Manejo de datos en PHP

Comillas **4**
Comillas simples 4
Escapar caracteres reservados 4
Comillas dobles 4
Uso de expresiones complejas en strings 5
Objetos **5**
Acceso a la información de objetos 5
Variables variables **6**
Strings **6**
Extracción de los caracteres de un string 6
Extracción de los espacios de un string 7
Convertir todos los caracteres de un string en minúsculas. 7
Convertir todos los caracteres de un string en mayúsculas. 7
Capitalizar string. 7
Primera letra en minúscula 8
Reemplazar 8
Formatear string 8
Eliminar etiquetas HTML de un string 9
Almacenamiento de cadenas multibyte. **9**
Uso de expresiones regulares **9**
PHPUnit **10**
Variables por referencia **11**
Funciones anónimas **11**
Arrays **12**
Tipos de Arrays 12
Simple 12
Array asociativo 13
Funciones para arrays 13
array_walk() 13
sort() 14
rsort() 14
ksort() 14
krsort() 14
array_slice() 14
array_chunk() 14
array_shift() 14

array_pop() 14
array_unshift() 14
array_push() 15
array_flip() 15
array_diff() 15
array_merge() 15
list() 15

Comillas

Comillas simples

Las comillas simples nos permiten guardar texto simple de una o varias líneas:

Código	<?php echo 'Línea de texto'; ?>
Output	Línea de texto

Escapar caracteres reservados

Como podemos ver, las comillas simples son elementos reservados en los que podemos asignar strings. Si quisiéramos incluir una comilla simple en nuestro string tendremos que indicar de alguna manera que no queremos cerrar el string. Esto lo hacemos con el uso de backslash (\).

Código	<?php echo 'Entonces Tomás dijo: \'¡Señor mío!\''; ?>
Output	Entonces Tomás dijo: '¡Señor mío!'

Si quisiéramos hacer uso de un backslash en un string, podemos añadir un doble backslash:

Código	<?php echo 'Hacemos uso de \\ para escapar texto'; ?>
Output	Hacemos uso de \ para escapar texto

Comillas dobles

Cuando el string está delimitado con comillas dobles, PHP es capaz de interpretar una mayor lógica dentro de su composición.

Código	<?php $nombre = 'Rick'; echo "Mi nombre es $nombre"; ?>
Output	Mi nombre es Rick

Uso de expresiones complejas en strings

Cuando a un string se le anida sintaxis de PHP compleja (como atributos de objetos) hace falta encapsularla de manera que no se pueda confundir con el string, esto lo hacemos con el uso de corchetes.

Código	<?php $series = ["serie1" => ["nombre" => "Breaking Bad", "IMDB" => 9], "serie2"=>["nombre" => "Ozark", "IMDB" => 8.8]]; echo "La serie número 1 es: {$series['serie1']['nombre']}"; ?>
Output	La serie número 1 es: Breaking Bad

Objetos

Acceso a la información de objetos

Podemos acceder a los atributos de un objeto haciendo uso de “->”:

Código	<?php
	class Persona
	{
	public $nombre = "José";
	}
	$jose = new persona;
	
	echo "$jose->nombre quiere hacer cosas cool.";
	?>

Output	José quiere hacer cosas cool.

Variables variables

Podemos crear variables dinámicas a través del nombramiento de variables a partir de otras variables:

Código	<?php $variable1 = "variable2"; $variable2 = "Buenos días, estás accediendo a mí a través de una variable variable :)"; echo $variable1; echo " "; echo $$variable1; ?>
Output	variable2 Buenos días, estás accediendo a mí a través de una variable variable :)

Strings

Extracción de los caracteres de un string

Para poder extraer el carácter concreto de un string, podemos tratarlo como si de un array se tratara.

Código	<?php $data = "Lorem ipsum dolor"; echo $data[0]; ?>
Output	L

Además, podemos hacer uso de la función substr() para extraer un intervalo de caracteres.

Código	<?php $data = "Lorem ipsum dolor sit amet, consectetur adipiscing elit. In vehicula lacus eros, in porta lacus volutpat a. Sed malesuada porta leo non consequat"; echo substr($data, 0, 50) . '... ver más.'; ?>
Output	Lorem ipsum dolor sit amet, consectetur adipiscing... ver más.

Extracción de los espacios de un string

Podemos eliminar todos los espacios iniciales o finales de un string con la función trim().

Código	<?php $data = " hola "; echo trim($data); ?>
Output	hola

Convertir todos los caracteres de un string en minúsculas.

Podemos convertir un string en minúsculas con el uso de la función strtolower()

Código	<?php $data = "Este es UN TEXTO."; echo strtolower($data); ?>
Output	este es un texto.

Convertir todos los caracteres de un string en mayúsculas.

Podemos convertir un string en minúsculas con el uso de la función strtoupper()

Código	<?php $data = "Este es UN TEXTO."; echo strtoupper($data); ?>
Output	ESTE ES UN TEXTO.

Capitalizar string.

Podemos convertir la primera letra de un string en mayúscula con el uso de la función ucfirst()

Código	<?php $data = "era un día como cualquier otro"; echo ucfirst($data); ?>
Output	Era un día como cualquier otro

Primera letra en minúscula

Podemos convertir la primera letra de un string en minúscula con el uso de la función lcfirst()

Código	<?php $data = "Buenos días"; echo lcfirst($data); ?>
Output	buenos días

Reemplazar

Podemos reemplazar texto de un string con la función str_replace()

Código	<?php $data = "php es un lenguaje"; echo str_replace(' ', '-', $data); ?>
Output	php-es-un-lenguaje

Formatear string

Podemos rellenar la cantidad de caracteres que tiene un string para cumplir con un formato con la función str_pad().

Código	<?php
	$code = 39;
	echo str_pad($code, 9, '#');
	echo "
	";
	echo str_pad($code, 9, '#', STR_PAD_BOTH);
	echo "
	";
	echo str_pad($code, 9, '#', STR_PAD_LEFT);
	echo "
	";
	echo str_pad($code, 9, '#', STR_PAD_RIGHT);
	?>

Output	39####### ###39#### #######39 39#######

Eliminar etiquetas HTML de un string

Podemos filtrar y eliminar todas las etiquetas HTML de un string con la función strip_tags().

Código	<?php $html = " Esto es texto con HTML "; echo strip_tags($html); ?>
Output	Esto es texto con HTML

Almacenamiento de cadenas multibyte.

El número total de caracteres en inglés no es mayor a 256, así que cada carácter puede ser representado usando diferentes secuencias de 8 bits (es decir, 1 byte).

En el caso del español, que tiene signos especiales, no es posible hacerse con únicamente 256 bits:

Código	<?php $html = "Texto con ñ y con acento ó"; echo strtoupper($html); ?>
Output	TEXTO CON ñ Y CON ACENTO ó

Así, según el tipo de variables, PHP puede almacenar múltiples bytes por cada carácter o únicamente uno.

Esta especificación la podemos hacer añadiendo el prefijo “mb_” a a algunas funciones:

Código	<?php $html = "Texto con ñ y con acento ó"; echo mb_strtoupper($html); ?>
Output	TEXTO CON Ñ Y CON ACENTO Ó

Uso de expresiones regulares

Las expresiones regulares nos permite validar si un dato cumple ciertos requisitos o no. Por ejemplo, podemos validar si una oración comienza por un conjunto concreto de caracteres, o validar si un número es lo suficientemente grande o pequeño. Para ello contamos con una sintaxis concreta referente al mundo de las expresiones regulares:

Símbolo	Significado
/	Marca el inicio y final del contenedor de la expresión regular
^	Indica que la expresión debe coincidir al principio del valor a validar
$	Indica que la expresión debe coincidir al final del valor a validar
-	Indica los rangos a los que aplican las expresión regular.
[]	Patrón de la expresión regular.
{}	Condiciones que valida la expresión regular

Por regla general, una expresión regular nos retornará un 1 cuando se cumplan los parámetros y un 0, cuando no. En el caso de PHP podemos hacer uso de expresiones regulares con la función preg_match().

Código	<?php $password = "123456"; echo preg_match('/^[0-9]{6,9}$/', $password); ?>
Output	1

En el ejemplo anterior validamos que la variable $password esté compuesta por valores del 1 al 9 y que tenga de 6 a 9 carácteres de longitud. Al cumplirse, se nos retorna un 1.

PHPUnit

PHPUnit es un framework de testing de PHP que nos permite programar pruebas automatizadas para no tener que hacer tests experimentales.

Por ejemplo, si tenemos un formulario de registro, en vez de probarlo a través del frontend utilizando el navegador, podemos añadir información ejemplar que hará la prueba automáticamente por nosotros con PHPUnit.

Esto será útil a largo plazo cuando los métodos se actualicen o interactúen entre sí, permitiéndonos reconocer errores de manera mucho más fácil y rápida.

Variables por referencia

Cuando se le envía un parámetro a una función, la función realiza una copia de esa variable y las manipulaciones que se le realicen únicamente existirán dentro de la función. Es decir, la variable, fuera de la función permanecerá inalterada.

Para que los cambios que hagamos desde la función se realicen también en la variable global, debemos enviar el argumento por referencia. (Usando & antes de definir el parámetro).

Código	<?php
	$valor = 3;
	
	function suma(&$valor){
	  $valor += $valor;
	}
	
	suma($valor);
	
	echo $valor;
	?>

Output	6

Funciones anónimas

Las funciones anónimas son aquellas a las que no se les atribuye un nombre. Resultan especialmente útiles cuando se envían como parámetros de otras funciones.

Cuando esperamos que una función vaya a recibir como parámetro una función anónima, es una buena práctica hacer uso de la palabra reservada “Closure”, porque de no hacerlo, la función aceptará otros posibles argumentos que podrían prestarse a errores futuros.

Código	<?php
	function greet(Closure $lang, $name){
	  return $lang($name);
	}
	
	$es = function ($name){
	  return "Hola, $name";
	};
	
	$en = function ($name){
	  return "Hello, $name";
	};
	
	echo greet($es, 'José');

	echo " "; echo greet($en, 'José'); ?>
Output	Hola, José Hello, María

Arrays

Tipos de Arrays

Existen dos tipos de arrays: Simples y asociativos.

Simple

Puede crearse de dos maneras. Haciendo uso de la función array:

Código	<?php $series = array(“Breaking Bad”, “Ozark”, “Bojack Horseman”); ?>

O haciendo uso de corchetes:

Código	<?php $series = [“Breaking Bad”, “Ozark”, “Bojack Horseman”]; ?>

La característica principal del array simple es que auto asigna un key numérico ascendente para cada valor que añadamos. En el anterior ejemplo “Breaking bad” tendrá el valor 0, Ozark el valor 1 y así sucesivamente.

Por lo tanto, podemos obtener sus valores de esta manera:

Código	<?php echo $series[0]; ?>
Output	Breaking Bad

Array asociativo

Es similar al array simple, pero en vez de que se asigne un key numérico por defecto, podemos asignar cualquier key.

Podemos crearlo de la misma manera con la que crearíamos un array simple:

Código	<?php $series = array(“serie1”=>”Breaking Bad”, “serie2”=>”Ozark”); ?>

O usando corchetes:

Código	<?php $series = [“serie1”=>”Breaking Bad”, “serie2”=>”Ozark”]; ?>

Y podríamos acceder a sus datos de la siguiente manera:

Código	<?php echo $series['serie1']; ?>
Output	Breaking Bad

Funciones para arrays

array_walk()

Array walk es una función que permite la manipulación de arrays.

Código	<?php $tvshows = [['name' => 'Breaking Bad', 'IMDB' => 9.5], ['name' => 'Ozark', 'IMDB' => 8.2], ['name' => 'The Walking Dead', 'IMDB' => 8.2]];

	function isItGood($tvshow){
	  if($tvshow['IMDB'] > 8.2)
	  {
	  echo "Yep. {$tvshow['name']} is fucking good.
	";
	  }
	  else
	  {
	  echo "Nope. {$tvshow['name']} is trash. Sorry.
	";
	  }
	}
	
	array_walk($tvshows, 'isItGood');
	
	?>

Output	Yep. Breaking Bad is fucking good. Nope. Ozark is trash. Sorry. Nope. The Walking Dead is trash. Sorry.

sort()
Podemos usar la función sort() para ordenar un array en orden ascendente

rsort()
Podemos usar rsort() para ordenar un array descendentemente.

ksort()
Podemos usar ksort() para ordenar un array ascendentemente por su key.

krsort()
Podemos usar ksort() para ordenar un array descendentemente por su key.

array_slice()
Podemos usar array_slice($array, elemento) para remover un elemento en particular de un array.

array_chunk()
Podemos usar array_chunk() para poder dividir un array en trozos de nuevos arrays.

array_shift()
Elimina el primer elemento de un array.

array_pop()
Elimina el último elemento de un array.

array_unshift()
Añade al inicio de un array uno o más elementos.

array_push()
Añade al final de un array uno o más elementos.

array_flip()
Intercambia todas las claves de un array con sus valores asociados.

array_diff()
Nos permite comparar valores semejantes entre dos arrays.

array_merge()
Nos permite fusionar y ordenar dos arrays en uno.

Podemos usar array_merge_recursive en caso de que los keys de ambos arrays sean iguales.

Array_combine nos permite fusionar dos arrays de manera que un array represente los keys y otro, el valor.

list()
List es una herramienta que nos permite guardar cada valor de un array en una variable independiente.

Código	<?php list($variable1, $variable2) = [“Perro”, “Gato”]; echo $variable1; echo “ ”; echo $variable2; ?>
Output	Perro Gato