Programming Web

Dr. Jamil Alagha
College of Engineering
University of Palestine

Introduction to PHP Fundamental

Part II
Back-End Platform

Outline

- Introduction
- Syntax
- Comment in PHP
- Case Sensitivity
- Variables
- Data types
- String Functions
- Constants
- Operators
- Conditional Statements
- Functions
- Compound Data types
- Arrays

HTML

- **Hyper Text Markup Language**
- HTML used to describe the contents of a web page. It is **not** a **programming language**.
- You simply use HTML to indicate what a certain chunk of text is-such as a paragraph, a heading or specially formatted text.
- All HTML directives are specified using matched sets of angle brackets and are usually called **tags**.
- HTML documents need to have certain tags in order for them to be considered "correct".

JavaScript

- JavaScript is a client-side scripting language
- which means the source code is processed by the client's web browser rather than on the web server.
- This means JavaScript functions can run after a webpage has loaded **without** communicating with the **server**.
- For example, a JavaScript function may check a web form before it is submitted to make sure all the required fields have been filled out.

JavaScript

- Extension to HTML (client side)
- object-based scripting language
- is event-driven

- What is for
    - Interactivity with the page user:

      input - processing - output
- Many web developers now prefer to use JavaScript libraries like **jQuery** to add more advanced dynamic elements to websites.

CSS

- Stands for **"Cascading Style Sheet.“**
- Cascading style sheets are used to format the **layout** of Web pages.
- They can be used to define text styles, table sizes, and other aspects of Web pages that previously could only be defined in a page's HTML.

PHP

- **PHP document Consist:**
    - HTML + JavaScript + CSS + PHP code
- HTML + JavaScript + CSS ➔ rendered at **client side**
- PHP code interpreted at **server side**➔ result any thing, but usually html tags + JavaScript + CSS ➔ rendered at **client side**
- embedded within a
  <?php ... ?> tag

Notes

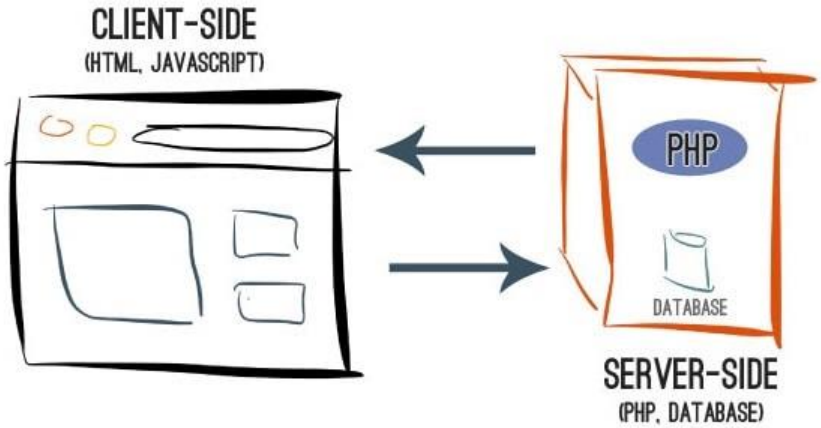
- Focus on server side and database access
    - From client reforming query to be passed to the database server
    - Retrieve data from the database and forward it to the client
- Ensure that the connection is secure
- Transfer data from DBMS to another (JSON) or (XML)

PHP Example

```
<html>
    <head>
        <title>PHP test</title>
    </head>
    <body>
        <?php echo “<p>Hello World</p>";   ?>
    </body>
</html>
```

Client-Side VS. Server-Side

Server side	Vs	Client side
PHP	Vs	HTML + CSS + javascript
Apache Server	Vs	Browser



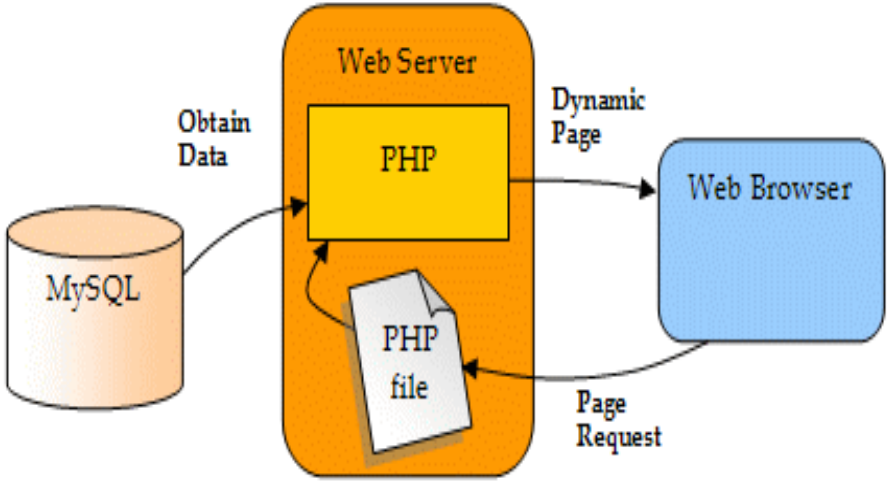
Client-Side VS. Server-Side

- Client-side scripts
  - Validate user input
  - Reduce requests needed to be passed to server
  - Access browser
  - Enhance Web pages with HTML, JavaScript, and CSS
- Server-side scripts
  - Executed on server
  - Generate custom response for clients
  - Wide range of programmatic capabilities
  - Access to server-side software that extends server functionality

Database

- A **database** is an organized collection of data whose content must be **quickly and easily** (Accessed, Managed, Updated)
- A **relational database** is one whose data are split up into **tables**, sometimes called **relations**
- Structured Query Language (**SQL**)
    - Allows for complete
        - Table Creation, Deletion, Editing
        - Data extraction (Queries)
- **MySQL server** – DBMS
    - PHP connected with to access the data

PHP with Database



Accessing Web Servers

- **Request documents from Web servers**
  - Host names
  - Local Web servers
    - Access through machine name or **localhost**
  - Remote Web servers
    - Access through machine name
  - **Domain name** or **Internet Protocol** (IP) address
    - **Domain name server** (DNS)
      - Computer that maintains a database of host names and their corresponding IP address

Apache Web Server

Apache Web Server is Currently the most popular Web server

- Stability
- Efficiency
- Portability
- Open-source

Apache Alternatives

- Amazon Web Services (AWS)
- IBM
- Microsoft
- Oracle
- Eclipse
- Google
- Red Hat
- SAP
- Xampp
- Tomcat

PHP basics

What is PHP ?

- PHP is an acronym for "**P**HP **H**ypertext **P**reprocessor“
- PHP is a widely-used, **open source scripting language**
- PHP scripts are executed on **the server**
- PHP is **free** to download and use
- Suited for web development and can be embedded into HTML.

PHP File

- PHP files can contain text, HTML, CSS, JavaScript, and PHP code
- PHP code are executed on the server, and the result is returned to the browser as plain HTML
- PHP files have extension ".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 **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, PDF files, and even Flash movies**. You can also output any text, **such as XHTML and XML**.

Basic 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

?>
```

- The default file extension for PHP files is ".php".
- A PHP file normally contains HTML tags, and some PHP scripting code.
- **Note:** PHP statements end with a semicolon (;).

Basic 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:

```
<html>
        <body>

            <h1>My first PHP page</h1>

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

        </body>
</html>
```

Comments in PHP

- A comment in PHP code is a line that is not read/executed as 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 what you are doing
    - 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

Comments in PHP

- PHP supports several ways of commenting:

```
<html>
        <body>
                <?php
                        // This is a single-line comment
                        # This is also a single-line comment
                        /*
                        This is a multiple-lines comment block
                        that spans over multiple
                        lines
                        */
                ?>
        </body>
</html>
```

PHP Case Sensitivity

- In PHP, **NO** keywords (e.g. if, else, while, echo, etc.)
- classes, functions, and user-defined functions are case-sensitive.
- In the example below, all three echo statements below are legal (and equal):

```
<html>
        <body>
                <?php
                        ECHO "Hello World!<br>";
                        echo "Hello World!<br>";
                        EcHo "Hello World!<br>";
                ?>
        </body>
</html>
```

PHP Case Sensitivity

- However; all **variable** names are **case-sensitive**.
- In 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):

```
<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>
```

PHP Case Sensitivity

- The output will be

```
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:
- 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**.

PHP Variables

- **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.
- **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)**

Question

Which one is valid variable ?

a. my_function

b. !counter

c. Size

d. _somewhere

e. 4ward

F. This that

PHP Variable - Output

- The PHP **echo, print, printf** statement is often used to output data to the screen.
- Example

```
<?php
        $txt = “PHP";
        echo "I love $txt!";
?>
```

- The same output

```
<?php
        $txt = “PHP";
        echo "I love " . $txt . "!";
?>
```

PHP Variables

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

```
<?php

        print "<h2>PHP is Fun!</h2>";
        print "Hello world!<br>";
        print "I'm about to learn PHP!";
?>
```

Programming Question

- Write an PHP script with three variables: the first with initial value your name and the second with initial value your department name and the last one your average. The script to produce the following

Name	Department	Average
Tala N.	WDMN	85.8

Solution

```
<?php
        $myname = “ali N.";
        $department = "WDMN";
        $average = 85.8;
                echo"<table border=\"1\"><thead> <tr><th>";
                echo"name</th><th>department</th><th>average";
                echo"</th></tr> </thead><tbody><tr><td>";
                echo"$myname</td><td>$department</td><td>$average";
                echo"</td></tr></tbody></table>";
?>
```

PHP Variables Scope

- PHP has three different variable scopes:
  - local
  - Global
  - Static
- A variable declared **outside** a function has a GLOBAL SCOPE and can only be accessed outside a function:

```
<?php
        $x = 5; // global scope
        function myTest() {
        echo "<p>Variable x inside function is: $x</p>";
        }
        myTest();
        echo "<p>Variable x outside function is: $x</p>";
?>
```

PHP Variables Scope

- A variable declared **within** a function has a LOCAL SCOPE and can only be accessed within that function:

```
<?php
        function myTest() {
            $x = 5; // local scope
            echo "<p>Variable x inside function is: $x</p>";
        }
        myTest();
        echo "<p>Variable x outside function is: $x</p>";
 ?>
```

PHP Variable Scope

- Note : 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.

PHP Variable Scope

- PHP The static Keyword
- Normally, when a function is completed/executed, **all of its variables are deleted**. However, sometimes we want a local variable **NOT** to be deleted. We need it for a further job.
- To do this, use the static keyword when you first declare the variable:

```
<?php
        function myTest() {
            static $x = 0;
            echo $x;
            $x++;
        }
        myTest();
        myTest();
        myTest();
?>
```

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

PHP Data Types- 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:

```
<?php
        $x = "Hello world!";
        $y = 'Hello world!';

        echo $x;
        echo "<br>";
        echo $y;
?>
```

PHP Data Types- PHP Integer -

- An integer data type is a non-decimal number between -2,147,483,648 and 2,147,483,647.

- In the following example $x is an integer. The PHP var_dump() function returns the data type and value:

```
<?php
        $x = 5985;
        var_dump($x);
?>
```

- The output of this example is **int(5985)**

PHP Data Types- 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.

```
<?php
        $x = 10.365;
        var_dump($x);
?>
```

The output of this example is **float(10.365)**

PHP Data Types- PHP Binary-

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

```
$x = true;
$y = false;
```

- Booleans are often used in conditional testing.

PHP Data Types - PHP Array -

- An array stores multiple values in one single variable.
- In the following example $cars is an array.

```
<?php
        $cars = array("Volvo","BMW","Toyota");
        var_dump($cars);
?>
```

- **Output:**

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

PHP Data Types - PHP NULL Value -

- Null is a special data type which can have only one value: NULL.
- A variable of data type NULL is a variable that has no value assigned to it.
- **Tip:** If a variable is created without a value, it is automatically assigned a value of NULL.
- Variables can also be emptied by setting the value to NULL:

```
<?php
        $x = "Hello world!";
        $x = null;
        var_dump($x);
?>
```

Github

- Login and create repository
- Open Git bash here
- **Git init**
- **Git add .**
- **Git commit –m ‘first commit’**
- **Git remote add origin *rep url***
- **Git push origin main** OR **git push –f origin main**

- **To download project**
- Open Git bash here
- **Git clone *rep url***

Assignment -1-

- Q1 : Write a PHP script to display the following strings.

  Sample String :

- ' Welcome To PHP'

  'Tomorrow I 'll learn PHP global variables.'

  Expected Output :

  Welcome To PHP

  Tomorrow I 'll learn PHP global variables.

Assignment -1-

- Q2: $var = 'PHP Tutorial'. Put this variable into the title section, h3 tag and as an anchor text within an HTML document.

Sample Output :

- PHP Tutorial
  PHP, an acronym for Hypertext Preprocessor, is a widely-used open source general-purpose scripting language. It is a cross-platform, HTML embedded server-side scripting language and is especially suited for web development. Go to the PHP Tutorial.

PHP String Functions

- In this chapter we will look at some commonly used functions to manipulate strings.
- **Get The Length of a String**
- The PHP **strlen()** function returns the length of a string.
- The example below returns the length of the string "Hello world!":

```
<?php
  echo strlen("Hello world!");
?>
```

- The output of the code above will be: 12.

PHP String Functions

- **Count The Number of Words in a String**
- The PHP **str_word_count()** function counts the number of words in a string:
- Example:

```
<?php
        echo str_word_count("Hello world!");
?>
```

- The output of the code above will be: 2.

PHP String Functions

- **Reverse a String**
- The PHP **strrev()** function reverses a string:

```
<?php
        echo strrev("Hello world!");
?>
```

- The output of the code above will be: !dlrow olleH.

PHP String Functions

- **Search For a Specific Text Within a String**
- The PHP **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**.
- The example below searches for the text "world" in the string "Hello world!":

```
<?php
        echo strpos("Hello world!", "world"); // outputs 6
?>
```

PHP String Functions

- **Replace Text Within a String**
- The PHP **str_replace()** function replaces some characters with some other characters in a string.
- The example below replaces the text "world" with “PHP":

```
<?php
    echo str_replace("world", “PHP", "Hello world!");
?>
```

- The output of the code above will be: Hello PHP!

PHP String Functions Assignment #2

- **Give a description of those String functions with example:**

1. rtrim()
2. similar_text()
3. str_split()
4. str_word_count()
5. str_shuffle()
6. strtolower()
7. substr()

Escaping characters

- characters with special meanings that might be interpreted incorrectly
- $text = 'My spelling's atroshus'; // Erroneous syntax
- To correct this, add a backslash

  $text = 'My spelling\'s still atroshus’;
- to insert tabs, newlines, and carriage returns. by \t, \n, and \r

  $heading = "Date\tName\tPayment";

PHP Constants

- Constants are like variables except that once they are defined they **cannot** be **changed** or **undefined**.
- A **valid** constant name starts with a **letter** or **underscore** (no $ sign before the constant name).
- **Create a PHP Constant**
    - To create a constant, use the **define()** function.
- **Syntax**
    - **define(name, value, case-insensitive)**
- **Parameters:**
    - **name**: Specifies the name of the constant
    - **value**: Specifies the value of the constant
    - **case-insensitive**: Specifies whether the constant name should be case-insensitive. Default is false

PHP Constants

- The example below creates a constant with a **case sensitive** name:

```
<?php
    define("GREETING", "Welcome to W3Schools.com!");
    echo GREETING;
?>
```

- The example below creates a constant with a **case-insensitive** name:

```
<?php
  define("GREETING", "Welcome to W3Schools.com!", true);
  echo greeting;
?>
```

Predefined Constants

_LINE__	The current line number of the file.
__FILE__	The full path and filename of the file
__DIR__	The directory of the file.
__FUNCTION__	The function name.
__CLASS__	The class name.
__METHOD__	The class method name.
__NAMESPACE__	The name of the current namespace (case-sensitive

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

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 (Introduced in PHP 5.6)

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
x = y	x = y	The left operand gets set to the value of the expression on the right
x += y	x = x + y	Addition
x -= y	x = x - y	Subtraction
x *= y	x = x * y	Multiplication
x /= y	x = x / y	Division
x %= y	x = x % y	Modulus

PHP Comparison Operators

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

Operator	Name	Example	Result
==	Equal	$x == $y	Returns true if $x is equal to $y
===	Identical	$x === $y	Returns true if $x is equal to $y, and they are of the same type
!=	Not equal	$x != $y	Returns true if $x is not equal to $y
<>	Not equal	$x <> $y	Returns true if $x is not equal to $y
!==	Not identical	$x !== $y	Returns true if $x is not equal to $y, or they are not of the same type
>	Greater than	$x > $y	Returns true if $x is greater than $y
<	Less than	$x < $y	Returns true if $x is less than $y
>=	Greater than or equal to	$x >= $y	Returns true if $x is greater than or equal to $y
<=	Less than or equal to	$x <= $y	Returns true if $x is less than or equal to $y

PHP Increment / Decrement Operators

- The PHP increment operators are used to increment a variable's value.
- The PHP decrement operators are used to decrement a variable's value.

Operator	Name	Description
++$x	Pre-increment	Increments $x by one, then returns $x
$x++	Post-increment	Returns $x, then increments $x by one
--$x	Pre-decrement	Decrements $x by one, then returns $x
$x--	Post-decrement	Returns $x, then decrements $x by one

PHP Logical Operators

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

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

PHP String Operators

- PHP has **two operators** that are specially designed for **strings**.

Operator	Name	Example	Result
.	Concatenation	$txt1 . $txt2	Concatenation of $txt1 and $txt2

```
<?php
        $txt1 = "Hello";
        $txt2 = " world!";
        echo $txt1 . $txt2;// output: Hello world!
?>
```

PHP Array Operators

- The PHP array operators are used to compare arrays.

Operator	Name	Example	Result
+	Union	$x + $y	Union of $x and $y
==	Equality	$x == $y	Returns true if $x and $y have the same key/value pairs
===	Identity	$x === $y	Returns true if $x and $y have the same key/value pairs in the same order and of the same types
!=	Inequality	$x != $y	Returns true if $x is not equal to $y
<>	Inequality	$x <> $y	Returns true if $x is not equal to $y
!==	Non-identity	$x !== $y	Returns true if $x is not identical to $y

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 Conditional Statements

- **PHP - The if Statement**
  - The if statement executes some code if one condition is true.
- **Syntax**

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

- **Example**

```
<?php
    $t = 7;
        if ($t < 20) {
            echo "Have a good day!";
         }
 ?>
```

PHP Conditional Statements

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

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

- **Example:**

```
<?php
        $t = 7;
        if ($t < 20) {echo "Have a good day!";
        } else {echo "Have a good night!";
        }
?>
```

PHP Conditional Statements

- **The Ternary Operator**
- The **ternary operator** provides a **shorthand way of writing the if...else statements**. The ternary operator is represented by the **question mark (?) symbol** and it takes three operands: **a condition to check, a result for true, and a result for false**.
- **Example**

```
<?php
        if($age < 18){
                echo 'Child';
        } else{
                echo 'Adult';}
?>
```

PHP Conditional Statements

```
<?php echo ($age < 18) ? 'Child' : 'Adult'; ?>
```

- The ternary operator in the example above selects the value on the left of the colon (i.e. 'Child') if the condition evaluates to true (i.e. if $age is less than 18), and selects the value on the right of the colon (i.e. 'Adult') if the condition evaluates to false.

PHP Conditional Statements

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

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

PHP Conditional Statements

- **The PHP 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 Conditional Statements

- **Example**

```
<?php
        $favcolor = "red";
        switch ($favcolor) {
                case "red":
                        echo "Your favorite color is red!";
                        break;
                case "blue":
                        echo "Your favorite color is blue!";
                        break;
                case "green":
                        echo "Your favorite color is green!";
                        break;
                default:
                        echo "Your favorite color is neither red, blue,
                nor green!";
                }
?>
```

PHP Conditional Statements

- **PHP Loops**
  - Often when you write code, you want the same block of code to run over and over again in a row. Instead of adding several almost equal code-lines in a script, we can use loops to perform a task like this.
- **In PHP, we have the following looping statements:**
  - **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

PHP Conditional Statements

- **The PHP 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;
}
```

- **Example**

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

PHP Conditional Statements

- **The PHP 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);
```

PHP Conditional Statements

- **Example**

```
<?php
    $x = 1;


        do {
            echo "The number is: $x <br>";
            $x++;
        } while ($x <= 5);
?>
```

PHP Conditional Statements

- Notice that in a **do while** loop the condition is tested **AFTER** executing the statements within the loop. **This means that the do while loop would execute its statements at least once, even if the condition is false the first time.**
- **Example**

```
<?php
    $x = 6;

    do {
        echo "The number is: $x <br>";
        $x++;
    } while ($x <= 5);
?>
```

PHP Conditional Statements

- **The PHP 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;
}
```

- **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

PHP Conditional Statements

- **Example**

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

PHP Conditional Statements

- **The PHP 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.

PHP Conditional Statements

- **Example**

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

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

PHP Functions

- **PHP User Defined Functions**
  - Besides the built-in PHP functions, we can create our own functions.
  - A function is a block of statements that can be used repeatedly in a program.
  - A function will not execute immediately when a page loads.
  - A function will be executed by a call to the function.
- **Create a User Defined Function in PHP**
  - A user-defined function declaration starts with the word **function**
- **Syntax**

```
function functionName() {
    code to be executed;
}
```

PHP Functions

- **Note:** A function name can start with a letter or underscore (not a number).
- **Tip:** Give the function a name that reflects what the function does!
- Function names are NOT case-sensitive.
- **Example**

```
<?php
    function writeMsg() {
            echo "Hello world!";
    }

    writeMsg(); // call the function
 ?>
```

PHP 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**.
- **Example**

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

        familyName("Jani");
?>
```

PHP Default Argument Value

- **Example**

```
<?php
    function setHeight($minheight = 50) {
        echo "The height is : $minheight <br>";
    }

    setHeight(350);
    setHeight(); // will use the default value of 50
    setHeight(135);
    setHeight(80);
?>
```

PHP Functions - Returning values

- Example

```
<?php
    function sum($x, $y) {
            $z = $x + $y;
            return $z;
    }

    echo "5 + 10 = " . sum(5, 10) . "<br>";
    echo "2 + 4 = " . sum(2, 4);

    ?>
```

Compound Data Types

- Allow for **multiple items** to be aggregated under a single representative entity.
  - Array
  - Object

What is an Array?

- An array is **a special variable**, which can hold more than one value at a time.
- If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

```
$cars1 = "Volvo";
$cars2 = "BMW";
$cars3 = "Toyota";
```

- However, what if you want to loop through the cars and find a specific one? And what if you had not 3 cars, but 300?
- The solution is to create an **array**!

What is an Array?

- An array can hold **many values** under a single name, and you can access the values by referring to an index.
- **Type of Arrays**
  - **Indexed arrays:** Arrays with a numeric index.
  - **Associative array:** Arrays with named keys.
- Create an Array in PHP
  - In PHP, the array() function is used to create an array:
    - array();

Indexed Arrays

- **There are two ways to create indexed arrays:**
- The index can be assigned automatically (index always starts at 0), like this:
    - $cars = array("Volvo", "BMW", "Toyota");

- or the index can be assigned manually:
    - $cars[0] = "Volvo";
      $cars[1] = "BMW";
      $cars[2] = "Toyota";

Example

- Define an array (name it stdGrads) with initial values of student's grades in registered courses. (up to 5 courses)

```
<?php

        $stdGrads[0] = 78.8;

        $stdGrads[1] = 80.5;

        $stdGrads[2] = 81.2;

        $stdGrads[3] = 90.8;

        $stdGrads[4] = 76.6;



    $stdGrads = array (78.8, 80.5, 81.2, 90.8, 76.6);

?>
```

Indexed Array

- To add new value to the array

```
<?php
  $stdGrads[5] = 88.7;
?>
```

Example

- The following example creates an indexed array named $cars, assigns three elements to it, and then prints a text containing the array values:

```
<?php
    $cars = array("Volvo", "BMW", "Toyota");
    echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
?>
```

PHP Associative Arrays

- **Associative arrays** are arrays that use named keys that you assign to them.
- There are two ways to create an associative array:
    - $age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
- or:
    - $age['Peter'] = "35";
      $age['Ben'] = "37";
      $age['Joe'] = "43";

PHP Associative Arrays

- Example

```
<?php
    $age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
    echo "Peter is " . $age['Peter'] . " years old.";
?>
```

Example

- Create an associative array Fname to hold the names of the oldest three brothers. The Indices are the order of each and the values are the names.

```
<?php
        $Fname["oldborother "] = "Ali";
        $Fname["midborother "] = " abed-alrhman";
        $Fname["yangborother "] = "Ahmad";
        $Fname = array (" oldborother "=> "Ali",
                        "midborother "=> "abed-alrhman",
                         "yangborother " => "Ahmad");
?>
```

Dealing with the Array

- **Get The Length of an Array**
- The **count()** function is used to **return the length** (the number of elements) of an array:

- **Example**

```
<?php
    $cars = array("Volvo", "BMW", "Toyota");
    echo count($cars);
?>
```

Dealing with the Array

- **Loop Through an Indexed Array**

Using for loop Print out the contents of $stdGrads

- $stdGrads = array (78.8, 80.5, 81.2, 90.8, 76.6);

```
for ($i=0; $i< count($stdGrads ); $i++){

    echo $stdGrads[i]. "</br>";

  }
```

-
```
foreach ($ stdGrads  as $value) {
    echo "$value <br>";
  }
```

Dealing with the Array

- **Loop Through an Associative Array**

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

foreach($age as $x => $x_value) {
    echo "Key=" . $x . ", Value=" . $x_value;
    echo "</br>";
}
```

Array - Example

- Use array function to create an array of the departments heads as associative array. The department names used as Indices and the head name as value

Solution

```
$departments = array(
                    "cs"=>“IyadSh",
                    "its"=>“Omer",
                    "sdev"=>“Basem",
                   "wdmm"=>" Ashraf“,
                    “mobl” =>”IyadAg”
                  );
```

Programming Question

- Create PHP script that print the figure

1th	2th	3th	4th	5th
78.8	80.5	81.2	90.8	76.6

Solution 1

```
<?php
$stdGrads = array (78.8, 80.5, 81.2, 90.8, 76.6);?>
<table border="1">
    <tr>
        <?php
        for ($i=1;$i<count($stdGrads)+1;$i++){?>
            <td>
                <?php
                echo $i."th"; ?>
            </td>
        <?php }?>
    </tr>
  <tr>
<?php
foreach ($stdGrads as $sg){?>
        <td>
        <?php
        echo $sg; ?>
        </td>
       <?php }?>
</tr>
</table>
```

Solution 2

```php
<?php
echo "<br>";
$stdGrads = array (78.8, 80.5, 81.2, 90.8, 76.6);

echo "<table border='1'>";
echo "<tr>";
for ($i=1;$i<count($stdGrads)+1;$i++){

 echo "<td> $i th</td>";}
echo "</tr> ";

echo "<tr>";
foreach ($stdGrads as $sg){

    echo "<td> $sg</td>";}
echo "</tr> ";

echo "</table>";
```

Array Functions

- **Array Functions**
  - array()
  - count
  - array_key_exists, array_search
  - Key, current, next, prev, reset
  - Sort, rsort

Array Functions

- key() Fetches a key from an array
- current() - Return the current element in an array
- next() - Advance the internal pointer of an array
- Prev() — Rewind the internal array pointer
- Reset() — Set the internal pointer of an array to its first element

Sort Functions For Arrays

Function	Description
sort()	sort arrays in ascending order
rsort()	sort arrays in descending order
asort()	sort associative arrays in ascending order, according to the value
ksort()	sort associative arrays in ascending order, according to the key
arsort()	sort associative arrays in descending order, according to the value
krsort()	sort associative arrays in descending order, according to the key

Example

```
$grades = array(42, 98, 100, 100, 43, 12);
sort($grades);
Print_r($grades);
```

Output:

```
Array ( [0] => 12 [1] => 42 [2] => 43 [3] => 98 [4] => 100 [5] => 100 )
```

Multidimensional Array

- Sometimes you want to **store values** with **more** than **one key**.
- This can be stored in **multidimensional arrays**.
- A multidimensional array is an **array containing one or more arrays**.
- PHP understands multidimensional arrays that are two, three, four, five, or more levels deep.

Multidimensional Array

- A two-dimensional array is an array of arrays.
- First, take a look at the following table:

Name	Stock	Sold
Volvo	22	18
BMW	15	13
Saab	5	2
Land Rover	17	15

Multidimensional Array

- We can store the data from the table above in a two-dimensional array, like this:

```
$cars = array
 (
   array("Volvo",22,18),
   array("BMW",15,13),
   array("Saab",5,2),
   array("Land Rover",17,15)
 );
```

Multidimensional Array

- Now the two-dimensional $cars array contains four arrays, and it has two indices: row and column.
- To get access to the elements of the $cars array we must point to the two indices (row and column):

```
<?php
 echo $cars[0][0].": In stock: ".$cars[0][1].", sold: ".$cars[0][2].".<br>";
 echo $cars[1][0].": In stock: ".$cars[1][1].", sold: ".$cars[1][2].".<br>";
 echo $cars[2][0].": In stock: ".$cars[2][1].", sold: ".$cars[2][2].".<br>";
 echo $cars[3][0].": In stock: ".$cars[3][1].", sold: ".$cars[3][2].".<br>";
?>
```

Multidimensional Array

- We can also put a for loop inside another for loop to get the elements of the $cars array (we still have to point to the two indices):

```
<?php
        for ($row = 0; $row < 4; $row++) {
            echo "<p><b>Row number $row</b></p>";
            echo "<ul>";
           for ($col = 0; $col < 3; $col++) {
                    echo "<li>".$cars[$row][$col]."</li>";
           }
            echo "</ul>";
        }
?>
```

END