



# TEMA 2.3

Arrays en PHP

## Arrays en PHP: Características

- Un array es una estructura de datos que permite almacenar **múltiples valores**.
- En PHP el tipado de datos es dinámico, es decir, una variable no tiene predefinido un tipo de datos. Con los arrays pasa lo mismo.... ¡ A nivel elemento del array !
- Lo anterior implica que, en PHP, los distintos elementos de un array **pueden tener distintos tipos de datos**.
- Podemos acceder a los datos de un array por un índice, un número entero que implica posición del elemento. Al igual que en C y en Java, las posiciones empiezan a contarse desde cero.
- Podemos asignar un valor concreto a una posición concreta del array, de ahí que dos elementos de un mismo array puedan tener distinto tipo de datos
- Para crear un array en PHP, basta con asignar a una variable una información englobada entre corchetes, o bien con la función **array**
- La manipulación de arrays en PHP es muy sencilla.

# Arrays en PHP: Creación y asignación de información

Hay varias formas de crear un array en PHP:

- Asignando directamente la información entre corchetes

```
$frutas = ["naranja","manzana","pera"];
```

- Asignando la información mediante la función array

```
$frutas = array("naranja","manzana","pera");
```

- Indicando el array vacío, ya se rellenará más tarde:

```
$frutas = [ ]; // Se crea un array vacío
```

- Para asignar un elemento a una posición concreta de un array, ponemos el índice de la posición entre corchetes

```
$semillas[0] = "nuez"; // Primera posición del array
```

- Para asignar un elemento al final de un array, basta con no indicar índice entre los corchetes:

```
$semillas[] = "avellana"; // Se añade al final
```

```
<?php
$frutas = ["naranja","manzana","pera"];
$verduras = array("acelga","brócoli");
$verduras[] = "coliflor";
$semillas = [];
$semillas[0] = "nuez";
$semillas[] = "avellana";
$semillas[] = "pipa";
```

## Arrays en PHP: Count y foreach

- **count:** Para obtener la longitud de un array, se utiliza la función count (contar)

```
echo count($verduras); // Mostrará cuántos elementos tiene el array $verduras
```

- PHP no puede convertir un array en un string. Por tanto, para mostrarlo no podemos hacer: echo \$miArray. Es necesario recorrerlo entero para mostrar cada elemento.
- **foreach:** Para recorrer un array podemos utilizar foreach. Foreach es un bucle, muy parecido a un for, que ejecuta el código entre llaves **{ }** para cada elemento del array.

```
foreach ($miArray as $elemento) { echo $elemento . "<br/>"; }
```

- Entre llaves **{ }** está el código que se ejecuta para cada elemento del array.
- Entre paréntesis **( )** se indica el nombre del array (**\$miArray**), luego la palabra "**as**", y luego la variable donde, en cada vuelta del bucle, se almacena el valor del elemento correspondiente (**\$elemento**)

## Ejemplos con count y foreach

```
<?php
$frutas = ["naranja","manzana","pera"];
$verduras = array("acelga","brócoli");
$verduras[] = "coliflor";
$semillas = [];
$semillas[0] = "nuez";
$semillas[] = "avellana";
$semillas[] = "pipa";

$nVerduras = count($verduras);
for ($i=0;$i<$nVerduras;$i++)
    echo "Verdura numero " . $i + 1 . ": $verduras[$i]<br/>";

foreach ($semillas as $semilla)
    echo "Semilla: $semilla <br/>";
?>
```

El array `$verduras` se crea con dos elementos. Posteriormente se añade otro.

El array `$semillas` se crea vacío. Luego se añaden el elemento nuez indicando el índice entre corchetes (posición), y el resto de elementos añadiéndolos al final.

El array `$verduras` se recorre mediante un bucle `for`, previamente se ha averiguado su longitud.

El array `$semillas` se recorre mediante un bucle `foreach`, que se encarga de averiguar la longitud y obtener un elemento en cada vuelta de bucle correspondiente.



## Arrays Asociativos: clave => valor

```
<?php
$capitales = array (
    "España" => "Madrid",
    "Francia" => "París",
    "Italia" => "Roma"
);

$telefonos = [
    "Pedro" => "915732525",
    "Juan" => "630524178",
    "Carlos" => "627586944",
];

$telefonos[] = ["Nacho" => "914458963"];
```

- En PHP existe un tipo de array llamado **asociativo**.
- Cada elemento del array consta de una pareja **"clave" y "valor"**.
- La **clave** es una expresión que nos permite acceder a un elemento, en vez de por un índice, por una descripción. El **valor** sería el valor del elemento asociado a esa clave.
- Al igual que los arrays clásicos de PHP, podemos crear estos arrays con la función array o bien directamente escribiendo entre corchetes los elementos.
- También podemos añadir un elemento al final, aunque *no funcionará como esperamos*. En el ejemplo se añade un array "anidado", es decir, el valor del elemento añadido es otro array.

## Arrays Asociativos: Añadir elementos al final

```
$telefonos = [  
    "Pedro" => "915732525",  
    "Juan" => "630524178",  
    "Carlos" => "627586944",  
];  
$telefonos["Lola"] = "621258974";  
$telefonos[] = ["Nacho" => "914458963"];  
$telefonos["mascota"] = 3;
```

### Array creado:

```
[Pedro] => "915732525"  
[Juan] => "630524178"  
[Carlos] => "627586944"  
[Lola] => "621258974"  
[0] => Array ( [Nacho] => 914458963 )  
[mascota] => 3
```

1. Creamos el array por primera vez. PHP asigna el tipo array a lo contenido entre corchetes [ ]
2. Al escribir \$telefonos["Lola"], PHP busca el elemento cuya clave es Lola, y al no encontrarlo, crea un nuevo elemento con la clave Lola y el valor asignado.
3. Al escribir \$telefonos[], va a añadir un elemento al final. Sin embargo, al asignar algo entre corchetes, PHP añade un elemento de clave autogenerada ("0"), y el valor será un elemento de tipo array.
4. Aquí PHP añadirá al final del array un nuevo elemento, de clave "mascota", y de valor 3, que en este caso será de tipo entero.
5. Al final, el array creado contendrá los elementos que vemos a la izquierda. Habrá:
  - 4 elementos de tipo string
  - Un elemento de tipo array asociativo
  - Otro elemento de tipo integer.

# Manipulación de arrays asociativos

- Para obtener el valor de un elemento accedemos por su clave:

```
$unValor = $miArray["unaClave"];
```

- Podemos recorrer un array asociativo con foreach, tanto para recuperar los valores como para recuperar las parejas clave valor:

```
foreach ($miArray as $valor) // En cada iteración tendremos el valor del elemento
```

```
foreach ($miArray as $clave => $valor) // En cada iteración tendremos la clave y el valor
```

- La conversión a string sería elemento a elemento, lo cual no da error.

```
echo "Teléfono de Carlos: " . $telefonos["Carlos"] . "<br/>";  
echo "<h1>Capitales</h1>";  
foreach ($capitales as $valor)  
| echo "Capital: $valor <br/>";  
echo "<h1>Países y sus capitales</h1>";  
foreach ($capitales as $clave => $valor)  
| echo "La capital de $clave es $valor<br/>";
```



## Funciones: print\_r

```
<?php
$capitales = array (
    "España" => "Madrid",
    "Francia" => "París",
    "Italia" => "Roma"
);
$telefonos = [
    "Pedro" => "915732525",
    "Juan" => "630524178",
    "Carlos" => "627586944",
];
$telefonos["Lola"] = "621258974";
$telefonos[] = ["Nacho" => "914458963"];
$telefonos["mascota"] = 3;

echo("<br/><b> Capitales : </b><br/>");
print_r($capitales);
echo("<br/><b> Telefonos :</b> <br/>");
print_r($telefonos);
```

- **print\_r**: Es una función que nos permite mostrar el contenido de las variables en PHP. Suele utilizarse para depuración, porque no permite configurar la visualización.
- Con arrays, muestra todo el contenido de los mismos. Con ellos **print\_r** muestra los pares clave => valor, así como el contenido de los arrays anidados si los hubiera.

Salida

**Capitales :**

Array ( [España] => Madrid [Francia] => París [Italia] => Roma )

**Telefonos :**

Array ( [Pedro] => 915732525 [Juan] => 630524178 [Carlos] => 627586944 [Lola] => 621258974 [0] => Array ( [Nacho] => 914458963 ) [mascota] => 3 )

## Funciones: var\_dump

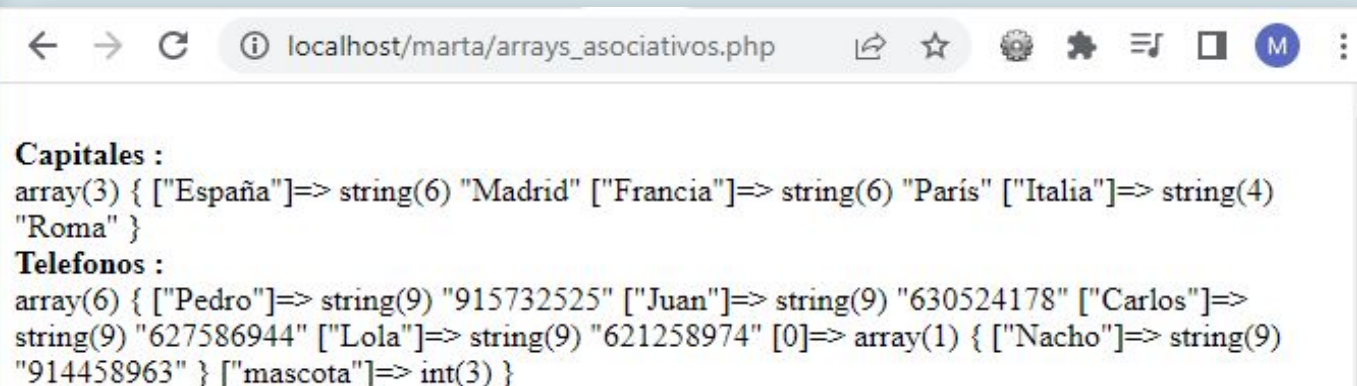
```
$capitales = array (
    "España" => "Madrid",
    "Francia" => "París",
    "Italia" => "Roma"
);
$telefonos = [
    "Pedro" => "915732525",
    "Juan" => "630524178",
    "Carlos" => "627586944",
];
$telefonos["Lola"] = "621258974";
$telefonos[] = ["Nacho" => "914458963"];
$telefonos["mascota"] = 3;

echo("<br/><b> Capitales : </b><br/>");
var_dump($capitales);
echo("<br/><b> Telefonos :</b> <br/>");
var_dump($telefonos);
```

### Salida:

En el caso de **var\_dump** se muestran los tipos de datos de cada valor, y las longitudes de arrays y strings.

- **var\_dump**: Esta función también muestra el contenido de la variable, mostrando no sólo el contenido, como hacía `print_r`, sino también los tipos de datos
- Con arrays, también el número de elementos del array, y las longitudes en caso de que se tenga un string.



```
Capitales :
array(3) { ["España"]=> string(6) "Madrid" ["Francia"]=> string(6) "París" ["Italia"]=> string(4)
"Roma" }
Telefonos :
array(6) { ["Pedro"]=> string(9) "915732525" ["Juan"]=> string(9) "630524178" ["Carlos"]=>
string(9) "627586944" ["Lola"]=> string(9) "621258974" [0]=> array(1) { ["Nacho"]=> string(9)
"914458963" } ["mascota"]=> int(3) }
```

## Funciones: array\_pop, array\_push, in\_array

- **array\_pop**: Elimina el último elemento del array: `array_pop($miArray)`
- **array\_push**: Añade uno o varios elementos al final del array:  
`array_push($miArray, $elemento1, $elemento2, . . . );`
- **in\_array**: Averigua si un elemento está en el array: `in_array($elemento, $miArray)`

```
$amigos=["Pedro", "Juan", "Carlos"];  
print_r($amigos);  
echo "<br/>";  
array_pop($amigos);  
print_r($amigos);  
echo "<br/>";  
  
if (!in_array("Pedro",$amigos))  
    array_push($amigos,"Pedro");  
print_r($amigos);  
echo "<br/>";  
if (!in_array("Nacho",$amigos))  
    array_push($amigos,"Nacho","Nieves");  
print_r($amigos);  
echo "<br/>";
```

```
Array ( [0] => Pedro [1] => Juan [2] => Carlos )  
Array ( [0] => Pedro [1] => Juan )  
Array ( [0] => Pedro [1] => Juan )  
Array ( [0] => Pedro [1] => Juan [2] => Nacho [3] => Nieves )
```

### Salida:

1. El primer **array\_pop** elimina el elemento "Carlos" (índice 2), porque es el último del array
2. El **in\_array** comprueba si está "Pedro", y como sí está no lo añade otra vez con **array\_push**
3. El **in\_array** comprueba si está "Nacho", y como no está, añade a "Nacho" y a "Nieves" al final del array con **array\_push**

## Más Funciones:

- La función **array\_keys** retorna las claves de los pares clave=>valor de un array asociativo  
`$claves = array_keys($capitales);`
- La función **array\_key\_exists** indica si existe una clave en mi archivo  
`array_key_exists($clave, $miArray)`
- La función **sort** ordena los elementos de un array. Si es asociativo, *pierde las claves*. El array mantendría los valores, asociados a un índice posicional.
- La función **ksort** ordena un array asociativo por clave
- La función **asort** ordena un array asociativo por valor, en este caso *no pierde las claves*.
- La función **shuffle** mezcla los valores de un array en un orden aleatorio. Si el array es asociativo *pierde las claves*, que son sustituidas por el índice posicional. Para hacerlo con un array asociativo, tendríamos que programarlo nosotros.
- La función **is\_array** nos indica si una variable es de tipo array.
- La función **unset** elimina un elemento concreto de un array, bien por índice bien por clave:  
`unset( $miArrayAsociativo["clave"] )` o bien `unset( $miArray[2] )`
- La función **isset** indica si en un array existe una clave o índice de un array:  
`isset( $miArrayAsociativo[$clave] )` o bien `isset ( $miArray[1] )`

# Ejercicio 1:



## Manipulando arrays.

1. Crea un documento arrays.php y crea dos arrays, uno asociativo y otro no asociativo.
2. Añade al menos cinco elementos a ambos arrays.
3. Utiliza todas las funciones de la página anterior con el array *asociativo*, utilizando var\_dump para comprobar cómo está quedando el array.
4. Utiliza todas las funciones de la página anterior con el array *no asociativo*, utilizando var\_dump para comprobar cómo está quedando el array



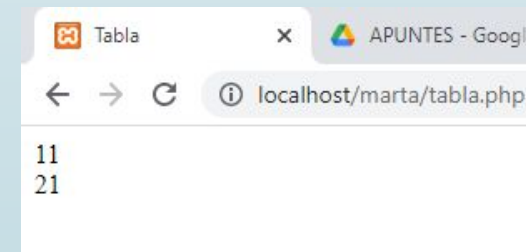
## Arrays de arrays: Anidación.

- ❑ En PHP podemos utilizar un array para guardar otros arrays. Esto es especialmente útil porque nos permite almacenar información enviada por la web, por ejemplo, en formato JSON (<https://www.json.org/json-es.html>)
- ❑ En realidad, *no* estamos creando un array bidimensional. Es un array unidimensional, cuyos elementos son arrays a su vez. Esto debe quedar claro, porque podrían almacenar también otro tipo de elementos, por ejemplo un string o un integer.
- ❑ Para recuperar los elementos de arrays anidados, se utiliza una sintaxis similar a la del array bidimensional en otros lenguajes:

```
$persona["nombre"] = "Bruce Wayne"; // Este elemento es de tipo string
$persona["profesion"] = ["dia"=>"Filántropo", "noche"=>"Caballero Oscuro"]; // Este elemento es un array
echo "Me llamo " . $persona["nombre"] . " y de noche trabajo como " . $persona["profesion"]["noche"];
```

- ❑ Podemos utilizar esta estructura para simular arrays bidimensionales para almacenar una tabla. Sería un array de elementos (filas), todos del mismo tipo de datos: array, que hace las veces de columna. En este caso el programador controlaría que las columnas tuvieran la misma longitud. Por ejemplo:

```
$tabla = [
    "fila1" => ["11","12","13"],
    "fila2" => ["21","22","23"]
];
echo $tabla["fila1"][0] . "<br/>";
echo $tabla["fila2"][0] . "<br/>";
```



## Ejercicio 2:

### Simulando una tabla

11	12	13
21	22	23
31	32	33
41	42	43

1. Almacena en un array la tabla de la izquierda.
2. Utilizarás un array no asociativo, cuyos elementos serán a su vez arrays no asociativos.
3. Mediante el uso de foreach, mostrarás el resultado tal y como se visualiza a la izquierda.

**PISTA:** Utiliza el <table> de html para la visualización de la salida

# Formar arrays dividiendo strings: split y explode

Existen varias funciones que permiten dividir un string en trozos más pequeños, y convertirlo en un array. Dos de ellas son:

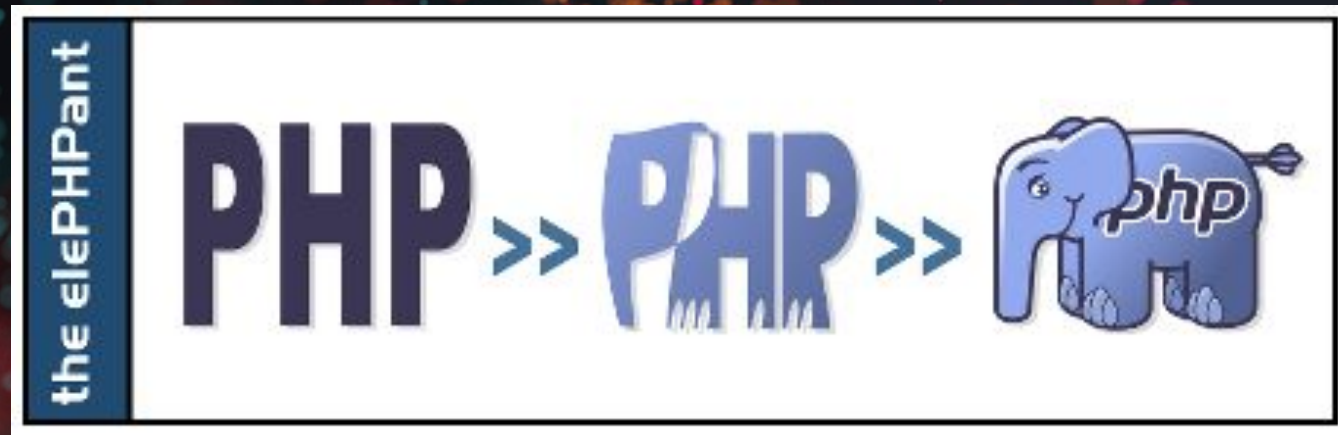
- ❑ **str\_split:** Convierte un string en un array de strings. Puede indicarse la longitud de cada elemento del array. Por defecto será uno.
- ❑ **explode:** Convierte un string en un array de strings, separando la cadena original por el delimitador indicado.

```
$pizza = "porcion1,porcion2,porcion3,porcion4";  
// Con explode, dividimos en elementos por el delimitador ","  
$porciones = explode(",", $pizza);  
echo "<br/>" . print_r($porciones) . "<br/>";  
  
$saludos = "Hola amigos";  
// Con str_split, dividimos en elementos de longitud 1 - por defecto  
$arraySaludos_1 = str_split($saludos);  
echo "<br/> Sin longitud: " . print_r($arraySaludos_1) . "<br/>";  
// Con str_split, dividimos en elementos de longitud 3  
$arraySaludos_3 = str_split($saludos, 3);  
echo "<br/> Sin longitud: " . print_r($arraySaludos_3) . "<br/>";
```

```
arraySaludos_1(11) {  
[0]=> string(1) "H"  
[1]=> string(1) "o"  
[2]=> string(1) "l"  
[3]=> string(1) "a"  
[4]=> string(1) ""  
[5]=> string(1) "a"  
[6]=> string(1) "m"  
[7]=> string(1) "i"  
[8]=> string(1) "g"  
[9]=> string(1) "o"  
[10]=> string(1) "s"  
}
```

```
porciones(4) {  
[0] => string(8) "porcion1"  
[1] => string(8) "porcion2"  
[2] => string(8) "porcion3"  
[3] => string(8) "porcion4"  
}
```

```
arraySaludos_3(4) {  
[0] => string(3) "Hol"  
[1] => string(3) "a a"  
[2] => string(3) "mig"  
[3] => string(3) "os"  
}
```



<https://www.php.net/docs.php>