

Manual de iniciación a PHP

MANUAL DE INICIACIÓN A PHP

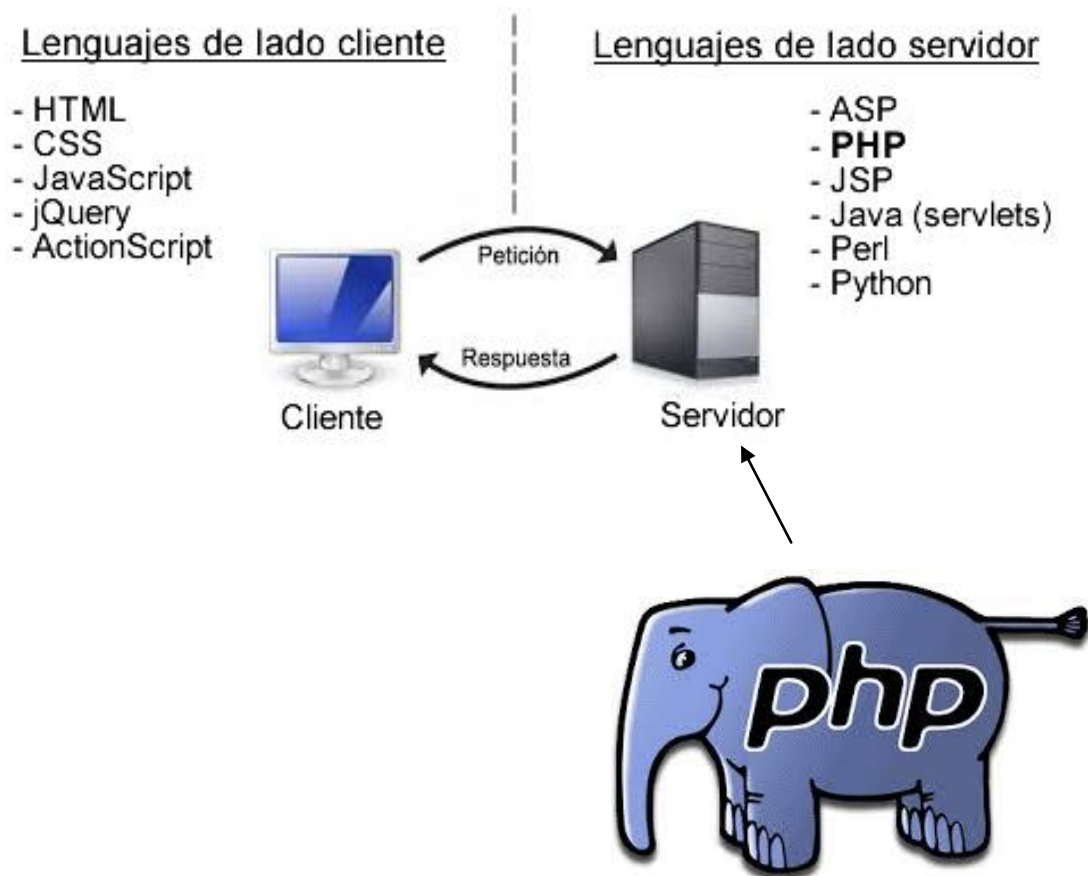
INDICE

1. DEFINICIÓN.....	2
2. ETIQUETAS PHP	3
3. INSTRUCCIONES PHP.....	4
4. EJECUCIÓN DE UN SCRIPT PHP.....	5
5. VARIABLES	6
6. ÁMBITO DE VARIABLES	8
7. VARIABLES SUPERGLOBALES.....	9
8. CONSTANTES.....	10
9. VISUALIZACIÓN.....	11
10. PALABRAS RESERVADAS.....	13
11. OPERADORES.....	14
12. TIPOS DE DATOS.....	15

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
```

[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**.

Ejemplo:

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

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:

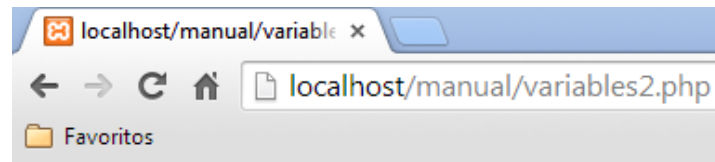
```
<?php
    $nombre1 = "Ana";    // variable de tipo
string
    $nombre2 = "Juan";  // variable de tipo
string
    $nivel = 24;         // variable de
tipo integer
    $profe = TRUE;      // variable de tipo
boolean
    $_sueldo = 'poco';   // variable de
tipo string
```

[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>";
```

[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.

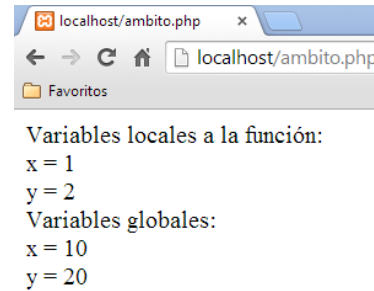
El ámbito de una variable es la parte del código donde puede ser usada, PHP tiene los siguientes ámbitos para las variables:

- 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>";
?>
```

[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:

```
<?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>";
}
```

[mod_global.php](#)



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

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 "\$".

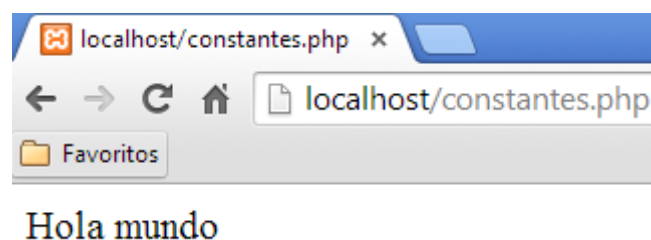
El resultado del script es el mismo que en el ejemplo anterior con el modificador "global".

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](localhost/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:

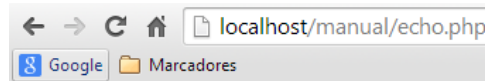
void **echo** cadena1, cadena2, ... , cadenaN

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

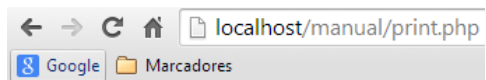
La variable \$x vale: 10

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](#)

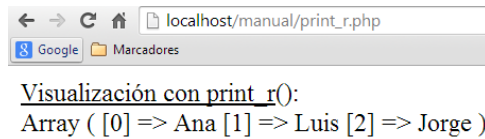
9.2. print_r

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.

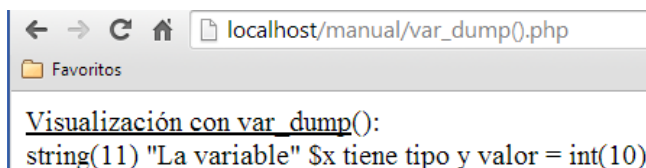
9.4. var_dump()

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](#)



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

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	Multipliación	
\$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.

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

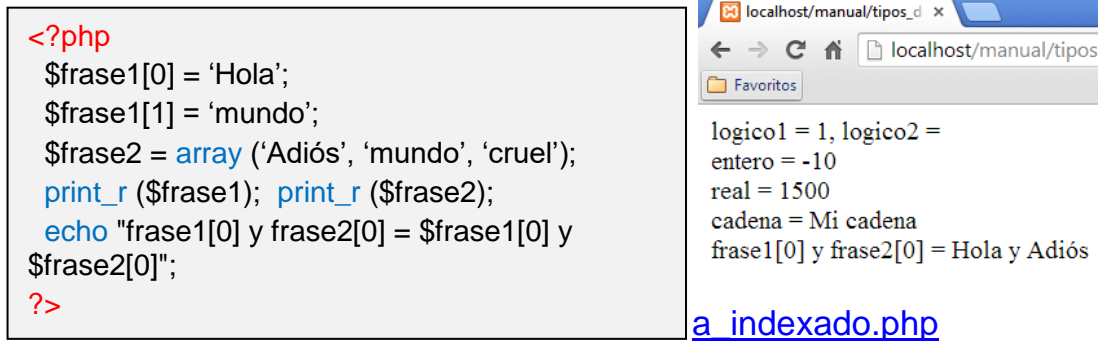
```
<?php
$logico1 = true;    $logico2 = false;
$entero = -10;     $real = 1.5e3;
$cadena = 'Tipos de datos simples';
echo "logico1 = $logico1, logico2 = $logico2
<br>";
echo "entero = $entero <br>";
echo "real = $real <br>";
echo "cadena = $cadena <br>";
?>
```

[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)



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:

