

Chapter 1

PHP Introduction



By: Asst. Lect. Montadhar Moslem

2026-09-26

PHP Introduction



- ▶ 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.
- ▶ **What is PHP?**
 - PHP is an acronym for "PHP: Hypertext Preprocessor"
 - PHP is a widely-used, open source scripting language
 - PHP scripts are executed on the server
 - PHP is free to download and use

▶ PHP Introduction

▶ **What is a PHP File?**

- ▶ 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 Introduction

▶ **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
 - PHP can be used to control user-access
 - PHP can encrypt data
- ▶ With PHP you are not limited to output HTML. You can output images or PDF files. You can also output any text, such as XHTML and XML.

PHP Introduction

► Why 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 is executed on the server, and the plain HTML result is sent back to the browser.
- ▶ A PHP script can be placed anywhere in the document.
- ▶ A PHP script starts with **<?php** and ends with **?>**
- ▶ The default file extension for PHP files is ".php".
- ▶ A PHP file normally contains HTML tags, and some PHP scripting code.

```
<?php
```

```
// PHP code goes here
```

```
?>
```

PHP Syntax

- Below, we have an example of a simple PHP file, with a PHP script that uses a built-in PHP function **"echo"** to output the text "Hello World!" on a web page:

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

<h1>My first PHP page</h1>

<?php
echo "Hello World!";
?>

</body>
</html>
```

PHP Comments

- ▶ A comment in PHP code is a line that is not executed as a part of the program. Its only purpose is to be read by someone who is looking at the code.
- ▶ Comments can be used to:
 - Let others understand your code
 - Remind yourself of what you did - Most programmers have experienced coming back to their own work a year or two later and having to re-figure out what they did. Comments can remind you of what you were thinking when you wrote the code
 - Leave out some parts of your code
- ▶ PHP supports several ways of commenting:

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

<?php
// This is a single-line comment

# This is also a single-line comment

/* This is a
multi-line comment */
?>

</body>
</html>
|
```

PHP Case Sensitivity

- ▶ In PHP, keywords (e.g. if, else, while, echo, etc.), classes, functions, and user-defined functions are not case-sensitive.
- ▶ In the example below, all three echo statements below are equal and legal:
- ▶ **ECHO** is the same as **echo**:

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<?php
```

```
ECHO "Hello Ali<br>";
```

```
echo "Hello Hassan<br>";
```

```
EcHo "Hello Sami<br>";
```

```
?>
```

```
</body>
```

```
</html>
```



Hello Ali
Hello Hassan
Hello Sami

PHP Case Sensitivity

- ▶ All variable names are case-sensitive!
- ▶ Look at the example below; only the first statement will display the value of the `$color` variable! This is because `$color`, `$COLOR`, and `$coLOR` are treated as three **different** variables:
- ▶ `$COLOR` is not same as `$color`:

```
<html>
<body>

<?php
$color = "red";
echo "My car is " . $color . "<br>";
echo "My house is " . $COLOR . "<br>";
echo "My boat is " . $coLOR . "<br>";
?>

</body>
</html>
```

```
My car is red
My house is
My boat is
```

PHP Variables

- ▶ Variables are "containers" for storing information.
- ▶ In PHP, a variable starts with the **\$** sign, followed by the name of the variable:
- ▶ In the example above, the variable **\$x** will hold the value **5**, and the variable **\$y** will hold the value **"John"**.
- ▶ Note: When you assign a text value to a variable, put quotes around the value.
- ▶ A variable can have a short name (like **\$x** and **\$y**) or a more descriptive name (**\$age**, **\$carname**, **\$total_volume**).

```
$x = 5;  
$y = "John";
```

PHP Variables Rules

- ▶ 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)

Chapter 2

echo & Data Types



By: Asst. Lect. Montadhar Moslem

2026-09-26

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

```
echo "Hello";  
//same as:  
echo("Hello");
```

Display Text

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

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

<?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.";
?>

</body>
</html>
```

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:

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

<?php
$txt1 = "Learn PHP";
$txt2 = "W3Schools.com";

echo "<h2>$txt1</h2>";
echo "<p>Study PHP at $txt2</p>";
?>

</body>
</html>
```

Learn PHP

Study PHP at [W3Schools.com](https://www.w3schools.com)

Using Single Quotes

- ▶ Strings are surrounded by quotes, but there is a difference between single and double quotes in PHP.
- ▶ When using double quotes, variables can be inserted to the string as in the example above.
- ▶ When using single quotes, variables have to be inserted using the `.` operator, like this:

```
$txt1 = "Learn PHP";  
$txt2 = "W3Schools.com";  
  
echo '<h2>' . $txt1 . '</h2>';  
echo '<p>Study PHP at ' . $txt2 . '</p>';
```



Learn PHP

Study PHP at W3Schools.com

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

▶ Getting the Data Type

- ▶ You can get the data type of any object by using the **var_dump()** function.

The **var_dump()** function returns the data type and the value:

```
$x = 5;  
var_dump($x);
```



int(5)

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:

```
$x = "Hello world!";  
$y = 'Hello world!';
```

```
var_dump($x);  
echo "<br>";  
var_dump($y);
```



```
string(12) "Hello world!"  
string(12) "Hello world!"
```

Double or Single Quotes?

- ▶ You can use double or single quotes, but you should be aware of the differences between the two.
- ▶ Double quoted strings perform action on special characters.
- ▶ E.g. when there is a variable in the string, it returns the value of the variable:

Double quoted string literals perform operations for special characters:

```
$x = "John";  
echo "Hello $x";
```



Hello John

Single quoted string literals returns the string as it is:

```
$x = "John";  
echo 'Hello $x';
```



Hello \$x

PHP Integer

- ▶ An integer data type is a non-decimal number between -2,147,483,648 and 2,147,483,647.
- ▶ Rules for integers:
 - An integer must have at least one digit
 - An integer must not have a decimal point
 - An integer can be either positive or negative
 - Integers can be specified in: decimal (base 10), hexadecimal (base 16), octal (base 8), or binary (base 2) notation
- ▶ In the following example **\$x** is an integer. The PHP **var_dump()** function returns the data type and value:

```
$x = 5985;  
var_dump($x);
```



```
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:

```
$x = 10.365;  
var_dump($x);
```



float(10.365)

▶ PHP Boolean

- ▶ A Boolean represents two possible states: TRUE or FALSE.

```
$x = true;  
var_dump($x);
```



bool(true)

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:

```
$cars = array("Volvo", "BMW", "Toyota");  
var_dump($cars);
```



```
array(3) {  
    [0]=>  
        string(5) "Volvo"  
    [1]=>  
        string(3) "BMW"  
    [2]=>  
        string(6) "Toyota"  
}
```

Chapter 3

PHP Operators



By: Asst. Lect. Montadhar Moslem

PHP Operators

- ▶ Operators are used to perform operations on variables and values.
- ▶ PHP divides the operators in the following groups:
 - Arithmetic operators
 - Assignment operators
 - Comparison operators
 - Increment/Decrement operators
 - Logical operators
 - String operators
 - Array operators
 - Conditional assignment operators

PHP Arithmetic Operators

- ▶ The PHP arithmetic operators are used with numeric values to perform common arithmetical operations, such as addition, subtraction, multiplication etc.

Operator	Name	Example	Result
+	Addition	$\$x + \y	Sum of $\$x$ and $\$y$
-	Subtraction	$\$x - \y	Difference of $\$x$ and $\$y$
*	Multiplication	$\$x * \y	Product of $\$x$ and $\$y$
/	Division	$\$x / \y	Quotient of $\$x$ and $\$y$
%	Modulus	$\$x \% \y	Remainder of $\$x$ divided by $\$y$
**	Exponentiation	$\$x ** \y	Result of raising $\$x$ to the $\$y$ 'th power

PHP Assignment Operators

- ▶ The PHP assignment operators are used with numeric values to write a value to a variable.
- ▶ The basic assignment operator in PHP is "=". It means that the left operand gets set to the value of the assignment expression on the right.

Assignment	Same as...	Description
<code>x = y</code>	<code>x = y</code>	The left operand gets set to the value of the expression on the right
<code>x += y</code>	<code>x = x + y</code>	Addition
<code>x -= y</code>	<code>x = x - y</code>	Subtraction
<code>x *= y</code>	<code>x = x * y</code>	Multiplication
<code>x /= y</code>	<code>x = x / y</code>	Division
<code>x %= y</code>	<code>x = x % y</code>	Modulus

PHP Comparison Operators

- ▶ The PHP comparison operators are used to compare two values (number or string):

Operator	Name	Example	Result
==	Equal	<code>\$x == \$y</code>	Returns true if \$x is equal to \$y
===	Identical	<code>\$x === \$y</code>	Returns true if \$x is equal to \$y, and they are of the same type
!=	Not equal	<code>\$x != \$y</code>	Returns true if \$x is not equal to \$y
<>	Not equal	<code>\$x <> \$y</code>	Returns true if \$x is not equal to \$y
!==	Not identical	<code>\$x !== \$y</code>	Returns true if \$x is not equal to \$y, or they are not of the same type
>	Greater than	<code>\$x > \$y</code>	Returns true if \$x is greater than \$y
<	Less than	<code>\$x < \$y</code>	Returns true if \$x is less than \$y
>=	Greater than or equal to	<code>\$x >= \$y</code>	Returns true if \$x is greater than or equal to \$y
<=	Less than or equal to	<code>\$x <= \$y</code>	Returns true if \$x is less than or equal to \$y
<=>	Spaceship	<code>\$x <=> \$y</code>	Returns an integer less than, equal to, or greater than zero, depending on if \$x is less than, equal to, or greater than \$y. Introduced in PHP 7.

Increment / Decrement Operators

- ▶ The PHP increment (زيادة) operators are used to increment a variable's value.
- ▶ The PHP decrement (نقصان) operators are used to decrement a variable's value.

Operator	Same as...	Description
<code>++\$x</code>	Pre-increment	Increments <code>\$x</code> by one, then returns <code>\$x</code>
<code>\$x++</code>	Post-increment	Returns <code>\$x</code> , then increments <code>\$x</code> by one
<code>--\$x</code>	Pre-decrement	Decrements <code>\$x</code> by one, then returns <code>\$x</code>
<code>\$x--</code>	Post-decrement	Returns <code>\$x</code> , then decrements <code>\$x</code> by one

PHP Logical Operators

- ▶ The PHP logical operators are used to combine conditional statements.

Operator	Name	Example	Result
and	And	<code>\$x and \$y</code>	True if both <code>\$x</code> and <code>\$y</code> are true
or	Or	<code>\$x or \$y</code>	True if either <code>\$x</code> or <code>\$y</code> is true
xor	Xor	<code>\$x xor \$y</code>	True if either <code>\$x</code> or <code>\$y</code> is true, but not both
<code>&&</code>	And	<code>\$x && \$y</code>	True if both <code>\$x</code> and <code>\$y</code> are true
<code> </code>	Or	<code>\$x \$y</code>	True if either <code>\$x</code> or <code>\$y</code> is true
<code>!</code>	Not	<code>!\$x</code>	True if <code>\$x</code> is not true

PHP Conditional Statements

- ▶ Very often when you write code, you want to perform different actions for different conditions. You can use conditional statements in your code to do this.
- ▶ In PHP we have the following conditional statements:
 - **if** statement - executes some code if one condition is true
 - **if...else statement** - executes some code if a condition is true and another code if that condition is false
 - **if...elseif...else statement** - executes different codes for more than two conditions
 - **switch** statement - selects one of many blocks of code to be executed

PHP - if Statement

- ▶ **if statement** executes some code if one condition is true.

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

```
<?php  
$t = 14;  
  
if ($t < 20) {  
    echo "Have a good day!";  
}  
?>
```

Have a good day!

```
<?php  
if (5 > 3) {  
    echo "Have a good day!";  
}  
?>
```

Have a good day!

PHP - if Statement

- ▶ The if statement executes some code if one condition is true.

```
$t = 14;  
  
if ($t == 14) {  
    echo "Have a good day!";  
}
```



Have a good day!

Comparison Operators

Operator	Name	Result
==	Equal	Returns true if the values are equal
===	Identical	Returns true if the values and data types are identical
!=	Not equal	Returns true if the values are not equal
<>	Not equal	Returns true if the values are not equal
!==	Not identical	Returns true if the values or data types are not identical
>	Greater than	Returns true if the first value is greater than the second value
<	Less than	Returns true if the first value is less than the second value
>=	Greater than or equal to	Returns true if the first value is greater than, or equal to, the second value
<=	Less than or equal to	Returns true if the first value is less than, or equal to, the second value

Logical Operators

- ▶ To check more than one condition, we can use logical operators, like the && operator:

Check if `$a` is greater than `$b`, AND if `$a` is less than `$c` :

```
$a = 200;  
$b = 33;  
$c = 500;
```

```
if ( $a > $b && $a < $c ) {  
    echo "Both conditions are true";  
}
```



Both conditions are true

Logical Operators

- ▶ Here are the PHP logical operators to use in **if** statements:

Operator	Name	Description
and	And	True if both conditions are true
&&	And	True if both conditions are true
or	Or	True if either condition is true
	Or	True if either condition is true
xor	Xor	True if either condition is true, but not both
!	Not	True if condition is not true

PHP if...else Statements

- ▶ The **if...else** statement executes some code if a condition is true and another code if that condition is false.

```
if (condition) {  
    // code to be executed if condition is true;  
} else {  
    // code to be executed if condition is false;  
}
```

- ▶ The **if...elseif...else** statement executes different codes for more than two conditions.

```
if (condition) {  
    code to be executed if this condition is true;  
} elseif (condition) {  
    // code to be executed if first condition is false and this condition is true;  
} else {  
    // code to be executed if all conditions are false;  
}
```

PHP if...else Statements

- ▶ The **if...else** statement executes some code if a condition is true and another code if that condition is false.

```
$a = 13;  
  
if ($a > 10) {  
    echo "Above 10";  
    if ($a > 20) {  
        echo " and also above 20";  
    } else {  
        echo " but not above 20";  
    }  
}
```



Above 10 but not above 20

PHP switch Statement

- ▶ The **switch** statement is used to perform different actions based on different conditions.
- ▶ Use the **switch** statement to select one of many blocks of code to be executed.

```
switch (expression) {  
    case label1:  
        //code block  
        break;  
    case label2:  
        //code block;  
        break;  
    case label3:  
        //code block  
        break;  
    default:  
        //code block  
}
```

```
<?php  
$d = 4;  
  
switch ($d) {  
    case 6:  
        echo "Today is Saturday";  
        break;  
    case 0:  
        echo "Today is Sunday";  
        break;  
    default:  
        echo "Looking forward to the Weekend";  
}  
?>
```

Looking forward to the Weekend



Chapter 4

Loop & Functions



By: Asst. Lect. Montadhar Moslem

PHP Loops

- ▶ Often when you write code, you want the same block of code to run over and over again a certain number of times. So, instead of adding several almost equal code-lines in a script, we can use loops.
- ▶ Loops are used to execute the same block of code again and again, as long as a certain condition is true.
- ▶ In PHP, we have the following loop types:
 - **for** - loops through a block of code a specified number of times
 - **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
 - **foreach** - loops through a block of code for each element in an array

for Loop

- ▶ The for loop - Loops through a block of code a specified number of times.
- ▶ The for loop is used when you know how many times the script should run.

Syntax

```
for (expression1, expression2, expression3) {  
  // code block  
}
```

- ▶ This is how it works:
 - expression1 is evaluated once
 - expression2 is evaluated before each iteration
 - expression3 is evaluated after each iteration

Example

- ▶ Print the numbers from 0 to 10:

```
for ($x = 0; $x <= 10; $x++) {  
    echo "The number is: $x <br>";  
}
```



The number is: 0
The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5
The number is: 6
The number is: 7
The number is: 8
The number is: 9
The number is: 10

- ▶ Example Explained

- The first expression, `$x = 0;`, is evaluated once and sets a counter to 0.
- The second expression, `$x <= 10;`, is evaluated before each iteration, and the code block is only executed if this expression evaluates to true. In this example the expression is true as long as `$x` is less than, or equal to, 10.
- The third expression, `$x++;`, is evaluated after each iteration, and in this example, the expression increases the value of `$x` by one at each iteration.

The break Statement

- ▶ With the **break** statement we can stop the loop even if the condition is still true:

```
<?php
  for ($x = 0; $x <= 10; $x++) {
    if ($x == 3) break;
    echo "The number is: $x <br>";
  }
?>
```



```
The number is: 0
The number is: 1
The number is: 2
```

while Loop

- ▶ The **while** loop - Loops through a block of code as long as the specified condition is true.
- ▶ Example: Print **\$i** as long as **\$i** is less than 6:

```
$i = 1;  
while ($i < 6) {  
    echo $i;  
    $i++;  
}
```



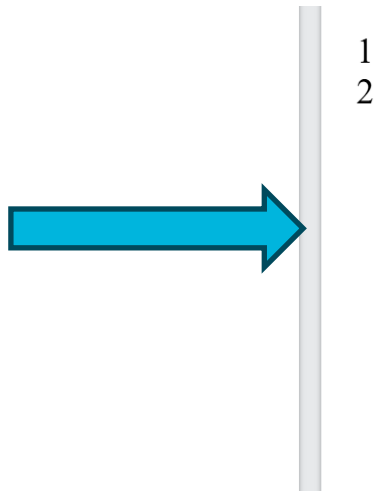
12345

while Loop

- ▶ The while loop does not run a specific number of times, but checks after each iteration if the condition is still true.
- ▶ With the break statement we can stop the loop even if the condition is still true:

```
<?php
    $i = 1;

    while ($i < 6) {
        if ($i == 3){
            break;
        }
        echo $i."<br>";
        $i++;
    }
?>
```



do while Loop

- ▶ The do...while loop - Loops through a block of code once, and then repeats the loop as long as the specified condition is true.
- ▶ Example: Print `$i` as long as `$i` is less than 6:

```
<?php
    $i = 1;
    do {
        echo $i."<br>";
        $i++;
    } while ($i < 6);
?>
```



1
2
3
4
5

foreach Loop

- ▶ The most common use of the **foreach** loop, is to loop through the items of an array.
- ▶ Example: Loop through the items of an indexed array:

```
<?php
    $colors = array("red", "green", "blue", "yellow");

    foreach ($colors as $x) {
        echo "$x <br>";
    }
?>
```

```
red
green
blue
yellow
```

- ▶ For every loop iteration, the value of the current array element is assigned to the variable `$x`. The iteration continues until it reaches the last array element.

Keys and Values

- ▶ The array (in the previous slide) is an indexed array, where the first item has the key 0, the second has the key 1, and so on.
- ▶ Associative arrays are different, associative arrays use named keys that you assign to them, and when looping through associative arrays, you might want to keep the key as well as the value.
- ▶ This can be done by specifying both the key and value in the foreach definition, like this:

```
$members = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
```

```
foreach ($members as $x => $y) {  
    echo "$x : $y <br>";  
}
```



Peter : 35

Ben : 37

Joe : 43

Functions

- ▶ The real power of PHP comes from its functions.
 - ▶ PHP has more than 1000 built-in functions, and in addition you can create your own custom functions.
 - ▶ Besides the built-in PHP functions, it is possible to create your own functions.
-
- A function is a block of statements that can be used repeatedly in a program.
 - A function will not execute automatically when a page loads.
 - A function will be executed by a call to the function.
-
- ▶ Create a Function: A user-defined function declaration starts with the keyword function, followed by the name of the function:

```
function myMessage() {  
    echo "Hello world!";  
}
```

Call a Function

- ▶ To call the function, just write its name followed by parentheses ():

```
function myMessage() {  
    echo "Hello world!";  
}
```

```
myMessage();
```



Hello world!

- ▶ In our example, we create a function named myMessage().
- ▶ The opening curly brace { indicates the beginning of the function code, and the closing curly brace } indicates the end of the function.
- ▶ The function outputs "Hello world!".

Function Arguments

- ▶ Information can be passed to functions through arguments. An argument is just like a variable.
- ▶ Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.
- ▶ The following example has a function with one argument (`$fname`). When the `familyName()` function is called, we also pass along a name, e.g. ("Jani"), and the name is used inside the function, which outputs several different first names, but an equal last name:

```
function familyName($fname) {  
    echo "$fname Refsnes.<br>";  
}
```

```
familyName("Jani");  
familyName("Hege");  
familyName("Stale");  
familyName("Kai Jim");  
familyName("Borge");
```



Jani Refsnes.
Hege Refsnes.
Stale Refsnes.
Kai Jim Refsnes.
Borge Refsnes.

Function Arguments

- ▶ The following example has a function with two arguments (`$fname`, `$year`):

```
function familyName($fname, $year) {  
    echo "$fname Refsnes. Born in $year <br>";  
}  
  
familyName("Hege", "1975");  
familyName("Stale", "1978");  
familyName("Kai Jim", "1983");
```



Hege Refsnes. Born in 1975
Stale Refsnes. Born in 1978
Kai Jim Refsnes. Born in 1983

Default Argument Value

- ▶ The following example shows how to use a default parameter. If we call the function `setHeight()` without arguments it takes the default value as argument:

```
function setHeight($minheight = 50) {  
    echo "The height is : $minheight <br>";  
}  
  
setHeight(350);  
setHeight(); // will use the default value of 50  
setHeight(135);  
setHeight(80);
```



The height is : 350
The height is : 50
The height is : 135
The height is : 80

► Functions – Returning values

- To let a function return a value, use the **return** statement:

```
function sum($x, $y) {  
    $z = $x + $y;  
    return $z;  
}  
  
echo "5 + 10 = " . sum(5, 10) . "<br>";  
echo "7 + 13 = " . sum(7, 13) . "<br>";  
echo "2 + 4 = " . sum(2, 4);
```



5 + 10 = 15

7 + 13 = 20

2 + 4 = 6

Chapter 5

Array

By: Asst. Lect. Montadhar Moslem



Arrays

- ▶ What is an Array?
- ▶ An array stores multiple values in one single variable.
- ▶ An array is a special variable that can hold many values under a single name, and you can access the values by referring to an index number or name.

```
$cars = array("Volvo", "BMW", "Toyota");
```



```
array(3) {  
    [0]=>  
    string(5) "Volvo"  
    [1]=>  
    string(3) "BMW"  
    [2]=>  
    string(6) "Toyota"  
}
```

Create Array

- ▶ You can create arrays by using the `array()` function:

```
$cars = array("Volvo", "BMW", "Toyota");
```

- ▶ You can also use a shorter syntax by using the `[]` brackets:

```
$cars = ["Volvo", "BMW", "Toyota"];
```

- ▶ Multiple Lines: Line breaks are not important, so an array declaration can span multiple lines:

```
$cars = [  
    "Volvo",  
    "BMW",  
    "Toyota"  
];
```

Create Array

- ▶ Trailing Comma: A comma after the last item is allowed:

```
$cars = [  
    "Volvo",  
    "BMW",  
    "Toyota",  
];
```

- ▶ Array Keys: When creating indexed arrays the keys are given automatically, starting at 0 and increased by 1 for each item, so the array above could also be created with keys:

```
$cars = [  
    0 => "Volvo",  
    1 => "BMW",  
    2 => "Toyota"  
];
```

Exercise

Exercise[?]

What is a correct syntax for creating arrays in PHP?

- `$fruits = ('Apple', 'Banana', 'Orange');`
- `$fruits['Apple', 'Banana', 'Orange'];`
- `$fruits = array('Apple', 'Banana', 'Orange');`

Array Types

- ▶ In PHP, there are three types of arrays:
 - **Indexed arrays** - Arrays with a numeric index
 - **Associative arrays** - Arrays with named keys
 - **Multidimensional arrays** - Arrays containing one or more arrays
- ▶ Array Items
 - Array items can be of any data type.
 - The most common are strings and numbers (int, float), but array items can also be objects, functions or even arrays.
 - You can have different data types in the same array.

Indexed Arrays

- ▶ In indexed arrays each item has an index number.
- ▶ By default, the first item has index 0, the second item has item 1, etc.

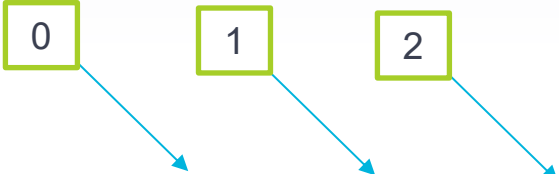
```
$cars = array("Volvo", "BMW", "Toyota");  
var_dump($cars);
```



```
array(3) {  
    [0]=>  
    string(5) "Volvo"  
    [1]=>  
    string(3) "BMW"  
    [2]=>  
    string(6) "Toyota"  
}
```

Access Indexed Arrays

- ▶ To access an array item you can refer to the index number.



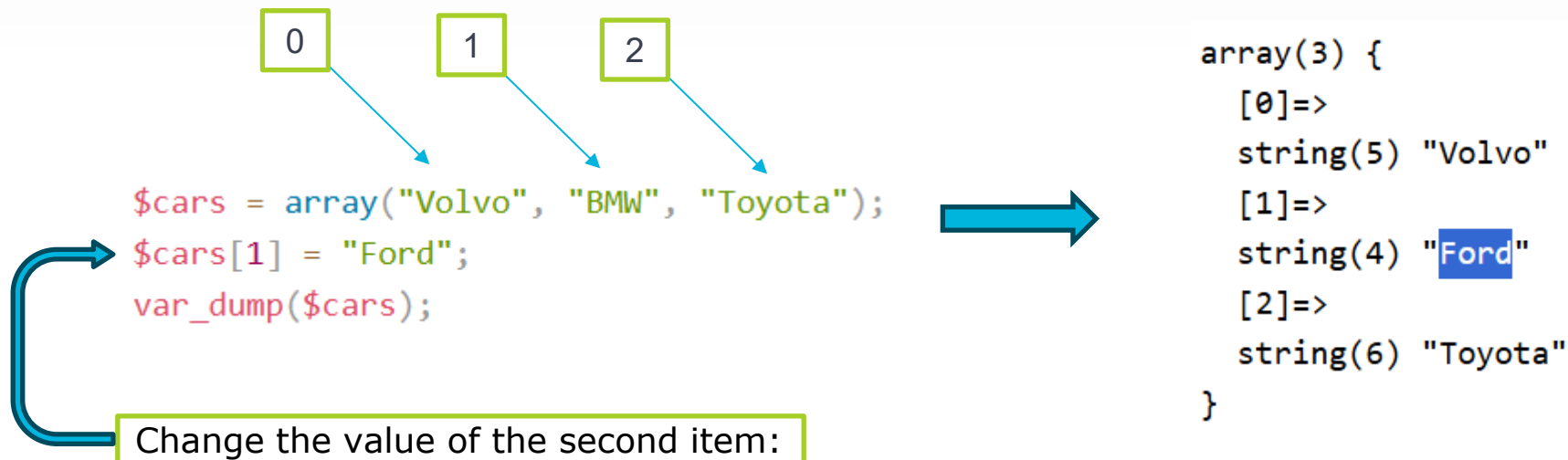
```
$cars = array("Volvo", "BMW", "Toyota");  
echo $cars[0];
```



Volvo

Change Value

- ▶ To change the value of an array item, use the index number:



▶ Loop Through an Indexed Array

- ▶ To loop through and print all the values of an indexed array, you could use a foreach loop, like this:

```
$cars = array("Volvo", "BMW", "Toyota");  
foreach ($cars as $x) {  
    echo "$x <br>";  
}
```



Volvo
BMW
Toyota

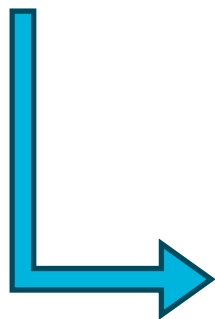
Associative Arrays



► Associative Arrays

- Associative arrays are arrays that use named keys that you assign to them.

```
$car = array("brand"=>"Ford", "model"=>"Mustang", "year"=>1964);  
var_dump($car);
```

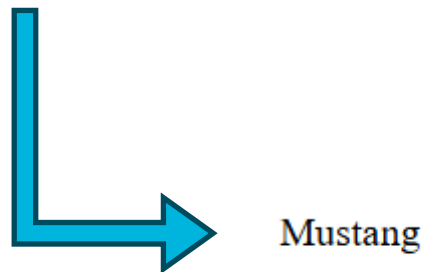


```
array(3) {  
    ["brand"]=>  
    string(4) "Ford"  
    ["model"]=>  
    string(7) "Mustang"  
    ["year"]=>  
    int(1964)  
}
```

Access Associative Arrays

- ▶ To access an array item you can refer to the key name.

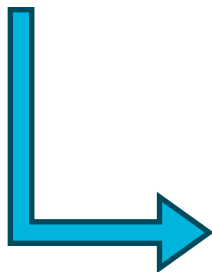
```
$car = array("brand"=>"Ford", "model"=>"Mustang", "year"=>1964);  
echo $car["model"];
```



Access Associative Arrays

- ▶ Double or Single Quotes: You can use both double and single quotes when accessing an array:

```
$cars = array("brand" => "Ford", "model" => "Mustang", "year" => 1964);  
echo $cars["model"];  
echo $cars['model'];
```

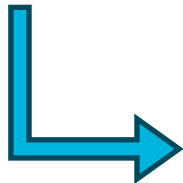


Mustang
Mustang

Change Value

- ▶ To change the value of an array item, use the key name:

```
$car = array("brand"=>"Ford", "model"=>"Mustang", "year"=>1964);  
$car["year"] = 2024;  
var_dump($car);
```



```
array(3) {  
    ["brand"]=>  
        string(4) "Ford"  
    ["model"]=>  
        string(7) "Mustang"  
    ["year"]=>  
        int(2024)  
}
```

▶ Loop Through an Associative Array

- ▶ To loop through and print all the values of an associative array, you could use a `foreach` loop, like this:

```
$car = array("brand"=>"Ford", "model"=>"Mustang", "year"=>1964);
```

```
foreach ($car as $x => $y) {  
    echo "$x: $y <br>";  
}
```



```
brand: Ford  
model: Mustang  
year: 1964
```

Exercise

What will the following array look like after the update:

```
$fruits = array('Apple', 'Banana', 'Orange');  
$fruits[1] = 'Kiwi';
```

- ('Kiwi', 'Banana', 'Orange');
- ('Apple', 'Kiwi', 'Orange');
- ('Kiwi', 'Apple', 'Banana', 'Orange');
- ('Apple', 'Kiwi', 'Banana', 'Orange');

► Add Array Items

- To add items to an existing array, you can use the bracket `[]` syntax.

Add one more item to the `fruits` array:

```
<?php
$fruits = array("Apple", "Banana", "Cherry");
$fruits[] = "Orange";

var_dump($fruits);
?>
```



```
array(4) {
    [0]=>
        string(5) "Apple"
    [1]=>
        string(6) "Banana"
    [2]=>
        string(6) "Cherry"
    [3]=>
        string(6) "Orange"
}
```

Add Array Items

- ▶ To add items to an **associative** array, or key/value array, use brackets `[]` for the key, and assign value with the `=` operator.

```
<?php
$cars = array("brand" => "Ford", "model" => "Mustang");
$cars["color"] = "Red";

var_dump($cars);
?>
```



```
array(3) {
    ["brand"]=>
    string(4) "Ford"
    ["model"]=>
    string(7) "Mustang"
    ["color"]=>
    string(3) "Red"
}
```

Exercise

For the following array:

```
$fruits = array('Apple', 'Banana', 'Orange');
```

What will be a correct syntax to add a new array item?

- `$fruits[] = 'Pineapple';`
- `add_item($fruits, 'Pineapple');`
- `$fruits.add_item('Pineapple');`

Delete Array Items

- ▶ Using the `unset()` Function

Example

Remove the second item:

```
$cars = array("Volvo", "BMW", "Toyota");  
unset($cars[1]);
```



```
array(2) {  
    [0]=>  
        string(5) "Volvo"  
    [2]=>  
        string(6) "Toyota"  
}
```

Delete Array Items

- ▶ The `unset()` function takes a unlimited number of arguments, and can therefore be used to delete multiple array items:

```
$cars = array("Volvo", "BMW", "Toyota");  
unset($cars[0], $cars[1]);
```

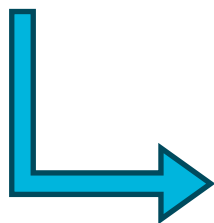


```
array(1) {  
    [2]=>  
    string(6) "Toyota"  
}
```

Delete Array Items

- ▶ Remove Item From an Associative Array
- ▶ To remove items from an associative array, you can use the unset() function.
- ▶ Specify the key of the item you want to delete.

```
$cars = array("brand" => "Ford", "model" => "Mustang", "year" => 1964);  
unset($cars["model"]);
```



```
array(2) {  
    ["brand"]=>  
        string(4) "Ford"  
    ["year"]=>  
        int(1964)  
}
```

Chapter 6

String



By: Asst. Lect. Montadhar Moslem

PHP Strings

- ▶ A string is a sequence of characters, like "Hello world!".
- ▶ Strings in PHP are surrounded by either double quotation marks, or single quotation marks.
- ▶ **Link:** https://www.w3schools.com/php/php_string.asp

Double or Single Quotes?

- ▶ You can use double or single quotes, but you should be aware of the differences between the two.
- ▶ Double quoted strings perform action on special characters.
- ▶ E.g. when there is a variable in the string, it returns the value of the variable:

```
<?php
  $x = "John";
  echo "Hello $x"; ➡ Hello John

  echo 'Hello $x'; ➡ Hello $x
?>
```

String Functions



String Length

- ▶ **strlen()** function returns the length of a string.

```
<?php
    $x = "Salam";
    echo strlen($x);
?>
```



5

Word Count

- ▶ **str_word_count()** function counts the number of words in a string.

```
<?php
    $x = "Hi my name is ali";
    echo str_word_count($x);
?>
```

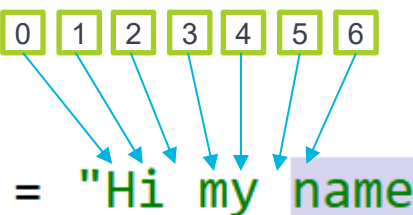


5

Search For Text Within a String

- ▶ **strpos()** function searches for a specific text within a string.
- ▶ If a match is found, the function returns the character position of the first match. If no match is found, it will return FALSE.

```
<?php
    0 1 2 3 4 5 6
    $x = "Hi my name is ali";
    $word = "name";
    echo strpos($x , $word);
?>
```



6

- ▶ Search for the text "name" in the string "Hi my name is ali"

Exercise?

What will be the output of the following code:

```
$x = 5;  
echo 'The price is $x';
```

- ☐ The price is 5
- ☐ The price is
- ☐ The price is \$x

Upper Case

- ▶ **strtoupper()** function returns the string in upper case:

```
<?php
```

```
    $word = "ali";
```

```
    echo strtoupper($word);
```

```
?>
```

ALI



▶ Lower Case

- ▶ **strtolower()** function returns the string in lower case:

```
<?php
    $word = "ALi";
    echo strtolower($word);
```



ali

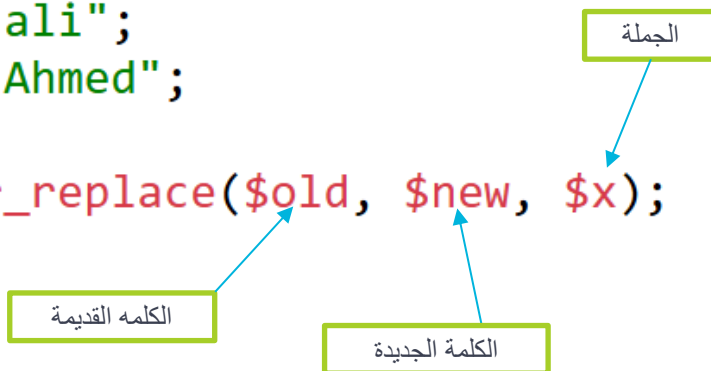
?>

Replace String

- ▶ **str_replace()** function replaces some characters with some other characters in a string.
- ▶ Replace the text "ali" with "Ahmed":

```
<?php
    $x = "Hi my name is ali";
    $old = "ali";
    $new = "Ahmed";

    echo str_replace($old, $new, $x);
?>
```



Hi my name is Ahmed



Reverse a String

- ▶ **strrev()** function reverses a string.

```
<?php
    $x = "Ahmed";
    echo strrev($x);
?>
```



demhA

Remove Whitespace

- ▶ **trim()** function removes any whitespace from the beginning or the end.
- ▶ Whitespace is the space before and/or after the actual text, and very often you want to remove this space.

```
<?php
    $x = " Ahmed ";
    echo trim($x);
?>
```



Ahmed

Convert String into Array

- ▶ **explode()** function splits a string into an array.
- ▶ The first parameter of the explode() function represents the "separator". The "separator" specifies where to split the string.
- ▶ **Note:** The separator is required.
- ▶ **Example:** Split the string into an array. Use the space character as separator:

```
<?php
    $x = "Green Black Red";
    $y = explode(" ", $x);
    var_dump($y);
?>
```



```
array(3) {
    [0]=>
    string(5) "Green"
    [1]=>
    string(5) "Black"
    [2]=>
    string(3) "Red"
}
```

String Concatenation

- ▶ To concatenate, or combine, two strings you can use the `.` Operator or double quotes.

```
<?php
  $x = "Hello";
  $y = "Ali";
  $z = $x . $y;
  echo $z;
?>
```

Diagram: A blue arrow points from the output of the PHP code to the text "HelloAli".

```
<?php
  $x = "Hello";
  $y = "Ali";
  $z = "$x $y";
  echo $z;
?>
```

Diagram: A blue arrow points from the output of the PHP code to the text "Hello Ali".

```
<?php
  $x = "Hello";
  $y = "Ali";
  $z = $x . " " . $y;
  echo $z;
?>
```

Diagram: A blue arrow points from the output of the PHP code to the text "Hello Ali".

▶ Slicing Strings

- ▶ You can return a range of characters by using the **substr()** function.
- ▶ Specify the start index and the number of characters you want to return.
- ▶ Note The first character has index 0.
- ▶ Example: Start the slice at index **6** and end the slice **5** positions later.

```
echo substr($word, $start, $length);
```

Two blue arrows originate from the text in the previous block. One arrow points from the red number '6' to the red parameter '\$start' in the code. The other arrow points from the red number '5' to the red parameter '\$length' in the code.

- ▶ **Slice to the End:** By leaving out the length parameter, the range will go to the end

Example

```
<?php
    $x = "My Name Is Ali";
    echo substr($x, 0, 2);
    echo substr($x, 3, 4);
    echo substr($x, 3);
    echo substr($x, 8, 2);
    echo substr($x, 11, 3);
?>
```

My
Name
Name Is Ali
Is
Ali

Escape Character

- ▶ To insert characters that are illegal in a string, use an escape character.
- ▶ An escape character is a backslash `\` followed by the character you want to insert.

```
<?php
    echo "hi  \$ali  salem"; —————> hi $ali salem
    echo "hi  \"ali\"  salem"; —————> hi "ali" salem
?>
```

Code	Result
\'	Single Quote
\"	Double Quote
\\$	PHP variables

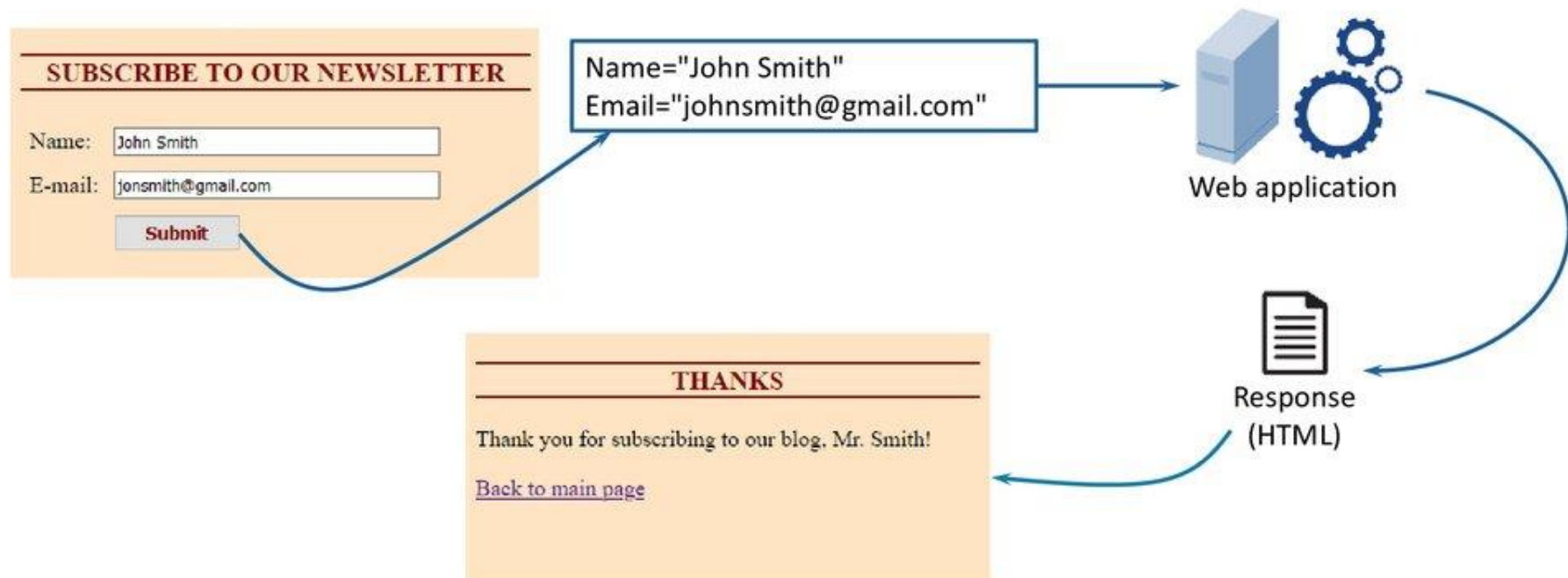
Chapter 7

Form



By: Asst. Lect. Montadhar Moslem

Forma va server



PHP Form Handling

- ▶ The PHP superglobals **`$_GET`** and **`$_POST`** are used to collect form data.
- ▶ A Simple HTML Form: The example below displays a simple HTML form with two input fields and a submit button:
- ▶ When the user fills out the form and clicks the submit button, the form data is sent for processing to a PHP file named "**`welcome.php`**". The form data is sent with the HTTP **`POST`** method.

```
<html>
```

```
<body>
```

```
<form action="welcome.php" method="POST">
```

```
Name: <input type="text" name="name"><br>
```

```
E-mail: <input type="text" name="email"><br>
```

```
<input type="submit">
```

```
</form>
```

```
</body>
```

```
</html>
```



Name:

E-mail:

PHP Form Handling

- ▶ The "`welcome.php`" looks like this:

```
<html>
<body>

Welcome <?php echo $_POST["name"]; ?><br>
Your email address is: <?php echo $_POST["email"]; ?>

</body>
</html>
```

The output could be something like this:

```
Welcome John
Your email address is john.doe@example.com
```

GET Method

- ▶ The same result could also be achieved using the HTTP GET method:

Same example, but the method is set to GET instead of POST:

```
<html>
<body>

<form action="welcome_get.php" method="GET">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>

</body>
</html>
```

and "welcome_get.php" looks like this:

```
<html>
<body>

Welcome <?php echo $_GET["name"]; ?><br>
Your email address is: <?php echo $_GET["email"]; ?>

</body>
</html>
```

▶ When to use GET?

- ▶ Information sent from a **form** with the **GET** method is **Visible To Everyone (مرئية)** (all variable names and values are displayed in the URL).
- ▶ GET also has limits on the amount of information to send. The limitation is about 2000 characters.
- ▶ However, because the variables are displayed in the URL, it is possible to bookmark the page. This can be useful in some cases.
- ▶ GET may be used for sending non-sensitive data.
- ▶ **Note:** GET should NEVER be used for sending passwords or other sensitive information!

▶ When to use POST?

- ▶ Information sent from a form with the POST method is **Invisible To Others** (غير مرئي) (all names/values are embedded within the body of the HTTP request) and has no limits on the amount of information to send.
- ▶ Moreover POST supports advanced functionality such as support for multi-part binary input while uploading files to server.
- ▶ However, because the variables are not displayed in the URL, it is not possible to bookmark the page.

Exercise

Exercise?

What HTML attribute is used to specify the HTTP method of a form?

- ☐ httpmethod
- ☐ action
- ☐ method

Example

- Write a PHP program that takes three numbers from a **form** then calculates their sum, and displays the result.

```
<html>
<body>

    <form action="sum.php" method="Get">

        Number 1: <input type="text" name="n1"> <br>
        Number 2: <input type="text" name="n2"> <br>
        Number 3: <input type="text" name="n3"> <br>

        <input type="submit">

    </form>

</body>
</html>
```

الصفحة التي سيتم نقل البيانات اليها

طريقة نقل البيانات اما
POST or GET

اسم مربع الادخال

زر ارسال البيانات

```
<html>
<body>
```

```
<?php
```

اسم متغير من اختيار المبرمج

```
$num1 = $_GET["n1"];
$num2 = $_GET["n2"];
$num3 = $_GET["n3"];
```

```
$sum = $num1 + $num2 + $num3;
```

```
echo $sum;
```

```
?>
```

```
</body>
</html>
```

الكود الخاص بصفحة

sum.php

Form with radio and checkbox



Form With radio Button Input

► Home.php

```
<html>
<body>

<h2>Choose your favorite Web language:</h2>

<form action="chk.php" method="POST">
  <input type="radio" name="fav_language" value="HTML">
  <label>HTML</label><br>

  <input type="radio" name="fav_language" value="CSS">
  <label>CSS</label><br>

  <input type="radio" name="fav_language" value="JavaScript">
  <label>JavaScript</label><br>

  <input type="submit" value="Save">
</form>

</body>
</html>
```



Choose your favorite Web language:

- ☐ HTML
- ☐ CSS
- ☒ JavaScript

Save

Form With radio Button Input

► chk.php

```
<?php
```

```
if (isset($_POST['fav_language'])) {  
    $x = $_POST["fav_language"];  
    echo "My favorite Web language <b>$x</b>";  
} else {  
    echo "No.";  
}
```

```
?>
```



No.

My favorite Web language **HTML**

My favorite Web language **CSS**

My favorite Web language **JavaScript**

Form With checkbox Input

► Home.php

```
<html>
<body>

<h2>Choose your favorite Web language:</h2>

<form action="chk.php" method="POST">
  <input type="checkbox" name="html_lang" value="HTML">
  <label>HTML</label><br>

  <input type="checkbox" name="css_lang" value="CSS">
  <label>CSS</label><br>

  <input type="checkbox" name="js_lang" value="JavaScript">
  <label>JavaScript</label><br>

  <input type="submit" value="Save">
</form>

</body>
</html>
```



Choose your favorite Web language:

- ☒ HTML
- ☒ CSS
- ☒ JavaScript

Save

Form With checkbox Input

► chk.php

```
<?php
```

```
$x = $_POST["html_lang"];  
$y = $_POST["css_lang"];  
$z = $_POST["js_lang"];
```

```
echo "My favorite Web language: <br>";  
echo "$x <br>";  
echo "$y <br>";  
echo "$z <br>";
```

```
?>
```



My favorite Web language:
HTML
CSS
JavaScript

Example

▶ Home.php

```
<html>
  <body>
    <h2>Enter 2 Numbers:</h2>
    <form action="chk.php" method="POST">
      Number 1: <input type="text" name="num1"> <br>
      Number 2: <input type="text" name="num2"> <br>

      <input type="radio" name="operation" value="sum">
      <label>Sum +</label><br>

      <input type="radio" name="operation" value="mult">
      <label>mult *</label><br>

      <input type="submit" value="go">
    </form>
  </body>
</html>
```



Enter 2 Numbers:

Number 1:

Number 2:

☐ Sum +

☐ mult *

Example

- ▶ chk.php

```
<?php
```

```
$n1 = $_POST["num1"];  
$n2 = $_POST["num2"];  
$op = $_POST["operation"];
```

```
if ($op == "sum"){  
    $sum = $n1+$n2;  
    echo "number1 + number2 = <b>$sum</b>";  
} elseif($op == "mult") {  
    $mul = $n1*$n2;  
    echo "number1 * number2 = <b>$mul</b>";  
}
```

```
?>
```

Example

Enter 2 Numbers:

Number 1:

Number 2:

☒ Sum +
☐ mult *



$$\text{number1} + \text{number2} = 7$$

Enter 2 Numbers:

Number 1:

Number 2:

☐ Sum +
☒ mult *



$$\text{number1} * \text{number2} = 10$$

اسئلة محلولة



By: Asst. Lect. Montadhar Moslem

Question 1

- Write a PHP program to calculate the sum of the elements in an array.

كيفية يمكنك إيجاد مجموع عناصر مصفوفة

```
<?php
$numbers = [10, 20, 30, 40, 50]; // مصفوفة تحتوي على أرقام
$sum = 0; // متغير لتخزين المجموع

foreach ($numbers as $number) {
    $sum = $sum + $number; // إضافة كل عنصر إلى المجموع
}

echo " SUM = " . $sum;

?>
```

Question 2

- Write a PHP program that prints the elements of an array inside an HTML order list.

كيف يمكنك طباعة عناصر المصفوفة داخل لسته مرتبة

```
<!DOCTYPE html>
<html>
<body>
  <ol>
    <?php
      $colors = array("red", "green", "blue", "yellow");

      foreach ($colors as $x) {
        echo "<li> $x </li>";
      }
    <?>
  </ol>
</body>
</html>
```



1. red
2. green
3. blue
4. yellow

Question 3

- Write a PHP program to check if a specific number exists in an array or not.

كيفية التحقق مما إذا كانت قيمة موجودة في المصفوفة

```
<?php
$array = array(10, 20, 30, 40, 50); // Define the array
$number = 30; // Define the number to search for
$found = false; // Flag to track if the number is found

foreach ($array as $value) {
    if ($value == $number) {
        $found = true;
        break; // Exit the loop once the number is found
    }
}

// Check if the number was found
if ($found) {
    echo "The number $number exists in the array.";
} else {
    echo "The number $number does not exist in the array.";
}
?>
```



The number 30 exists in the array.

واجبات



By: Asst. Lect. Montadhar Moslem

Question 4

▶ Write a PHP Program

▶ لديك المصفوفة التي تحتوي على الارقام التالية: { 6, 8, 9, 31, 3, 5, 12, 9, 4 } اكتب برنامج يقوم بايجاد التالي:

▶ 1. معدل الارقام.

▶ 2. قم بطباعة ناتج المعدل بشكل:

▶ • مائل اذا كان المعدل اكبر من 15

▶ • وتحت خط اذا كان المعدل اقل من 15

▶ • وبشكل Bold اذا كان المعدل يساوي 15

Question 5

- ▶ Write a PHP Program.

لديك المصفوفة التي تحتوي على الاسماء التالية: { "Mohammed", "Jawad", "Ahmed", "Zainab", "Hasan", "Ali" }

اكتب برنامج يقوم بايجاد التالي:

1. قم بتغيير اسم Ahmed الى اسم اخر

2. قم بحذف اسم Jawad

3. قم بطباعة المصفوفة داخل لسته غير مرتبه

فيديو تشغيل السيرفر اضغط هنا

او

https://youtu.be/96CJWVH-_3s