INTRODUCTION TO PHP

Lecture 1

2021-2022

Lecturer: Dezheen Hussein Abdulazeez

E-mail: dezheen.abdulazeez@uod.ac

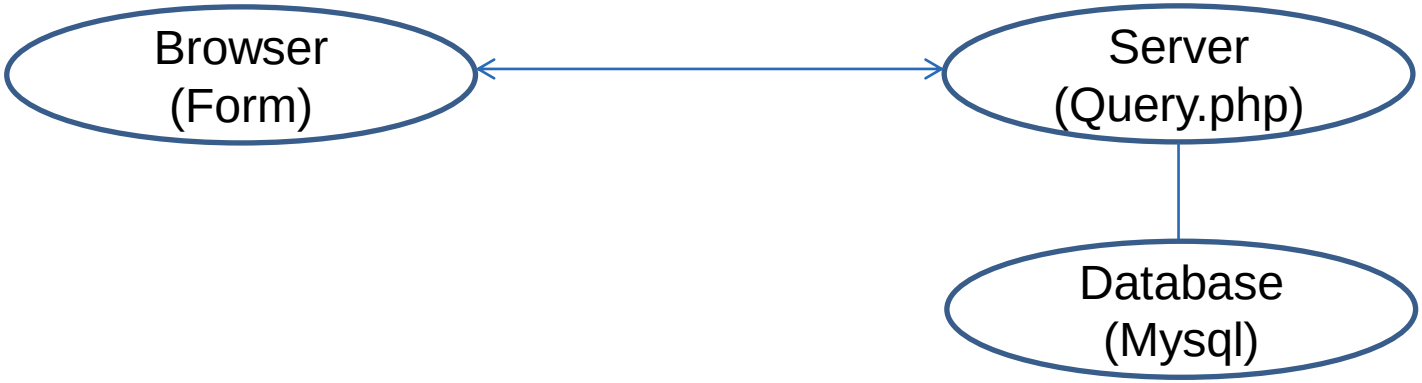
Computer science Dept_UOD

OUTLINES

- What is PHP?
- HOW does A PHP FILE LOOLK LIKE?
- What Can PHP Do?
- Basic PHP Syntax.
- Variables in PHP.
- Conditional statement.
- Loop Types.

WHAT IS PHP?

- PHP is a powerful server-side scripting language designed for web development to produce dynamic web pages.
- It stands for **Personal Home Page.**



HOW DOES A PHP FILE LOOLK LIKE?

- PHP files may contain text, HTML tags and scripts.
- PHP files are returned to the browser as plain HTML
- PHP files have a file extension of ".php“.

What Can PHP Do?

- PHP can generate dynamic page content.
- PHP can create, open, read, write, delete, and close files on the server.
- PHP can collect form data.
- PHP can send and receive cookies.
- PHP can add, delete, modify data in your database.

Basic PHP Syntax

- A PHP scripting block always starts with <?php and ends with ?>.
- Echo is a command for showing output.
- Example1: simple PHP script

```
<html> <body>
        <h1>My first PHP page</h1>
        <?php
            echo"Hello World";
        ?>
    </body></html>
```

VARIABLES IN PHP

- Variables are used for storing a values, like text strings, numbers or arrays.
- PHP is a loosely typed language.
- PHP variables are case-sensitive.
- All variables in PHP start with a $ sign symbol.
- Example2: PHP variables

```
<html> <body>          <?php
                $txt="Hello World";
                 echo $txt;
                        ?>
</body></html>
```

Variable Naming Rules

- A variable name must start with a letter or an underscore "_".
- A variable name can only contain alpha-numeric characters and underscores (a-Z,0-9, and _)
- A variable name should not contain spaces.
- Examples:

```
$name="valid"    // valid name

$_name="valid"    // valid name

$1name="invalid"    // invalid name, starts with a number

$f name ="invalid"    // invalid name, contains a space
```

PHP Case Sensitivity(variables)

Example3:

```
<html><body>
        <?php
// $color, $COLOR, and $coLOR are three different variables
            $age = "22";
            echo "My Age is " . $age. "<br>";
            echo "My Age is " . $AGE . "<br>";
            echo "My Age is " . $AgE . "<br>";
 ?>
</body></html>
```

PHP Case Sensitivity(keywords)

All keywords (e.g. if, else, while, echo, etc.) , classes and functions are NOT case-sensitive.

Example4:

```
<html><body>
<?php
// All three echo statements below are equal
ECHO "Hello World!<br>";
echo "Hello World!<br>";
EcHo "Hello World!<br>";
?>
</body></html>
```

CONCATENATION AND COMMENTS

- To concatenate two or more variables together use the (.) dot operator.
- We use // to make a single line comment or /* and */ to make a large comment block.
- Example5: Concatenation

```
<html> <body>          <?php
        // This is a single-line comment
        $txt1="Hello World";
        $txt2="12345“;
        echo $txt1. " ". $txt2;   ?>          </body></html>
```

Output: Hello World 12345

CONCATENATION

Example6: <html> <body> <?php

```
$a="University of";
$b="Duhok";
echo $a. $b ;
echo "<br>";
echo $a." ". $b ;
?> </body></html>
```

Output: University ofDuhok

University of Duhok

THE PRINT() STATEMENT

- We could use the print() statement for the same purposes as echo().
- Echo prototype : Void echo(string argument1 [,… string argumentN]).
- Print prototype : int print(argument)
- Which faster echo or print()?

  -The answer is that the echo() function is a faster because it returns nothing , whereas print() will return 1 if the statement successfully output.

- Example7:

```
<?php
        $university = "Duhok";
        print ("<p> University of $university .</p>");
?>
```

Arithmetic Operators

y=5

Operator	Description	Example	Result
+	Addition	x=y+2	x=7
-	Subtraction	x=y-2	x=3
*	Multiplication	x=y*2	x=10
/	Division	x=y/2	x=2.5
%	Modulus	x=y%2	x=1
++	Increment	x=++y	x=6
--	Decrement	x=--y	x=4

Comparison and Logical Operators

x=3, y=5

Operator	Description	Example
==	is equal to	x==y is false
!=	is not equal	x!=y is true
>	is greater than	x>y is false
<	is less than	x<y is true
>=	is greater than or equal to	x>=y is false
<=	is less than or equal to	x<=y is true
&&	and	(x<y &&y>1) is false
||	or	(x==5y || y==5) is false
!	not	!(x==y)is true

CONDITIONAL STATEMENT

- Example8: If statement

```
<html> <body>
        <?php
            $name="Azad";

          if ($name=="Azad")
            {
                echo "Welcome ". $name  ;
              }
          else {
              echo "Please register ";
           }        ?>
</body></html>
```

CONDITIONAL STATEMENT

- Example9:

```
<html> <body>
        <?php
                $x=2;
    switch($x)
 {
                case 1:    echo "Number 1";
                break;
                case 2:    echo "Number 2";
                break;
                case 3:   echo "Number 3";
                break;
}
        ?>        </body></html>
```

PHP LOOPING

- **For -** loops through a block of code a specified number of times.
- **While -** loops through a block of code if and as long as a specified condition is true.
- **Do...while -** loops through a block of code once, and then repeats the loop as long as a special condition is true.
- **Foreach -** loops through a block of code for each element in an array.

The for Statement

Example10: For Loop

```
<?php
        for ($i = 0; $i <= 5; $i++) {
         echo "The number is: $i <br>";
  }
  ?>
```

The do...while Statement

Example11: While Loop

```
<?php
        $i=1;
        while($i<=5)
            {
                echo "The number is " . $i . "<br />";
                    $i++;
            }
?>
```

Do While Loop

Example12: Do While Loop

```
<?php
  $i=0;
        do
          {
              $i++;
              echo "The number is " . $i . "<br />";

          } while ($i<5);
?>
```

The foreach Statement

- The foreach statement is used to loop through arrays.
- 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.

The foreach Statement

Example13: Foreach Loop

```
<?php
    $colors = array ("red","green","blue","yellow");
    foreach ($colors as $value) {
        echo "Color is: " . $value . "<br />";
      }
    ?>
```

Output: Color is: red
Color is: green
Color is: blue
Color is: yellow

HTML +PHP

Example14: HTML+ PHP

```
<?php
 $Fruit=array("Apple","Orange",“Figs");
    echo "<table border='1'>";
     echo "<tr><td colspan='3'> Fruit </td></tr>";
                 echo "<tr>";
        foreach($Fruit as $value)
       {  echo"<td>$value</td>"; }
 echo  "</tr>";
echo "</table>";  ?> ?>
```

Fruit		
Apple	Orange	Figs

EXERCISE

Q: Display the following output?

1	2	3	4	5	6	7	8	9	10	11	12
2	4	6	8	10	12	14	16	18	20	22	24
3	6	9	12	15	18	21	24	27	30	33	36
4	8	12	16	20	24	28	32	36	40	44	48
5	10	15	20	25	30	35	40	45	50	55	60
6	12	18	24	30	36	42	48	54	60	66	72
7	14	21	28	35	42	49	56	63	70	77	84
8	16	24	32	40	48	56	64	72	80	88	96
9	18	27	36	45	54	63	72	81	90	99	108
10	20	30	40	50	60	70	80	90	100	110	120
11	22	33	44	55	66	77	88	99	110	121	132
12	24	36	48	60	72	84	96	108	120	132	144