ESTRUCTURAS DE DATOS EN PHP

LOS ARRAYS

Módulo: Desarrollo Web en Entorno Servidor | Ciclo: Desarrollo Aplicaciones Web

Introducción

¿El Problema?

Las variables simples solo almacenan un valor a la vez.

```
$nombre = "Juan";

$usuario = "admin";
```

¿Cómo almacenamos grandes cantidades de información (ej. una lista de usuarios, productos, o menús) de forma ordenada?

¿La Solución?

Los **arrays**. Permiten almacenar múltiples valores en una sola variable.

En PHP, los arrays son extremadamente flexibles y se basan en un sistema de:

```
clave => valor
```

¿Qué es un Array en PHP?

> Un array en PHP es en realidad un **mapa ordenado**. Un mapa es un tipo de datos que asocia **valores** con **claves**.
>
> — Manual Oficial de PHP (php.net)

Creación de Arrays: Dos Sintaxis

Sintaxis Clásica (array())

La forma tradicional, compatible con todas las versiones de PHP.

```
$personas = array( 1 => "Paco", 2 => "Manolo", 3 => "Lucia" );
```

Sintaxis Moderna ([ ])

Introducida en PHP 5.4. Es más corta y la más usada hoy en día.

```
$personas = [ 1 => "Paco", 2 => "Manolo", 3 => "Lucia" ];
```

El Par Clave (Key) => Valor (Value)

Es el concepto fundamental. Cada valor almacenado tiene una clave única para poder acceder a él.

Claves Automáticas (Arrays Indexados)

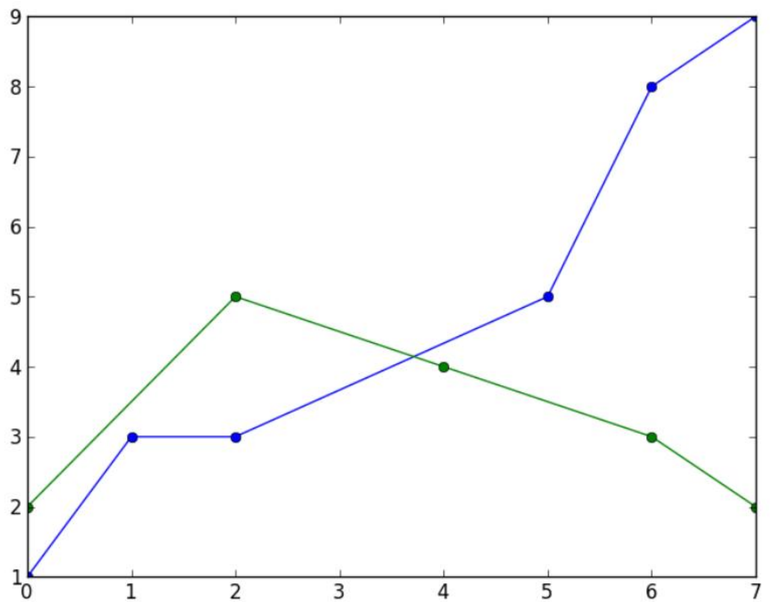
Si no especificas una clave, PHP asigna automáticamente claves numéricas (enteros), **comenzando desde 0.**

Código:

```
$personas = ["Paco", "Manolo", "Lucia"];
```

Equivalente a:

```
$personas = [ 0 => "Paco", 1 => "Manolo", 2 => "Lucia" ];
```



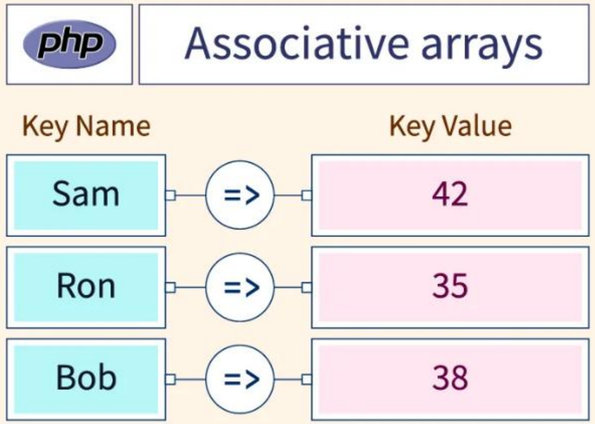
Claves Mixtas (Arrays Asociativos)

Las claves pueden ser de tipo **integer** (número) o **string** (texto).

¡Puedes mezclarlas en el mismo array! Esto le da una gran flexibilidad a PHP.

Código de ejemplo:

```
$datos = [ 1 => "Paco", "jefe" => "Manolo", 3 => "Lucia" ];
```



Acceso a la Información

Array Indexado (Clave numérica)

Accedemos usando el índice numérico (empezando en 0 si es automático).

```
$personas = ["Paco", "Manolo", "Lucia"]; echo $personas[0]; // Muestra "Paco"
echo $personas[2]; // Muestra "Lucia"
```

Array Asociativo (Clave de texto)

Accedemos usando la clave de texto (string) que hemos definido.

```
$datos = ["jefe" => "Manolo", "admin" => "Lucia"]; echo $datos["jefe"]; //
Muestra "Manolo"
```

Modificar Contenido del Array

Modificar Elemento

Se accede por su clave y se asigna un nuevo valor.

```
$alturas[2] = 178;
```

Añadir Elemento

Usa corchetes vacíos `[]`. PHP lo añade al final con una nueva clave numérica.

```
$alturas[] = 189;
```

Eliminar Elemento

Usa la función `unset()` para borrar un elemento (clave y valor).

```
unset($alturas[2]);
```

Arrays Multidimensionales

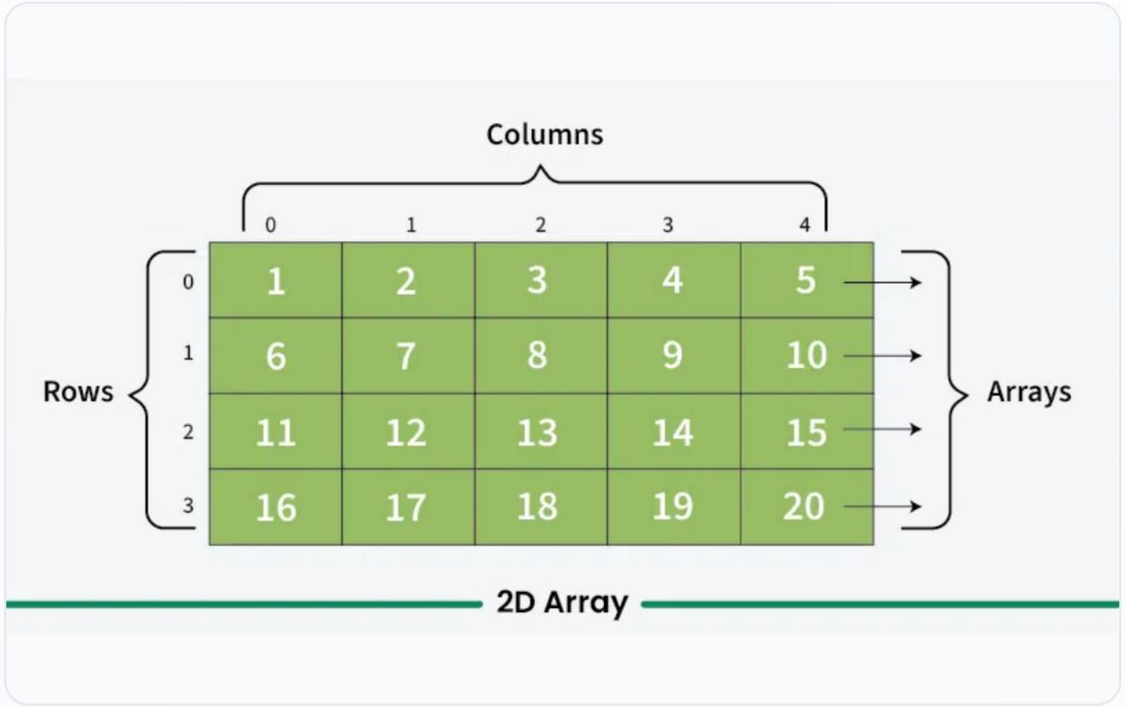
Un array puede contener otros arrays. Esto permite crear estructuras complejas como matrices o árboles de datos.

Código:

```
$clases = [ "clase1" => ["Paco", "Lucia", "Marcos"], "clase2" => ["Pedro", "Juan"] ];
```

Acceso:

```
// Muestra "Paco" echo $clases["clase1"][0];
// Muestra "Juan" echo $clases["clase2"][1];
```



Recorrer Arrays

¿Cómo mostramos *todos* los elementos de un array sin saber cuántos hay? Usamos bucles.

Opción 1: El Bucle `for`

¿Para qué?

Ideal para arrays con claves numéricas **secuenciales** (0, 1, 2, 3...).

Necesitamos saber el tamaño del array con la función `count()`.

Ejemplo de Uso

Generar una lista HTML de un menú.

```
      $menu = ["Home", "Nosotros",
"Productos"]; for ($i = 0; $i <
count($menu); $i++) { echo "

- ".$menu[$i]."

"; }
```

Opción 2: El Bucle `foreach` (Recomendado)

¿Para qué?

¡La forma más fácil y recomendada!

Funciona con **cualquier** array (indexado, asociativo o mixto).

No necesitas `count()` ni gestionar un índice `$i`.

Sintaxis 1 (Solo Valor)

Cuando solo te importa el valor.

```
$menu = ["Home", "Nosotros", "Productos"]; foreach ($menu as $item) { echo "
```

- ".$item."

```
"; }
```

`foreach` (Sintaxis Completa)

¿Para qué?

Para cuando necesitas tanto la **clave** como el **valor**.

Esencial para recorrer arrays asociativos.

Sintaxis 2 (Clave y Valor)

Obtenemos ambas variables en cada iteración.

```
      $datos = ["jefe" => "Manolo", "admin"
=> "Lucia"]; foreach ($datos as $puesto =>
$nombre) { echo $puesto . ": " . $nombre; }
```

Funciones Útiles sobre Arrays (1/2)

Función	Descripción
`count($array)`	Cuenta el número de elementos que hay en el array.
`sort($array)`	Ordena un array por valor. (Pierde las claves asociativas).
`asort($array)`	Ordena un array por valor, manteniendo la asociación de claves.
`ksort($array)`	Ordena un array por clave.

Funciones Útiles sobre Arrays (2/2)

Función	Descripción
`shuffle($array)`	Mezcla un array aleatoriamente.
`array_merge($a1, $a2)`	Combina (fusiona) dos o más arrays en uno solo.
`array_keys($array)`	Devuelve un nuevo array con todas las claves del array original.
`in_array("valor", $array)`	Comprueba si un valor existe en el array. Devuelve true o false.

Visualizar Arrays (Para Depurar)

`print_r($array);`

Muestra el array de forma legible para humanos. Ideal para ver la estructura de claves y valores rápidamente.

`var_dump($array);`

Más detallado. Muestra la estructura del array, el tipo de dato de cada valor y su longitud (ej. `string(5) "Manolo"`).

¡Importante!

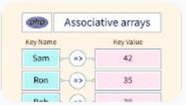
Estas funciones son solo para **depuración (debug)**. Nunca deben dejarse en el código final de producción.

¿Preguntas?

Image Sources

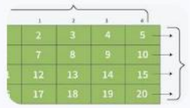
https://i.sstatic.net/qn0zh.png

Source: stackoverflow.com



https://www.scaler.com/topics/images/associative-array-in-php-thumbnail.webp

Source: www.scaler.com



https://media.geeksforgeeks.org/wp-content/uploads/20240916191406/2d-array-in-c.webp

Source: www.geeksforgeeks.org