Full stack
Bootcamp

Php characteristics

Php variable:

In PHP, a variable is declared using a $ sign followed by the variable name.

- As PHP is a loosely typed language, so we do not need to declare the data types of the variables. It automatically analyzes the values and makes conversions to its correct datatype.
- After declaring a variable, it can be reused throughout the code.
- Assignment Operator (=) is used to assign the value to a variable.

Syntax of declaring a variable in PHP:

```
$name= ”Abdalla”;
```

Rules for declaring PHP variable:

- variable must start with a dollar ($) sign, followed by the variable name.
- can only contain alpha-numeric character and underscore (A-z, 0-9, _).
- variable name must start with a letter or underscore (_) character.
- PHP variable name cannot contain spaces.
- one thing to be kept in mind that the variable name cannot start with a number or special symbols.
- HP variables **are case-sensitive**, so $name and $NAME both are treated as different variable.

PHP Variable: case sensitive

In PHP, variable names are case sensitive. So variable name "color" is different from Color.

```
<?php
    $color= "red";
    $Color = 'green'
    echo "My car is " . $color . "<br>" ;      // red
    echo "My house is " . $Color . "<br> ;    // green

    ?>
```

PHP: Loosely typed language

PHP is a loosely typed language, it means PHP automatically converts the variable to its correct data type.

PHP Variable Scope

PHP has three types of variable scopes:

1. Local variable
2. Global variable
3. Static variable

The variables that are declared within a function are called local variables for that function. These local variables have their scope only in that particular function in which they are declared. This means that these variables cannot be accessed outside the function, as they have local scope.

A variable declaration outside the function with the same name is completely different from the variable declared inside the function.

```php
<?php
    $num = 50; // global variable
    function local_var()
    {
        $num = 45;  //local variable
        echo "Local variable declared inside the function is: ".
$num;                       //45
    }
    local_var();
    echo $num;    // 50
?>
```

Global variable

The global variables are the variables that are declared outside the function. These variables can be accessed anywhere in the program. To access the global variable within a function, use the GLOBAL or use(variable name) keyword before the variable. However, these variables can be directly accessed or used outside the function without any keyword. Therefore there is no need to use any keyword to access a global variable outside the function.

```php
<?php
$name = "Abdalla Alsaeed";        //Global Variable
function global_var()
{
    global $name;
    echo "Variable inside the function: ". $name;
                                        // Abdalla Alsaeed
    echo "</br>";
}
global_var();
echo "Variable outside the function: ". $name;
                                         // Abdalla Alsaeed
?>
```

If two variables, local and global, have the same name, then the local variable has higher priority than the global variable inside the function.

```php
<?php
$x = 5;
function mytest()
{
    $x = 7;
    echo "value of x: " .$x;   // 7
}
mytest();
?>
```

Static variable

It is a feature of PHP to delete the variable, once it completes its execution and memory is freed. Sometimes we need to store a variable even after completion of function execution. Therefore, another important feature of variable scoping is static variable. We use the static keyword before the variable to define a variable, and this variable is called as **static variable**.

Static variables exist only in a local function, but **it does not free its memory after the program execution leaves the scope**.

```
    <?php
    function static_var()
    {
        static $num1 = 3;       //static variable
        $num2 = 6;          //Non-static variable
        //increment in non-static variable
        $num1++;
        //increment in static variable
        $num2++;
        echo "Static: " .$num1 ."</br>";
        echo "Non-static: " .$num2 ."</br>";
    }

//first function call
    static_var();
                    // num1 4
                    // num2 7
    //second function call
    static_var();
             // num1 5
            // num2 7

?>
```

You have to notice that $num1 regularly increments after each function call, whereas $num2 does not. This is why because $num1 is not a static variable, so it freed its memory after the execution of each function call.

PHP Functions

PHP function is a piece of code that can be reused many times. It can take input as argument list and return value.

Advantage of PHP Functions

Code Reusability: PHP functions are defined only once and can be invoked many times, like in other programming languages.

Less Code: It saves a lot of code because you don't need to write the logic many times. By the use of function, you can write the logic only once and reuse it.

Easy to understand: PHP functions separate the programming logic. So it is easier to understand the flow of the application because every logic is divided in the form of functions.

PHP User-defined Functions

Syntax:

```
<?php

function functionname(){
        //code to be executed
        }
  ?>
```

PHP Function Arguments

We can pass the information in PHP function through arguments which is separated by comma.

PHP supports **Call by Value** (default), **Call by Reference**, **Default argument values** and **Variable-length argument list**.

```
<?php
function sayHello($name){
echo "Hello $name<br/>";
}
sayHello("Abdalla");    // Abdalla
sayHello("Saeed");      // Saeed
sayHello("hassan");     // hassan
?>
```

PHP Call By Reference

Value passed to the function doesn't modify the actual value by default (call by value). But we can do so by passing value as a reference.

By default, value passed to the function is call by value. To pass value as a reference, you need to use ampersand (&) symbol before the argument name.

Let's see a simple example of call by reference in PHP.

```
<?php
 function adder(&$str2)
 {
     $str2 .= 'Call By Reference';
 }
 $str = 'Hello ';
 adder($str);
 echo $str;     // 'Call By Reference
 ?>
```

PHP Function: Default Argument Value

We can specify a default argument value in function. While calling PHP function if you don't specify any argument, it will take the default argument. Let's see a simple example of using default argument value in PHP function.

```
<?php
function sayHello($name="Sonoo"){
echo "Hello $name<br/>";
}
sayHello("Rajesh");
sayHello();//passing no value
sayHello("John");     // John
?>
```

PHP Function: Returning Value

```
<?php
function cube($n){
return $n*$n*$n;
}
echo "Cube of 3 is: ".cube(3);   // 27
?>
```

In PHP function/method names are case-insensitive

What are PHP Keywords?

PHP keywords are predefined, reserved words in PHP that are used to perform specific functions. These keywords are reserved by PHP and cannot be used as variable names, function names, or class names. PHP keywords are case-insensitive, meaning that they can be written in either upper or lower case letters.

Usage of PHP Keywords

Keywords are used in PHP to define certain statements, constructs, and functions. For example, the if keyword is used to define a conditional statement, the while keyword is used to define a loop, and the function keyword is used to define a function.

```php
<?php

$x = 7;
// Define a conditional statement
if ($x == 5) {
    echo "x is equal to 5.";
}

// Define a loop
for ($i = 0; $i < 10; $i++) {
    echo $i;
}

// Define a function
function add($a, $b)
{
    return $a + $b;
}
```

Conclusion

Keywords are an important part of any programming language, including PHP. By understanding the meaning and usage of PHP keywords, you can write more effective and efficient code. In addition, using keywords correctly can help prevent errors and make your code more readable and maintainable.