جامعة الانبار

كلية علوم الحاسوب وتكنولوجيا المعلومات

قسم أنظمة شبكات الحاسوب

برمجة صفحات الويب PHP

المرحلة الثالثة

الفصل الدراسي الاول

مدرس المادة

م.د. سميه عبدالله حمد

Dr. Sumaya Abdulla Hamad

٢٠٢٢ – ٢٠٢٣

Lectures:

١. lec١: PHP Fundamentals (اساسيات برمجة صفحات الانترنت)

٢. lec٢: PHP Data Types (أنواع البيانات)

٣. lec٣: PHP Arrays (المصفوفات)

٤. lec٤: PHP Logic Control Structures (عبارات السيطرة)

٥. lec٥: PHP Loops (عبارات التكرار)

٦. lec٦: PHP Strings PHP String Functions (السلاسل ودوالها)

٧. lec٧: PHP Function (الدوال)

٨. lec٨: PHP Date() & Time Function (دوال الوقت والتاريخ)

٩. lec٩: Regular Expressions (التعابير المنتظمة)

١٠. lec١٠: PHP File Handling & Functions (دوال وتحميل الملفات)

١١. lec١١: PHP Session & PHP Cookies (تعريف وملفات الجلسات (الارتباط

١٢. lec١٢: How to Send Email using PHP mail (كيفية ارسال الايميل)

Dr. Sumaya Abdulla Hamad   ٢٠٢٣ – ٢٠٢٢

3- PHP Array: Associative, Multidimensional

What is a PHP Array?

A PHP array is a variable that stores more than one piece of related data in a single variable.

Think of an array as a box of chocolates with slots inside.

The box represents the array itself while the spaces containing chocolates represent the values stored in the arrays.

The diagram below illustrates the above syntax.

Numeric Arrays

Numeric arrays use number as access keys.

An access key is a reference to a memory slot in an array variable.

The access key is used whenever we want to read or assign a new value an array element.

Below is the syntax for creating numeric array in php.

Array Example

```php
<?php
$variable_name[n] = value;
?>
Or
<?php
$variable_name = array(n => value, …);
?>
```

HERE,

"$variable_name…" is the name of the variable

"[n]" is the access index number of the element

"value" is the value assigned to the array element.

Let's now look at an example of a numeric array.

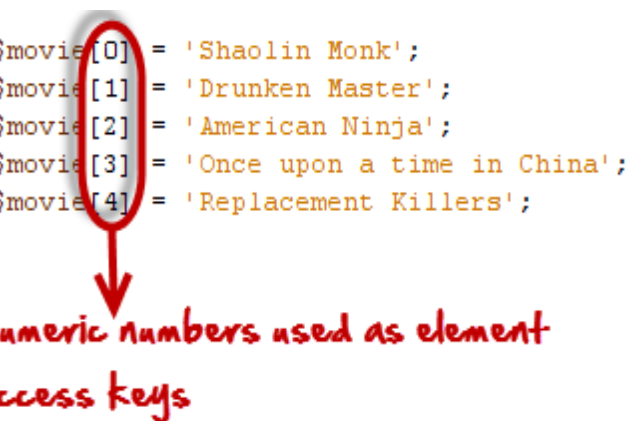Suppose we have 5 movies that we want to store in array variables.

We can use the example shown below to do that.

```php
<?php
$movie[0] = 'Shaolin Monk';
$movie[1] = 'Drunken Master';
$movie[2] = 'American Ninja';
$movie[3] = 'Once upon a time in China';
$movie[4] = 'Replacement Killers';
?>
```

Here,

```php
$movie[0] = 'Shaolin Monk';
$movie[1] = 'Drunken Master';
$movie[2] = 'American Ninja';
$movie[3] = 'Once upon a time in China';
$movie[4] = 'Replacement Killers';
```

Numeric numbers used as element access keys

Each movie is given an index number that is used to retrieve or modify its value. Observe the following code-

```php
<?php
$movie[0]="Shaolin Monk";
$movie[1]="Drunken Master";
$movie[2]="American Ninja";
$movie[3]="Once upon a time in China";
$movie[4]="Replacement Killers";
echo $movie[3];
$movie[3] = " Eastern Condors";
echo $movie[3];
?>
```

Output:

Once upon a time in China Eastern Condors

As you can see from the above examples, working with arrays in PHP when dealing with multiple values of the same nature is very easy and flexible.

Alternatively, the above array variables can also be created using the following code.

```php
<?php
$movie = array(0 => "Shaolin Monk",
        1 => "Drunken Master",
        2 => "American Ninja",
        3 => "Once upon a time in China",
        4 =>"Replacement Killers" );
echo $movie[4];
?>
```

Output:

Replacement Killers

PHP Associative Array

Associative array differ from numeric array in the sense that associative arrays use descriptive names for id keys.

Below is the syntax for creating associative array in php.

```php
<?php
$variable_name['key_name'] = value;

$variable_name = array('keyname' => value);
?>
```

HERE,

- "$variable_name…" is the name of the variable
- "['key_name']" is the access index number of the element
- "value" is the value assigned to the array element.

Let's suppose that we have a group of persons, and we want to assign the gender of each person against their names.

We can use an associative array to do that. The code below helps us to do that.

```php
<?php
$persons = array("Mary" => "Female", "John" => "Male",
"Mirriam" => "Female");
print_r($persons);
echo "";
echo "Mary is a " . $persons["Mary"];
?>
```

HERE,

Descriptive captions used as array element access key

Output:

Array ([Mary] => Female [John] => Male [Mirriam] => Female) Mary is a Female
Associative array are also very useful when retrieving data from the database.

The field names are used as id keys.

PHP Multi-dimensional arrays

These are arrays that contain other nested arrays.

The advantage of multidimensional arrays is that they allow us to group related data together.

Let's now look at a practical example that implements a php multidimensional array.

The table below shows a list of movies by category.

Movie title	Category
Pink Panther	Comedy
John English	Comedy
Die Hard	Action
Expendables	Action
The Lord of the rings	Epic
Romeo and Juliet	Romance
See no evil hear no evil	Comedy

The above information can be represented as a multidimensional array. The code below shows the implementation.

```php
<?php
$movies =array(
"comedy" => array("Pink Panther", "John English", "See no evil
hear no evil"),
"action" => array("Die Hard", "Expendables"),
"epic" => array("The Lord of the rings"),
"Romance" => array("Romeo and Juliet")
);
print_r($movies);
?>
```

HERE,
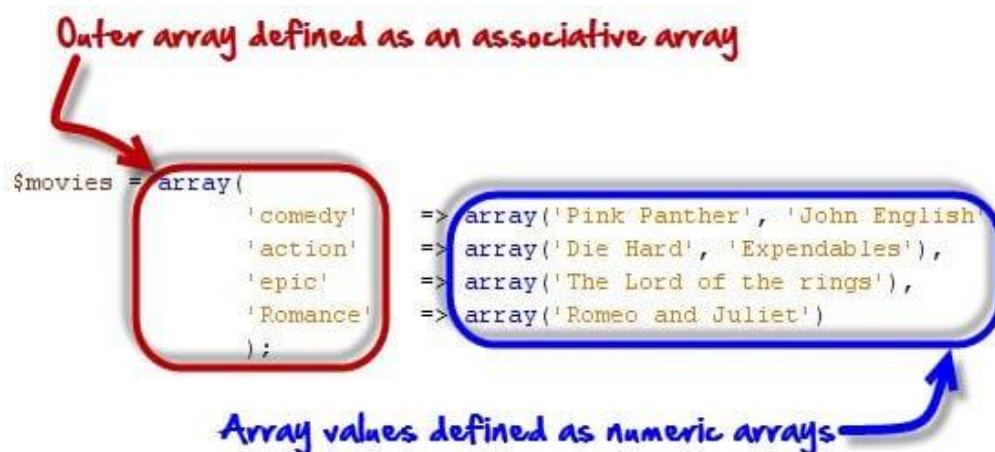
Output:

Array ([comedy] => Array ([0] => Pink Panther [1] => John English [2] => See no evil hear no evil) [action] => Array ([0] => Die Hard [1] => Expendables) [epic] => Array ([0] => The Lord of the rings) [Romance] => Array ([0] => Romeo and Juliet))

Another way to define the same array is as follows

```php
<?php
$film=array(

        "comedy" => array(

                0 => "Pink Panther",

                1 => "john English",

                2 => "See no evil hear no evil"

                ),

        "action" => array (

                0 => "Die Hard",

                1 => "Expendables"

                ),

        "epic" => array (

                0 => "The Lord of the rings"

                ),

        "Romance" => array

                (

                0 => "Romeo and Juliet"

                )

        );
        echo $film["comedy"][0];
        ?>
```

Output:
Pink Panther

Note: the movies numeric array has been nested inside the categories associative array

Operator	Name	Description	How to do it	Output
x + y	Union	Combines elements from both arrays	```<?php $x = array('id' => 1); $y = array('value' => 10); $z = $x + $y; ?>```	Array([id] => 1 [value] => 10)
X == y	Equal	Compares two arrays if they are equal and returns true if yes.	```<?php $x = array("id" => 1); $y = array("id" => "1"); if($x == $y) { echo "true"; } else { echo "false"; } ?>```	True or 1
X === y	Identical	Compares both the values and data types	```<?php $x = array("id" => 1); $y = array("id" => "1"); if($x === $y) { echo "true"; } else { echo "false"; } ?>```	False or 0
X != y, x <> y	Not equal		```<?php $x = array("id" => 1); $y = array("id" => "1"); if($x != $y) { echo "true"; } else { echo "false"; } ?>```	False or 0

X !== y	Non identical		```php
			<?php
			$x = array("id" => 1);
			
			$y = array("id" => "1");
			
			if($x !== $y)
			{
			echo "true";
			}
			else
			{
			echo "false";
			}
			?>
			``
				True or 1

PHP Arrays: Operators

PHP Array Functions

Count function

The count function is used to count the number of elements that an php array contains. The code below shows the implementation.

```php
<?php
$lecturers = array("Mr. Jones", "Mr. Banda", "Mrs. Smith");
echo count($lecturers);
?>
```

Output:

3

is_array function

The is_array function is used to determine if a variable is an array or not. Let's now look at an example that implements the is_array functions.

```php
<?php
$lecturers = array("Mr. Jones", "Mr. Banda", "Mrs. Smith");
echo is_array($lecturers);
?>
```

Output:

1

Sort

This function is used to sort arrays by the values. If the values are alphanumeric, it sorts them in alphabetical order. If the values are numeric, it sorts them in ascending order.

It removes the existing access keys and add new numeric keys. The output of this function is a numeric array

```php
<?php
$persons = array("Mary" => "Female", "John" => "Male", "Mirriam" => "Female");
sort($persons);
print_r($persons);
?>
```
Output:

Array ([0] => Female [1] => Female [2] => Male)

ksort

This function is used to sort the array using the key. The following example illustrates its usage.

```php
<?php
$persons = array("Mary" => "Female", "John" => "Male", "Mirriam" => "Female");
ksort($persons);
print_r($persons);
?>
```
Output:

Array ([John] => Male [Mary] => Female [Mirriam] => Female)

asort

This function is used to sort the array using the values. The following example illustrates its usage.

```php
<?php
$persons = array("Mary" => "Female", "John" => "Male", "Mirriam"
=> "Female");
asort($persons);
print_r($persons);
?>
```

Output:

Array ([Mary] => Female [Mirriam] => Female [John] => Male)

Why use arrays?

- Contents of Arrays can be stretched,
- Arrays easily help group related information such as server login details together
- Arrays help write cleaner code.

Summary

- Arrays are special variables with the capacity to store multi values.
- Arrays are flexibility and can be easily stretched to accommodate more values
- Numeric arrays use numbers for the array keys
- PHP Associative array use descriptive names for array keys
- Multidimensional arrays contain other arrays inside them.
- The count function is used to get the number of items that have been stored in an array
- The is_array function is used to determine whether a variable is a valid array or not.
- Other array functions include sort, ksort, assort etc.