echo "Variables locales a la función: <br>";
    echo "x = $x <br>";   echo "y = $y <br>";
}
ambito();
echo "Variables globales: <br>";
echo "x = $x <br>"; echo "y = $y <br>";
?>
```



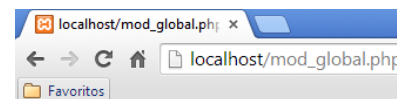
```
Variables locales a la función:
x = 1
y = 2
Variables globales:
x = 10
y = 20
```

[ambito.php](#)

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:

```
<?php
$x=10; $y=20;           // ámbito global

function ambito() {
    $x=1; $y=2;          // ámbito local a la función
    echo "Variables locales a la función: <br>";
    echo "x = $x <br>";   echo "y = $y <br>";
    global $x, $y;
    echo "Variables globales: <br>";
    echo "x = $x <br>"; echo "y = $y <br>";
}
ambito();
?>
```



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

[mod_global.php](#)

7. 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]`.

```
<?php
    $x=10; $y=20;           // variables globales

    function ambito() {
        $x=1; $y=2;         // variables locales
        echo "Variables locales a la función: <br>";
        echo "x = $x <br>";   echo "y = $y <br>";
        echo "Variables globales: <br>";
        echo "x = $GLOBALS[x] <br>";
        echo "y = $GLOBALS[y] <br>";
    }
    ambito();
?>
```

[globales.php](#)

Como se puede observar, 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".

8. 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:

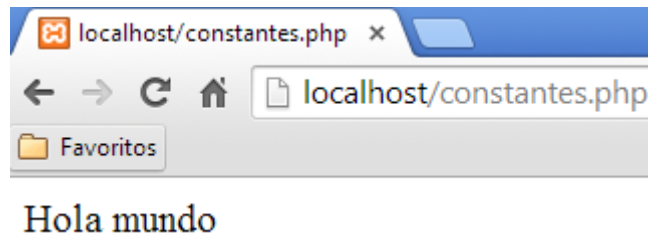
`int define (string nombre, mixed valor)`

- 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, se utiliza la función `constant()`

```
<?php
define ("Constante1", "Hola");           // constante declarada mediante define()
const Constante2 = "mundo";             // constante declarada mediante const
echo Constante1, " ", Constante2;

?>
```

[constantes.php](#)



9. VISUALIZACIÓN

9.1. 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:

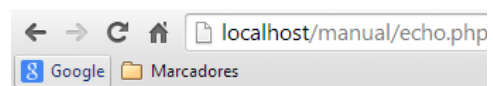
void **echo** cadena1, cadena2, ... , cadenaN

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 { }

```
<?php
    $x=10;
    $mensaje ="La variable";
    $nombres =['Luis', 'Ana', 'Jorge'];

    echo "<u>Visualización con echo</u>: <br>";
    echo "\$x = $x <br>";
    echo "$mensaje", ' $x vale: ', "$x", "<br>";
    echo "$nombres[0] $nombres[1] $nombres[2]";
?>
```



Visualización con echo:

\$x = 10

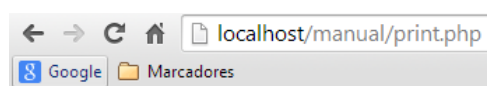
La variable \$x vale: 10

Luis Ana Jorge

[echo.php](#)

```
<?php
    $x=10;
    $mensaje ="La variable";
    $nombres =['Luis', 'Ana', 'Jorge'];

    print "<u>Visualización con print</u>: <br>";
    print "\$x = $x <br>";
    print "$mensaje". ' $x vale: '. "$x". "<br>";
    print "$nombres[0]" . " $nombres[1]" .
        " $nombres[2]";
?>
```



Visualización con print:

\$x = 10

La variable \$x vale: 10

Luis Ana Jorge

[print.php](#)

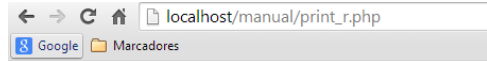
9.2. print_r

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.

```
<?php
    $nombres= ['Luis', 'Ana', 'Jorge'];

    echo "<u>Visualización con print_r</u>: <br>";
    print_r "$nombres";
?>
```



Visualización con print_r():

Array ([0] => Ana [1] => Luis [2] => Jorge)

[print_r.php](#)

9.3. var_export()

Visualiza el contenido de una variable.

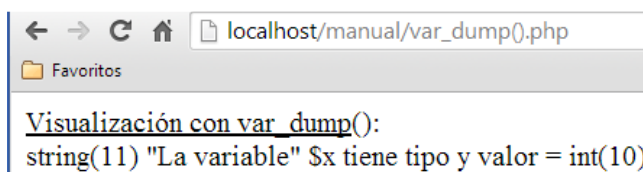
9.4. var_dump()

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

```
<?php
    $x=10;
    $mensaje ="La variable";

    echo "<u>Visualización con var_dump</u>(): <br>";
    var_dump ($mensaje);
    echo '$x tiene tipo y valor = ' ;
    var_dump ($x); echo "<br>";
?>
```

[var_dump.php](#)



10. PALABRAS RESERVADAS

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

- No se pueden usar como constantes, nombres de clase, nombres de funciones o de métodos
- Se pueden usar como nombres de variables, pero no se recomienda
- Con las construcciones del lenguaje, en general, no se requiere el uso de paréntesis
- 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>

11. OPERADORES

- Asignación**

Operador	Nombre	Resultado
\$a = 7;	Asignación	

- Aritméticos**

\$a + \$b	Suma	
\$a - \$b	Resta	
\$a * \$b	Multiplicación	
\$a / \$b	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 \$a && \$b	Y	Cierto si \$a y \$b son ciertos
\$a or \$b \$a \$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.

12. 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’)

Ejemplo 1: Boolean, integer, float, string

```
<?php
$logico1 = true;   $logico2 = false;
$entero = -10; $real = 1.5e3;
$cadena = 'Tipos de datos simples';
echo "logico1 = $logico1, logico2 = $logico2 <br>";
echo "entero = $entero <br>";
echo "real = $real <br>";
echo "cadena = $cadena <br>";
?>
```

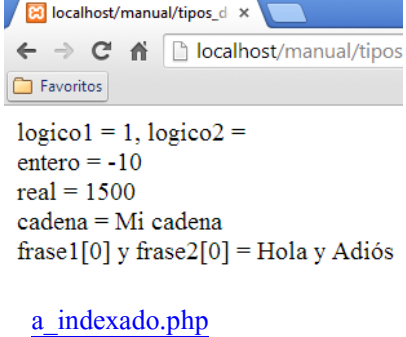
[tipos.php](#)

Ejemplo 2: Array indexado

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

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

```
<?php
$frase1[0] = 'Hola';
$frase1[1] = 'mundo';
$frase2 = array ('Adiós', 'mundo', 'cruel');
print_r ($frase1); print_r ($frase2);
echo "frase1[0] y frase2[0] = $frase1[0] y $frase2[0]";
?>
```



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

a\_indexado.php
```

Ejemplo 3: Array asociativo

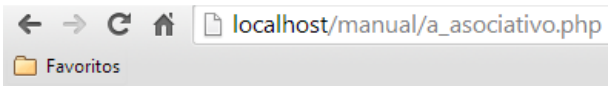
Es un array que en lugar de utilizar un índice de tipo entero, utiliza una clave de tipo cadena de caracteres.

Como antes, 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 “.”

Ejemplo:

```
<?php
$países = array ('it'=>'Italia', 'es'=>'España', 'fr'=>'Francia');
$platos ['it'] = 'Pizza';
$platos ['es'] = 'Paella';
$platos ['fr'] = 'Quiche';
echo 'En '.$países['es'].' la '.$platos['es']; echo '<br>';
echo 'En '.$países['it'].' la '.$platos['it']; echo '<br>';
echo 'En '.$países['fr'].' la '.$platos['fr']; echo '<br>';
?>
```



```
En España la Paella
En Italia la Pizza
En Francia la Quiche

a\_asociativo.php
```

13. ESTRUCTURAS DE CONTROL

- Condicionales: **if – elseif – else**

```
<?php
$nota = 5;
if ($nota < 5) {
    echo "Vuelve a intentarlo";
} elseif ($nota == 5) {
    echo "Uff";
} else {
    echo "Bien hecho!";
}
?>
```

switch

```
<?php
$genero = 'F';
switch ($genero) {
    case 'M': echo 'Masculino';
        break;
    case 'F': echo 'Femenino';
        break;
    default: echo "Neutro";
}
?>
```

- Bucles: **while**

```
<?php
$i = 0;
while ($i < 10) {
    echo "contador = $i";
    $i++;
}
?>
```

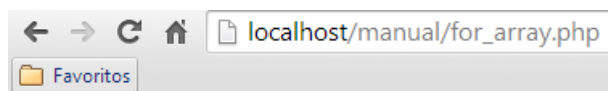
do while

```
<?php
$i = 0;
do {
    echo "contador = $i";
    $i++;
} while ($i < 10);
?>
```

- Bucle: **for** (para arrays indexados)

```
<?php
for ($i=0; $i<10; $i++) {
    echo "contador = $i";
}
?>
```

```
<?php
$países = array ('Italia', 'España', 'Francia');
$longitud = count ($países);
for ($i=0; $i<$longitud; $i++) {
    echo "país[$i] = ".$países[$i] <br>";
}
?>
```



```
país[0] = Italia
país[1] = España
país[2] = Francia
```


- Bucle: **foreach** (solo para arrays asociativos, matrices y objetos)

Hay dos formas posibles de usar el bucle foreach:

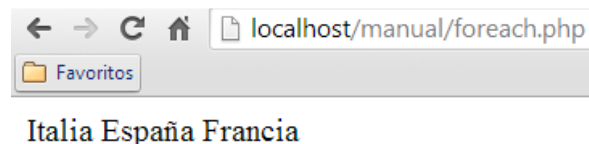
1. `foreach ($nombre_array as $valor) {`
 sentencias; `}`

Este bucle recorre el 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.

```
<?php
$países = array ('it'=>'Italia', 'es'=>'España', 'fr'=>'Francia');
foreach ($países as $valor) {
    echo "$valor <br>";
}
?>
```

[foreach1.php](#)



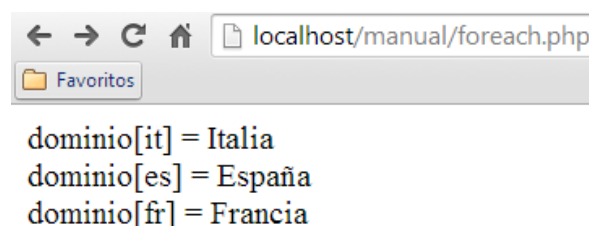
2. `foreach ($nombre_array as $clave=>$valor) {`
 sentencias; `}`

Este bucle recorre el array indicado en \$nombre_array.

En cada iteración el elemento actual del array se guarda en \$valor, la clave se guarda en \$clave y se incrementa el puntero interno del array.

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

[foreach2.php](#)



- **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)

```
<?php
    echo '<a href="/require.php">require</a> --
        <a href="/include.php">include</a> --
        <a href="/require_once.php">require_once</a> --
        <a href="/include_once.php">include_once</a> --';
?>
```

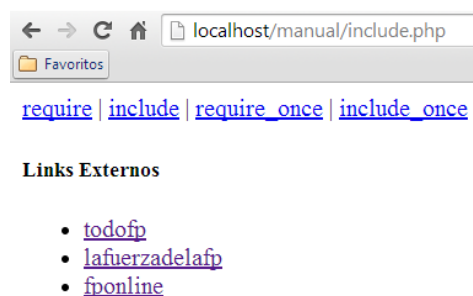
[header.php](#)

```
<div id="sectionLinks">
    <h2> Links Externos </h2>
    <ul>
        <li><a href="http://todofp.es/">todofp</a></li>
        <li><a href="http://lafuerzadelafp.es">lafuerzadelafp</a></li>
        <li><a href="http://www.educacion.gob.es/fponline/">fponline</a></li>
    </ul>
</div>
```

[menu.html](#)

```
<?php
    require 'header.php';
    include 'menu.html';
?>
```

[include.php](#)



14. FUNCIONES

Sintaxis:

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

- Las funciones no se ejecutan inmediatamente al cargar la página php en el servidor, solo 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

```
<?php
function sumar ($a1, $a2) { // paso de parámetros por valor
    return $a1 + $a2;
}
$s = sumar (2, 4);
echo "La suma es: $s";
?>
```

[sumar.php](#)

Ejemplo 2: Paso de parámetros por referencia (añadiendo "&" al argumento)

```
<?php
function acumular (&$a, $incremento) {
    $a = $a + $incremento;
}
$acum = 1;
echo "acumulador = $acum <br>";
for ($i=1; $i<4; $i++){
    acumular ($acum, 4);
    echo "acumulador = $acum <br>";
}
?>
```

← → ↺ ↻ 🏠 📄 localhost/manual/acum.php
Favoritos

```
acumulador = 1
acumulador = 5
acumulador = 9
acumulador = 13
```

[acum.php](#)

El paso de parámetros por referencia permite a una función cambiar el valor del parámetro. En el ejemplo 2 el parámetro \$acum es modificado por referencia dentro de la función, al cambiar el valor de \$a.

Es decir, \$acum → &\$a (\$acum apunta a la dirección de \$a).

Actividad:

Quita el símbolo “&” del primer argumento y comprueba como \$acum no varía.

Ejemplo 3: paso de parámetros por defecto

```
<?php
function soñar ($a = 'ser rico') {      // paso de parámetro por defecto
    return "Quiero $a";
}
echo soñar ('un Volvo V40 <br>');
echo soñar ();
?>
```

[defecto.php](#)

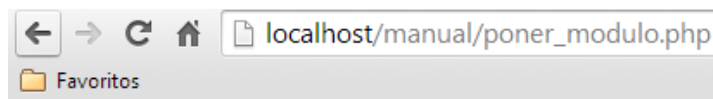
Si la llamada no tiene parámetros se usa el valor por defecto definido en la función.

Ejemplo 4: cantidad variable de parámetros

PHP dispone de las funciones `func_num_args ()`, `func_get_arg ()` y `func_get_args()` para el manejo de funciones con un número variable de parámetros.

```
<?php
function poner_modulo ( ) {
    for ($i=1; $i<func_num_args(); $i++) {
        echo func_get_arg($i);
    }
    echo "<br>";
}
poner_modulo ('DES');           // 1 parámetro
poner_modulo ('Desarrollo', 'Entorno', 'Servidor'); // 3 parámetros
?>
```

[poner_modulo.php](#)



DES
Desarrollo Entorno Servidor

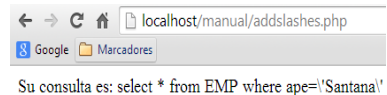
15. MANEJO DE CADENAS

Vamos a ver algunas funciones predefinidas en PHP para manejo de cadenas:

- Cuando hacemos consultas a una Base de Datos, usamos los caracteres especiales: Comillas simples ('), Comillas dobles ("), Barra invertida (\) y NULL

La función **addslashes** (string) añade un carácter de barra invertida (\) a los anteriores, para que el intérprete de PHP no lo tome como un carácter significativo.

```
<?php
$consulta1 = "select * from EMP where ape='Santana' ";
echo 'Su consulta es: ' . addslashes ($consulta1);
?>
```

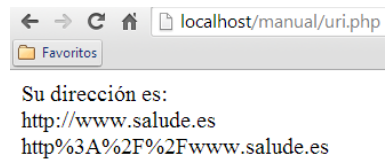


Su consulta es: select * from EMP where ape='\'Santana\''

[addslashes.php](localhost/manual/addslashes.php)

- Si una cadena en PHP va a ser utilizada para consultar una URI en HTML, necesitamos que se mantengan los espacios y los caracteres alfanuméricos. La función **urlencode** (string) reemplaza los caracteres alfanuméricos por el símbolo % y dos dígitos que representan el carácter, y los espacios por el símbolo +.

```
<?php
$uri = "http://www.salude.es";
echo "Su dirección es: $uri";
echo "Su dirección codificada es: " . urlencode ($uri);
?>
```



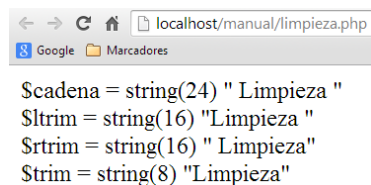
Su dirección es:
http://www.salude.es
http%3A%2F%2Fwww.salude.es

[uri.php](localhost/manual/uri.php)

- Para limpieza de cadenas.
 - **rtrim** (string) elimina caracteres predefinidos por la derecha de la cadena
 - **ltrim** (string) elimina caracteres predefinidos por la izquierda de la cadena
 - **trim** (string) elimina caracteres predefinidos por ambos lados de la cadena
 - **strip_tags** (string) elimina etiquetas HTML de la cadena

```
<?php
$cadena = " \0 \r \n Limpieza \n \r \0 ";
$ltrim = ltrim($cadena);
$rtrim = rtrim ($cadena);
$trim = trim ($cadena);

echo '$cadena = '; var_dump ($cadena); echo "<br>";
echo '$ltrim = '; var_dump ($ltrim); echo "<br>";
echo '$rtrim = '; var_dump ($rtrim); echo "<br>";
echo '$trim = '; var_dump ($trim); echo "<br>";
?>
```



\$cadena = string(24) " Limpieza "
\$ltrim = string(16) "Limpieza "
\$rtrim = string(16) " Limpieza"
\$trim = string(8) "Limpieza"

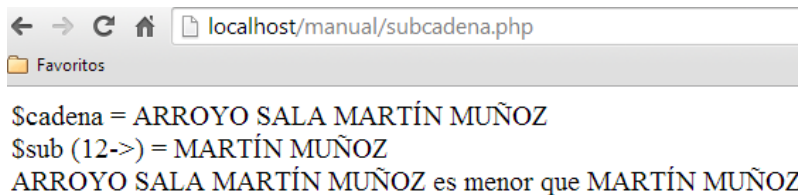
[limpieza.php](localhost/manual/limpieza.php)

- Para manejo de subcadenas
 - **strlen** (string) proporciona el n° de caracteres (longitud) de la cadena
 - **substr** (string, inicio, [n]) proporciona la subcadena que empieza en la posición “inicio” y tiene longitud “n” (parámetro opcional).
 - **strcmp** (string1, string2) devuelve un entero negativo si string1 es menor que string2, positivo si string1 es mayor que string2 y 0 si son iguales. La comparación se realiza carácter a carácter empezando por la izquierda y en case sensitive (distingue mayúsculas de minúsculas).

```
<?php
$cadena = "ARROYO SALA MARTÍN MUÑOZ";
$sub = substr ($cadena, 12);
$cmp = strcmp ($cadena, $sub);

echo "$cadena = $cadena <br>";
echo "\$sub (12->) = $sub <br>";
if ($cmp>0)
    echo "$cadena es mayor que $sub";
else if ($cmp<0)
    echo "$cadena es menor que $sub";
else
    echo "Las cadenas son iguales";
?>
```

[subcadena.php](#)



```
Scadena = ARROYO SALA MARTÍN MUÑOZ
$sub (12->) = MARTÍN MUÑOZ
ARROYO SALA MARTÍN MUÑOZ es menor que MARTÍN MUÑOZ
```

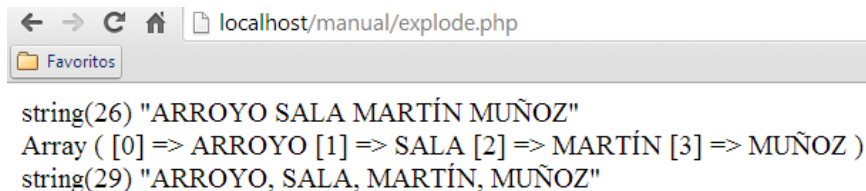
- Conversión de cadenas en arrays

- **explode** (separador, string, [límite]) divide una cadena de caracteres según un “separador” y la convierte en un array con un número “límite” de elementos.
- **implode** (separador, string) convierte un array de varios elementos en una sola cadena separada por el “separador”.

```
<?php
$cadena = 'ARROYO SALA MARTÍN MUÑOZ';
$array = explode(' ', $cadena);
$cadena2 = implode(' ', $array);

var_dump($cadena); echo '<br>';
print_r($array); echo '<br>';
var_dump($cadena2);

?>
```

[explode.php](#)


```
string(26) "ARROYO SALA MARTÍN MUÑOZ"
Array ( [0] => ARROYO [1] => SALA [2] => MARTÍN [3] => MUÑOZ )
string(29) "ARROYO, SALA, MARTÍN, MUÑOZ"
```

- Sintaxis heredoc

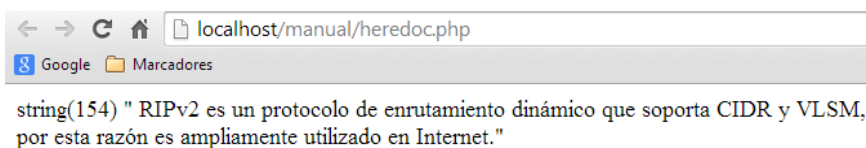
Otra forma de declarar variables de tipo string es utilizar la sintaxis heredoc y nowdoc, y resulta muy útil cuando el texto es largo. Se usa el operador <<< seguido de un delimitador, después la cadena y por último el mismo delimitador;

```
<?php
$cadena = <<<EOF
    RIPv2 es un protocolo de enrutamiento dinámico
    que soporta CIDR y VLSM, <br> por esta razón es
    ampliamente utilizado en Internet.
EOF;
var_dump($cadena);

?>
```

[heredoc.php](#)

Observación: el delimitador final no puede llevar otros caracteres de ningún tipo, ni sangría, excepto el carácter punto y coma “;”



```
string(154) " RIPv2 es un protocolo de enrutamiento dinámico que soporta CIDR y VLSM,
por esta razón es ampliamente utilizado en Internet."
```

16. FUNCIONES PARA MANEJO DE FECHAS

En PHP no existe el tipo de datos fecha, se trabaja con cadenas de caracteres y funciones que extraen la fecha y hora del sistema.

- **date** (formato, [timestamp]) devuelve una cadena con el formato especificado y que contiene la fecha indicada por el entero “timestamp”.

“timestamp” es opcional y si se omite la función toma la fecha/hora del sistema.

- **strtotime** (string) devuelve un entero que representa la fecha indicada en la cadena

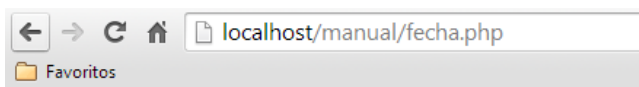
- **mktime** (h, i, sa, m, d, Y) devuelve un entero que representa la fecha indicada por los argumentos

(h=horas, i=minutos, sa=segundos, m=mes, d=día, Y=año)

```
<?php
$fechaSis = date (" l d/m/Y");
$cadena = "11:40am April 16 2011";
$fecha1=strtotime ($cadena);
$fecha2=mktime(14, 0, 0, 7, 10, 2014);

echo 'Hoy es'. $fechaSis. "<br>";
echo 'La hora es '. date("h:i:sa").'<br>';
echo '<br> Formato completo RFC: <br>';
echo date (DATE_RFC2822);
echo '<br><br>Joel nació el '.date("d/m/Y h:i:sa", $fecha1);
echo '<br><br>Las vacaciones comienzan el'.date("d/m/Y h:i:sa", $fecha2);
?>
```

[fecha.php](#)



Hoy es Friday 30/05/2014

La hora es 05:34:55pm

Formato completo RFC:

Fri, 30 May 2014 17:34:55 +0200

Joel nació el 16/04/2011 11:40:00am

Las vacaciones comienzan el 10/07/2014 02:00:00pm

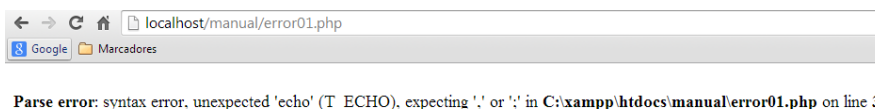
ANEXO 1. TIPOS DE ERRORES

Este es un listado de los errores más comunes:

Archivo	Tipo de error	Nivel PHP	Descripción
error01.php	Error sintáctico	E_PARSE	Falta un ;
error02.php	Error lógico	E_NOTICE	Se envía a la salida estándar una variable no inicializada
error03.php	Error semántico	E_WARNING	División por 0
error05.php	Error fatal	E_ERROR	Se ha usado () para acceder a elementos de un array
error07.php	Error conceptual	E_STRICT	Un método no estático es llamado estáticamente

```
<?php
echo "Hola"
echo "Adiós"
?>
```

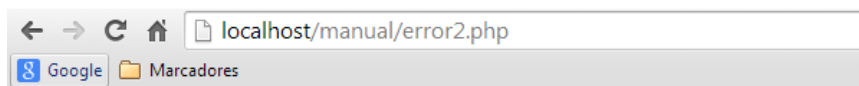
[error01.php](#)



Parse error: syntax error, unexpected 'echo' (T_ECHO), expecting ';' or ':' in C:\xampp\htdocs\manual\error01.php on line 3

```
<?php
echo "$a";
?>
```

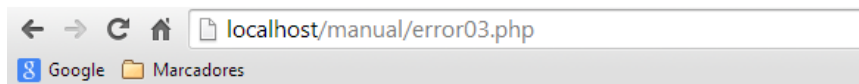
[error02.php](#)



Notice: Undefined variable: a in C:\xampp\htdocs\manual\error2.php on line 2

```
<?php
$a = 1; $b = 0;
$c = $a / $b;
echo "$c";
?>
```

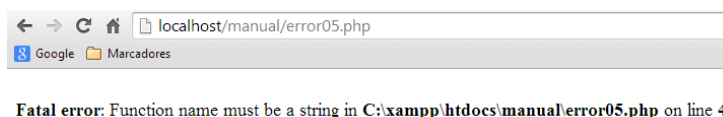
[error03.php](#)



Warning: Division by zero in C:\xampp\htdocs\manual\error3.php on line 4

```
<?php
$lista = [1, 2, 3];
for ($i = 0; $i <= 2; $i++) {
    echo $lista($i);
}
```

[error05.php](#)



Fatal error: Function name must be a string in C:\xampp\htdocs\manual\error05.php on line 4