PHP Variables

A variable can have a short name (like x and y) or a more descriptive name (age, car name, total_volume).

Rules for PHP variables:

- A variable starts with the $ sign, followed by the name of the variable
- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- Variable names are case-sensitive ($age and $AGE are two different variables)

Output Variables

The PHP echo statement is often used to output data to the screen.

The following example will show how to output text and a variable:

Example

```
<html>
<body>
<?php
$txt = "W3Schools.com";
echo "I love $txt!";
?>
</body>
</html>
```

Output: I love W3Schools.com!

The following example will produce the same output as the example above:

Example

```
<?php
$txt = "W3Schools.com";
echo "I love " . $txt . "!";
?>
```

Output: I love W3Schools.com!

The following example will output the sum of two variables:

Example

```
<?php
$x = 5;
$y = 4;
echo $x + $y;
?>
output : 9
```

PHP is a Loosely Typed Language

In the example above, notice that we did not have to tell PHP which data type the variable is.PHP automatically converts the variable to the correct data type, depending on its value.

In other languages such as C, C++, and Java, the programmer must declare the name and type of the variable before using it.

PHP Variables Scope

In PHP, variables can be declared anywhere in the script.
The scope of a variable is the part of the script where the variable can be referenced/used.
PHP has three different variable scopes:
- local
- global
- static

Global and Local Scope

A variable declared **outside** a function has a GLOBAL SCOPE and can only be accessed outside a function:

Example

```php
<?php
$x = 5; // global scope
function myTest() {
   // using x inside this function will generate an error
   echo "<p>Variable x inside function is: $x</p>";
}
myTest();
echo "<p>Variable x outside function is: $x</p>";
?>
```
Output: Variable x inside function is:
           Variable x outside function is: 5

A variable declared **within** a function has a LOCAL SCOPE and can only be accessed within that function:

Example

```php
<?php
function myTest() {
   $x = 5; // local scope
   echo "<p>Variable x inside function is: $x</p>";}
myTest();
// using x outside the function will generate an error
echo "<p>Variable x outside function is: $x</p>";
?>
 Output: Variable x inside function is: 5
         Variable x outside function is:
```

You can have local variables with the same name in different functions, because local variables are only recognized by the function in which they are declared.

PHP The global Keyword

The global keyword is used to access a global variable from within a function.To do this, use the global keyword before the variables (inside the function):

Example

```php
<?php
$x = 5;
$y = 10;
function myTest() {
   global $x, $y;
   $y = $x + $y;
}
myTest();
echo $y; // outputs 15
?>
```

PHP also stores all global variables in an array called $GLOBALS[*index*]. The *index* holds the name of the variable. This array is also accessible from within functions and can be used to update global variables directly.

The example above can be rewritten like this:

PHP The static Keyword

Normally, when a function is completed/executed, all of its variables are deleted. However, sometimes we want a local variable NOT to be deleted. We need it for a further job.To do this, use the **static** keyword when you first declare the variable:

Example

```php
<?php
function myTest() {
   static $x = 0;
   echo $x;
   $x++;
}
myTest();
myTest();
myTest();
?>
Output: 0
          1
          2
```

Then, each time the function is called, that variable will still have the information it contained from the last time the function was called.

Note: The variable is still local to the function.

PHP echo and print Statements

In PHP there are two basic ways to get output: echo and print.
In this tutorial we use echo (and print) in almost every example. So, this chapter contains a little more info about those two output statements.

PHP echo and print Statements

echo and print are more or less the same. They are both used to output data to the screen.

The differences are small: echo has no return value while print has a return value of 1 so it can be used in expressions. echo can take multiple parameters (although such usage is rare) while print can take one argument. echo is marginally faster than print.

The PHP echo Statement

The echo statement can be used with or without parentheses: echo or echo().

Display Text

The following example shows how to output text with the echo command (notice that the text can contain HTML markup):

Example

```php
<?php
echo "<h2>PHP is Fun!</h2>";
echo "Hello world!<br>";
echo "I'm about to learn PHP!<br>";
echo "This ", "string ", "was ", "made ", "with multiple parameters.";
?>
```

Output: **PHP is Fun!**

Hello world!
I'm about to learn PHP!
This string was made with multiple parameters.

Display Variables

The following example shows how to output text and variables with the echo statement:

Example

```php
<?php
$txt1 = "Learn PHP";
$txt2 = "W3Schools.com";
$x = 5;
$y = 4;
echo "<h2>$txt1</h2>";
echo "Study PHP at $txt2<br>";
echo $x + $y;
?>
```

Output: **Learn PHP**

Study PHP at   W3Schools.com
9

The PHP print Statement

The print statement can be used with or without parentheses: print or print().
Display Text

The following example shows how to output text with the print command

Example

```php
<?php
print "<h2>PHP is Fun!</h2>";
print "Hello world!<br>";
print "I'm about to learn PHP!";
?>
Output:
```

PHP is Fun!
Hello world!
I'm about to learn PHP!

Display Variables

The following example shows how to output text and variables with the print statement:

Example

```php
<?php
$txt1 = "Learn PHP";
$txt2 = "W3Schools.com";
$x = 5;
$y = 4;
print "<h2>$txt1</h2>";
print "Study PHP at $txt2<br>";
print $x + $y;
?>
```
Output: Learn PHP
Study PHP at W3Schools.com
9

PHP Data Types

Variables can store data of different types, and different data types can do different things.
PHP supports the following data types:
- String
- Integer
- Float (floating point numbers - also called double)
- Boolean
- Array
- Object
- NULL
- Resource

PHP String

A string is a sequence of characters, like "Hello world!".
A string can be any text inside quotes. You can use single or double quotes:

Example

```
<?php
$x = "Hello world!";
$y = 'Hello world!';
echo $x;
echo "<br>";
echo $y;
?>
```
Output:

Hello world!

Hello world!

PHP Integer

An integer is a whole number (without decimals). It is a number between -2,147,483,648 and +2,147,483,647.
Rules for integers:
- An integer must have at least one digit (0-9)
- An integer cannot contain comma or blanks
- An integer must not have a decimal point
- An integer can be either positive or negative
- Integers can be specified in three formats: decimal (10-based), hexadecimal (16-based - prefixed with 0x) or octal (8-based - prefixed with 0)

In the following example $x is an integer. The PHP var_dump () function returns the data type and value:

Example

```
<?php
$x = 5985;
var_dump($x);
?>
Output :  int(5985)
```

PHP Float

A float (floating point number) is a number with a decimal point or a number in exponential form.

In the following example $x is a float. The PHP var_dump() function returns the data type and value:

Example

```
<?php
$x = 10.365;
var_dump($x);
?>
Output:  float(10.365)
```

PHP Boolean
A Boolean represents two possible states: TRUE or FALSE.

$x = true;

$y = false;

Booleans are often used in conditional testing. You will learn more about conditional testing in a later chapter of this tutorial.

PHP Array

An array stores multiple values in one single variable.

In the following example $cars is an array. The PHP var_dump() function returns the data type and value:

Example

```
<?php
$cars = array("Volvo","BMW","Toyota");
var_dump($cars);
?>
   Output : array(3) { [0]=> string(5) "Volvo" [1]=> string(3) "BMW" [2]=>
                              string(6) "Toyota" }
```

PHP Object

An object is a data type which stores data and information on how to process that data.
In PHP, an object must be explicitly declared.
First we must declare a class of object. For this, we use the class keyword. A class is a structure that can contain properties and methods:

Example

```php
<?php
class Car {
    function Car() {
        $this->model = "VW";
    }
}
// create an object
$herbie = new Car();
// show object properties
echo $herbie->model;
?>
Output : VW
```

PHP NULL Value

Null is a special data type which can have only one value: NULL.
A variable of data type NULL is a variable that has no value assigned to it.
Tip: If a variable is created without a value, it is automatically assigned a value of NULL.
Variables can also be emptied by setting the value to NULL:

Example

```php
<?php
$x = "Hello world!";
$x = null;
var_dump($x);
?>
Output : NULL
```