PHP ARRAY

En este nuevo capítulo de PHP desde Cero estudiaremos los **array**, un tipo de variable que nos permite juntar o agrupar diferentes elementos y manejarlos como si fueran una sola variable. En otros términos un array de una lista de variables.

Una vez que que un array sea creado, se pueden agregar, editar eliminar sus elementos. Los elementos de un array pueden ser de cualquier tipo y se pueden mezclar, es decir no todos los elementos de un array deben ser del mismo tipo. Los elementos en un array se acceden usando una llave, para esto existen dos tipos de llaves y por lo tanto dos tipos de arrays en PHP, los numéricos y los asociativos.

Los arrays numéricos son a los que se acceden con números a los datos que guardan, el primer dato tendría como numero el 0, el segundo el 1 y así sucesivamente. Los arrays asociativos poseen un nombre sobre cada dato y con ese nombre es que se accede al dato específico.

**Crear un Array**

Para crear un array se hace uso de la función array(). Esta puede tomar de cero a varios parámetros y retorna un array el cual es asignado a una variable con el operado de igualación =. Si la función recibe parámetros, entonces el array será creado con estos valores. Los arrays en PHP cambian su tamaño dinámicamente dependiendo la cantidad de datos que tengan, por lo tanto no es necesario establecer un tamaño al momento de su creación como en otros lenguaje.

**<?php**

**$arreglo1 = array();**

**// creación de array vacío**

**$arreglo2 = array('php', 'python', 'ruby', 'java');**

**//ó**

**$arreglo2 = ['php', 'python', 'ruby', 'java'];**

**// creación de array con parámetros.**

**// Los parámetros serán los primeros datos del array.**

**?>**

**Accediendo a un Elemento del Array**

Para acceder a un elemento de un array de tipo numérico se debe especificar el indice del elemento deseado. Para esto se colocar el nombre del array seguido del indice del elemento dentro de unos corchetes []. Recordemos que el primer elemento siempre es el 0.

<?php

$arraymio = array('php', 'python', 'ruby', 'java');

echo $arraymio[1];

// Esto devolverá el valor python, el cual se encuentra en la posición 1?>

**Creando un Array Asociativo**

Ahora que tenemos un array asociativo creado veamos como utilizarlo. Para acceder a cualquier valor del array debemos hacerlo con el nombre de la llave, por lo tanto ponemos el nombre de la variable y entre corchetes el nombre de la llave. Veamos un ejemplo con el array anterior.

<?php

$persona = array('nombre'=>'Juan', 'apellido'=>'valdez', 'direccion'=>'calle 1', 'nacionalidad'=>'venezuela');

echo $persona['apellido'];

// esto nos devolverá el valor de valdez

?>

**Creando un Array Multidimensional**

Un array multidimensional no es más que un array cuyos valores son otros arrays. Un array multidimensional puede ser como una tabla, en donde cada valor del array padre son las filas y cada array interno son las columnas de cada fila. Veamos un ejemplo de una lista de libros con diferentes datos de cada libro

<?php

$libros = array();

$libros[0] = array('titulo'=>'Aprendiendo PHP', 'autor'=>'Ramses Velasquez');

$libros[1] = array('titulo'=>'Aprendiendo a desarrollar', 'autor'=>'CodeHero');

?>

**Accediendo a Elementos de un Array Multidimensional**

Sigamos con la analogía anterior de nuestra lista de libros con filas y columnas, para poder acceder a un dato de esta tabla primero debemos especificar qué libro queremos (la fila) y luego que dato del libro queremos (la columna). Por lo tanto cada vez que queramos acceder a un dato primero colocamos la fila dentro de corchetes y luego la columna dentro de corchetes. Veamos un ejemplo con los datos anteriores:

<?php

$libros = array();

$libros[0] = array('titulo'=>'Aprendiendo PHP', 'autor'=>'Ramses Velasquez');

$libros[1] = array('titulo'=>'Aprendiendo a desarrollar', 'autor'=>'CodeHero');

echo $libros[1]['autor'];

// esto devolver el valor CodeHero, ya que especificamos la fila numero 1 y la columna autor

?>

**Editando, Agregando y Borrando Elementos de un Array**

Un elemento puede ser editado con el operador de asignación y especificando la posición, por ejemplo vamos a editar el segundo elemento de un arreglo:

<?php

$figuras = array('cuadrado', 'triángulo', 'circulo');

$figuras[1] = 'rectángulo';

?>

Para agregar un elemento al final podemos utilizar la función array\_push(). Esta función recibe dos parámetros, el primero es el arreglo en donde se va a insertar y el segundo es el nuevo elemento.

<?php

$figuras = array('cuadrado', 'triángulo', 'circulo');

array\_push($figuras, 'pentagon');

// ahora el array es ('cuadrado', 'triángulo', 'circulo', 'pentágono')

?>

Para insertar al principio del arreglo utilizamos la función **array\_unshift()**, esta también recibe dos parámetros. El primero es el arreglo en donde se va a insertar y el segundo es el nuevo elemento.

<?php

$figuras = array('cuadrado', 'triángulo', 'circulo');

array\_unshift($figuras ,'pentagono');

// ahora el array es ( 'pentágono', 'cuadrado', 'triángulo', 'circulo')

?>

También podemos eliminar el primero y el último de los valores del array con las funciones **array\_pop()** para el ultimo y **array\_shift()** para el primero. Estas funciones solo reciben como parámetro el array a modificar.

<?php

$posiciones = array('Primera', 'Segunda', 'Tercera', 'Ultima');

array\_pop($posiciones);

// el array queda así array('Primera', 'Segunda', 'Tercera')

array\_shift($posiciones);

// el array queda así array('Segunda', 'Tercera')

?>

**Recorrer Arrays**

Muchas veces necesitaremos recorrer todos los elementos de un array para leer los valores o editarlos. Para hacer esto existen varias maneras pero la mejor y más sencilla es utilizar el bucle **foreach().** Este bucle funciona como un **while** o un **for** pero es especial para los arrays y pasa por cada elemento del array. Hay dos maneras de utilizar este bucle: La primera manera es asignar el valor actual del array a una variable que se puede acceder solo dentro del bucle, la sintaxis es la siguiente **foreach( $arreglo as $elemento).** Veamos un ejemplo de esto:

<?php

$colores = array('rojo', 'verde', 'azul');

//Recorro todos los elementos con un bucle fot

for($i=0; $i< count($colores); $i++)

{

//saco el valor de cada elemento

echo $colores[$i];

echo "<br>";

}

// recorrer todos los elemento con bucle foreach

foreach( $colores as $color){

echo 'Color actual '. $color ;

}

// Esto imprimirá lo siguiente

// Color actual rojo

// Color actual verde

// Color actual azul

?>

La segunda forma es para los arrays asociativos y permite obtener en una variable el valor actual y en otra variable la clave de ese valor con la siguiente sintaxis **foreach ($arreglo as $llave => $elemento).** Veamos un ejemplo de esto:

<?php

$persona = array('nombre'=>'Juan', 'apellido'=>'valdez', 'direccion'=>'calle 1', 'nacionalidad'=>'venezuela');

foreach( $persona as $llave => $elemento){

echo 'Llave - '. $llave;

echo 'Elemento - '. $elemento;

}

// Esto imprimira lo siguiente

// Llave - nombre

// Elemento - Juan

// Llave - apellido

// Elemento - valdez

// Llave - direccion

// Elemento - calle 1

// Llave - nacionalidad

// Elemento - venezuela?>

Ejemplo de array bidimensional y cómo recorrerlo:

$equipo\_futbol = array (array("Rooney","Chicharito","Gigs"),

array("Suarez"),

array("Torres","Terry","Etoo")

);

foreach($equipo\_futbol as $equipo)

{

echo "En este equipo juegan: ";

foreach($equipo as $jugador)

{

echo $jugador ." ";

}

echo "<br>";

}

**Otras Funciones para Arrays**

A continuación veremos una lista de algunas funciones que son muy útiles a la hora de trabaje con arrays en PHP.

* **sort():** Sirve para ordenar un array no asociativo, recibe dos parámetros. El primero es el array que se va a modificar y el segundo es el tipo de algoritmo que se va aplicar ( SORT\_NUMERIC, SORT\_STRING, SORT\_REGULAR, estos son los más utilizados).
* **ksort():** Es lo mismo que sort pero funciona para ordenar arrays asociativos por la clave.
* **asort():** Es lo mismo que sort pero funciona para ordenar arrays asociativos por el valor.
* **print\_r():** Imprime todos los elementos del array que recibe como parámetro.
* **in\_array():** Busca un valor(primer parámetro) en un array (segundo parametro) y devuelve True si lo consigue o False si no.
* **array\_keys():** Recibe como parametro un array asociativo y retorna un array solo con las llaves.
* **array\_search():** Busca un valor (primer parametro) en un array (segundo parámetro) y devuelve la posición si lo consigue.