**MANUAL DE INICIACIÓN A PHP**

**INDICE**

[1. DEFINICIÓN 1](#_Toc390377883)

[2. ETIQUETAS PHP 2](#_Toc390377884)

[3. INSTRUCCIONES PHP 3](#_Toc390377885)

[4. EJECUCIÓN DE UN SCRIPT PHP 4](#_Toc390377886)

[5. VARIABLES 5](#_Toc390377887)

[6. ÁMBITO DE VARIABLES 6](#_Toc390377888)

[7. VARIABLES SUPERGLOBALES 7](#_Toc390377889)

[8. CONSTANTES 8](#_Toc390377890)

[9. VISUALIZACIÓN 9](#_Toc390377891)

[10. PALABRAS RESERVADAS 11](#_Toc390377892)

[11. OPERADORES 12](#_Toc390377893)

[12. TIPOS DE DATOS 13](#_Toc390377894)

[13. ESTRUCTURAS DE CONTROL 15](#_Toc390377895)

[14. FUNCIONES 18](#_Toc390377896)

[15. MANEJO DE CADENAS 20](#_Toc390377897)

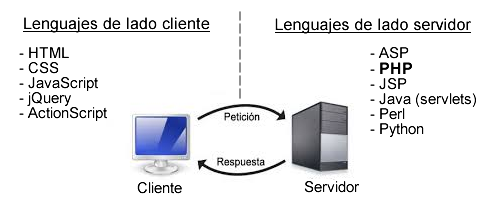
[16. FUNCIONES PARA MANEJO DE FECHAS 23](#_Toc390377898)

[ANEXO 1. TIPOS DE ERRORES 24](#_Toc390377899)

## 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 **<?** ………. **?>**
* 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](http://localhost/manual/primer.php)

## 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](http://localhost/manual/prueba.php)

## 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:

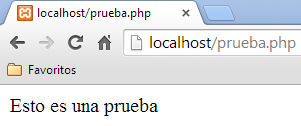
- Para pruebas locales <http://localhost/manual/prueba.php>

- 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



## 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](http://localhost/manual/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>";

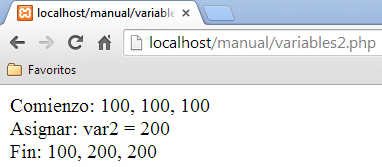
$var2 = 200; // modifica el valor de $var2 y por referencia a $var3

echo "Asignar: var2 = $var2 <br>";

echo "Fin: $var1, $var2, $var3";

?>

[variables2.php](http://localhost/manual/variables2.php)



## 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>";

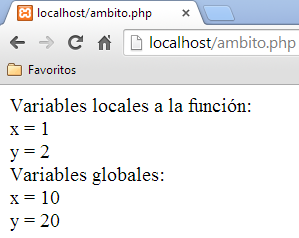
}

ambito();

echo "Variables globales: <br>";

echo "x = $x <br>"; echo "y = $y <br>";

?>



[ambito.php](http://localhost/manual/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;

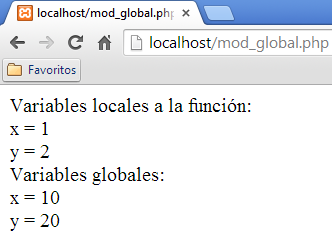
echo "Variables globales: <br>";

echo "x = $x <br>"; echo "y = $y <br>";

}

ambito();

?>

****

[mod\_global.php](http://localhost/manual/mod_global.php)

## 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](http://localhost/manual/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”.

## 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

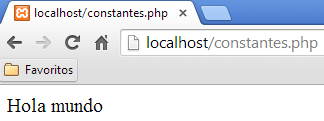
define ("Constante1", "Hola"); // constante declarada mediante define()

const Constante2 = "mundo"; // constante declarada mediante const

echo Constante1, " " , Constante2;

?>

[constantes.php](http://localhost/manual/constantes.php)



## VISUALIZACIÓN

* 1. **echo y print**

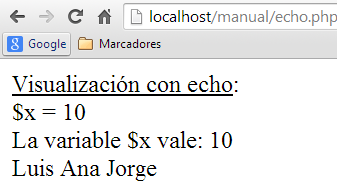
Son construcciones del lenguaje no funciones, por lo que se deben utilizar sin paréntesis, soportan etiquetas HTML y se aplican sobre cadenas de caracteres.

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]";

?>

[echo.php](http://localhost/manual/echo.php)

<?php

$x=10;

$mensaje ="La variable";

$nombres =['Luis', 'Ana', 'Jorge'];

print "<u>Visualización con print</u>: <br>";

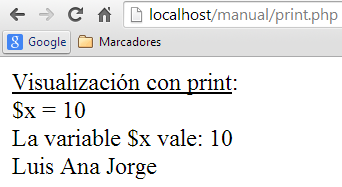
print "\$x = $x <br>";

print "$mensaje". ' $x vale: '. "$x"."<br>";

print "$nombres[0]" . " $nombres[1]" .

" $nombres[2]";

?>



[print.php](http://localhost/manual/print.php)

* 1. **print\_r**

Visualiza el contenido de una variable [string](http://es1.php.net/manual/es/language.types.string.php), [integer](http://es1.php.net/manual/es/language.types.integer.php) o [float](http://es1.php.net/manual/es/language.types.float.php).

Si es un [array](http://es1.php.net/manual/es/language.types.array.php), los valores se presentan de forma ordenada primero lo índices o claves y después los elementos.

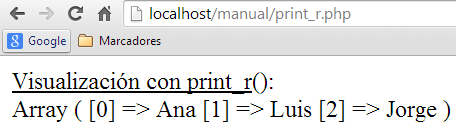
<?php

$nombres= ['Luis', 'Ana', 'Jorge'];

echo "<u>Visualización con print\_r</u>: <br>";

print\_r "$nombres";

?>

****

[print\_r.php](http://localhost/manual/print_r.php)

* 1. **var\_export()**

Visualiza el contenido de una variable.

* 1. **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>";

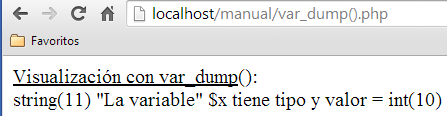
var\_dump ($mensaje);

echo ‘$x tiene tipo y valor = ‘;

var\_dump ($x); echo “<br>”;

?>

[var\_dump.php](http://localhost/manual/var_dump.php)

****

## 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>

## 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 | 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.

* **Llaves**

El operador { } sirve para indicar al intérprete cual es la parte que debe procesar, dentro de una cadena entre comillas dobles. Se puede utilizar para visualizar matrices, funciones, métodos de objetos, etc. dentro de cadenas.

## TIPOS DE DATOS

PHP soporta 8 tipos de datos primitivos:

1. **Boolean**

Para declarar un literal booleano se utilizan las palabras reservadas “true/false”.

1. **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).

1. **Float**

Es un número real expresado con decimales o en notación exponencial.

1. **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.

1. **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.

1. **Object**

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

1. **Resource**

Es una variable especial que apunta a un recurso externo

1. **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>";

?>

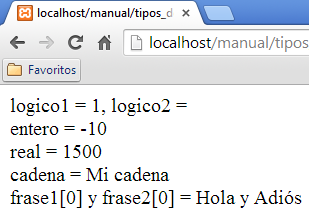
[tipos.php](http://localhost/manual/tipos_de_datos.php)

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]";

?>

[a\_indexado.php](http://localhost/manual/tipos_de_datos.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

$paises = array (‘it’=>’Italia’, ‘es’=>’España’, ‘fr’=>’Francia’);

$platos [‘it’] = ‘Pizza’;

$platos [‘es’] = ‘Paella’;

$platos [‘fr’] = ‘Quiche’;

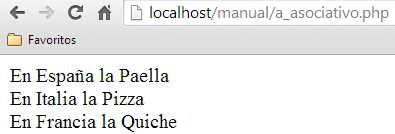
echo ‘En ‘.$paises[‘es’].’ la ‘.$platos[‘es’]; echo ‘<br>’;

echo ‘En ‘.$paises[‘it’].’ la ‘.$platos[‘it’]; echo ‘<br>’;

echo ‘En ‘.$paises[‘fr’].’ la ‘.$platos[‘fr’]; echo ‘<br>’;

?>

[a\_asociativo.php](http://localhost/manual/a_asociativo.php)



## ESTRUCTURAS DE CONTROL

* Condicionales: **if – elseif – else** **switch**

<?php

$genero = 'F';

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

<?php

$nota = 5;

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

* Bucles: **while**  **do while**

<?php

$i = 0;

do {  
  echo “contador = $i”;

$i++;  
  } while ($i<10);  
?>

<?php

$i = 0;

while ($i<10) {  
  echo “contador = $i”;

$i++;  
  }  
?>

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

<?php

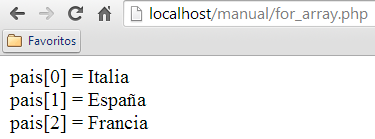
$paises = array (‘Italia’, ’España’, ‘Francia’);

$longitud = count ($paises);

for ($i=0; $i<$longitud; $i++) {  
  echo "pais[$i] = "."$paises[$i] <br>";  
  }  
?>

<?php

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



* 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.

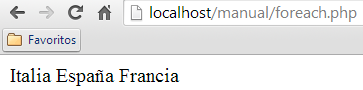
En cada iteración el elemento actual del array se guarda en $valor y se incrementa el puntero interno del array.

<?php

$paises = array (‘it’=>’Italia’, ‘es’=>’España’, ‘fr’=>’Francia’);

foreach ($paises as $valor) {  
  echo "$valor <br>";  
  }  
?>

[foreach1.php](http://localhost/manual/foreach1.php)



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

sentencias; }

Este bucle recorre el array indicado en $nombre\_array.

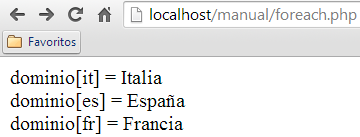
En cada iteración el elemento actual del array se guarda en $valor, la clave se guarda en $clave y se incrementa el puntero interno del array.

<?php

$dominios = array (‘it’=>’Italia’, ‘es’=>’España’, ‘fr’=>’Francia’);

foreach ($dominios as $clave=>$valor) {  
  echo "dominio[$clave] = "."$valor <br>";  
  }  
?>

[foreach2.php](http://localhost/manual/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](http://localhost/manual/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>

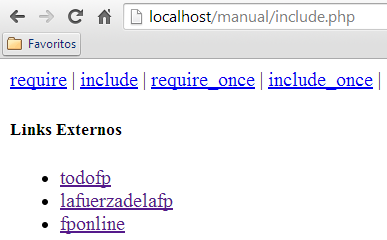
[menu.html](http://localhost/manual/menu.html)

<?php

require ‘header.php’;

include ‘menu.html’;  
?>

[include.php](http://localhost/manual/include.php)



## 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](http://localhost/manual/sumar.php)

Ejemplo 2: **Paso de parámetros por referencia** (añadiendo “&” al argumento)

<?php

function acumular (&$a, $incremento) {

$a = $a + $incremento;

}

$acum = 1;

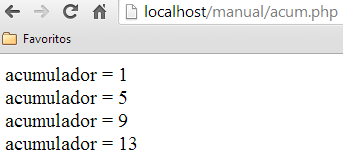
echo "acumulador = $acum <br>";

for ($i=1; $i<4; $i++){

acumular ($acum, 4);

echo "acumulador = $acum <br>";

}  
?>



[acum.php](http://localhost/manual/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](http://localhost/manual/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);

}

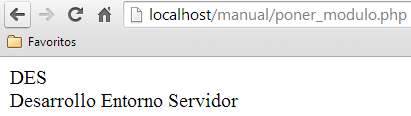
echo “<br>”;

}

poner\_modulo (‘DES’); // 1 parámetro

poner\_modulo (‘Desarrollo’, ‘Entorno’, ‘Servidor’); // 3 parámetros  
?>

[poner\_modulo.php](http://localhost/manual/poner_modulo.php)



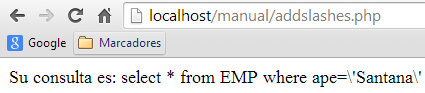
## MANEJO DE CADENAS

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

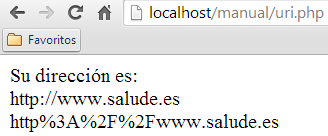
$consulta1 = "select \* from EMP where ape=’Santana’ ";

echo ‘Su consulta es: ‘ . addslashes ($consulta1);  
?>

[addslashes.php](http://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);  
?>

[uri.php](http://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);

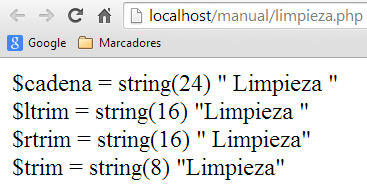
$trim = trim ($cadena);

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>";  
?>



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

* Para manejo de subcadenas
* **strlen** (string) proporciona el nº 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";

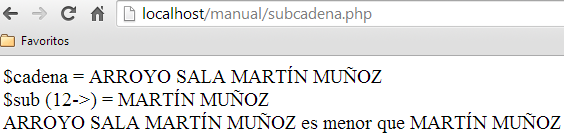
else if ($cmp<0)

echo "$cadena es menor que $sub";

else

echo "Las cadenas son iguales";  
?>

[subcadena.php](http://localhost/manual/subcadena.php)



* 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);

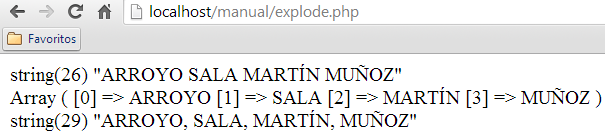
var\_dump ($cadena); echo ‘<br>’;

print\_r ($array); echo ‘<br>’;

var\_dump ($cadena2);

?>

[explode.php](http://localhost/manual/explode.php)



* 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.

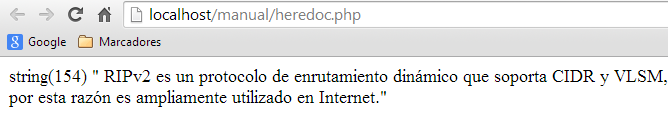
EOF;

var\_dump ($cadena);

?>

[heredoc.php](http://localhost/manual/heredoc.php)

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

****

## 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>’;

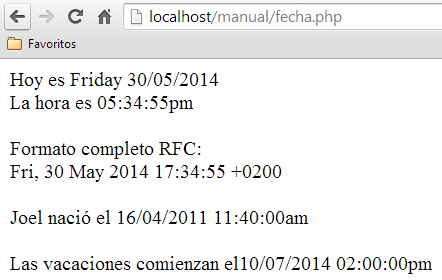
echo date (DATE\_RFC2822);

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](http://localhost/manual/fecha.php)

****

## 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 |

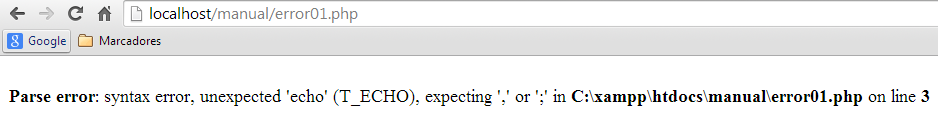
[error01.php](http://localhost/manual/error01.php)

<?php

echo "Hola"

echo "Adiós"

?>

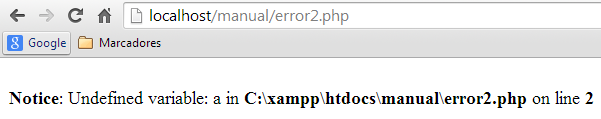


[error02.php](http://localhost/manual/error02.php)

<?php

echo "$a";

?>



[error03.php](http://localhost/manual/error03.php)

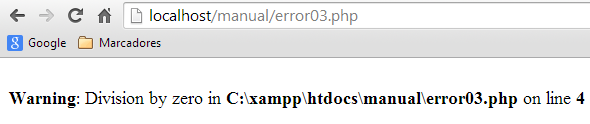
<?php

$a = 1; $b = 0;

$c = $a / $b;

echo "$c";

?>



[error05.php](http://localhost/manual/error05.php)

<?php

$lista = [1, 2, 3];

for ($i = 0; $i <= 2; $i++) {

echo $lista($i);

}

?>

