

Laboratorio: ARRAY EN PHP

Un mapa es un tipo de datos que asocia valores con claves. Este tipo es optimizado para varios usos diferentes; puede ser usado como una matriz real, una lista (vector), una tabla asociativa (una implementación de un mapa), diccionario, colección, pila, cola, y posiblemente más. Ya que los valores de un array pueden ser otros arrays, árboles y también son posibles arrays multidimensionales.

Sintaxis

Un array puede ser creado usando el constructor del lenguaje `array()`. Éste toma un cierto número de parejas clave => valor como argumentos.

```
array(  
    clave => valor,  
    clave2 => valor2,  
    clave3 => valor3,  
    ...  
)
```

Ejemplo #2 Ejemplo de moldeado de tipo y sobrescritura

```
<?php  
$array = array(  
    1      => "a",  
    "1"    => "b",  
    1.5    => "c",  
    true   => "d",  
);  
var_dump($array);  
?>
```

El resultado del ejemplo sería:

```
array(1) { [1]=> string(1) "d"}
```

Ejemplo #3 Claves mixtas integer y string

```
<?php  
$array = array(  
    "foo" => "bar",  
    "bar" => "foo",  
    100   => -100,  
    -100  => 100,  
);  
var_dump($array);  
?>
```

El resultado del ejemplo sería:

```
array(4) {  
    ["foo"]=>  
    string(3) "bar"  
    ["bar"]=>
```

```
    string(3) "foo"  
    [100]=>  
    int(-100)  
    [-100]=>  
    int(100)  
}
```

Ejemplo #4 Arrays indexados sin clave

```
<?php  
$array = array("foo", "bar", "hallo", "world");  
var_dump($array);  
?>
```

El resultado del ejemplo sería:

```
array(4) {  
    [0]=>  
    string(3) "foo"  
    [1]=>  
    string(3) "bar"  
    [2]=>  
    string(5) "hallo"  
    [3]=>  
    string(5) "world"  
}
```

Es posible especificar la clave sólo para algunos de los elementos y dejar por fuera a los demás:

Ejemplo #5 Claves no en todos los elementos

```
<?php  
$array = array(  
    "a",  
    "b",  
    6 => "c",  
    "d",  
);  
var_dump($array);  
?>
```

El resultado del ejemplo sería:

```
array(4) {  
    [0]=>  
    string(1) "a"  
    [1]=>  
    string(1) "b"  
    [2]=>  
    int(6)  
    [3]=>  
    string(1) "d"
```

```
[6]=>
string(1) "c"
[7]=>
string(1) "d"
}
```

Como se puede ver el último valor "d" se le asignó la clave 7. Esto es debido a que la mayor clave integer era 6.

Acceso a elementos de array con la sintaxis de corchete

Los elementos de array se pueden acceder utilizando la sintaxis *array[key]*.

Ejemplo #6 Acceso a elementos de array

```
<?php
$array = array(
    "foo" => "bar",
    42    => 24,
    "multi" => array(
        "dimensional" => array(
            "array" => "foo"
        )
    )
);

var_dump($array["foo"]);
var_dump($array[42]);
var_dump($array["multi"]["dimensional"]["array"]);
?>
```

El resultado del ejemplo sería:

```
string(3) "bar"
int(24)
string(3) "foo"
```

Nota:

Tanto los corchetes como las llaves pueden ser utilizadas de forma indiferente para acceder a elementos de un array (por ejemplo: `$array[42]` and `$array{42}` harán lo mismo en el ejemplo anterior).

A partir de PHP 5.4 es posible hacer referencia al array del resultado de una llamada a una función o método directamente. Antes sólo era posible utilizando una variable temporal.

Desde PHP 5.5 es posible hacer referencia directa un elemento de un array literal.

Ejemplo #7 Hacer referencia al resultado array de funciones

```
<?php
function getArray() {
    return array(1, 2, 3);
}

// en PHP 5.4
$secondElement = getArray()[1];

// anteriormente
$tmp = getArray();
$secondElement = $tmp[1];

// o
list(, $secondElement) = getArray();
?>
```

Nota:

Si intenta acceder a una clave del array que no se ha definido es lo mismo que el acceso a cualquier otra variable no definida: se emitirá un mensaje de error de nivel `E_NOTICE`, y el resultado será `NULL`.

Creación/modificación con sintaxis de corchete

Un [array](#) existente puede ser modificado al definir valores explícitamente en éste.

Esto se realiza mediante la asignación de valores al [array](#), especificando la clave en corchetes. La clave también se puede omitir, resultando en un par de corchetes vacíos (`[]`).

```
$arr[clave] = valor;
$arr[] = valor;
// clave puede ser un integer o string
// valor puede ser cualquier valor de cualquier tipo
```

Si `$arr` aún no existe, se creará, así que esto es también una forma alternativa de crear un [array](#). Sin embargo, esta práctica es desaconsejada ya que si `$arr` ya contiene algún valor (p.ej. un [string](#) de una variable de petición), entonces este valor estará en su lugar y `[]` puede significar realmente el [operador de acceso a cadenas](#). Siempre es mejor inicializar variables mediante una asignación directa.

Para cambiar un valor determinado, se debe asignar un nuevo valor a ese elemento con su clave. Para quitar un par clave/valor, se debe llamar la función [unset\(\)](#) en éste.

```
<?php
$arr = array(5 => 1, 12 => 2);

$arr[] = 56;    // Esto es lo mismo que $arr[13] = 56;
                // en este punto de el script

$arr["x"] = 42; // Esto agrega un nuevo elemento a
                // el array con la clave "x"

unset($arr[5]); // Esto elimina el elemento del array

unset($arr);    // Esto elimina el array completo
?>
```

Nota:

Como se mencionó anteriormente, si no se especifica una clave, se toma el máximo de los índices [integer](#) existentes, y la nueva clave será ese valor máximo más 1 (aunque al menos 0). Si todavía no existen índices [integer](#), la clave será 0 (cero).

Tenga en cuenta que la clave integer máxima utilizada para éste no es necesario que actualmente exista en el [array](#). Ésta sólo debe haber existido en el [array](#) en algún momento desde la última vez que el [array](#) fué re-indexado. El siguiente ejemplo ilustra este comportamiento:

```
<?php
// Crear un array simple.
$array = array(1, 2, 3, 4, 5);
print_r($array);

// Ahora elimina cada elemento, pero deja el mismo array intacto:
foreach ($array as $i => $value) {
    unset($array[$i]);
}
print_r($array);

// Agregar un elemento (note que la nueva clave es 5, en lugar de 0).
$array[] = 6;
print_r($array);

// Re-indexar:
$array = array_values($array);
$array[] = 7;
print_r($array);
?>
```

El resultado del ejemplo sería:

```
Array(  
    [0] => 1  
    [1] => 2  
    [2] => 3  
    [3] => 4  
    [4] => 5  
)  
Array(  
)  
Array(  
    [5] => 6  
)  
Array(  
    [0] => 6  
    [1] => 7  
)
```

Ejemplo de PHP array asociativo

Este es un array cuyos valores se asignan mediante claves:

```
1 $variable = array(clave1=>valor1, clave2=>valor2, clave3=>valor3...);
```

- **\$variable**: Elemento donde se quedará almacenado el array.
- **\$clave1**, **\$clave2** y **\$clave3**: Claves por las cuales se asignarán y asociarán los valores 1, 2 y 3.

```
1 $equipo = array(portero=>'Cech', defensa=>'Terry', medio=>'Lampard', delantero=>'Torres');
2
3 foreach($equipo as $posicion=>$jugador)
4     {
5         echo "El " . $posicion . " es " . $jugador;
6         echo "<br>";
7     }
```

En el código superior he asignado a cada clave (puesto) un valor (nombre del jugador). Luego recorro los valores insertados con un bucle [PHP foreach\(\)](#) y saco los valores.

```
1 //así estaría accediendo al valor de la clave delantero
2 $equipo['delantero'];
```

Ejemplo de PHP array bidimensional (multidimensional)

```
1 $equipo_futbol = array
2
3     (
4         array("Rooney", "Chicharito", "Gigs"),
5         array("Suarez"),
6         array("Torres", "Terry", "Etoo")
7     );
8 foreach($equipo_futbol as $equipo)
9     {
10        echo "En este equipo juegan: ";
11        foreach($equipo as $jugador)
12            {
13                echo $jugador . " ";
14            }
15        echo "<br>";
16    }
```

- **\$equipo_futbol**: Es un array contenedor de otros 3 arrays con jugadores de futbol.
- Luego recorro el array con [PHP foreach\(\)](#) **\$equipo_futbol** y a su vez recorro cada array que encuentro para sacar los jugadores.

En caso de necesitar acceder a un elemento en cuestión, este es el código:

```
1 $equipo_futbol[0][1];
```

De esta manera hemos accedido al jugador 2 del primer equipo.

Ejemplo de PHP array tridimensional (multidimensional)

Al igual que el anterior es un array de arrays, en este caso es de 3 dimensiones. Aquí el ejemplo:

```
1 $datos = array(
2     array(array(0, 0, 0),
3         array(0, 0, 1),
4         array(0, 0, 2)
5     ),
6     array(array(0, 1, 0),
7         array(0, 1, 1),
8         array(0, 1, 2)
9     ),
10    array(array(0, 2, 0),
11        array(0, 2, 1),
12        array(0, 2, 2)
13    )
14 );
```

en la **matriz** del ejemplo hay una profundidad de 3 arrays, es decir, el primero contiene un segundo y el segundo de 3 terceros.

```
1 foreach($datos as $datos2)
2 {
3     foreach($datos2 as $datos3)
4     {
5         foreach($datos3 as $dato)
6         {
7             echo "$dato ";
8         }
9         echo "<br>";
10    }
11    echo "<br>";
12 }
```

Crear un array en PHP

En el siguiente código PHP te mostramos cómo crear un array en PHP y la forma de acceder a los valores que contiene (debemos indicar el índice o posición en la que se encuentra el dato, teniendo en cuenta que comienzan desde cero):

```
<?php
```

```
    // Array de 7 elementos con el nombre de un día de la semana en cada uno de ellos:
    $aDias = array("Lunes", "Martes", "Miércoles", "Jueves", "Viernes", "Sábado",
    "Domingo");
```

```
    // Array de 4 elementos con un número contenido en cada posición:
    $aNumeros = array(33, 12, 83, 55);
```



```
// Array de 5 elementos, sin asignar valor en ninguna posición:
$aNombres = array(5);

// Array vacío, sin posiciones definidas:
$aDatos = array();

// Mostrar datos de un array:
echo "El primer día es el: ".$aDias[0]."<br/>";
echo "El segundo número es el: ".$aNumeros[1]."<br/>";

echo "-----<br/>";
echo "Fin del ejemplo.";

?>
```

Obtener el número de elementos en un array

Para obtener el número de elementos de un array en PHP usaremos la función `count()`:

```
<?php

$xDias = array("Lunes", "Martes", "Miércoles", "Jueves", "Viernes", "Sábado", "Domingo");

echo count($xDias); // Devuelve 7

?>
```

Recorrer un array

Para recorrer los elementos en un array se suele usar el bucle `for()`, teniendo en cuenta que las posiciones o índices comienzan desde cero.

A continuación te mostramos un ejemplo en el que recorremos un array en PHP de 7 elementos, mostrando cada posición y su valor:

```
<?php

$xDias = array("Lunes", "Martes", "Miércoles", "Jueves", "Viernes", "Sábado", "Domingo");

$contador;

for( $contador=0; $contador < 7; $contador++ ) {

    echo "El valor de la posición [".$contador."] es [".$xDias[$contador]."]<br/>";

}

?>
```

```
El valor de la posición [0] es [Lunes]
El valor de la posición [1] es [Martes]
El valor de la posición [2] es [Miércoles]
El valor de la posición [3] es [Jueves]
El valor de la posición [4] es [Viernes]
El valor de la posición [5] es [Sábado]
El valor de la posición [6] es [Domingo]
```

Otras funciones de PHP que podemos utilizar para recorrer un array son:

- **reset()**: mueve el puntero interno del array al primer elemento. Si el array está vacío devuelve **false**.
- **prev()**: mueve el puntero interno del array al elemento anterior. Si el puntero se encontrara en el primer elemento devolvería **false**.

- **next()**: mueve el puntero interno del array al siguiente elemento. Si el puntero se encontrara en el último elemento devolvería **false**
- **end()**: mueve el puntero interno del array al último elemento. Si el array está vacío devuelve **false**.
- **current()**: Devuelve el elemento actual en un array.

```
<?php
```

```
$aNombres = array("Pedro", "Juan", "Roberta", "Alfredo", "Lola");  
  
echo "-> ".current($aNombres)."<br />"; // Devuelve: "Pedro"  
  
echo "-> ".next($aNombres)."<br />"; // Devuelve: "Juan"  
  
echo "-> ".current($aNombres)."<br />"; // Devuelve: "Juan"  
  
echo "-> ".prev($aNombres)."<br />"; // Devuelve: "Pedro"  
  
echo "-> ".end($aNombres)."<br />"; // Devuelve: "Lola"  
  
echo "-> ".current($aNombres)."<br />"; // Devuelve: "Lola"
```

```
?>
```

Buscar datos en un array

Una vez hemos visto cómo recorrer un array en PHP, para buscar si existe un determinado dato en él deberemos comprobar los valores guardados en cada una de sus posiciones.

En el siguiente ejemplo comprobamos si el nombre 'PEPE' existe en el array '\$aDias', quedando la variable '\$posicion' con un valor de '-1' si no se ha encontrado, o conteniendo la posición en la que se ha encontrado:

```
<?php
```

```
$aDias = array("MARIA", "JUAN", "PEDRO", "ISABEL", "PEPE", "FERNANDO", "ROBERTO");  
  
$contador;  
  
$posicion = -1;  
  
for( $contador=0; $contador < count($aDias); $contador++ )  
{  
  
    if( $aDias[$contador] == "PEPE" ) {
```

```
$posicion = $contador;

break;

}

}


if( $posicion == -1 )

    echo "No se ha encontrado el nombre";

else

    echo "PEPE está en la posición [".$encontrado."]";

?>
```

 **Descargar ejemplo**

PEPE está en la posición [4]

Con **break** forzamos el que se salga del bucle si se encuentra el dato (no será necesario continuar la búsqueda).

También podemos **buscar un dato en un array** usando la función de PHP **in_array()**, la cual devolverá **true** si se ha encontrado o bien **false** en caso contrario:

```
<?php

$Colores1 = array( "color1" => "rojo", "color2" => "verde", "color3" => "azul", "color4" =>
"verde" );

$Colores2 = array( 37, "Pedro", "color1" => "rojo", "color2" => "verde", "color3" => "azul" );

if( in_array( "rojo", $Colores1 ) == true )

    echo "Se ha encontrado el valor 'rojo' en \">$Colores1<br/>";

if( in_array( "magenta", $Colores1 ) == false )

    echo "NO se ha encontrado el valor 'magenta' en \ \"$Colores1<br/>";

if( in_array( 37, $Colores2 ) == true )

    echo "Se ha encontrado el número 37 en \ \"$Colores2<br/>";

if( in_array( 85, $Colores2 ) == false )
```

```
echo "NO se ha encontrado el número 85 en \$aColores2<br/>";  
?>
```

```
Se ha encontrado el valor 'rojo' en $aColores1.  
NO se ha encontrado el valor 'magenta' en $aColores1.  
Se ha encontrado el número 37 en $aColores2.  
NO se ha encontrado el número 85 en $aColores2.
```

Nota: en caso de buscar una cadena de texto la comparación se realizará diferenciando entre mayúsculas y minúsculas.

Insertar elementos en un array

Usaremos la función `array_push()` para insertar elementos al final de un array y `array_unshift()` para insertar elementos al principio del mismo. Estas funciones de PHP devuelven el número de elementos en el array tras la inserción.

```
<?php  
  
$aDias = array("Martes", "Miércoles", "Jueves");  
  
array_unshift( $aDias, "Lunes" );  
  
$num = array_push( $aDias, "Viernes" );  
  
// El array quedaría: "Lunes", "Martes", "Miércoles", "Jueves", "Viernes"  
  
print_r( $aDias );  
  
echo "<p/>".$num."<br/>"; // Devuelve 5  
  
echo count($aDias)."<br />"; // Devuelve 5  
  
?>
```

Con la función `print_r()` mostramos los valores contenidos en las diferentes posiciones del array.

Para insertar elementos en un array disponemos también de la función de PHP `array_splice()`: en el segundo parámetro especificamos en qué posición insertar el elemento (recuerda que comienzan desde cero), en el tercer parámetro pondremos cero (puesto que esta función es usada también para borrar elementos en un array, como veremos más adelante) y en el cuarto parámetro indicaremos el valor que deseamos guardar en dicha posición.

```
<?php  
  
$aDias = array("Lunes", "Martes", "Miércoles", "Jueves", "Viernes", "Sábado", "Domingo");  
  
// En la posición 2 insertamos el valor 88
```

```
array_splice( $aDias, 2, 0, 88 );

// El array quedaría: "Lunes", "Martes", 88, "Miércoles", "Jueves", "Viernes", "Sábado",
"Domingo";

print_r( $aDias );

?>
```

Observa que los valores de las posiciones existentes, desde la que hemos insertado el nuevo elemento, se han desplazado a la derecha.

Como hemos comentado, se puede usar `array_splice()` para borrar elementos de un array, así pues podemos combinar las dos acciones: si en el tercer parámetro escribimos un número que no sea cero se insertará el elemento deseado y también se borrarán las posiciones indicadas:

```
<?php

$aDias = array("Lunes", "Martes", "Miércoles", "Jueves", "Viernes", "Sábado", "Domingo");

// En la posición 2 insertamos el valor 88, y eliminamos la siguiente posición

array_splice( $aDias, 2, 1, 88 );

// El array quedaría: "Lunes", "Martes", 88, "Jueves", "Viernes", "Sábado", "Domingo";

print_r( $aDias );

?>
```

Borrar elementos en un array

Usando `array_shift()` eliminaremos el primer elemento del array, y con `array_pop()` el último. Ambas funciones devuelven el valor del elemento borrado:

```
<?php

$aDias = array("Lunes", "Martes", "Miércoles", "Jueves", "Viernes", "Sábado", "Domingo");

echo count($aDias)."<br />"; // Devuelve 7

$primero = array_shift( $aDias );
```

```
print_r( $aDias );          // El array queda: "Martes", "Miércoles", "Jueves", "Viernes", "Sábado",
"Domingo"

echo "<br />";

$ultimo = array_pop( $aDias );

print_r( $aDias );          // El array queda: "Martes", "Miércoles", "Jueves", "Viernes", "Sábado"

echo "<br />";

echo count($aDias)."<p />";   // Devuelve 5

echo "El primer elemento era: ".$primero."<br />"; // Devuelve "Lunes"

echo "El último elemento era: ".$ultimo."<br />"; // Devuelve "Domingo"

?>
```

Otra función que podemos usar es `array_splice()`, con la que podemos eliminar un determinado número de posiciones a partir de la indicada: en el segundo parámetro indicamos a partir desde qué posición borrar (recuerda que comienzan desde cero) y en el segundo parámetro cuántos elementos deseamos eliminar (si se pone cero, no se eliminará ninguno).

```
<?php

$aDias = array("Lunes", "Martes", "Miércoles", "Jueves", "Viernes", "Sábado", "Domingo");

// Desde la posición 3 borramos 2 elementos (el actual y el siguiente, las posiciones 2 y 3)

array_splice( $aDias, 3, 2);

// El array queda: "Lunes", "Martes", "Miércoles", "Sábado", "Domingo"

print_r( $aDias );

?>
```

Asimismo disponemos también la función `array_unique()`, que devolverá un array con el resultado de **eliminar valores duplicados en un array**:

```
<?php

$aDias = array("Lunes", "Martes", "Miércoles", "Lunes", "Miércoles");
```

```
$resultado = array_unique( $aDias );
```

```
// Devuelve: Array ( [0] => Lunes [1] => Martes [2] => Miércoles )
```

```
print_r( $resultado );
```

```
?>
```

Por último, para vaciar un array lo crearemos de nuevo con el mismo nombre, o bien usando la función de PHP `unset()` eliminaremos el array definitivamente:

```
<?php
```

```
$aDias = array("Lunes", "Martes", "Miércoles"); // Crear el array
```

```
$aDias = array(); // Borrar todos los elementos
```

```
unset($aDias); // Eliminar el array de la memoria
```

```
?>
```


RESOLVER:

1. Diseñar un formulario en HTML y que permita ingresar un número y genere dentro de un TextArea una serie de los múltiplos de ese número. Ejemplo:

numero=7

Entonces se almacena 7 elementos: $1*7$, $2*7$, $3*7$, ..., $7*7$

2. En un formulario se muestra una lista de útiles de escritorio: Lápiz, cuaderno, borrador, Temperas, Papel A4, etc. Cada vez que se recorre la lista se muestra el precio de cada producto independiente.

Se requiere desarrollar este formulario para crear cotizaciones de precios cuando se consultan las listas de productos. Almacenar los productos elegidos por el usuario en un vector y luego mostrar esta información en una página "resultados.php".