Programación Web con PHP y MySQL– Nivel 2

Unidad 4: Introducción a PHP

Contacto: consultas@elearning-total.com
Web: www.elearning-total.com

Indice

Unidad 4: Introducción a PHP - Variables

Objetivos

Que el alumno logre:

- Aprender los conceptos de tabla, columna, fila o registro y valor de un dato dentro de la tabla.
- Aprender a definir una llave o clave primaria y cuál es el propósito de la misma.

Variables, Constantes y Operadores

VARIABLES

Una variable consiste en un elemento al cual le damos un nombre y le atribuimos un determinado tipo de información. Las variables pueden ser consideradas como la base de la programación.

De este modo podríamos escribir en un lenguaje ficticio:

a="perro"

b="muerde"

La variable que nosotros llamamos "a" posee un elemento de información de tipo texto que es "perro". Asimismo, la variable "b" contiene el valor "muerde".

Podríamos definir una tercera variable que fuese la suma de estas dos:

c=a+b

Si introdujésemos una petición de impresión de esta variable en nuestro lenguaje ficticio:

imprimir(c)

El resultado podría ser:

perro muerde

Podríamos de la misma forma trabajar con variables que contuviesen números y construir nuestro programa:

a=3

b=4

c=a+b

imprimir(c)

El resultado de nuestro programa sería:

7

Profundizando nuestra definición anterior, una variable en PHP es un espacio de memoria (en el caso de un lenguaje de lado de servidor, un espacio en la memoria del servidor) que guarda un valor, y tiene un identificador (el nombre de la variable) y un tipo de dato (enteros, reales, cadenas, arrays, objetos).

No es necesario indicar de qué tipo de dato es una variable, el intérprete de PHP se da cuenta solo según su contenido, y en muchos casos también "convierte" el tipo de dato de una variable a otro si es conveniente según el contexto. ¿Para qué usamos variables? Para guardar información en ellas que luego usaremos varias veces a lo largo del código de la página.

En PHP las variables se representan como un signo de pesos seguido por el nombre de la variable. El nombre de la variable es sensible a minúsculas y mayúsculas.

Los nombres de variables siguen las mismas reglas que otras etiquetas en PHP. Un nombre de variable válido tiene que empezar con una letra o un guión bajo (underscore), seguido de cualquier número de letras, números y rayas.

Veamos algunos ejemplos de uso de variables:

```
<?php
/* Caso 1 */
$Cadena = "Hola Mundo"; //Asignación de una variable tipo string
/* Caso 2 */
echo $Cadena."<br/>"; //Muestra por pantalla el valor de $Cadena
/* Caso 3 */
$NumeroUno = 1; //Asignación de una variable de tipo int.
/* Caso 4 */
$NumeroDos = "2"; //Asignación de una variable tipo string.
/* Caso 5 */
echo $NumeroUno + $NumeroDos; //Muestra por pantalla '3'.
/* Caso 6 */
echo $NumeroDos + $Cadena; //Muestra por pantalla '2', no realiza la suma.
?>
```

En el **CASO 1** vemos un ejemplo simple de asignación de una variable con el operador de asignación **=**.

En el **CASO 2**, vemos un ejemplo de uso de la instrucción **echo** con una variable en vez de texto y otra cosa a destacar: **concatenación de variables.**

La concatenación de variables sirve para unir el valor de una variable con el valor de otra mediante el operador **”.”** y que el resultado de sus uniones sea una cadena formada por sus valores.

En el **CASO 2**, echo estaría mostrando “Hola Mundo
”, provocando que después de mostrar “Hola Mundo” por pantalla, haya un salto de línea.

En el **CASO 3** se muestra un ejemplo de asignación de un número entero a una variable.

En el **CASO 4** se muestra un ejemplo de asignación de una cadena te texto a una variable.

El **CASO 5** y el **CASO 6** muestran que el intérprete de PHP se da cuenta de cuándo puede operar entre números y strings y cuándo no.

Tipos de datos que guardan las variables

PHP soporta ocho tipos primitivos.

Cuatro tipos escalares o simples:

- boolean (boleano)
- integer (entero)
- float (número de punto-flotante, también conocido como 'double o doble')
- string (cadena de caracteres)

Dos tipos compuestos:

- array (arreglos o vectores)
- object (objetos)

Y finalmente dos tipos especiales:

- resource (recurso)
- NULL (nulo)

Analizaremos ahora los cuatro primeros tipos, los más usados en PHP, más adelante veremos el resto.

También puede encontrar algunas referencias al tipo "double". Considere al tipo doublé como el mismo que float, los dos nombres existen solo por razones históricas.

El tipo de una variable usualmente no es declarado por el programador; en cambio, es decidido en tiempo de compilación por PHP dependiendo del contexto en el que es usada la variable.

Si desea chequear el tipo y valor de una cierta expresión, use **var_dump()**. Por ejemplo, partiendo del siguiente código:

```
<?php
$b = 3.1;
$c = true;
var_dump($b, $c);
?>
```

El resultado del ejemplo sería:

```
float(3.1)
bool(true)
```

Si tan solo se desea saber el tipo de la variable, use **gettype()**. Para chequear por un cierto tipo, no use **gettype()**; en su lugar utilice las funciones ***is_type()*** junto con un condicional "if" (lo veremos más adelante).

Algunos ejemplos:

```
<?php
$bool = TRUE; // un valor booleano
$str = "foo"; // una cadena
$int = 12; // un entero
echo gettype($bool); // imprime "boolean"
echo gettype($str); // imprime "string"
echo gettype($int); // imprime "integer"
?>
```

Si quisiera forzar la conversión de una variable a cierto tipo, puede moldear la variable, o usar la función **settype()** sobre ella.

Una variable puede ser evaluada con valores diferentes en ciertas situaciones, dependiendo del tipo que posee en cada momento.

```
setType($variable,"nuevo_tipo");
```

La función setType() actualiza el tipo de $variable a "nuevo_tipo" y devuelve un boleano indicando si hubo éxito o no en la conversión.

Entre "nuevo_tipo" tenemos:

- "integer"
- "double"
- "string"
- "array"
- "object"

También podemos hacer que una variable se comporte como un tipo determinado forzándola, de la misma manera a como se hace en el lenguaje C.

$variable = "23";

$variable = (int) $variable;

Los forzados permitidos son:

- (int), (integer) - fuerza a entero (integer)
- (real), (double), (float) - fuerza a doble (double)
- (string) - fuerza a cadena (string)
- (array) - fuerza a array (array)
- (object) - fuerza a objeto (object)

Booleanos

Este es el tipo más simple. Un boolean expresa un valor de verdad. Puede ser TRUE or FALSE (verdadero o falso).

Nota: El tipo booleano fue introducido en PHP 4.

Sintaxis

Para especificar un literal booleano, use alguna de las palabras clave TRUE o FALSE.

Ambas son insensibles a mayúsculas y minúsculas.

```
<?php
$foo = True; // asignar el valor TRUE a $foo
?>
```

Cuando se quiere mostrar el valor de una variable del tipo boolean -1 es considerado TRUE, como cualquier otro número diferente a cero (ya sea negativo o positivo)

Enteros

Un integer o entero, es un número del conjunto Z = {..., -2, -1, 0, 1, 2, ...}.

Sintaxis

Los enteros pueden ser especificados en notación decimal (base-10), hexadecimal (base-16) u octal (base-8), opcionalmente precedidos por un signo (- o +).

Si usa la notación octal, debe preceder el número con un *0* (cero), para usar la notación hexadecimal, preceda el número con *0x*.

Ejemplo. Literales tipo entero

```
<?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)
?>
```

El tamaño de un entero es dependiente de la plataforma, aunque un valor máximo de aproximadamente dos billones es el valor usual (lo que es un valor de 32 bits con signo). PHP no soporta enteros sin signo.

Desbordamiento de enteros

Si se especifica un número más allá de los límites del tipo integer, será interpretado en su lugar como un float.

Asimismo, si se realiza una operación que resulta en un número más allá de los límites del tipo integer, un float es retornado en su lugar.

```
<?php
$numero_grande = 2147483647;
var_dump($numero_grande);
// salida: int(2147483647)
$numero_grande = 2147483648;
var_dump($numero_grande);
// salida: float(2147483648)
// esto no ocurre con los enteros indicados como hexadecimales:
var_dump( 0x100000000 );
// salida: int(2147483647)
$millon = 1000000;
$numero_grande = 50000 * $millon;
var_dump($numero_grande);
// salida: float(50000000000)
?>
```

Números de punto flotante

Los números de punto flotante (también conocidos como "flotantes", "dobles" o

"números reales") pueden ser especificados usando cualquiera de las siguientes sintaxis:

```
<?php
$a = 1.234;
$b = 1.2e3;
$c = 7E-10;
?>
```

El tamaño de un flotante depende de la plataforma, aunque un valor común consiste en un máximo de ~1.8e308 con una precisión de aproximadamente 14 dígitos decimales (lo que es un valor de 64 bits en formato IEEE).

Precisión del punto flotante

Es bastante común que algunas fracciones decimales simples como *0.1* o *0.7* no puedan ser convertidas a su representación binaria interna sin perder un poco de precisión. Esto puede llevar a resultados confusos: por ejemplo, *floor((0.1+0.7)*10)* usualmente devolverá *7* en lugar del esperado *8* ya que el resultado de la representación interna es en realidad algo como *7.9999999999...*.

Esto se encuentra relacionado al hecho de que es imposible expresar de forma exacta algunas fracciones en notación decimal con un número finito de dígitos. Por ejemplo,

1/3 en forma decimal se convierte en *0.3333333. . ..*

Así que nunca confíe en resultados de números flotantes hasta el último dígito, y nunca compare números de punto flotante para conocer si son equivalentes.

Cadenas

Un valor string es una serie de caracteres. En PHP, un caracter es lo mismo que un byte, es decir, hay exactamente 256 tipos de caracteres diferentes. Esto implica también que PHP no tiene soporte nativo de Unicode. Vea utf8_encode() y utf8_decode() para conocer sobre el soporte Unicode.

El que una cadena se haga muy grande no es un problema. PHP no impone límite práctico alguno sobre el tamaño de las cadenas, así que no hay ninguna razón para preocuparse sobre las cadenas largas.

Sintaxis

Un literal de cadena puede especificarse en tres formas diferentes.

- comillas simples
- comillas dobles
- sintaxis heredoc

Comillas simples

La forma más simple de especificar una cadena sencilla es rodearla de comillas simples (el caracter ').

Para especificar una comilla sencilla literal, necesita escaparla con una barra invertida (\), como en muchos otros lenguajes.

Si una barra invertida necesita aparecer antes de una comilla sencilla o al final de la cadena, necesitará doblarla. Por lo general, no hay necesidad de escapar la barra invertida misma.

A diferencia de las otras dos sintaxis, las variables y secuencias de escape para caracteres especiales no serán expandidas cuando ocurren al interior de cadenas entre comillas sencillas.

```
<?php
echo 'esta es una cadena simple';
echo 'También puede tener saltos de línea embebidos en las cadenas de esta forma, ya que es válido';

echo 'Arnold dijo una vez: "I\'ll be back"';
// Imprime: Arnold dijo una vez: "I'll be back"
echo 'Ha eliminado C:\\*.*?';
// Imprime: Ha eliminado C:\*.*?
echo 'Ha eliminado C:\*.*?';
// Imprime: Ha eliminado C:\*.*?
echo 'Esto no va a expandirse: \n una nueva línea';
// Imprime: Esto no va a expandirse: \n una nueva línea
echo 'Las variables no se $expanden $tampoco';
// Imprime: Las variables no se $expanden $tampoco
?>
```

Comillas Dobles

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

Tabla: Caracteres escapados

secuencia	*significado*
\n	alimentación de línea (LF o 0x0A (10) en ASCII)
\r	retorno de carro (CR o 0x0D (13) en ASCII)
\t	tabulación horizontal (HT o 0x09 (9) en

	ASCII)
\\	barra invertida
\$	signo de pesos
\"	comilla-doble
\[0-7]{1,3}	la secuencia de caracteres que coincide con la expresión regular es un caracter en notación octal
\x[0-9A-Faf]{1,2}	la secuencia de caracteres que coincide con la expresión regular es un caracter en notación hexadecimal

Pero la característica más importante de las cadenas entre comillas dobles es el hecho de que los nombres de variables serán expandidos.

```
<?php
$ahora="ahora";
$expanden="expanden";
echo "Las variables $ahora si se $expanden";
// Imprime: Las variables ahora si se expanden
?>
```

Heredoc

Otra forma de delimitar cadenas es mediante el uso de la sintaxis heredoc ("<<<"). Debe indicarse un identificador después de la secuencia <<<, luego la cadena, y luego el mismo identificador para cerrar la cita.

El identificador de cierre debe comenzar en la primera columna de la línea.

Asimismo, el identificador usado debe seguir las mismas reglas que cualquier otra etiqueta en PHP: debe contener solo caracteres alfanuméricos y de subrayado, y debe iniciar con un caracter no-dígito o de subrayado.

Aviso
Es muy importante notar que la línea con el identificador de cierre no contenga otros caracteres, excepto quizás por un punto-y-coma (;).

Esto quiere decir en especial que el identificador no debe usar sangría, y no debe haber espacios o tabuladores antes o después del punto-y-coma.

Es importante también notar que el primer caracter antes del identificador de cierre debe ser un salto de línea, tal y como lo defina su sistema operativo. Esto quiere decir \r en Macintosh, por ejemplo.

Si esta regla es rota y el identificador de cierre no es "limpio", entonces no se considera un identificador de cierre y PHP continuará en busca de uno.

Si, en tal caso, no se encuentra un identificador de cierre apropiado, entonces un error del analizador sintáctico resultará con el número de línea apuntando al final del script.

El texto heredoc se comporta tal como una cadena entre comillas dobles, sin las comillas dobles. Esto quiere decir que no necesita escapar tales comillas en sus bloques heredoc, pero aún puede usar los códigos de escape listados anteriormente.

Las variables son expandidas, aunque debe tenerse el mismo cuidado cuando se expresen variables complejas al interior de un segmento heredoc, al igual que con otras cadenas.

Ejemplo de uso de una cadena heredoc:

```
<?php
$cadena = <<<FIN
Ejemplo de una cadena
que se extiende por varias líneas
usando la sintaxis heredoc.
FIN;
?>
```

Nota: El soporte heredoc fue agregado en PHP 4.

Cambio del tipo de las variables en PHP

PHP no requiere que indiquemos el tipo que va a contener una variable, sino que lo deduce del valor que asignemos a la variable. Asimismo, se encarga de actualizar automáticamente el tipo de la variable cada vez que le asignamos un nuevo valor.

Por ello, para cambiar el tipo de una variable simplemente le asignamos un valor con un nuevo tipo.

Forzado

En cualquier caso, podemos forzar una variable para que cambie de tipo con la función

```
setType().
setType ($variable,"nuevo_tipo");
```

La función **setType()** actualiza el tipo de $variable a "nuevo_tipo" y devuelve un boleano indicando si hubo éxito o no en la conversión.

Entre "nuevo_tipo" tenemos:

- "integer"
- "double"
- "string"
- "array"
- "object"

También podemos hacer que una variable se comporte como un tipo determinado forzándola, de la misma manera a como se hace en el lenguaje C.

```
$variable = "23";
$variable = (int) $variable;
```

Los forzados permitidos son:

- (int), (integer) - fuerza a entero (integer)
- (real), (double), (float) - fuerza a doble (double)
- (string) - fuerza a cadena (string)
- (array) - fuerza a array (array)
- (object) - fuerza a objeto (object)

VARIABLES DE SISTEMA EN PHP

Dada su naturaleza de lenguaje de lado servidor, PHP es capaz de darnos acceso a toda una serie de variables que nos informan sobre nuestro servidor y sobre el cliente. La información de estas variables es atribuida por el servidor y en ningún caso nos es posible modificar sus valores directamente mediante el script. Para hacerlo es necesario influir directamente sobre la propiedad que definen.

Existen multitud de variables de este tipo, algunas sin utilidad aparente y otras realmente interesantes y con una aplicación directa para nuestro sitio web. Aquí os enumeramos algunas de estas variables y la información que nos aportan:

Las nombramos a continuación:

$HTTP_USER_AGENT

Nos informa principalmente sobre el sistema operativo y tipo y versión de navegador utilizado por el internauta. Su principal utilidad radica en que, a partir de esta información, podemos redireccionar nuestros usuarios hacia páginas optimizadas para su navegador o realizar cualquier otro tipo de acción en el contexto de un navegador determinado.

$HTTP_ACCEPT_LANGUAGE

Nos devuelve la o las abreviaciones de la lengua considerada como principal por el navegador. Esta lengua o lenguas principales pueden ser elegidas en el menú de opciones del navegador. Esta variable resulta también extremadamente útil para enviar al internauta a las páginas escritas en su lengua, si es que existen.

$HTTP_REFERER

Nos indica la URL desde la cual el internauta ha tenido acceso a la página. Muy interesante para generar botones de "Atrás" dinámicos o para crear nuestros propios sistemas de estadísticas de visitas.

$PHP_SELF

Nos devuelve una cadena con la URL del script que está siendo ejecutado. Muy interesante para crear botones para recargar la página.

$HTTP_GET_VARS

Se trata de un array que almacena los nombres y contenidos de las variables enviadas al script por URL o por formularios GET.

$HTTP_POST_VARS

Se trata de un array que almacena los nombres y contenidos de las variables enviadas al script por medio de un formulario POST.

$HTTP_COOKIES_VARS

Se trata de un array que almacena los nombres y contenidos de las cookies. Veremos qué son más adelante.

$PHP_AUTH_USER

Almacena la variable usuario cuando se efectúa la entrada a páginas de acceso restringido. Combinado con $PHP_AUTH_PW resulta ideal para controlar el acceso a las páginas internas del sitio.

$PHP_AUTH_PW

Almacena la variable password cuando se efectúa la entrada a páginas de acceso restringido. Combinado con $PHP_AUTH_USER resulta ideal para controlar el acceso a las páginas internas del sitio.

$REMOTE_ADDR

Muestra la dirección IP del visitante.

$DOCUMENT_ROOT

Nos devuelve el path físico en el que se encuentra alojada la página en el servidor.

$PHPSESSID

Guarda el identificador de sesión del usuario. Veremos más adelante en qué consisten las sesiones. No todas estas variables están disponibles en la totalidad de servidores o en determinadas versiones de un mismo servidor. Además, algunas de ellas han de ser previamente activadas o definidas por medio de algún acontecimiento. Así, por ejemplo, la variable $HTTP_REFERER no estará definida a menos que el internauta acceda al script a partir de un enlace desde otra página.

Variables superglobales

A partir de PHP 4.1.0, se dispone de un conjunto de varibles de tipo array que mantienen información del sistema, llamadas superglobales porque se definen automáticamente en un ámbito global.

Estas variables hacen referencia a las mismas que se accedían antes por medio de los arrays del tipo $HTTP_*_VARS. Éstas todavía existen, aunque a partir de PHP 5.0.0 se pueden desactivar con la directiva register_long_arrays.

La lista de estas variables, extraída directamente de la documentación de PHP es la siguiente:

$GLOBALS
Contiene una referencia a cada variable disponible en el espectro de las variables del script. Las llaves de esta matriz son los nombres de las variables globales. $GLOBALS existe dese PHP 3.

$_SERVER
Variables definidas por el servidor web ó directamente relacionadas con el entorno en don el script se está ejecutando. Análoga a la antigua matriz $HTTP_SERVER_VARS (la cual está todavía disponible, aunque no se use).

$_GET
Variables proporcionadas al script por medio de HTTP GET. Análoga a la antigua matriz $HTTP_GET_VARS (la cual está todavía disponible, aunque no se use).

$_POST
Variables proporcionadas al script por medio de HTTP POST. Análoga a la antigua matriz $HTTP_POST_VARS (la cual está todavía disponible, aunque no se use).

$_COOKIE
Variables proporcionadas al script por medio de HTTP cookies. Análoga a la antigua matriz $HTTP_COOKIE_VARS (la cual está todavía disponible, aunque no se use).

$_FILES
Variables proporcionadas al script por medio de la subida de ficheros vía HTTP .

Análoga a la antigua matriz $HTTP_POST_FILES (la cual está todavía disponible, aunque no se use). Vea también Subiendo ficheros por método POST para más información.

$_ENV

Variables proporcionadas al script por medio del entorno. Análoga a la antigua matriz $HTTP_ENV_VARS (la cual está todavía disponible, aunque no se use).

$_REQUEST

Variables proporcionadas al script por medio de cualquier mecanismo de entrada del usuario y por lo tanto no se puede confiar en ellas. La presencia y el orden en que aparecen las variables en esta matriz es definido por la directiva de configuración variables_order.

Esta matriz no tiene un análogo en versiones anteriores a PHP 4.1.0. Vea también import_request_variables().

$_SESSION

Variables registradas en la sesión del script. Análoga a la antigua matriz **$HTTP_SESSION_VARS** (la cual está todavía disponible, aunque no se use.)

CONSTANTES

Una constante es un identificador para expresar un valor simple. Como el nombre sugiere, este valor no puede variar durante la ejecución del script.

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

El nombre de una constante sigue las mismas reglas que cualquier etiqueta en PHP. Un nombre de constante válido empieza con una letra o un caracter de subrayado, seguido por cualquier número de letras, números, o subrayados.

Se podrían expresar mediante la siguiente expresión regular: *[a-zA-Z_\x7f-\xff][a-zAZ0-*

9_\x7f-\xff

El alcance de una constante es global, Es decir, es posible acceder a ellas sin preocuparse por el ámbito de alcance.

Sintaxis

Se puede definir una constante usando la función define(). Una vez definida, no puede ser modificada ni eliminada.

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

Para obtener el valor de una constante solo es necesario especificar su nombre. A diferencia de las variables, no se tiene que especificar el prefijo *$*.

También se puede utilizar la función constant(), para obtener el valor de una constante, en el caso de que queramos expresarla de forma dinámica Usa la función get_defined_constants() parar obtener una lista de todas las constantes definidas.

> **Nota:** Las constantes y las variables (globales) se encuentran en un espacio de nombres distinto. Esto implica que por ejemplo *TRUE* y *$TRUE* son diferentes.

Si usas una constante todavía no definida, PHP asume que estás refiriéndote al nombre de la constante en sí. Se lanzará un aviso si esto sucede. Usa la función defined() para comprobar la existencia de dicha constante.

Estas son las diferencias entre constantes y variables:

- Las constantes no son precedidas por un símbolo de pesos ($)
- Las contantes solo pueden ser definidas usando la función **define()** , nunca por simple asignación
- Las constantes pueden ser definidas y accedidas sin tener en cuenta las reglas de alcance del ámbito.
- Las constantes no pueden ser redefinidas o eliminadas despues de establecerse;
- Las constantes solo puede albergar valores escalares

Ejemplo. Definiendo constantes:

```
<?php
define("CONSTANTE", "Hola Mundo.");
echo CONSTANTE; // muestra " Hola Mundo."
echo Constante; // muestra "Constante" y un aviso del sistema.
?>
```

OPERADORES

Las variables, como base de información de un lenguaje, pueden ser creadas, modificadas y comparadas con otras por medio de los llamados operadores. En las clases anteriores hemos utilizado en nuestros ejemplos algunos de ellos.

Ahora pretendemos listar los más importantes y así dar constancia de ellos para futuros ejemplos.

Según el manual oficial de PHP, un operador es algo a lo que usted entrega uno o más valores (o expresiones, en jerga de programación) y produce otro valor (de modo que la construcción misma se convierte en una expresión).

Operadores de asignación

Operador	Función
=	Asigna el valor de la derecha al de la izquierda
+=	Suma el valor de la derecha con el de la izquierda, y asigna el total al de la izquierda
-=	Resta el valor de la derecha con el de la izquierda, y asigna la diferencia de la izquierda
*=	Multiplica el valor de la derecha con el de la izquierda, y asigna el producto al de la izquierda
/=	Divide el valor de la derecha con el de la izquierda, y asigna el cuociente al de la izquierda
%=	Divide el valor de la derecha con el de la izquierda, y asigna el resto al de la izquierda
.=	Concatena la cadena de la derecha con la de la izquierda, y asigna el conjunto al de la izquierda

El operador más simple es el operador de asignación, el cual ya lo hemos visto, consta del signo igual "=" y sirve para asignar o cambiar el valor de una variable.

A primera vista, se podría pensar en él como "es igual a". No es así.

Lo que quiere decir en realidad es que el operando de la izquierda recibe el valor de la expresión a la derecha (es decir, "se define a").

El valor de una expresión de asignación es el valor que se asigna. Es decir, el valor de "$a = 3" es 3.

Esto le permite hacer alguna que otra cosa curiosa:

```
<?php
```

```
$a = ($b = 4) + 5; // $a es igual a 9 ahora, y $b ha sido definido a 4.
?>
```

En conjunto con el operador básico de asignación, existen "operadores combinados" para todas las operaciones de aritmética binaria y de cadenas, que le permiten usar un valor en una expresión y luego definir su valor como el resultado de esa expresión. Por ejemplo:

```
<?php
$a = 3;
$a += 5; // define $a como 8, como si hubiésemos dicho: $a = $a + 5;
$b = "Hola ";
$b .= "a todos!"; // define $b como "Hola a todos!", tal como $b = $b . "a todos!";
?>
```

El operador ".=" es muy utilizado, ya que permite concatenar (o unir) dos o más variables. Se usa preferentemente con las variables de texto, aunque puede usarse con cualquier variable, incluso entre variables y textos entrecomillados. Los números los trata como si fuesen textos, poniendo unos a continuación de otros. Veamos cómo funciona:

```
<?php
$cadena_uno = "Lorem ipsum";
$cadena_dos = " dolor sit amet";
$cadena_uno .= $cadena_dos;
echo $cadena_uno;
?>
```

Operadores Aritméticos

Los operadores aritméticos en el PHP nos permiten trabajar matemáticamente con valores numéricos. Así, mediante reglas básicas aritméticas y la sintaxis del PHP podemos usar y realizar cálculos a voluntad con los valores obtenidos.

Operador	Función
+	Suma dos valores numéricos
-	Resta dos valores numéricos
*	Multiplica dos valores numéricos
/	Divide dos valores numéricos
%	Obtiene el resto al dividir dos valores
( )	Agrupa valores.
++	Aumenta un valor numérico en una unidad
--	Disminuye un valor numérico en una unidad

Ejemplos:

```
<?php
$calculo1=12/4; // 12/4=3
$calculo2=3*2; // 3*2=6
$calculo3=20%3; // 20/3=6, resto=2
echo $calculo1+$calculo2-$calculo3;
?>
```

```
<?php
$numero=24;
echo $numero."<br/>";
echo $numero++; // Lo muestra primero y luego lo incrementa, por ello es 24
echo "<br/>";
echo $numero; // Volvemos a mostrarlo y ya vale 25
echo "<br/>";
echo $numero--; // Lo muestra primero y luego lo decrementa, por ello es 25
echo "<br/>";
echo $numero; // Volvemos a mostrarlo y ya vale 24
echo "<br/>";
echo ++$numero; // Lo incrementa primero y luego lo muestra, por ello es 25
echo "<br/>";
echo --$numero; // Lo decrementa primero y luego lo muestra, por ello es 24
echo "<br/>";
?>
```

Si queremos lograr alguna prioridad de algún calculo, debemos asociarlo entre paréntesis.

```php
<?php
$uno = 1;
$dos = 10%8; // 10/8=1, en lo que el resto es 2
$tres = 3;
$cuatro = 4;
$uno++; // la variable $uno ahora vale 2, ya que 1+1=2
$dos--; // la variable $dos ahora vale 1, ya que 2-1=1
$calculo = 1 + 5 * 3; // Da como resultado 16
echo $calculo;
$calculo2 = 1 + (5 * 3); // Da como resultado 18
echo $calculo2;
?>
```

Operadores de Comparación

Los operadores de comparación, como su nombre indica, le permiten comparar dos valores. Si compara un entero con una cadena, la cadena es convertida a un número. Si compara dos cadenas numéricas, ellas son comparadas como enteros.

Ejemplo	Nombre	Resultado
$a == $b	Igual	**TRUE** si $a es igual a $b.
$a === $b	Idéntico	**TRUE** si $a es igual a $b, y son del mismo tipo. (A partir de PHP 4)
$a != $b	Diferente	**TRUE** si $a no es igual a $b.
$a <> $b	Diferente	**TRUE** si $a no es igual a $b.
$a !== $b	No idénticos	**TRUE** si $a no es igual a $b, o si no son del mismo tipo. (A partir de PHP 4)
$a < $b	Menor que	**TRUE** si $a es escrictamente 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.

```php
<?php
$a = 8;
$b = 3;
```

```
$c = 3;
echo $a == $b,"<br>"; //imprime 1 porque es verdadero
echo $a != $b,"<br>"; //imprime 1 porque es verdadero
echo $a < $b,"<br>"; //imprime vacío " " porque es falso
echo $a > $b,"<br>"; //imprime 1 porque es verdadero
echo $a >= $c,"<br>"; //imprime 1 porque es verdadero
echo $b <= $c,"<br>"; //imprime 1 porque es verdadero
?>
```

Operadores Lógicos

Los operadores lógicos son usados para evaluar varias comparaciones, combinando los posibles valores de estas.

Ejemplo	Nombre	Resultado
$a and $b	Y	**TRUE** si tanto $a como $b son **TRUE**.
$a or $b	O	**TRUE** si cualquiera de $a o $b es **TRUE**.
$a xor $b	O exclusivo (Xor)	**TRUE** si $a o $b es **TRUE**, pero no ambos.
! $a	No	**TRUE** si $a no es **TRUE**.
$a && $b	Y	**TRUE** si tanto $a como $b son **TRUE**.
$a || $b	O	**TRUE** si cualquiera de $a o $b es **TRUE**.

Resumen

En esta Unidad…

En la presente unidad trabajamos con los conceptos básicos del lenguaje PHP

En la próxima Unidad…

En la próxima unidad trabajaremos con variables y operadores.