

TEMA 2

PHP BÁSICO

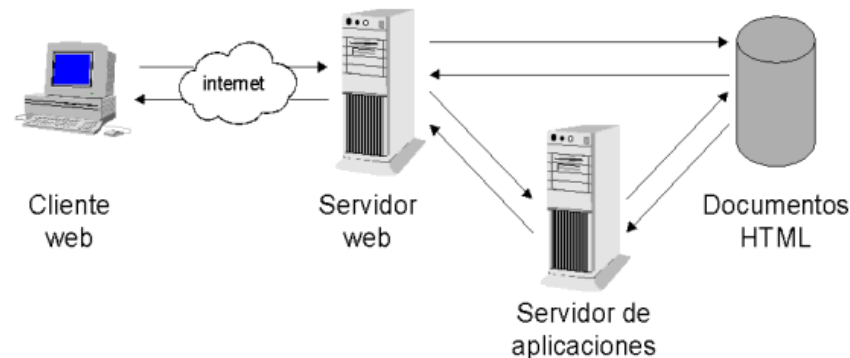
1.- INTRODUCCIÓN

PHP (Hypertext Preprocessor") es un lenguaje de "código abierto" interpretado, de alto nivel, embebido en páginas HTML.

El código PHP es ejecutado en el servidor. El cliente solo recibe el resultado de la ejecución del script PHP en el servidor, sin ninguna posibilidad de determinar qué código ha producido el resultado recibido.

PHP permite procesar la información de formularios , generar páginas con contenidos dinámicos, o enviar y recibir cookies.

Es portable y multiplataforma y su sintaxis es parecida a C. Proporciona alta conectividad con la mayoría de Sistemas de Gestión de Bases de datos y apoyo para diferentes protocolos de comunicación de Internet (HTTP, IMAP, FTP,...)



Es potente, fácil de aprender , de libre distribución, permite el acceso a bases de datos y otras funcionalidades orientadas en la red, además de que dispone de abundante apoyo en la Web.

Requisitos:

- Servidor web Apache (www.apache.org) con el módulo PHP (www.php.net).
- base de datos MySQL (www.mysql.com) si se desea crear páginas dinámicas

Otras utilidades:

- Herramientas para la gestión de MySQL , como PHPMyAdmin (www.phpmyadmin.net).
- Editores de PHP , como DevPHP (www.sourceforge.net), Netbeans, Notepad ++
- Editores de HTML como Dreamweaver, etc
- Manuales de PHP y MySQL

2.- PREPARACIÓN DEL ENTORNO

En la hora de instalar Apache + PHP + MYSQL tenemos dos opciones:

- Instalación manual e independiente de Apache , PHP y MYSQL
- Instalación automatizada mediante la utilización de los paquetes WAMP, XAMPP o AppServer, bajo entornos windows, o LAMP para entornos Linux

Si no queremos tener problemas de instalación y configuración podemos decantarnos por la segunda opción, y en este caso , podemos realizar la descarga desde .:

- <http://www.wampserver.com/>
- <http://www.appservnetwork.com/>
- <http://www.apachefriends.org/en/xampp.html>.
- [http://es.wikibooks.org/wiki/programaci%C3%B3n en PHP#PHP b.C3.A1sico](http://es.wikibooks.org/wiki/programaci%C3%B3n_en_PHP#PHP_b.C3.A1sico)

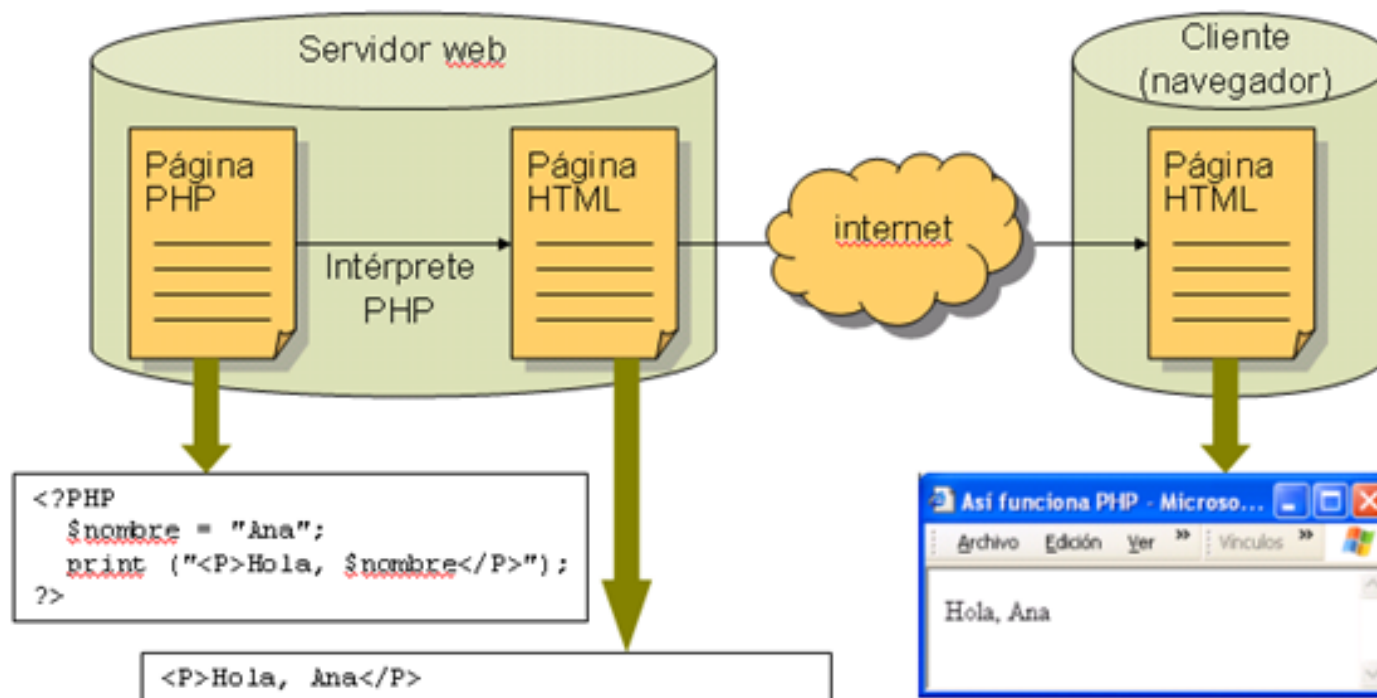
3.- SINTAXIS BÁSICA

Enlace de interés: <http://php.net/manual/es/index.php>

Un párrafo PHP tiene que delimitarse mediante las etiquetas **<?php y ?>**.

Cuando el cliente solicita el documento HTML el servidor entregará el archivo al módulo de PHP y este buscará esas etiquetas, sustituyéndolas junto con su contenido, por el resultado de la sentencia php.

Como resultado, el navegador recibirá el mismo archivo pero sustituyendo las líneas por su resultado.



ESTRUCTURA DE UNA PÁGINA PHP

Fragmentos PHP y fragmentos HTML

Una página PHP es un archivo de texto que contiene uno o varios fragmentos de código **PHP** y que también puede contener fragmentos de código **HTML**.

Los fragmentos de código PHP están delimitados por las etiquetas:

- **<?php** (etiqueta inicial)
- **?>** (etiqueta final).

Nota:

Otras maneras de delimitar el código php: en función del fichero de configuración **php.ini**:

- `short_open_tag = Dónde` ◇ **<?.....?>**
- `asp_tags = Dónde` ◇ **<%.....%>**
- También se pueden utilizar las etiquetas **<script language = "php">.....</script>**

pero se recomienda utilizar exclusivamente **<?php ... ?>**

Los fragmentos de código HTML no están delimitados, en una página **PHP** todo el que no son fragmentos PHP son fragmentos **HTML**.

Cuando un navegador solicita una página PHP a un servidor, el servidor lee el archivo secuencialmente y va generando el resultado:

- si la página contiene fragmentos HTML, el código HTML se incluye sin modificados en el resultado.
- si la página contiene fragmentos PHP, se ejecutan las instrucciones que se encuentran en los fragmentos de código PHP. Si esas instrucciones generan texto, el texto se incluye en el resultado.

Toda las líneas finalizan con **punto y coma** en el final de línea, excepto si definimos una función o estamos if, else, while, etc.

Ejemplo:

<i>holaMundoHtml.php</i>	<i>holaMundoPHP.php</i>
<pre> <html> <head> <meta charset="UTF-8"> <title></title> </head> <body> <?php // put your code here echo "Hola Mundo"; ?> </body> </html> </pre>	<pre> <?php echo "Hola Mundo"; ?> </pre>

COMENTARIOS EN PHP

- Los comentarios **cortos** (una línea)

//

```
<?php
// La instrucción print escribe texto en la página web
print "Hola"; // El comentario se puede escribir al final de la línea
?>
```

#

```
<?php
# La instrucción print escribe texto en la página web
print "Hola"; # El comentario se puede escribir al final de la línea
?>
```

- Los comentarios **largos** (varias líneas)

/...../

```
<?php
# La instrucción print escribe texto en la página web
print "Hola"; # El comentario se puede escribir al final de la línea
?>
```

- <!-- -->

Ejemplo:

```
<?php
    echo 'Esto es una prueba'; // Esto es un comentario al estilo de c++ de una sola línea
    /* Esto es un comentario multilínea
       y otra línea de comentarios */
    echo 'Esto es otra prueba';
    echo 'Una prueba final'; # Esto es un comentario al estilo de consola de una sola línea
?>
```


ENLACES A HOJAS DE ESTILO

Lo veremos más adelante

Una página PHP puede enlazar a una hoja de estilo CSS exactamente igual que una página HTML, es decir, indicando la hoja de estilo mediante la etiqueta <link>.

Los siguientes ejemplos muestran el enlace a la hoja de estilo incluido en un fragmento HTML o generado por un fragmento PHP:

```
<?php print "<?xml version=\"1.0\" encoding=\"iso-8859-1\"?>\n"; ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta http-equiv="content-type" content="text/html; charset=iso-8859-1" />
<?php
print "    <link href="estilo.css" rel="stylesheet" type="text/css" title="Hoja de estilo" />\n";
print "    <title>Página válida XHTML 1.0 Strict</title>\n";
?>
</head>

<body>
<?php
print "    <p>El código HTML de esta página es XHTML 1.0 Strict válido</p>\n";
?>
</body>
</html>
```

CARACTERES DE 'ESCAPE'

Dentro del código PHP podemos usar el carácter `\` combinado con otro. Este carácter hace que no se tenga en cuenta el carácter que viene a continuación de él.

Algunos signos de escape (*escape*) más utilizados:

- `\n` crea una nueva línea
- `\r` crea un salto de párrafo.
- `\t` crea una tabulación

Este signo `\` también sirve para enmascarar signos especiales, es decir hacer que php no interprete el carácter que viene a continuación de `\`

- `\'` comillas simples
- `\$` firme dólar
- `\\` barra invertida
- `\"` comillas dobles

Ejemplo: No se interpretará el primer signo `$` del echo.

```
$a = 4;  
echo "\$a=" . $a;
```

FUNCIONES ÚTILES

Funciones print(), printf() y echo()

- **echo:** <http://php.net/manual/es/function.echo.php>
- **print:** <http://php.net/manual/es/function.print.php>

echo()	Es la más utilizada, muestra cadenas de caracteres en la salida estándar. No acepta formato de salida. Sintaxis: echo string arg1 [, string arg2]; Sintaxis: echo (string arg1 [, string arg2]);
print()	Es la más sencilla, y muestra el contenido de una cadena de caracteres en la salida estándar. No acepta formato de salida. Sintaxis: printf(string cadena);
printf()	Realiza la misma acción que la función anterior, con la diferencia que ésta si acepta argumentos de formato de salida. Sintaxis: printf(string formato [, cualquier valor , ...]);

- **printf**→ <http://php.net/manual/es/function.printf.php>

Formatos de salida para printf.

Elemento	Tipo de variable
%s	Cadena de caracteres.
%d	Número sin decimales.
%f	Número con decimales.
%c	Carácter ASCII.
Aunque existen otros tipos, estos son los más importantes.	

Veremos ejemplos más adelante

- **sprintf**: Devuelve un string con formato

<http://php.net/manual/es/function.sprintf.php>

4.- VARIABLES

Una variable de un lenguaje de programación es un elemento que permite almacenar información.

Las variables se identifican por su nombre.

En PHP el programador puede dar el nombre que quiera a las variables, con algunas restricciones:

- No es necesario **declarar** el tipo de variable . PHP reconoce automáticamente el tipo de variable cuando se le hace una asignación.
- Los nombres de las variables tienen que empezar por el carácter \$.
- A continuación tiene que haber una letra (mayúscula o minúscula) o un guion bajo (_).
- El resto de caracteres del nombre pueden ser números, letras o guiones bajos.
- PHP distingue entre **mayúsculas** y **minúsculas** cuando el identificador corresponde a una variable, mientras que no hace esa distinción en el resto de los casos, por ejemplo en los nombres de funciones o las propias instrucciones del lenguaje.
- Podemos elegir el **nombre** que queramos para las variables, pero es recomendable que tengan un significado descriptivo de para que servirá . (Por ejemplo, si se guardará en una variable la edad del usuario, un nombre adecuado de la variable sería por ejemplo \$edad)
- No podemos utilizar como nombre de variable ninguna **palabra reservada** del lenguaje PHP.
- Si el nombre de la variable contiene varias palabras, se aconseja utilizar la notación **camel caso**, es decir, escribir la primera palabra en minúsculas y la primera letra de las siguientes palabras en mayúsculas. (ejemplo \$edadUsuario, \$totalEuros).
- No es recomendable poner **acentos** o **caracteres especiales** en los nombres de variable (ç,ñ...), aunque podría funcionar correctamente.

USO DE VARIABLES

Para guardar un valor en una variable se utiliza el operador de asignación (=) escribiendo a la izquierda únicamente el nombre de la variable y a la derecha el valor que queremos guardar.

- Si queremos guardar un número, **no** hace falta poner comillas
- Si queremos guardar una cadena de texto **hay que** poner cometas (dobles o simples)

Ejemplo:

```
<HTML>
<HEAD><TITLE>Variables en PHP</TITLE></HEAD>
<BODY>
  <?php
    // inicialización de variables

    $nombre = 'Agustín';
    $email = "ayague@email.es";
    $edad=21;

    echo $nombre;
    echo "<br>";
    echo $email;
    echo "<br>";
    echo $edad;
  ?>
</BODY>
</HTML>
```

TIPOS BÁSICOS DE VARIABLES

- El tipo de una variable en PHP no se suele especificar.
- Se decide en tiempo de ejecución en función del contexto y puede variar

PHP soporta 8 tipos de datos **primitivos** y 2 tipos **compuestos**

TIPOS PRIMITIVOS DE DATOS

- Tipos escalares: *boolean, integer, double, string*
- Tipos compuestos: *array, object*
- Tipos especiales: *resource, NULL*

TIPOS COMPUESTOS

Array

Utilizado para almacenar conjunto de datos.

(Los trataremos más adelante).

Object

Utilizado para almacenar instancias de clases.

VARIABLES LÓGICAS (BOOLEAN)

Las variables de tipo lógico solo pueden tener el valor **TRUE** (verdadero) o **FALSE** (falso).

Comparando su valor con TRUE o FALSE	Empleádola directamente en la condición del if.
<pre><?php \$contestar = TRUE; if (\$contestar==TRUE) { echo "Contestar es TRUE"; } else { echo "Contestar es FALSE"; } ?></pre>	<pre><?php \$contestar = TRUE; if (\$contestar) { echo "Contestar es TRUE"; } else { echo "Contestar es FALSE"; } ?></pre>

VARIABLES ENTERAS (INTEGER)

Las variables de tipo entero pueden guardar números enteros (**positivos o negativos**).

Ejemplo 1:

```
<?php
$a = 1234; // numero decimal
$a = -123; // un numero negativo
$a = 0123; // numero octal (equivalente al 83 decimal)
$a = 0x1A; // numero hexadecimal (equivalente al 26 decimal)
?>
```

Ejemplo 2:

```
<?php
    $lado = 14;
    $area = $lado*$lado;
    echo "<STRONG>Un cuadrado de lado $lado cm"
        ." tiene un área de $area cm<sup>2</sup>.</STRONG>";
?>
```

VARIABLES REALES (FLOAT I DOUBLE)

Las variables de tipo decimal (*float*) pueden guardar números decimales (**positivos o negativos**). Cómo en las calculadoras, el separador de la parte entera y la parte decimal es el **punto** (.), no la coma (,)..

```
<?php
$lado = 14.5;
$area = $lado*$lado;

print "<p>Un cuadrado de lado $lado cm tiene un área de $area cm<sup>2</sup>.</p>";
?>
```

VARIABLES CADENA (STRING)

- Las variables de tipos **cadena** pueden guardar caracteres.
- PHP no impone ningún límite al tamaño de las cadenas. Las cadenas pueden ser todo lo largas que permita la memoria del servidor.
- El juego de caracteres que utiliza PHP viene determinado en principio por el juego de caracteres que utiliza el fichero fuente del programa. Pero hay que tener en cuenta que las funciones de tratamiento de cadenas no están preparadas para tratar la diversidad de juegos de caracteres: muchas suponen que cada carácter ocupa solo un byte, otras suponen un juego de caracteres determinado (UTF-8, por ejemplo), otras utilizan el juego de caracteres definido localmente, etc.
- Se puede acceder a caracteres **individuales** indicando la posición del carácter, como si se tratara de una matriz de una dimensión en la cual el primer carácter ocupa la posición 0.

USO DE CADENAS EN PHP

Ver los ejemplos de cadenas en el archivo ➔ *ejer_cadenas.php*:

En PHP se podan tanto las comillas **simples** (') como las **dobles** (") para representar una cadena, aunque hay algunas diferencias entre ambas.

Ver enlace <http://php.net/manual/es/language.types.string.php>

```
<?php
echo "Esto es una cadena sencilla";
?>
```

DIFERENCIAS ENTRE COMILLAS DOBLES " Y COMILLAS SIMPLES '

- El que hacen las comillas **dobles** es **reemplazar** las variables por su valor.
- En cambio las comillas **simples** no leen el valor, solo entregan texto plano.

Ejemplo:

```
$variable = "Cielo";  
  
// El Cielo es azul  
echo "El $variable es azul";  
  
// El $variable es azul  
echo 'El $variable es azul';
```

Salida:

El Cielo es azul
El \$variable es azul

COMILLAS SIMPLES

Las comillas simples entienden pocos caracteres de escape.

PHP tampoco sustituye algunos caracteres especiales (por ejemplo, el salto de línea `\n`) dentro de las comillas simples.

Ejemplo:

```
echo "<br>";  
print "<pre>Esto está en\ndos líneas.</pre>";  
  
print '<pre>Esto está en\ndos líneas.</pre>';
```

Salida:

```
Esto está en  
dos líneas.
```

```
Esto está en\ndos líneas.
```

COMILLAS DOBLES

Si la cadena se encuentra rodeada de comillas dobles ("), PHP entiende más secuencias de escape para caracteres especiales:

Caracteres escapados	
Secuencia	Significado
\n	avance de línea (LF o 0x0A (10) en ASCII)
\r	retorno de carro (CR o 0x0D (13) en ASCII)
\t	tabulador horizontal (HT o 0x09 (9) en ASCII)
\v	tabulador vertical (VT o 0x0B (11) en ASCII) (desde PHP 5.2.5)
\e	escape (ESC o 0x1B (27) en ASCII) (desde PHP 5.4.4)
\f	avance de página (FF o 0x0C (12) en ASCII) (desde PHP 5.2.5)
\\	barra invertida
\\$	signo de dólar
\"	comillas dobles

Si en una cadena tiene que haber comillas simples y dobles, se pueden escribir como \' o \". Es decir, se consigue visualizarlas mediante el carácter de escape \.

Como las cadenas de texto se pueden delimitar por comillas dobles o simples. Si una cadena está delimitada por comillas dobles, en su interior puede haber cualquier número de comillas simples, y viceversa.

Ejemplo:

```
<?php
// comillas doble entre comillas simples
$x=' Aquí puede haber " dentro de la cadena';
// comillas simples entre comillas dobles
$x=" Aquí puede haber ' dentro de la cadena";
?>
```

En el primer caso la comilla simple ' abre y valla la cadena por lo cual es posible meter una " comilla doble en esta.

Ejemplo:

Archivo *ejer_cadenas2.php*.

```
<?php
print "Esto es una comilla simple: '";
echo "<br>";

print 'Esto es una comilla doble: "';
echo "<br>";

print "Esto es una comilla simple: ' y esto una comilla doble: \" ";
echo "<br>";

print 'Esto es una comilla simple: \' y esto una comilla doble: " ';
echo "<br>";
?>
```

Salida:

```
Esto es una comilla simple: '
Esto es una comilla doble: "
Esto es una comilla simple: ' y esto una comilla doble: "
Esto es una comilla simple: ' y esto una comilla doble: "
```

El código HTML también puede tener comillas simples o dobles:

Ejemplo:

```
<?php
print "<strong style='color: red;'>Hola</strong>";
echo "<br>";

print '<strong style="color: red;">Hola</strong>';
echo "<br>";
?>
```

Salida:



Hola

Hola

OPERADOR CONCATENACIÓN DE CADENAS

El operador . (**punto**) permite concatenar dos o más cadenas

Enlace: <http://php.net/manual/es/language.operators.string.php>

```
<?php  
print "<p>Pasa"."tiempos</p>";  
?>
```

Salida:

Pasatiempos

CARACTERES DE ESCAPE

Dentro del código PHP podemos usar el carácter `\` combinado con otro. Este carácter hace que no se tenga en cuenta el carácter que viene a continuación de él.

Algunos signos de escape (**scape**) más utilizados:

- `\n`

crea una nueva línea

- `\r`

crea un salto de párrafo.

- `\t`

crea una tabulación

Este signo \ también sirve para **enmascarar** (escapar) **signos especiales**, es decir hacer que **php no interprete el carácter que viene a continuación de **

– \'

escapa la comilla simple

– \\$

escapa el signo dólar

– \\

escapa la barra invertida

– \"

escapa las comillas dobles

Ejemplo: No se interpretará el primer signo \$ del *echo*.

```
$a = 4;  
echo "\$a=".$a;
```

VARIABLES ARRAY

Se verán más adelante

CONVERSIONES DE TIPOS

Ver los ejemplos de TIPOS en el archivo

ejemp_tipo.php:

Enlace: <http://php.net/manual/es/language.types.type-juggling.php>

- **Conversión implícita**

Al realizar operaciones entre tipos diferentes se realizará una conversión implícita al tipo de mayor rango.

Ejemplo:

```
<?php
```

```
// Ejemplos de TIPOS
```

```
$num_entero = 10; // num es un entero
```

```
$num_real=10.25; // es un real
```

```
$num_entero =$num_real; // ahora $num_entero pasa a ser real
```

```
?>
```

- **Conversión Explícita (Casting)**

En php tenemos la posibilidad de realizar conversiones explícitas entre tipos. Una forma es utilizando la sintaxis del casto explícito del lenguaje C.

Sintaxis:

`$var1 = (tipo) $var2;`

Ejemplo:

```
<?php
// Ejemplos de CASTING
$num = 10; // num es un entero
$booleana = (boolean)$num; // $booleana es un boolean
?>
```

VARIABLES PREDEFINIDAS

<http://php.net/manual/es/reserved.variables.php>

PHP proporciona una enorme cantidad de variables predefinidas que están disponibles para cualquier script de código que se ejecute. Muchas de estas variables dependen de la configuración del servidor.

Varias variables predefinidas en PHP son "**superglobales**", lo cual quiere decir que están disponibles en todos los ámbitos a lo largo de un script.

Estas variables superglobales son:

- **\$GLOBALS**: Contiene una referencia a cada variable disponible en el espectro de las variables del script. Las claves de esta matriz son los nombres de las variables globales
- **\$_SERVER**: Información del servidor y el entorno de ejecución.
- **\$_GET**: Variables HTTP GET
- **\$_POST**: Variables HTTP PUESTO
- **\$_FILES**: Variables de Carga de Archivos HTTP
- **\$_COOKIE**: Cookies HTTP
- **\$_SESSION**: Variables de sesión.
- **\$_REQUEST**: Variables de petición HTTP
- **\$_ENV**: Variables de entorno.

VARIABLES DE VARIABLE


Se trata de una variable que contiene el nombre de otra variable.

- PHP nos permite cambiar el nombre de una variable de manera dinámica.
- Su funcionamiento consiste a utilizar el valor de una variable como nombre de otra.
- Puede ser útil por ejemplo para ir enumerando las variables de un formulario con un bucle.
- No conviene abusar de ellas, porque pueden dar lugar a código ofuscado.

Ejemplo:

```
<?php
// Ejemplos de vble de vble
$nombreVar= 'valor';
$$nombreVar = 200;
echo 'Contenido de la variable de vble es: ';
echo $valor;
echo '<br>';
echo 'Equivalente a: '.$$nombreVar;
?>
```

Salida:



localhost/PHP1/ejemp_vbleDevble.php

Contenido de la variable de vble es: 200
Equivalente a: 200

ENTRADA DE DATOS DESDE UN NAVEGADOR (DESDE LA URL)

La forma más sencilla de pasar datos de entrada a un script de php desde un cliente web es a través del QUERY_STRING que tendrá una forma:

```
http://localhost/script.php?dato1=valor1&dato2=valor2&dato3=valor3
```

Ejemplo

`http://localhost/script.php?nombre=Juanjo&edad=38`

Fijémonos que después del nombre del fichero o script que ejecutaremos se pone el carácter `?` y después de los diferentes datos que queremos pasarle al script separados por el carácter `&`.

En el script podremos acceder al valor del nombre a través de la variable superglobal `$_GET` con la sintaxis `$_GET['dato1']` como ya hemos comentado.

Ejemplo

`$_GET['nombre']` se evaluará a la cadena "Juanjo" y `$_GET['edad']` se evaluará al entero 38.

ENTRADA DESDE UN FORMULARIO

Lo veremos en el tema siguiente

FUNCIONES ASOCIADAS A VARIABLES

Lo veremos más adelante

- **gettype()** devuelve el tipo de una variable
- Las funciones **is_type** comprueban si una variable es de un tipo dado:
 - **is_array()**
 - **is_bool()**
 - **is_float()**
 - **is_integer()**
 - **is_null()**
 - **is_numeric()**
 - **is_object()**
 - **is_scalar()**
 - **is_string()**
- Funciones que comprueban si una variable está asignada:
 - **isset(variable)**: devuelve TRUE si la variable pasada está ya definida. FALSE en caso contrario
 - **unset(variable)**: permite eliminar una variable definida con anterioridad
 - **empty(variable)**: Funcionamiento inverso a **isset**, devolviéndome 1(TRUE) si la variable no está definida y 0(FALSE) en caso contrario
- Funciones que asocia tipos a una variable:
 - **settype(variable, tipo)**: recibe como parámetros una referencia a una variable y una cadena de caracteres, cambia el tipo de la variable referenciada al indicado por esta cadena

Ejemplo:

```
echo 'Hago $a=10;<br>';  
$a=10;  
echo 'isset($a)='.isset($a). '<br>';  
echo 'empty($a)='.empty($a). '<br>';  
echo 'Hago unset($a);<br>';  
unset($a);  
echo 'isset($a)='.isset($a). '<br>';  
echo 'empty($a)='.empty($a). '<br>';
```

Lo utilizaremos en Formularios

5.- CONSTANTES

Las **constantes** son similares a las variables, pero sin el signo dólar delante, y solo la podemos asignar una vez.

Para definir una constantes usaremos la función **define** ("constando",valor);

Ejemplo:

```
echo "CONSTANTES<br>";  
define ("SALUDO", "Hola Mundo");  
printf (SALUDO);  
// o también con echo  
echo SALUDO;
```

La constante es sensible a mayúsculas por defecto . Por convención , los identificadores de constantes suelen declararse en mayúsculas.

El alcance de una constante es **global**.

Solo se puede definir como constantes valores escalares (boolean, integer, float y string).

CONSTANTES PREDEFINIDAS

Existen muchas **constantes predefinidas**.

<http://www.php.net/manual/es/reserved.constants.php>

Ejemplo:

```
<?php
echo "Algunas <strong>constantes Predefinidas</strong> son:<br><br>";

echo "La versión actual de PHP es : ".PHP_VERSION."<br>";
echo "La última versión de PHP es: ".PHP_MAJOR_VERSION."<br>";
echo "La longitud máxima de nombre de fichero_directorio soportada es: ".PHP_MAXPATHLEN."<br>";
echo "El valor TRUE es:".TRUE."<br>"; // valor verdadero (1)
?>
```

Listado de constantes **predefinidas** :

<http://www.php-es.com/reserved.constants.html>

Veamos algunas de ellas:

PHP_VERSION ([string](#))

La versión actual de PHP en notación "mayor.menor.edición[extra]".

PHP_MAJOR_VERSION ([integer](#))

La versión "mayor" actual de PHP como valor integer (p.ej., int(5) en la versión "5.2.7-extra"). Disponible desde PHP 5.2.7.

PHP_MINOR_VERSION ([integer](#))

La versión "menor" actual de PHP como valor integer (p.ej., int(2) en la versión "5.2.7-extra"). Disponible desde PHP 5.2.7.

PHP_RELEASE_VERSION ([integer](#))

La versión de "publicación" (release) actual de PHP como valor integer (p.ej., int(7) en la versión "5.2.7-extra"). Disponible desde PHP 5.2.7.

6.- EXPRESIONES Y OPERACIONES

A continuación nos familiarizaremos con los diferentes tipos de operadores que nos ofrece PHP y como construir expresiones validas que nos permitirán:

- hacer cálculos.
- realizar asignaciones.
- establecer comparaciones lógicas.

OPERADORES

- aritméticos.
- de asignación .
- comparación.
- de control de errores .
- incremento y decremento.
- lógicos.
- de cadena .

OPERADORES ARITMÉTICOS

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

Permiten hacer cálculos básicos y son los siguientes:

Ejemplo	Nombre	Resultado
+\$a	Identidad	Conversión de \$a a <u>int</u> o <u>float</u> según el caso.
-\$a	Negación	Opuesto de \$a.
\$a + \$b	Adición	Suma de \$a y \$b.
\$a - \$b	Sustracción	Diferencia de \$a y \$b.
\$a * \$b	Multiplicación	Producto de \$a y \$b.
\$a / \$b	División	Cociente de \$a y \$b.
\$a % \$b	Módulo	Resto de \$a dividido por \$b.
\$a ** \$b	Exponenciación	Resultado de elevar \$a a la potencia \$bésima. Introducido en PHP 5.6.

Ejemplo:

```
<?php
    $a = 4;
    $b = 7;
    $c = $a + $b; //la variable $c será igual a 11 echo "\$a=".$a."<br>";
    // Véase cómo introducimos caracteres de escape para que no se
    // interprete el símbolo $
    echo "\$a=".$a."<br>";
    echo "\$b=".$b."<br><br>";
    echo "\$c=\$a+\$b = $c";
?>
```

OPERADORES DE ASIGNACIÓN.

En el ejemplo anterior hemos visto que con PHP las asignaciones básicas se hacen mediante el operador '='. Este operador hace que el operando de la izquierda coja el valor de la derecha.

Ver enlace: <http://www.php.net/manual/en/language.operators.arithmetic.php>

\$a = \$b	Asigna a \$a el contenido de \$b
\$a += \$b	Le suma a \$b a \$a
\$a -= \$b	Le resta a \$b a \$a
\$a *= \$b	Multiplica \$a por \$b y lo asigna a \$a
\$a /= \$b	Divide \$a por \$b y lo asigna a \$a
\$a .= \$b	Añade la cadena \$b a la cadena \$a

Ejemplo:

```
$total = 2500; //la variable $total tendrá el valor 2500
```

Los operadores combinados, que permiten asignar y operar en un solo paso:

```
$a = 4;
```

```
$a += 6; // esto es equivalente a $a = $a + 6
```

y también con cadenas

```
$cadena = "Hola ";
```

```
$cadena .= "José"; // $cadena tendrá el valor "Hola José"
```

OPERADORES DE COMPARACIÓN.

Enlace: <http://php.net/manual/es/language.operators.comparison.php>

\$a == \$b	Igual	TRUE si \$a es igual a \$b
\$a === \$b	Identidad	TRUE si \$a es igual a \$b i si son del mismo tipo
\$a != \$b	Distinto	TRUE si \$a no es igual a \$b
\$a < \$b	Menor que	TRUE si \$a es estrictamente menor que \$b
\$a > \$b	Mayor que	TRUE si \$a es estrictamente mayor que \$b
\$a <= \$b	Menor o igual que	TRUE si \$a es menor o igual que \$b
\$a >= \$b	Mayor o igual que	TRUE si \$a es mayor o igual que \$b

OPERADOR DE CONTROL DE ERRORES (@)

Enlace: <http://php.net/manual/es/language.operators.errorcontrol.php>

PHP ofrece apoyo para un operador de control de errores : el signo de arroba (@). Cuando es colocado a comienzos de una expresión en PHP, cualquier mensaje de error que pudiera generarse a causa de esa expresión será ignorado, es decir , suprime los errores implícitos mostrados por el sistema y en su lugar se muestran los mensajes creados por nosotros .

Si la característica ***track_errores*** está habilitada, cualquier mensaje de error generado por la expresión será almacenado en la variable ***\$php_errormsg***. La variable será sobrescrita en cada instancia de error .

```
<?php
    echo "División entre 0"."<br>";
    $numerador=2;
    $denominador=0;
    echo "El Operador @ hará que se oculte el mensaje de error al dividir<br>";
    $a = @($numerador/$denominador);

?>
```

Nota:

El operador @ trabaja solo sobre expresiones.

OPERADORES INCREMENTO / DECREMENTO

Permiten incrementar/decrementar el valor de una variable en una unidad.

El incremento/decremento puede hacerse **antes** o **después** de tomar el valor de la variable

ejemplo	nombre	efecto
++\$a	Preincremento	Incrementa \$a en uno y después devuelve \$a
\$a++	Postincremento	Devuelve \$a y después incrementa \$a en uno
--\$a	Predecremento	Decrementa \$a en uno y después devuelve \$a
\$a--	Postdecremento	Devuelve \$a y después decrementa \$a en uno

OPERADORES LÓGICOS

Enlace: <http://php.net/manual/es/language.operators.logical.php>

ejemplo	nombre	efecto
\$a and \$b	Y	TRUE si tanto \$a como \$b son ciertos
\$a or \$b	O	TRUE si \$a o \$b son ciertos
\$a xor \$b	O exclusiva	TRUE si \$a o \$b son ciertos, pero no los dos a vez
!\$a	Negación	TRUE si \$a no es cierto
\$a && \$b	Y	TRUE si tanto \$a como \$b son ciertos
\$a \$b	O	TRUE si \$a o \$b son ciertos

OPERADORES A NIVEL DE BIT:

Enlace: <http://www.php.net/manual/en/language.operators.bitwise.php>

Operadores bit a bit		
Ejemplo	Nombre	Resultado
<code>\$a & \$b</code>	And (y)	Los bits que están activos en ambos <code>\$a</code> y <code>\$b</code> son activados.
<code>\$a \$b</code>	Or (o inclusivo)	Los bits que están activos ya sea en <code>\$a</code> o en <code>\$b</code> son activados.
<code>\$a ^ \$b</code>	Xor (o exclusivo)	Los bits que están activos en <code>\$a</code> o en <code>\$b</code> , pero no en ambos, son activados.
<code>~ \$a</code>	Not (no)	Los bits que están activos en <code>\$a</code> son desactivados, y viceversa.
<code>\$a << \$b</code>	Shift left(desplazamiento a izquierda)	Desplaza los bits de <code>\$a</code> , <code>\$b</code> pasos a la izquierda (cada paso quiere decir "multiplicar por dos").
<code>\$a >> \$b</code>	Shift right (desplazamiento a derecha)	Desplaza los bits de <code>\$a</code> , <code>\$b</code> pasos a la derecha (cada paso quiere decir "dividir por dos").

PRECEDENCIA DE OPERADORES.

Precedencia de operadores (de mayor a menor)

++, --
*, /, %
+, -
<, <=, >, >=
==, !=
&&
||
and
or

7.- ESTRUCTURAS DE CONTROL

7.1.- ESTRUCTURAS DE CONTROL CONDICIONALES

Sentencias de selección

- `if-else`
- `switch`
- `condicional` `compacta`

Sentencias de iteración o repetitivas:

- `while`
- `for`
- `foreach`

SENTENCIAS DE SELECCIÓN

IF- IF/ELSE-IF/ELSEIF

```
if (condición)
    sentencia
```

```
if (condición)
    sentencia 1
else
    sentencia 2
```

```
if (condición1)
    sentencia 1
else if (condición2)
    sentencia 2
...
else if (condición n)
    sentencia n
else
    sentencia n+1
```

Ejemplo IF:

sintaxisIf.php

```
<?php
    $Numero = 10;
    $Otro = 15;
    if ($Numero < $Otro)
    {
        echo "$Numero es menor que $Otro";
    }
?>
```

Ejemplo IF-ELSE:

sintaxisIfElse.php

```
<?php
$dato1=25;
$dato2=35;
if ($dato1 > $dato2)
{
    echo "El mayor: <B> $dato1 </B>";
}
else
{
    echo "El mayor: <B> $dato2 </B>";
}
?>
```

Ejemplo **IF-ELSEIF**:

sintaxisIfElseIf.php

```
$grupo = "B";  
if ($grupo == "A")  
{  
    echo "El grupo es A";  
}  
    elseif ($grupo == "B")  
    {  
        echo "El grupo es B";  
    }  
    elseif ($grupo == "C")  
    {  
        echo "El grupo es C";  
    }  
Else  
{  
    echo "El grupo no es ni A ni B ni C";  
}
```

SWITCH-CASO:

```
switch (expresión)
{
    case valor 1:
        sentencia 1
        break;
    case valor 2:
        sentencia 2
        break;
    ...
    case valor n:
        sentencia n
        break;
    default
        sentencia n+1
}
```

La expresión del caso puede ser integer, float o string

Ejemplo:*sintaxisSwitch.php*

```
<?php
$extension="TXT"; // inicializamos para probar el funcionamiento
switch ($extension)
{
    case ("PDF"):
        $tipo = "Documento Adobe PDF";
        break;

    case ("TXT"):
        $tipo = "Documento de texto";
        break;

    case ("HTML"):
    case ("HTM"):
        $tipo = "Documento HTML";
        break;
    default: $tipo = "Archivo " . $extension;
}
print ($tipo);
?>
```

ESTRUCTURA CONDICIONAL COMPACTA

Sintaxis:

`$variable = (exp1) ? exp2 : exp3;`

primero se evalúa la expresión (exp1), si es cierta se asigna el resultado de evaluar la expresión (exp2) a la variable, en caso contrario se asigna a la variable el resultado de evaluar la expresión 3.

Ejemplo:

sintaxisIfCompacta.php

```
<?php
$a = 5;
$b = 8;
$c = 3;
// Si $a < $b $maximo = $b sino igual a $a
$maximo = ($a < $b) ? $b : $a;
// ya tenemos el mayor de $a y $b
$maximo = ($maximo < $c) ? $c : $maximo;
// ya tenemos el mayor del anterior y el último número
// Escribimos el resultado
echo "El máximo de los tres valores es: $maximo";
?>
```