4. PHP Variables

Variables are "containers" for storing information:

Example 1

```
<?php
$x=5;
$y=6;
$z=$x+$y;
echo $z;
?>
```

Much Like Algebra

```
x=5
y=6
z=x+y
```

In algebra we use letters (like x) to hold values (like 5).

From the expression z=x+y above, we can calculate the value of z to be 11.

In PHP these letters are called **variables.**

Think of variables as containers for storing data.

4.1. PHP Variables

As with algebra, PHP variables can be used to hold values (x=5) or expressions (z=x+y).

Variable can have short names (like x and y) or more descriptive names (*age, name, totalvolume*).

Rules for PHP variables:

- A variable starts with the $ sign, followed by the name of the variable
- A variable name must begin with a letter or the underscore character
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- A variable name should not contain spaces
- Variable names are case sensitive ($y and $Y are two different variables)

Both PHP statements and PHP variables are case-sensitive.

4.2. Creating (Declaring) PHP Variables

PHP has no command for declaring a variable.

A variable is created the moment you first assign a value to it:

```
$txt="Hello world!";
$x=5;
```

After the execution of the statements above, the variable **txt** will hold the value **Hello world!**, and the variable **x** will hold the value **5**.

Note: When you assign a text value to a variable, put quotes around the value.

4.3. 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 a strongly typed programming language, we will have to declare (define) the type and name of the variable before using it.

4.4. PHP Variable Scopes

The scope of a variable is the part of the script where the variable can be referenced/used.

PHP has four different variable scopes:

- local
- global
- static
- parameter

4.4.1 Local Scope

A variable declared **within** a PHP function is local and can only be accessed within that function:

Example 2

```
<?php
```

```
$x=5; // global scope

function myTest()
{
echo $x; // local scope
}

myTest();
?>
```

The script above will not produce any output because the echo statement refers to the local scope variable $x, which has not been assigned a value within this scope.

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.

Local variables are deleted as soon as the function is completed.

4.4.2. Global Scope

A variable that is defined outside of any function, has a global scope.

Global variables can be accessed from any part of the script, EXCEPT from within a function.

To access a global variable from within a function, use the **global** keyword:

Example 3

```
<?php
$x=5; // global scope
$y=10; // global scope

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:

Example 4

```
<?php
$x=5;
$y=10;

function myTest()
{
$GLOBALS['y']=$GLOBALS['x']+$GLOBALS['y'];
}

myTest();
echo $y;
?>
```

4.4.3. Static Scope

When a function is completed, all of its variables are normally deleted. However, sometimes you want a local variable to not be deleted.

To do this, use the **static** keyword when you first declare the variable:

Example 5

```
<?php

function myTest()
{
static $x=0;
echo $x;
$x++;
}

myTest();
myTest();
myTest();

?>
```

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.

4.4.4. Parameter Scope

A parameter is a local variable whose value is passed to the function by the calling code.

Parameters are declared in a parameter list as part of the function declaration:

Example 6

```
<?php

function myTest($x)
{
echo $x;
}

myTest(5);

?>
```

Parameters are also called arguments. We will discuss it in more details in our ***13 PHP Functions*** chapter.