Array and Array Functions in PHP

Introduction

In this lesson, we will be discussing the various array functions available in PHP. Arrays are a fundamental data structure in programming and are often used to store and organize large amounts of data. PHP provides a wide range of built-in functions that make it easy to manipulate arrays, such as adding, removing, or sorting elements. This lesson will cover the most commonly used array functions, as well as provide examples of how to use them in your own programs. By the end of this lesson, you will have a solid understanding of how to work with arrays in PHP, and be able to use the array functions to perform a variety of tasks.

Objectives

The objectives of this lesson are to provide a comprehensive understanding of the array functions available in PHP, and to give students the ability to use these functions to manipulate arrays in their own programs. By the end of this lesson, students should be able to:

1. Explain the purpose and usage of the most commonly used PHP array functions.
2. Use the array functions to perform various tasks such as sorting, adding, and removing elements in an array.

The array functions are essential in PHP, it will help manipulate the data in an organized way, by the end of this lesson, students will be able to navigate through array data easily and perform different tasks with it.

Lesson Proper

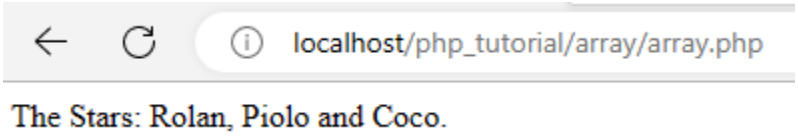
Overview of Arrays in PHP

In PHP, an array is a data structure that stores a collection of values or variables. These values can be of any data type, such as strings, integers, or even other arrays. Arrays in PHP are represented by the array keyword and are defined by enclosing a list of values in square brackets.

For example, here is how you would create an array of numbers in PHP:

```
<!DOCTYPE html>
<html>
<body>

<?php
$stars = array("Rolan", "Piolo", "Coco");
echo "The Stars: " . $stars[0] . ", " . $stars[1] . " and " . $stars[2] . ".";
?>

</body>
</html>
```

Output:



This code is written in PHP and HTML and it creates an array called $stars which contains three elements: "Rolan", "Piolo", and "Coco". Then, it uses the PHP echo statement to output a string that concatenates the elements of the array with a comma and a space.

The echo statement is used to output text or variables to the browser. In this case, it outputs the string "The Stars: " and then it concatenates (combines) the elements of the array $stars using the . operator. The array is accessed by its index, which is a numeric value representing the position of the element in the array. The first element has an index of 0, the second has an index of 1, and so on.

The resulting output of the code will be: "The Stars: Rolan, Piolo and Coco."

In PHP, arrays are also implemented as dynamic data structures, which means that they can grow or shrink as needed. This allows you to add or remove elements from an array as your program runs, making them an essential tool for managing data in your PHP applications.

There are different types of arrays in PHP, each one of them serves a different purpose and they are:

- Indexed arrays - arrays with a numeric index
- Associative arrays - arrays with named keys
- Multi-dimensional arrays - arrays containing one or more arrays

PHP arrays are powerful tools that allow you to organize and manipulate data in a variety of ways, and they are a fundamental part of many PHP programs.

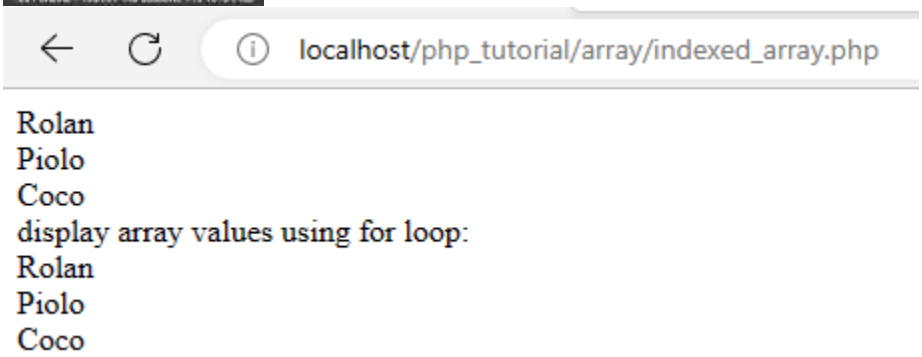
Indexed Array

In PHP, an indexed array is an array with a numeric index. The elements in an indexed array are stored in a linear fashion, with each element assigned a numerical key that can be used to access it. In the example code you provided, $stars is an indexed array that contains three elements: "Rolan", "Piolo", and "Coco". These elements are assigned the keys 0, 1, and 2, respectively. The echo statement is using the keys to access and display the values of the elements in the array. It is concatenating the values with the string "The Stars: " and ", " and " and " and "." to produce the final output: "The Stars: Rolan, Piolo and Coco."

Example

```
<!DOCTYPE html>
<html>
<body>

<?php
$stars = array("Rolan", "Piolo", "Coco");

echo $stars[0]; // Outputs "Rolan"
echo "</br>";
echo $stars[1]; // Outputs "Piolo"
echo "</br>";
echo $stars[2]; // Outputs "Coco"
echo "</br>";

echo "display array values using for loop:";
echo "</br>";
$arrlength = count($stars);

for($i = 0; $i < $arrlength; $i++) {
  echo $stars[$i];
  echo "<br>";
}

?>

</body>
</html>
```

Output:



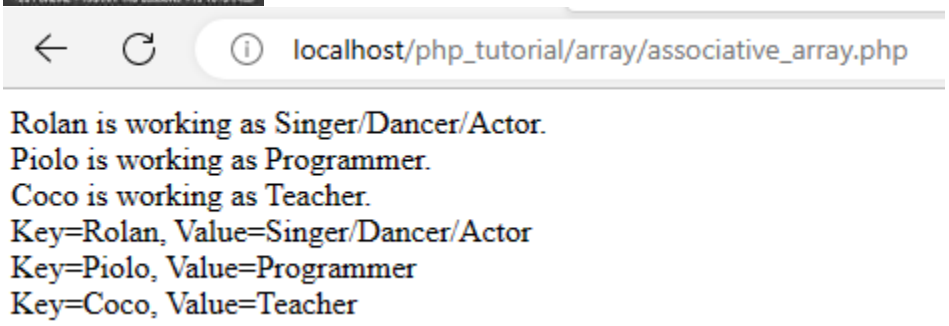
Associative Array

Associative arrays in PHP are arrays that use named keys instead of numeric indices. The keys are used to access the values stored in the array, rather than using the index of the value. This allows for more meaningful and intuitive access to the values stored in the array.

Example

```
<!DOCTYPE html>
<html>
<body>

<?php

$job = array("Rolan"=>"Singer/Dancer/Actor", "Piolo"=>"Programmer", "Coco"=>"Teacher");
echo "Rolan is working as " . $job['Rolan'] . ".";
echo "<br>";
echo "Piolo is working as " . $job['Piolo'] . ".";
echo "<br>";
echo "Coco is working as " . $job['Coco'] . ".";
echo "<br>";
foreach($job as $x => $x_value) {
  echo "Key=" . $x . ", Value=" . $x_value;
  echo "<br>";
}

?>

</body>
</html>
```

Output:



Explanation

The source code is written in PHP, and it uses an associative array to store information about the jobs of different people. The array is defined by the line:

$job = array("Rolan"=>"Singer/Dancer/Actor", "Piolo"=>"Programmer", "Coco"=>"Teacher");

The keys of the array are the names of the people: "Rolan", "Piolo", "Coco", and the values are their jobs: "Singer/Dancer/Actor", "Programmer", "Teacher".

The following lines of code use the echo statement to print the job of each person, by looking up the value associated with their name in the array. For example,

echo "Rolan is working as " . $job['Rolan'] . ".";

The last block of code is a foreach loop that iterates over the associative array, the loop variable $x will contain the key of the current element and $x_value will contain the value.

It will output the key and value of each element.

This code is showing how to use associative array and also how to iterate through it using foreach loop.

Multi-dimensional Array

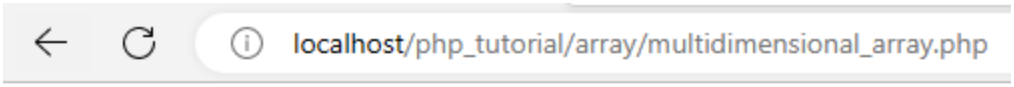
In PHP, a multi-dimensional array is an array that contains other arrays. These inner arrays can be of any type, including indexed arrays or associative arrays. These arrays can be nested to create multi-level arrays.

Multi-dimensional arrays are useful when you have a large amount of data that is best organized in a hierarchical manner. They are particularly useful when working with databases or other structured data.

Example

```
<!DOCTYPE html>
<html>
<body>

<?php

$stars = array(
    array("Rolan","Singer/Dancer/Actor","35"),
    array("Piolo","Programmer","49"),
    array("Coco","Teacher","42")
);

echo $stars[0][0].", Job: ".$stars[0][1].", age: ".$stars[0][2].".<br>";
echo $stars[1][0].", Job: ".$stars[1][1].", age: ".$stars[1][2].".<br>";
echo $stars[2][0].", Job: ".$stars[2][1].", age: ".$stars[2][2].".<br>";

echo "<br>";
echo "<table border='1'>";
echo "<tr> <th>Name</th> <th>Job</th> <th>Age</th> </tr>";

foreach($stars as $star) {
  echo "<tr>";
  echo "<td>".$star[0]."</td>";
  echo "<td>".$star[1]."</td>";
  echo "<td>".$star[2]."</td>";
  echo "</tr>";
}

echo "</table>";

?>
</body>
</html>
```

Output:



Rolan, Job: Singer/Dancer/Actor, age: 35.
Piolo, Job: Programmer, age: 49.
Coco, Job: Teacher, age: 42.

Name	Job	Age
Rolan	Singer/Dancer/Actor	35
Piolo	Programmer	49
Coco	Teacher	42

Explanation

This code is an example of how to use a multi-dimensional array in PHP to create a simple table. The array $stars is declared and initialized with three arrays containing the name, job, and age of three people.

The first part of the script echoes the name, job, and age of each person in a new line, using array indexes to access the individual elements of the sub-arrays.

The second part of the script starts by creating an HTML table with <table> and <tr> tags. Then, it creates a table header with <th> tags, displaying the column names "Name", "Job", and "Age".

Then it uses the foreach loop to iterate through the multi-dimensional array $stars and for each person, it creates a new row in the table with <tr> tags, and for each cell of the row it creates a new cell with <td> tag, and then it echoes the name, job, and age of the person.

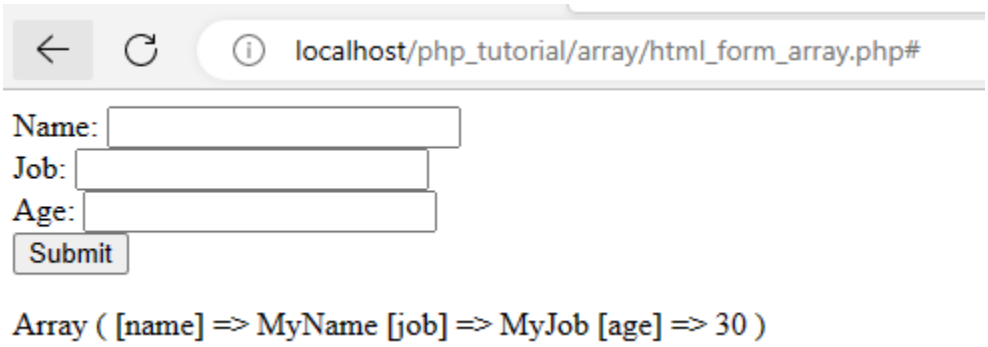
Finally, the script closes the table with the </table> tag. This will output an HTML table showing the name, job, and age of the three people in the array.

Array and HTML Form

Here is an example of a simple HTML form that accepts user input and stores it in a PHP array:

```
<html>
<body>

<form action="#" method="post">
Name: <input type="text" name="name"><br>
Job: <input type="text" name="job"><br>
Age: <input type="number" name="age"><br>
<input type="submit">
</form>

<?php

$user_data = array();

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $user_data["name"] = $_POST["name"];
    $user_data["job"] = $_POST["job"];
    $user_data["age"] = $_POST["age"];
}

print_r($user_data);

?>


</body>
</html>
```

Output:



Explanation

This code is a simple example of how to accept user input using an HTML form and store it in an array using PHP.

The HTML code creates a form with three input fields: "Name", "Job", and "Age". The form uses the "POST" method to send the data to the server when the user submits the form.

The PHP code begins by initializing an empty array called $user_data. It then checks to see if the request method is "POST", which would indicate that the form has been submitted. If the request method is "POST", the code assigns the values entered by the user in the "Name", "Job", and "Age" fields to the corresponding keys in the $user_data array.

After the form has been submitted and the array has been populated with user data, the code uses the print_r() function to display the contents of the $user_data array on the screen.

Summary

A PHP array is a data structure that stores a collection of elements, which can be of any data type, such as integers, strings, and objects. There are several types of arrays in PHP, including indexed arrays, associative arrays, and multi-dimensional arrays. In this article, we have provided examples for each type of array and explain how it works. The team will also provide a series of video tutorial on PHP, so don’t forget to visit our YouTube channel for updates.