PHP Variables

Presenter:
Agubata Odinaka

What is a Variable?

A variable is a memory location which contains some known or unknown values.

The value of a variable can change within a program.

Variables Definition

- ❖All variables in PHP are denoted with a leading dollar sign ($).
- ❖ The value of a variable is the value of its most recent assignment.
- ❖ Variables are assigned with the = operator, with the variable on the left-hand side and the expression to be evaluated on the right.

Variables Definition

- ❖Variables can, but do not need, to be declared before assignment.
- ❖ Variables used before they are assigned have default values.

Variable Naming Conventions

Variable names must begin with a letter or underscore character.

❖ A variable name can consist of numbers, letters, underscores but you cannot use characters like + , - , % , (,) . & , etc

There is no size limit for variables.

Variable Naming Conventions

- ❖Variables with more than one word should be separated with an underscore. E.g $my_var.
- ❖ Variables with more than one word can also be represented in camelCase. E.g $myVar.
- ❖ Names should describe the data in the variable.
- ❖ Single-character variables are most commonly used only for auxiliary variables; for instance, i, j, k for array index.

Variable Scope

What is Variable Scope?

Scope can be defined as the range of availability a variable has to the program in which it is declared.

PHP variables can be one of four scope types:

Variable Scope

Local Variables

A variable declared in a function is considered local; that is, it can be referenced solely in that function. Any assignment outside of that function will be considered to be an entirely different variable from the one contained in the function:

Variable Scope

```
$x = 4;
function assignx () {
    $x = 0;
    echo "x inside function is $x";
}
assignx();
echo "x outside of function is $x";
```

This will produce following result.

```
x inside function is 0.
x outside of function is 4
```

Variable Scope

Function Parameters

Function parameters are declared after the function name and inside parentheses. They are declared much like a typical variable would be:

```
function multiply ($value) {
    $value = $value * 10;
    return $value;
}
Multiply(3);
```

Variable Scope

Global Variables

In contrast to local variables, a global variable can be accessed in any part of the program. However, in order to be modified, a global variable must be explicitly declared to be global in the function in which it is to be modified.

Variable Scope

This is accomplished, conveniently enough, by placing the keyword GLOBAL in front of the variable that should be recognized as global. Placing this keyword in front of an already existing variable tells PHP to use the variable having that name. Consider an example:

Variable Scope

```
$somevar = 15;
function addit() {
     GLOBAL  $somevar;
     $somevar++;
     echo "Somevar is $somevar";
}
addit();
```

This will produce following result.

```
Somevar is 16
```

Variable Scope

Static Variables

The final type of variable is known as static. In contrast to the variables declared as function parameters, which are destroyed on the function's exit, a static variable will not lose its value when the function exits and will still hold that value should the function be called again.

Variable Scope

You can declare a variable to be static simply by placing the keyword **STATIC** in front of the variable name.

Variable Scope

```
function keep_track() {
   STATIC $count = 0;
   $count++;
   echo $count;
}
keep_track();
keep_track();
keep_track();
```

This will produce following result.

```
1, 2, 3
```

Loose typing in PHP

Some languages, such as Java, are ***strongly typed***: once you've created a variable, you can't change the type of data stored in it. PHP, in contrast, is ***loosely typed***: you can change a variable's data type at any time.

In the following PHP example, $x starts off by holding integer data (3). The next line appends the string "hello" to the data using the concatenation operator, which changes $x’s value to "3hello" (a string data type). Finally, 5.67 is assigned to $x, changing its data type to a float:

```
$x=3;
$x .= “Bishop”;
$x=4.5;
```

Exercise

```
<form action=“process.php" method="POST">
  <p>
    Username
    <input name="username" type="text"/>
  </p>
  <p>Password
    <input name="password" type="password" />
  </p>
  <p>
    <label for="button"></label>
    <input type="submit" name="submit" id="button"
value=“register" />
    </p>
</form>
```

Exercise

```
if(isset($_POST['submit']))
{
    $username = $_POST[“username"];
    $password = $_POST[“password”];
    echo “registration was successful”;
}else{
    echo “unable to register”;
}
```

Questions

And

answers