PHP - Variables

A variable in PHP is a named memory location that holds data belonging to one of the data types.

- PHP uses the convention of prefixing a dollar sign ($) to the name of a variable.
- Variable names in PHP are case-sensitive.
- Variable names follow the same rules as other labels in PHP. A valid variable name starts with a letter or underscore, followed by any number of letters, numbers, or underscores.
- As per the naming convention, "$name", "$rate_of_int", "$Age", "$mark1" are examples of **valid variable names** in PHP.
- **Invalid variable names:** "name" (not having $ prefix), "$rate of int" (whitespace not allowed), "$Age#1" (invalid character #), "$11" (name not starting with alphabet).

Variables are assigned with the "=" operator, with the variable on the left hand side and the expression to be evaluated on the right.

No Need to Specify the Type of a Variable

PHP is a dynamically typed language. There is no need to specify the type of a variable. On the contrary, the type of a variable is decided by the value assigned to it. The value of a variable is the value of its most recent assignment.

Take a look at this following **example** −

Open Compiler

```
<?php
   $x = 10;
   echo "Data type of x: " . gettype($x) . "\n";

   $x = 10.55;
   echo "Data type of x now: " . gettype($x) . "";
?>
```

It will produce the following **output** −

```
Data type of x: integer
Data type of x now: double
```

Automatic Type Conversion of Variables

PHP does a good job of automatically converting types from one to another when necessary. In the following code, PHP converts a string variable "y" to "int" to perform addition with another integer variable and print 30 as the result.

Take a look at this following **example** −

</> Open Compiler

```
<?php
   $x = 10;
   $y = "20";

   echo "x + y is: ", $x+$y;
?>
```

It will produce the following **output** −

```
x + y is: 30
```

Variables are Assigned by Value

In PHP, variables are always assigned by value. If an expression is assigned to a variable, the value of the original expression is copied into it. If the value of any of the variables in the expression changes after the assignment, it doesn’t have any effect on the assigned value.

</> Open Compiler

```
<?php
   $x = 10;
   $y = 20;
   $z = $x+$y;
   echo "(before) z = ". $z . "\n";

   $y=5;
   echo "(after) z = ". $z . "";
?>
```

It will produce the following **output** –

```
(before) z = 30
(after) z = 30
```

Assigning Values to PHP Variables by Reference

You can also use the way to assign values to PHP variables by reference. In this case, the new variable simply references or becomes an alias for or points to the original variable. Changes to the new variable affect the original and vice versa.

To assign by reference, simply prepend an ampersand (&) to the beginning of the variable which is being assigned (the source variable).

Take a look at this following **example** –

Open Compiler

```
<?php
   $x = 10;
   $y = &$x;
   $z = $x+$y;
   echo "x=". $x . " y=" . $y . " z = ". $z . "\n";

   $y=20;
   $z = $x+$y;
   echo "x=". $x . " y=" . $y . " z = ". $z . "";
?>
```

It will produce the following **output** –

```
x=10 y=10 z = 20
x=20 y=20 z = 40
```

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 −

- Local Variables
- Global Variables
- Static Variables
- Function Parameters

Variable Naming

Rules for naming a variable is −

- 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.