KTU NOTES

The learning companion.

KTU STUDY MATERIALS | SYLLABUS | LIVE NOTIFICATIONS | SOLVED QUESTION PAPERS

MODULE 3

PHP

PHP is a server scripting language, and a powerful tool for making dynamic and interactive Web pages.

PHP is a widely-used, free, and efficient alternative to competitors such as Microsoft's ASP.

- PHP files can contain text, HTML, CSS, JavaScript, and PHP code
- PHP code is executed on the server, and the result is returned to the browser as plain HTML
- PHP files have extension ".php"
- PHP runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- PHP supports a wide range of databases
- PHP is free. Download it from the official PHP resource: www.php.net
- PHP is easy to learn and runs efficiently on the server side.

PHP Syntax

A PHP script can be placed anywhere in the document.

A PHP script starts with <?php and ends with ?>:

```
<?php
// PHP code goes here
?>
```

Example

```
<!DOCTYPE html>
<html>
```

```
<body>
<?php
echo "My first PHP script!";
?>
</body>
</html>
```

PHP Variables

In PHP, a variable starts with the $ sign, followed by the name of the variable:

Example

```
<?php
$txt = "Hello world!";
$x = 5;
$y = 10.5;
?>
```


After the execution of the statements above, the variable $txt will hold the value Hello world!, the variable $x will hold the value 5, and the variable $y will hold the value 10.5.

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

Note: Unlike other programming languages, PHP has no command for declaring a variable. It is created the moment you first assign a value to it.

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

Remember that PHP variable names are case-sensitive!

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

WAYS TO CONVERT DATA TYPES IN PHP

1. CONVERTING DATA TYPES USING TYPE CASTING

Type casting can be achieved by placing the intended types in front of the variable to be cast. The different type casting operators that can be used to convert the variable from one type to another are given below.

cast operators → conversion

(array) → array

(bool) or (boolean) → boolean

(int) or (integer) → integer

(object) → object

(real) or (double) or (float) → float

(string) → string

Here are the some examples of converting one data type to another using type casting.

The expression $number=(float)16; converts the integer 16 to the float 16.0

2. CONVERTING DATA TYPES USING TYPE JUGGLING

Type juggling is the feature of PHP where the variables are automatically cast to best fit data type in the circumstances while manipulating with mathematical operators.

Type juggling converts automatically to the upper level data type that supports for the calculation. Following is the example that converts string into integer while adding to the integer.

```
<?php
$sum=15; //an integer
$increase="10"; // a string
$sum+=$increase; // $total=20, which is an integer.
?>
```

Here is an another example that shows the PHP's type juggling capabilities. Hence, the number used in string can be used to calculate with the integer.

```
<?php
$theory=45; //an integer
$practical="15 is the number obtained";    // a string
$sum=$theory+$practical;      // $total=20, which is an integer.
echo "The Total Number Obtained is " .$sum;
?>
```

OUTPUT:

The following output will be generated from the above PHP code.

[insert_php]
$theory=45;
$practical=”15 is the number obtained”;
$sum=$theory+$practical;
echo “The Total Number Obtained is ” .$sum;
[/insert_php]

PHP Operators

PHP Operator is a symbol i.e used to perform operations on operands. In simple words, operators are used to perform operations on variables or values. For example:

1. $num=10+20;//+ is the operator and 10,20 are operands

In the above example, + is the binary + operator, 10 and 20 are operands and $num is variable.

PHP Operators can be categorized in following forms:

- Arithmetic Operators
- Assignment Operators
- Bitwise Operators
- Comparison Operators
- Incrementing/Decrementing Operators
- Logical Operators
- String Operators
- Array Operators
- Type Operators
- Execution Operators

- Error Control Operators

We can also categorize operators on behalf of operands. They can be categorized in 3 forms:

- **Unary Operators:** works on single operands such as ++, -- etc.
- **Binary Operators:** works on two operands such as binary +, -, *, / etc.
- **Ternary Operators:** works on three operands such as "?:".

Flow-Control Statements

PHP supports a number of traditional programming constructs for controlling the flow of execution of a program.

Conditional statements, such as if/else and switch, allow a program to execute different pieces of code, or none at all, depending on some condition. Loops, such as while and for, support the repeated execution of particular code.

if

The if statement checks the truthfulness of an expression and, if the expression is true, evaluates a statement. An if statement looks like:

if (*expression*)

statement

To specify an alternative statement to execute when the expression is false, use the else keyword:

if (*expression*)

statement

else

statement

For example:

```
if ($user_validated)
  echo "Welcome!";
else
  echo "Access Forbidden!";
```

switch Statement

Use the switch statement to **select one of many blocks of code to be executed**.

Syntax

```
switch (n) {
  case label1:
    code to be executed if n=label1;
    break;
  case label2:
    code to be executed if n=label2;
    break;
  case label3:
    code to be executed if n=label3;
    break;
    ...
  default:
    code to be executed if n is different from all labels;
}
```

PHP Loops

In PHP, we have the following loop types:

- while - loops through a block of code as long as the specified condition is true
- do...while - loops through a block of code once, and then repeats the loop as long as the specified condition is true
- for - loops through a block of code a specified number of times
- foreach - loops through a block of code for each element in an array

while Loop

The while loop executes a block of code as long as the specified condition is true.

Syntax

```
while (condition is true) {
  code to be executed;
}
```

do...while Loop

The do...while loop will always execute the block of code once, it will then check the condition, and repeat the loop while the specified condition is true.

Syntax

```
do {
  code to be executed;
} while (condition is true);
```

for Loop

The for loop is used when you know in advance how many times the script should run.

Syntax

```
for (init counter; test counter; increment counter) {
  code to be executed for each iteration;
}
```

Parameters:

- *init counter*: Initialize the loop counter value
- *test counter*: Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends.
- *increment counter*: Increases the loop counter value

foreach Loop

The foreach loop works only on arrays, and is used to loop through each key/value pair in an array.

Syntax

```
foreach ($array as $value) {
  code to be executed;
}
```

For every loop iteration, the value of the current array element is assigned to $value and the array pointer is moved by one, until it reaches the last array element.