Funciones que trabajan con Arrays

Vamos en este tema a repasar algunas de las funciones que operan sobre matrices más importantes:

count (array)

Cuenta el número de elementos de la matriz. Devuelve un entero.

array_reverse (array, [boolean])

Invierte el orden de los elementos en la matriz. Si queremos mantener las claves de la matriz introducida, tenemos que pasar el segundo parámetro con un valor de TRUE. Devuelve la matriz entrada con los elementos invertidos.

sort (array)

Ordena los elementos de la matriz. Debemos tener cuidado, porque esta función, a diferencia de muchos otras, cambia directamente la matriz que se pasa como entrada, de manera que se perderá en su composición original.

Los valores se disponen en orden ascendente de acuerdo a los criterios que vimos cuando hablamos de operadores aritméticos, y los índices se pierden: después de la ordenación, la matriz tendrán índices numéricos a partir 0 de acuerdo con el nuevo orden. Esta función no devuelve nada.

rsort (array)

Ordena los elementos de la matriz en orden descendente. Esta función también cambia la matriz pasadaa y vuelve a asignar los índices numéricos desde 0. no devuelve nada.

asort (array)

Funciona como sort(), con la diferencia de que retiene los índices de elementos originarios. No devuelve nada.

arsort (array)

como rsort(), en orden descendente; pero conserva los índices originales. No devuelve nada.

in_array (valor, array)

Encuenta unvalor dentro de la matriz. Devuelve un valor booleano: verdadero o falso dependiendo de si el valor buscado está o no presente en la matriz.

array_key_exists (valor, array)

Busca el valor de los índices (y no entre los valores) de la matriz. Devuelve un valor booleano.

array_search (valor, array)

Busca el valor de la matriz y se indica su clave. Devuelve el índice del valor encontrado, o si la búsqueda no tiene éxito, el valor FALSE.

array_merge (array matriz [, array ...])

Fusiona los elementos de dos o más matrices. Los elementos con índices numéricos de un array se anexan a los del otro y se vuelven a reenumerar. Los índices asociativos son retenidos, y si hay más elementos en diferentes matrices con los mismos índices asociativos, los últimos sobrescriben los anteriores. Devuelve la matriz resultante de la fusión.

array_pop (array)

Extrae el último elemento de la matriz, acortando el array con un elemento menos. Devuelve el elemento situado en la parte inferior de la matriz, y cambiar simultáneamente la matriz de entrada quitando ese elemento.

array_push (array, valor [, valor ...])

Añade valores especificados en la matriz. Equivale a usar la instrucción $array[] = $valor, con la ventaja de que nos permite anexar varios valores a la vez. Devuelve el número de elementos de la matriz después tras añadir los elementos.

array_shift (array)

Extrae un elemento, como array_pop(), pero en este caso es al principio. También en este caso, la matriz se "acorta", y también los índices numéricos se vuelven a numerar. Permanece sin cambios los índices asociativos. Devuelve el elemento extraido de la matriz.

array_unshift (array, valor [, valor ...])

Insertar valores al principio de la matriz. Devuelve el número de elementos de la matriz después de la inserción.

implode (cadena, array)

Es la función opuesta de explode(), y sirve para reunir en una única cadena los valores del array. La cadena especificada como el primer parámetro se interpone entre todos los elementos de la matriz. Devuelve la cadena resultado de la agregación. Su sinónimo es join().

He aquí algunos ejemplos que utilizan estas funciones:

```
$arr
= array('Lucas', 'Juan', 'Mateo', 'Pablo', 'Antonio', 'Marcos', 'Jose');


$n = count($arr);              // $n devuelve 7

$arr1 = array_reverse($arr); // $arr1 tendrá los elementos invertidos,
desde 'Jose' a 'Lucas'

echo $arr[1], '<br>';          // 'Juan'

echo $arr1[1], '<br>';         // 'Marcos'


/* Ahora $arr será:

 * 'Antonio', 'Juan', 'Jose', 'Lucas', 'Marcos', ' Mateo', 'Pablo'

 */


sort($arr);


$a = in_array('Juan', $arr);       // $a es verdadero (TRUE)

$a = in_array('Francisco', $arr); // $a es falso (FALSE)

$ultimo = array_pop($arr);         // $ultimo es 'Pablo'

$ultimo = array_pop($arr);         // ahora $ultimo es 'Mateo', y en $arr
quedan 5 elementos

$primero = array_shift($arr);      // $primero es 'Antonio'


/* 'Mateo' e 'Antonio' se vuelven a colocar en la cabeza del array;

 * $a tiene 6 valores

 */


$a = array_unshift($arr, $ultimo, $primo);

 $cadena = implode(' ', $arr); // $cadena se convierte en 'Mateo Antonio
Juan Jose Lucas Marcos' */


/* $new_arr contendra 13 elementos:

 * 'Mateo', 'Antonio', 'Juan',

 * 'Jose', 'Lucas', 'Marcos' (estos son los seis provinientes de $arr),


 * 'Jose', 'Marcos',' Antonio', 'Pablo',

 * 'Mateo', 'Juan', 'Lucas' (estos son los 7 de $arr1). Los índices irán
de 0 a 12.

 */
```

```php
$new_arr = array_merge($arr, $arr1);


// Preparamos un array con claves asociativas:

$famila = array('padre' => 'Claudio', 'madre' => 'Paula', 'hijo'
=> 'Marcos', 'hija' => 'Elisa');


// Creamos una copia de nuestro array para poder hacer experimentos

$fam1 = $familia;


// ahora $fam1 será 'Paula', 'Marcos', 'Elisa', 'Claudio', con índices
de 0 a 3

rsort($fam1);


$fam1 = $familia; // restauramos la matriz original


/* de nuevo $fam1 será 'Paula', 'Marcos', 'Elisa', 'Claudio',
 * Pero cada uno con su llave original
 * ('madre', 'hijo', 'hija', 'padre')
 */

arsort($fam1);


$a = array_key_exists('hija', $fam1); // $a es TRUE

$a = array_key_exists('tio', $fam1);    // $a es FALSE

$a = array_search('Claudio', $fam1);    // $a es 'padre'

$a = array_search('Mateo', $fam1);     // $a es FALSE
```

Agradecimientos: Juan de la Torre Domingo.