B.C.A. Semester – 4

BCA-402
Building Application Using PHP

UNIT - 1
Introduction to PHP

B.C.A. Semester — IV BCA-402 : Building Application Using PHP
Unit 1 : Introduction to PHP

Web-Technologies :

- Web Technology refers to the various tools and techniques that are utilized in the process of communication between different types of devices over the internet. A web browser is used to access web pages. Web browsers can be defined as programs that display text, data, pictures, animation, and video on the Internet. Hyperlinked resources on the World Wide Web can be accessed using software interfaces provided by Web browsers.

Web Technology can be classified into the following sections:

I. HTML
- HTML stands for Hypertext Markup Language and is the standard markup language used for creating web pages.
- HTML is used to structure and display content on the web, including text, images, and links.
- HTML tags are used to define the structure and content of a web page.

II. CSS
- CSS stands for Cascading Style Sheets and is used to style and layout web pages.
- CSS is used to control the visual presentation of web pages, including font, color, spacing, and other design elements.
- CSS is separate from HTML and is typically linked to an HTML document, allowing for easy updates to the visual design without changing the underlying content.

III. JavaScript
- JavaScript is a programming language used to add dynamic behavior to web pages.
- JavaScript is used to create interactivity and animations, validate forms, and perform other actions on the client-side (in the user's web browser).
- JavaScript is often used in conjunction with HTML and CSS to create dynamic web pages that provide a more engaging user experience.

IV. XML
- XML stands for eXtensible Markup Language and is a markup language used for storing and exchanging data.
- XML is similar to HTML but is not used for displaying data on the web. Instead, it is used for transmitting and storing data between applications.
- XML is a flexible and versatile format that is widely used for data exchange between applications and for storing data in a structured format.

Server side technologies :

A form of web server technology in which users' requests are fulfilled by running a script directly on the web server to generate dynamic HTML pages. It is used to provide interactive web sites capable of interfacing with databases and other data stores.

- Server side technologies are programming languages and frameworks used to create applications that run on a web server. Examples of server side technologies include PHP, ASP.NET, Java, Ruby on Rails, and Node.js. These technologies are used to create dynamic web pages and applications that can interact with databases and other web services.

Unit 1 : Introduction to PHP

- **PHP**: As mentioned earlier, PHP is a popular server-side programming language that is often used for web development. It can be used to create dynamic web pages, access and manipulate databases, and perform other tasks on the server side.
- **ASP.NET:** This is a server-side web development framework developed by Microsoft. It uses the .NET programming language and allows developers to create dynamic websites and web applications.
- **Ruby on Rails:** This is a web development framework written in the Ruby programming language. It is known for its simplicity and convention-over-configuration approach, which makes it easy to build web applications quickly.
- **Java:** Java is a popular programming language that is widely used for building enterprise-level applications. It has several frameworks and libraries that can be used for web development, such as Spring and JavaServer Faces.

Web Server :
- A web server is a software program that runs on a computer and is responsible for accepting HTTP requests from clients (usually web browsers) and serving them HTTP responses along with optional data payloads.
- There are many different web servers available, including Apache, Nginx, and Microsoft IIS. Each web server has its own strengths and can be used in different contexts.
- In order to serve web content to the Internet, a web server must be running on a computer that is connected to the Internet and has a static IP address or domain name. When a client requests a web page, the request is sent to the IP address or domain name of the server, which then responds with the requested content.
- Web servers are an important component of the World Wide Web, as they are responsible for hosting and serving the web content that users access through their web browsers.

Web Browser :
- A Web browser is a software application that is used to access and view content on the World Wide Web. Some examples of popular web browsers include **Google Chrome, Mozilla Firefox, Microsoft Edge, and Safari.**
- Web browsers work by sending HTTP requests to web servers, which then respond with the requested content, such as HTML documents, images, and other media. The web browser then parses and renders the content, allowing the user to view and interact with it.
- Web browsers have many features that make it easier to browse the web, such as the ability to bookmark favorite sites, search for content, and access and manage multiple tabs. They also often include security features to protect users from malicious websites and online threats.
- Web browsers are an essential part of the modern Internet, as they allow users to access and interact with the vast amount of information and resources available on the web.

Introduction to PHP :

Unit 1 : Introduction to PHP

- PHP stands for "PHP: Hypertext Preprocessor". It is a popular programming language that is widely used for web development.
- PHP is a server-side language, which means that it is executed on the server and the results are sent to the client (usually a web browser). This is in contrast to client-side languages like JavaScript, which are executed on the client (the user's computer).
- PHP is an open-source language, which means that it is freely available for anyone to use and modify. This has contributed to its popularity, as it allows developers to build and customize applications without incurring licensing fees.
- PHP is widely used in conjunction with databases, as it can be used to retrieve, store, and manipulate data from a database. It is often used in combination with the MySQL database management system.
- PHP has a large and active community of developers, which has contributed to the development and evolution of the language. There are also many resources available online, such as tutorials, documentation, and forums, which can be helpful for learning and troubleshooting.

History of php :

- PHP is a programming language that was created in 1995 by Rasmus Lerdorf. Lerdorf initially developed PHP as a set of tools to assist him in maintaining his personal homepage. He released the source code for these tools publicly, and they quickly gained popularity among web developers.
- In 1997, two developers, Zeev Suraski and Andi Gutmans, rewrote the PHP parser, creating a new implementation of PHP that was more powerful and efficient than the original. This new version of PHP, known as PHP 3, was released in 1998.
- PHP 4 was released in 2000 and included additional features, such as support for object-oriented programming and the ability to interface with databases.
- PHP 5 was released in 2004 and included significant improvements, such as better support for object-oriented programming, improved performance, and more robust support for web services.
- PHP 6, which was intended to be a major update to the language, was in development for several years but was eventually abandoned. PHP 7, which included many of the features that were planned for PHP 6, was released in 2015 and has since become the most widely used version of PHP.

Future of PHP :

- PHP is a popular programming language that has been in use for more than two decades and is widely used for web development. While it is difficult to predict the exact future of any technology, it is likely that PHP will continue to be a widely used language for the foreseeable future. One reason for this is that PHP has a large and active user base, with millions of developers around the world using it to build websites and web applications. This user base ensures that PHP will continue to be supported and developed by the community, even as new technologies emerge. Another reason is that PHP has a number of features and capabilities that make it well-suited for web development, such as its support for databases, functions, and objects.

* Client-side scripting

Client-side scripting refers to programming languages and technologies that are executed on the client (the user's computer) as opposed to the server. This is in contrast to server-side technologies, which are executed on the server and deliver the results to the client.

Some examples of client-side programming languages include : **JavaScript, HTML CSS.**

Client-side scripting is an important aspect of modern web development, as it allows you to create interactive and dynamic experiences for users without requiring constant communication with the server.

* Server-side scripting

- Server-side scripting refers to programming languages and technologies that are executed on the server, as opposed to the client (the user's computer). This is in contrast to client-side technologies, which are executed on the client and deliver the results to the server.
- Some examples of server-side programming languages include:
- **PHP:** PHP is a popular server-side programming language that is widely used for web development. It allows you to create dynamic web pages, access and manipulate databases, and perform other tasks on the server side.
- **ASP.NET:** This is a server-side web development framework developed by Microsoft. It uses the .NET programming language and allows developers to create dynamic websites and web applications.
- Ruby on Rails: This is a web development framework written in the Ruby programming language. It is known for its simplicity and convention-over-configuration approach, which makes it easy to build web applications quickly.
- **Java:** Java is a popular programming language that is widely used for building enterprise-level applications. It has several frameworks and libraries that can be used for web development, such as Spring and JavaServer Faces.
- **Server-side scripting** is an important aspect of modern web development, as it allows you to create dynamic and interactive websites and web applications.

* Adding PHP to HTML Syntax and variable

- To add PHP to an HTML document, you can use the <?php and ?> tags to enclose your PHP code.

- Here is an example of how you might use PHP to output a simple message to an HTML page :

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Hello</title>
</head>
<body>
    <?php
        echo "Hello Idar";
    ?>
</body>
</html>                          Output : Hello Idar
```

* Variables :

In PHP, a variable is declared using a $ sign followed by the variable name. Here, some important points to know about variables

- As PHP is a loosely typed language, so we do not need to declare the data types of the variables. It automatically analyzes the values and makes conversions to its correct datatype.
- After declaring a variable, it can be reused throughout the code.
- Assignment Operator (=) is used to assign the value to a variable.
- PHP is a loosely typed language, it means PHP automatically converts the variable to its correct data type
- Syntax of declaring a variable in PHP is given below:

    $variablename=value;

Example:

```php
$name = "Idar" ;
$age = 20;
$user_id = 12345;
$is_admin = true;
```

* Rules for declaring PHP variable:

- A variable must start with a dollar ($) sign, followed by the variable name.
- It can only contain alpha-numeric characters and underscore (A-z, 0-9, _).
- A variable name must start with a letter or underscore (_) character.
- A PHP variable name cannot contain spaces.
- One thing to be kept in mind is that the variable name cannot start with a number or special symbols.
- PHP variables are case-sensitive, so $name and $NAME both are treated as different variables.

var_dump() function :

- You can use the var_dump() function to output the data type and value of a variable. For example:

```php
var_dump($name); // Output : string(4) "Idar"
var_dump($age); // Output : int(20)
var_dump($user_id); // Output : int(12345)
var_dump($is_admin); // Output : bool(true)
var_dump($fruits);

// Output : array(3) { [0]⇒ string(5) "apple" [1]⇒ string(6) "banana" [2]⇒ string(6) "orange" }
```

* Control and Functions.

- PHP has several control structures that allow you to create conditions and execute different blocks of code depending on the outcome. Here are some examples of control structures in PHP:
- **if statements :** These allow you to execute a block of code if a certain condition is true.

For example :

```php
$x = 5;

  if( $x > 3){
      echo "x is greater than 3";
  } else {
      echo "x is less than 3";
  }                                    // Output : x is greater than 3
```

- **if...else statements:** These allow you to execute a block of code if a certain condition is true, and a different block of code if the condition is false. For example:

```php
$x = 1;

  if( $x > 3){
      echo "x is greater than 3";
  } else {
      echo "x is less than 3";
  }                                    // Output : x is less than 3
```

switch statements : PHP switch statement is used to execute one statement from multiple conditions. It works like a PHP if-else-if statement.

Syntax :

```php
switch(expression){
case value1:
 //code to be executed
 break;
case value2:
 //code to be executed
 break;
     ......
default:
 code to be executed if all cases are not matched;
}
```

PHP Switch Example :

```php
<?php
$num = 20;
switch ($num) {
    case 10:
        echo ("number is equals to 10");
        break;
    case 20:
        echo ("number is equal to 20");
        break;
    case 30:
        echo ("number is equal to 30");
        break;
    default:
        echo ("number is not equal to 10, 20 or 30");
}
?>  // Output: number is equal to 20
```

* PHP For Loop :

PHP for loop can be used to traverse a set of code for the specified number of times.

It should be used if the number of iterations is known otherwise use a while loop. This means for loop is used when you already know how many times you want to execute a block of code.

It allows users to put all the loop related statements in one place. See in the syntax given below:

```php
for(initialization; condition; increment/decrement){

//code to be executed    }
```

For Loop Example :

```php
for ($n = 1; $n <= 10; $n++) {
    echo "$n ";
}  // Output: 1 2 3 4 5 6 7 8 9 10
```

* PHP For Each Loop :

The foreach loop is used to traverse the array elements. It works only on arrays and objects. It will issue an error if you try to use it with the variables of different datatypes.

The foreach loop works on element basis rather than index. It provides an easiest way to iterate the elements of an array.

In the foreach loop, we don't need to increment the value.

Syntax :

```
foreach( $array as $var ){
        //code to be executed
}
```

Example :

```php
<?php

$marvelHeroes = array("Iron Man", "Captain America", "Thor", "Hulk", "Black Widow");

    foreach ($marvelHeroes as $hero) {

    echo $hero . "<br>";

    }

?>
```

Output :

```
Iron Man
Captain America
Thor
Hulk
Black Widow
```

* PHP While Loop :

PHP while loop can be used to traverse a set of code like for loop. The while loop executes a block of code repeatedly until the condition is FALSE. Once the condition gets FALSE, it exits from the body of the loop.

- It should be used if the number of iterations is not known.

- The while loop is also called an **Entry control loop** because the condition is checked before entering the loop body. This means that first the condition is checked. If the condition is true, the block of code will be executed.

Syntax :

```
while(condition){
        //code to be executed
}
```

PHP While Loop Example :

```php
<?php

        $n = 1;
        while ($n <= 10) {
```

```php
        echo "$n ";
        $n++;
    }
?>
// Output : 1 2 3 4 5 6 7 8 9 10
```

Alternative Syntax :
```php
    while(condition) :
    //code to be executed
    Endwhile;
```

Example :
```php
<?php
    $n = 1;
    while ($n ≤ 10) :
        echo "$n ";
        $n++;
    endwhile;
?>
// Output : 1 2 3 4 5 6 7 8 9 1
```

Passing information between page

We often have to pass values of variables between pages in our site. These are required in many different ways. Sometimes we collect some value from a database and want to retain the value for the particular user throughout the site as the user moves between pages. There are different ways for passing such values of the variables. It can be passed within a site or even outside the site. We will discuss some of the ways with their possible uses.

- **Using query strings:** Data can be passed through the URL by appending key-value pairs to the URL as query strings.
- **Using forms:** Data can be passed from one page to another by submitting a form to a different page with superglobal variables $_GET, $_POST, $_REQUEST.
- **Using sessions:** Data can be stored in a session variable, which can be accessed across multiple pages.
- **Using cookies:** Data can be stored in a cookie on the client's browser, which can be accessed by the server on subsequent page loads.

```php
setcookie('name', 'Alice', time() + 86400);
```

This sets a cookie named name with the value "Alice" that will expire in 24 hours (86400 seconds). You can then retrieve the value of the cookie on the second page using the $_COOKIE superglobal array:

```php
$name = $_COOKIE['name'];
echo "Hello, $name!";
```

- **Using database:** Data can be stored in a database and retrieved on another page using database queries.
- **Using hidden fields :** Data can be passed between pages by adding hidden fields to a form which can be accessed on the next page.

```php
<form action="page2.php" method="post">
```

```html
<input type="hidden" name="name" value="Alice">
<input type="submit" value="Submit">
</form>
```

- You can then retrieve the value of name on the second page using the $_POST superglobal array:

```php
$name = $_POST['name'];
echo "Hello, $name!";
```

These are some of the ways of transferring data variables between pages. There is no clear rule to use a particular method to pass the variables between pages. Here are some standard ways of doing things with reasons. All variables need sanitization before using when they arrive from external sources.

Uses	Method	Reason
Login form with Userid and password	POST	Id and password should not be visible in query string
Sending Product id	GET	The url can be bookmarked for future uses.
Feedback form	Post	Several data can be posted
Favorite buying pattern of User	Cookies	Data stored in user computer
User Login status	Session	Secured data is maintained at server end

* Variable Function :

The PHP variable handling functions are part of the PHP core. No installation is required to use these functions.

1. **gettype():** This function returns the data type of a variable. Example:

```php
$num = 100;
$str = "Hello World";
$arr = array(1, 2, 3);
echo gettype($num); // outputs: integer
echo gettype($str); // outputs: string
echo gettype($arr); // outputs: array
```

2. **settype():** This function changes the data type of a variable. Example :

```php
$num = "100";
settype($num, "integer");
echo gettype($num); // outputs: integer
```

3. **isset():** This function checks if a variable is set and is not NULL. Example:

```php
$num = 100;
 if (isset($num)) {
```

```php
        echo "Variable is set.";
    }
```

4. **unset():** This function unset a variable and free up the memory. Example:

```php
    $num = 100;
    unset($num);
    if (!isset($num))
    {
            echo "Variable is unset.";
    }
```

5. **strval():** This function returns the string value of a variable. Example:

```php
    $num = 100;
    echo strval($num); // outputs: "100"
```

6. **floatval():** This function returns the float value of a variable. Example:

```php
    $num = "100.5";
    echo floatval($num); // outputs: 100.5
```

7. **intval():** This function returns the integer value of a variable. Example:

```php
    $num = "100";
    echo intval($num); // outputs: 100
```

8. **print_r() :** This function prints human-readable information about a variable. This is useful for debugging and understanding the structure of complex variables such as arrays and objects. Example:

- **For integer:**

```php
    $num = 100;
    print_r($num); // Output: 100
```

- **For string:**

```php
    $str = "Hello World";
    print_r($str); // Output: Hello World
```

- **For Array:**

```php
    $arr = array(1, 2, 3);
    print_r($arr); // Output: Array ( [0] ⇒ 1 [1] ⇒ 2 [2] ⇒ 3 )
```

- **For Boolean:**

```php
    $flag = true;
    print_r($flag); // Output: 1
```

- **For NULL:**

```php
    $var = null;
```

```
print_r($var); // Output: NULL
```

* PHP Arrays :

PHP array is an ordered map (contains value on the basis of key). It is used to hold multiple values of similar type in a single variable.

Advantage of PHP Array :

- **Less Code**: We don't need to define multiple variables.

- **Easy to traverse:** By the help of a single loop, we can traverse all the elements of an array.

- **Sorting:** We can sort the elements of an array.

PHP Array Types :

There are 3 types of array in PHP.

- Indexed Array

- Associative Array

- Multidimensional Array

PHP Indexed Array :

PHP index is represented by a number which starts from 0. We can store numbers, strings and objects in the PHP array. All PHP array elements are assigned to an index number by default.

Array syntax :

$arrayName = **array**(value1, value2, value3, ...);

Where arrayName is the name of the array and value1, value2, value3, ... are the values of the array.

There are two ways to define indexed array:

- 1st way:

```php
$marvelHeroes=array("IronMan","CaptainAmerica","Thor","Hulk","Black Widow");
```

- 2nd way:

```php
$marvelHeroes[0] = "IronMan";
$marvelHeroes[1] = "CaptainAmerica";
$marvelHeroes[2] = "Thor";
$marvelHeroes[3] = "Hulk";
$marvelHeroes[4] = "Black Widow";
```

- 3rd way :

```php
$marvelHeros = ["IronMan", "CaptainAmerica", "Thor", "Hulk", "Black Widow"];
```

Example :

```php
<?php
$marvelHeroes = array("IronMan", "CaptainAmerica", "Thor", "Hulk", "Black Widow");
```

```php
    echo $marvelHeroes[0] . "<br>";
    echo $marvelHeroes[1] . "<br>";
    echo $marvelHeroes[2] . "<br>";
    echo $marvelHeroes[3] . "<br>";
    echo $marvelHeroes[4] . "<br>";
?>
```

Output :

```
IronMan
CaptainAmerica
Thor
Hulk
Black Widow
```

* PHP Associative Array :

An associative array is a type of array in PHP that allows you to assign a key to each element, rather than using an index number.

We can associate names with each array element in PHP using => symbol.

You can create an associative array using the array() function or [] operator, like this:

```php
$marvelHeroes = array(
    "Iron Man" => "Tony Stark",
    "Captain America" => "Steve Rogers",
    "Thor" => "Thor Odinson",
    "Hulk" => "Bruce Banner",
    "Black Widow" => "Natasha Romanoff"
);
```

- In this example, the keys are the hero names and the values are the real names of the characters.

- You can access the individual elements of the array by their key, like this:

```php
echo $marvelHeroes["Iron Man"];
```

This will output "Tony Stark"

- You can also use a foreach loop to iterate through all elements of the array, like this:

```php
foreach ($marvelHeroes as $hero ⇒ $realName) {
    echo $hero . " is " . $realName . "<br>";
}
```

This will output each hero name and their real name one by one on a new line.

```
Iron Man is Tony Stark
Captain America is Steve Rogers
Thor is Thor Odinson
Hulk is Bruce Banner
```

```
Black Widow is Natasha Romanoff
```

- You can use associative arrays to store more complex data and to make it easier to access and manipulate specific elements of the array.

PHP Array Functions

PHP has a variety of built-in functions for working with arrays. Some of the most important and commonly used array functions include:

1. **array() or []:** This function creates a new array.
2. **count():** This function returns the number of elements in an array.
3. **array_push():** This function adds one or more elements to the end of an array.
4. **array_pop():** This function removes and returns the last element from an array.
5. **array_shift():** This function removes and returns the first element from an array.
6. **array_unshift():** This function adds one or more elements to the beginning of an array.
7. **array_slice():** This function returns a new array that contains a specified range of elements from the original array.
8. **array_splice():** This function removes and/or replaces elements in an array, and returns the removed elements.
9. **array_merge():** This function merges one or more arrays into a single array.
10. **array_keys():** This function returns an array containing all the keys of an array.
11. **array_values():** This function returns an array containing all the values of an array.
12. **array_combine():** This function creates an array by using one array for keys and another for its values
13. **array_reverse():** This function returns an array with elements in reverse order
14. **array_search():** This function search for a specified value in an array and returns the key if found
15. **sort():** This function sorts an array in ascending order, according to the value of each element
16. **rsort():** This function sorts an array in descending order, according to the value of each element

These are some of the most commonly used array functions in PHP. There are many more array functions available in PHP, each with its own specific use case.

Date functions

Unit 1 : Introduction to PHP

Function	Description
date_add()	Adds days, months, years, hours, minutes, and seconds to a date
date_create_from_format()	Returns a new DateTime object formatted according to a specified format
date_create()	Returns a new DateTime object
date_date_set()	Sets a new date
date_default_timezone_get()	Returns the default timezone used by all date/time functions
date_default_timezone_set()	Sets the default timezone used by all date/time functions
date_diff()	Returns the difference between two dates
date_format()	Returns a date formatted according to a specified format
date_modify()	Modifies the timestamp
date_offset_get()	Returns the timezone offset
date_parse_from_format()	Returns an associative array with detailed info about a specified date, according to a specified format
date_parse()	Returns an associative array with detailed info about a specified date
date_sub()	Subtracts days, months, years, hours, minutes, and seconds from a date
date_sun_info()	Returns an array containing info about sunset/sunrise and twilight begin/end, for a specified day and location
date_sunrise()	Returns the sunrise time for a specified day and location
date_sunset()	Returns the sunset time for a specified day and location
date_time_set()	Sets the time
date_timezone_get()	Returns the time zone of the given DateTime object
date_timezone_set()	Sets the time zone for the DateTime object
date()	Formats a local date and time

getdate()	Returns date/time information of a timestamp or the current local date/time
gettimeofday()	Returns the current time
gmdate()	Formats a GMT/UTC date and time
gmmktime()	Returns the Unix timestamp for a GMT date
gmstrftime()	Formats a GMT/UTC date and time according to locale settings
idate()	Formats a local time/date as integer
localtime()	Returns the local time
microtime()	Returns the current Unix timestamp with microseconds
mktime()	Returns the Unix timestamp for a date
strftime()	Formats a local time and/or date according to locale settings
strptime()	Parses a time/date generated with strftime()
strtotime()	Parses an English textual datetime into a Unix timestamp
time()	Returns the current time as a Unix timestamp
timezone_abbreviations_list()	Returns an associative array containing dst, offset, and the timezone name
timezone_identifiers_list()	Returns an indexed array with all timezone identifiers
timezone_location_get()	Returns location information for a specified timezone
timezone_name_from_ abbr()	Returns the timezone name from abbreviation
timezone_name_get()	Returns the name of the timezone
timezone_offset_get()	Returns the timezone offset from GMT
timezone_open()	Creates new DateTimeZone object
timezone_transitions_get()	Returns all transitions for the timezone

timezone_version_get()	Returns the version of the timezonedb

String in PHP

In PHP, a string is a sequence of characters that can be defined using single or double quotes. String can store alphanumeric and special characters, and can be manipulated using built-in functions.

There are 4 ways to specify a string literal in PHP.

1. single quoted
2. double quoted
3. heredoc syntax
4. newdoc syntax (since PHP 5.3)

For example:

```php
$str1 = "Hello World!";
$str2 = 'This is a string.';
```

You can also use the concatenation operator (.) to join two or more strings together. For example:

```php
$str1 = "Hello";
$str2 = "World";
$str3 = $str1 . " " . $str2 . "!";
```

This will set $str3 to "Hello World!".

PHP also provides a wide range of built-in string functions that can be used to manipulate strings such as substr, strlen, str_replace, etc.

* PHP String Functions :

PHP provides various string functions to access and manipulate strings.

Some commonly used PHP string functions include :

1. **strlen**() - returns the length of a string.
    ```php
    $str = "Hello, world!";
    echo strlen($str);  // Output: 13
    ```

2. **strtoupper**() - converts a string to uppercase.
    ```php
    $str = "Hello, world!";
    echo strtoupper($str); // Output: HELLO, WORLD!
    ```

3. **strtolower**() - converts a string to lowercase.
    ```php
    $str = "Hello, world!";
    ```

```php
echo strtolower($str); // Output: hello, world!
```

4. **substr**() - returns a portion of a string.
```php
$str = "Hello, world!";
echo substr($str, 7);   // Output: world!
```

5. **str_replace**() - replaces all occurrences of a search string with a replacement string
```php
$str = "Hello, world!";
echo str_replace("world", "there", $str); // Output: Hello, there!
```

6. **strpos**() - finds the position of the first occurrence of a substring in a string
```php
$str = "Hello, world!";
echo strpos($str, "world");   // Output: 7
```

7. **trim**() - removes whitespace or other characters from the beginning and end of a string
```php
$str = " Hello, world! ";
echo trim($str);   // Output: Hello, world!
```

8. **explode**() - splits a string into an array
```php
$str = "Hello, world!";
$arr = explode(" ", $str);
print_r($arr);   // Output: Array ( [0] ⇒ Hello, [1] ⇒ world! )
```

9. **implode**() - joins array elements with a string
```php
$arr = array("Hello,", "world!");
$str = implode(" ", $arr);
Echo $str;   // Output: Hello, world!
```

You can find the documentation of all string function on php.net

Working with Forms

Creating form :

- A form in PHP is used to collect input from users through different types of input fields, such as text fields, radio buttons, checkboxes, and drop-down menus.
- The data collected through the form can be processed and used to perform various actions on the server side, such as inserting the data into a database, sending an email, or updating a file.
- When creating a form in PHP, it's important to use the appropriate type of input field for the type of data you're collecting. For example, you should use a text field for short, single-line text inputs, a textarea for longer, multi-line text inputs, and a checkbox or radio button for multiple choice options.

Here is an example of a basic HTML form that can be used in a PHP script:

```html
<form action="process.php" method="GET">
    Name: <input type="text" name="name" />
```

```
    <input type="submit" value="signup" />
</form>
```

The form element has two attributes:

action: This attribute specifies the URL of the page that will process the form data. In this example, the form data will be sent to a page called process.php.

method: This attribute specifies the HTTP method used to submit the form data. In this example, the form data will be sent using the GET method.

Handling form

- Form handling refers to the process of receiving, validating, and processing user input submitted through an HTML form, including input validation, sanitization, error handling, and security measures.
- There are several sub-topics within form handling in PHP, some of which include:
    - **Input validation:** This involves checking that the input received from the form meets certain criteria, such as ensuring that required fields are filled in, that input matches a specific format (e.g. email addresses), or that it does not contain potentially harmful characters or code.
    - **Input sanitization:** This involves cleaning and removing any unwanted characters or data from the input in order to prevent security vulnerabilities such as SQL injection.
    - **Error handling:** This involves displaying error messages to the user if the input validation fails.
    - **Form processing:** This involves taking action based on the validated input, such as inserting it into a database or sending an email.
    - **File upload handling**: This involves handling file uploads through forms. Server-side validation of the file type, size, and other attributes are done to prevent unwanted files to be uploaded.
    - **Security:** This involves implementing security measures such as CSRF token, CAPTCHA etc to prevent malicious users from submitting fake data.
    - **Server-side form validation:** This involves validation of forms by server side scripting language like PHP in order to ensure the data is correct, complete and secure before sending it to the server for further processing.
    - **Client-side form validation:** This involves validation of forms by client side scripting language like JavaScript in order to ensure the data is correct, complete and secure before sending it to the server for further processing.

Form Validation in PHP

- Form validation in PHP is the process of checking that user input submitted through an HTML form is complete, accurate, and secure before it is sent to the server for further processing.
- This is typically done by writing server-side code in PHP that checks the input for specific conditions, such as ensuring that required fields are filled in, that input matches a specific format (e.g. email addresses), or that it does not contain potentially harmful characters or code.
- This helps to ensure that the data entered into the form is valid and can be safely used for the intended purpose.

There is no guarantee that the information provided by the user is always correct. PHP validates the data at the server-side, which is submitted by HTML form. You need to validate a few things:

1. **Empty String :**

```php
if (empty($name)) {
    $nameErr = "Name is required";
}
```

In the above example, we check if any of the form fields are empty by using the empty() function. For example, if (empty($name)) checks if the $name variable is empty. If it is, the $nameErr variable is set to an error message "Name is required". This can be displayed next to the form field to inform the user that this field is required.

2. **Validate String :**

```php
if (!preg_match("/^[a-zA-Z ]*$/", $name)) {
    $nameErr = "Only letters and white space allowed";
}
```

We use the preg_match() function to check if the input is a valid string. For example, if (!preg_match("/^[a-zA-Z]*$/",$name)) checks if the $name variable contains only letters and spaces. The regular expression /^[a-zA-Z]*$/ specifies that the input must start with a letter or space, and can contain any number of letters or spaces after that. If the input does not match this pattern, the $nameErr variable is set to the error message "Only letters and white space allowed".

3. **Validate Numbers:**

```php
if (!is_numeric($number)) {
    $numberErr = "Invalid number";
}
```

We use the is_numeric() function to check if the input is a valid number. For example, if (!is_numeric($number)) checks if the $number variable contains a valid number. If it does not, the $numberErr variable is set to the error message "Invalid number".

4. **Validate Email:**

```php
<?php
if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
    $emailErr = "Invalid email format";
}
```

We use the filter_var() function with the FILTER_VALIDATE_EMAIL filter to check if the input is a valid email address. For example, if (!filter_var($email, FILTER_VALIDATE_EMAIL)) checks if the $email variable contains a valid email address. If it does not, the $emailErr variable is set to the error message "Invalid email format".

5. **Input length:**

```php
<?php
if (strlen($name) > 30) {
    $nameErr = "Name should be less than 30 characters";
}
```

```
        }
```

We use the **strlen**() function to check if the input is within a specific length. For example, if (**strlen**($name) > 30) checks if the $name variable is more than 30 characters. If it is, the $nameErr variable is set to the error message "Name should be less than 30 characters".

Accessing Form Data

We can create and use forms in PHP. To get form data, we need to use PHP superglobals $_GET and $_POST.

The form request may be get or post. To retrieve data from a get request, we need to use $_GET, for post request **$_POST**.
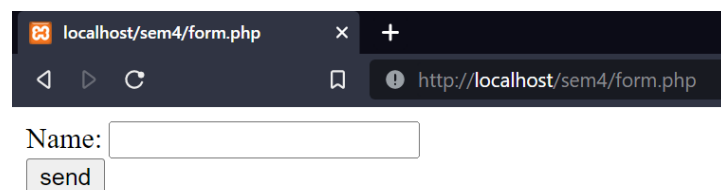
PHP Get Form :

Get request is the default form request. The data passed through get request is visible on the URL browser so it is not secured. You can send a limited amount of data through a GETrequest.
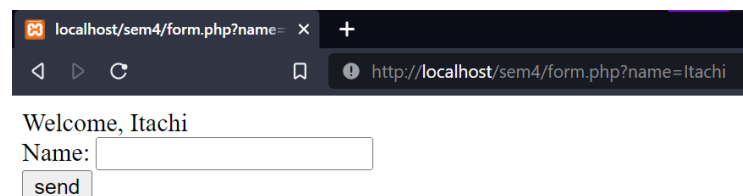
Let's see a simple example to receive data from a GET request in PHP.

```php
form.php
<?php
    $name = $_GET["name"]; //receiving name field value in $name variable
    echo "Welcome, $name";
?>
<form action="" method="get">
    Name: <input type="text" name="name" />
    <br>
    <input type="submit" value="send" />
</form>
```


After Send :

PHP Post Form :

Post request is widely used to submit forms that have a large amount of data such as file upload, image upload, login form, registration form etc.
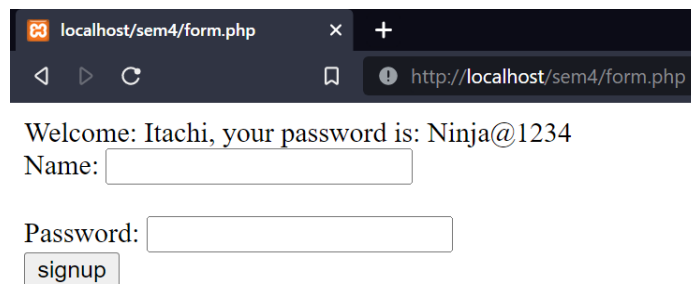
The data passed through post request is not visible on the URL browser so it is secured. You can send a large amount of data through a post request.

Let's see a simple example to receive data from a post request in PHP.

```php
file : form2.php
<?php

$name = $_POST["name"]; //receiving name field value in $name variable

$password = $_POST["password"]; //receiving password field value in $password variable

echo "Welcome: $name, your password is: $password";

?>

<form action="" method="POST">

    Name: <input type="text" name="name" />

    Password: <input type="password" name="password" />

    <input type="submit" value="signup" />

</form>
```


Use of Hidden fields to save State

- hidden field is an HTML form element that allows us to store information on a web page that the user can't see. Hidden fields are often used to save the state of a form between multiple page requests.

- Here's a simple example to demonstrate how hidden fields can be used to save the state of a form:

- Let's say we have a form that asks the user to select their favorite fruit from a dropdown list. We want to store their choice even if they submit the form multiple times.

```php
<?php session_start(); ?>

<form action="save_fruit.php" method="post">
    Select your favorite fruit:
    <select name="fruit">
            <option value="apple" <?php if ($_SESSION['fruit'] == 'apple') echo
'selected'; ?>>Apple</option>
            <option value="banana" <?php if ($_SESSION['fruit'] == 'banana') echo
'selected'; ?>>Banana</option>
```

```html
        <option value="grape" <?php if ($_SESSION['fruit'] == 'grape') echo
'selected'; ?>>Grape</option>
    </select>
    <input type="hidden" name="state" value="<?php echo $_SESSION['fruit']; ?>">
    <input type="submit" value="Submit">
</form>
```

Here is an example code for the **save_fruit.php** file that would handle the form submissions:

```php
<?php
session_start();
// Check if the form has been submitted
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    // Get the value of the selected fruit from the form
    $selected_fruit = $_POST['fruit'];

    // Store the selected fruit in a session variable
    $_SESSION['fruit'] = $selected_fruit;

    // Redirect the user back to the form page
    header("Location: test.php");
    exit;
}
```

In this code, we use a conditional statement to check if the value of the session variable "fruit" is equal to the value of the option, and if it is, we add the "selected" attribute to the option element. This will pre-select the previously selected option in the dropdown list.

Redirecting user

- User redirection in PHP is a process which allows a user to be automatically redirected from one page to another.
- It is often used when a form is submitted, so that the user is redirected back to the same page after the form is processed. It can also be used for security purposes, to prevent users from being able to access certain pages unless they have authentication.
- User redirection is achieved by using the **header()** function in PHP, and passing a parameter which contains the URL of the page to which the user should be redirected. This can be a relative URL, meaning a link to a file within the same directory, or an absolute URL, which includes the full URL of the page.

I. Importance of user redirection in web development

- User redirection is an important aspect of web development, as it helps to ensure that the user is directed to the correct page after a form has been submitted, or a specific action taken.
- In addition, it can be used to restrict access to certain pages, or to encourage users to take certain actions.
- For example, user redirection can be used to lead the user to a specific page after form submission, or it can be used to redirect the user to a different page if they attempt to access a restricted page. By taking

advantage of user redirection, developers can remove common sources of confusion, and improve the user experience when navigating a website.

II. Methods of Redirecting Users in PHP
A. Header() function
- The header() function is the most commonly used method for redirecting users in PHP. It is used to send a raw HTTP header to the browser.
- It will immediately redirect to the specified page, without providing any message or confirmation.
- To use the Header() function, you must specify the location of the page you wish to redirect to.

Syntax and usage in php.

> header(header, replace, http_response_code)

Usage :

```php
header("Location: http://www.example.com/");
    exit;
```

File Upload in php

Introduction

File upload is a common feature in web development that allows users to upload files, such as images, documents, and videos, to a website. This is an important aspect of many web applications, as it allows users to store and share files online.

- File uploads can be implemented using the PHP programming language, which is widely used for server-side scripting and dynamic web content generation.
- PHP provides a range of functions and libraries for handling file uploads, including the built-in **$_FILES** array and the **move_uploaded_file()** function.

Configuring PHP for file uploads:

To configure PHP for file uploads, you need to make a few modifications to your PHP configuration file (**php.ini**). The following are the key settings to consider:

file_uploads: This setting must be set to "On" to allow file uploads in PHP. By default, file uploads are enabled in PHP, but it's always a good idea to check this setting.

```
file_uploads = On
```

Creating an HTML Form for File Upload.

Creating an HTML form for file upload is the first step in the process of implementing file uploads in PHP. The form will allow users to select and upload files to the server. Here's an example of an HTML form for file upload:

Here's an explanation of the form elements:

1. **<form>**: This element defines the form and sets the form action to the PHP script that will handle the file upload.
2. **action**: This attribute specifies the URL of the PHP script that will handle the file upload. In this example, the form action is set to "upload.php".
3. **method**: This attribute specifies the HTTP method to use when submitting the form. The value "post" is used in this example, as file uploads require the POST method.
4. **enctype**: This attribute specifies the encoding type of the form data. The value "multipart/form-data" is used for file uploads, as it allows for the transmission of binary data.
5. <input type="file">: This element is a file input field that allows users to select a file for upload. The name attribute is used to give the input field a name, which will be used to access the uploaded file in the PHP script.
6. **<input type="submit">**: This element is a submit button that allows users to submit the form and upload the file to the server.

It's important to note that this is just a basic example and you may want to add additional form elements and validation to your file upload form based on your specific requirements.

A. Checking if a file was uploaded:

```php
// Check if the form has been submitted
if (isset($_FILES['file'])) {
    // continue with the file upload
} else {
    // handle the error, no file was uploaded
}
```

In this example, the code checks if the "file" input field in the form was set by checking if it exists in the $_FILES array. If it exists, the file was uploaded and the code can continue to process it. If it does not exist, an error occurs and the code can handle it accordingly.

Validating the uploaded file:

```php
<?php
if ($_FILES['file']['error'] === UPLOAD_ERR_OK) {
    $file_name = $_FILES['file']['name'];
    $file_size = $_FILES['file']['size'];
    $file_type = $_FILES['file']['type'];

    if ($file_size > 1000000) {
        // handle error, file size is too large
    }
```

```php
    $allowed_types = array("image/jpeg", "image/png", "image/gif");
    if (!in_array($file_type, $allowed_types)) {
        // handle error, invalid file type
    }


    // continue with file upload
} else {
    // handle error, file was not uploaded
}
?>
```

In this example, the code first checks the error code of the uploaded file. If the error code is equal to UPLOAD_ERR_OK, the file was uploaded successfully. The code then retrieves the file name, size, and type and checks if the file size is less than 1 MB and if the file type is one of the allowed types (JPEG, PNG, or GIF). If either of these conditions is not met, the code can handle the error.

Moving the uploaded file to a permanent location:

```php
<?php
$target_dir = "uploads/";
$target_file = $target_dir . basename($file_name);

if (move_uploaded_file($_FILES['file']['tmp_name'], $target_file)) {
    // file was uploaded successfully
} else {
    // handle error, file could not be uploaded
}
```

In this example, the code sets the target directory for the uploaded file to "uploads/". The target file is the full path to the file, including the target directory and the original file name. The move_uploaded_file() function is then used to move the uploaded file from its temporary location to the target file. If the function returns true, the file was uploaded successfully. If it returns false, an error occurs and the code can handle it accordingly.

Handling File Upload Errors:

A. Explanation of common file upload errors:

UPLOAD_ERR_INI_SIZE: The uploaded file exceeds the upload_max_filesize directive in the PHP configuration file.

UPLOAD_ERR_FORM_SIZE: The uploaded file exceeds the MAX_FILE_SIZE value specified in the HTML form.

UPLOAD_ERR_PARTIAL: The uploaded file was only partially uploaded.

UPLOAD_ERR_NO_FILE: No file was uploaded.

UPLOAD_ERR_NO_TMP_DIR: Missing a temporary folder.

UPLOAD_ERR_CANT_WRITE: Failed to write file to disk.

UPLOAD_ERR_EXTENSION: A PHP extension stopped the file upload.

Strategies for handling file upload errors:

- Check the error code: Before processing the uploaded file, check the error code in the $_FILES array to determine if there was an error. If there was an error, handle it appropriately.
- Validate the uploaded file: Validate the uploaded file to ensure that it is of the correct type and size and does not contain any malicious code.
- Display error messages to the user: Display error messages to the user so they can understand what went wrong and take action to correct the problem.
- Log the error: Log the error to a file or database so that you can track and debug the issue.
- Increase the upload_max_filesize and post_max_size directives in the PHP configuration file: If the error is due to the file size exceeding the maximum allowed size, increase these directives to allow larger file uploads.
- Ensure that the temporary folder exists: If the error is due to the absence of a temporary folder, create the folder and set the correct permissions to allow writing to it.
- Disable unnecessary PHP extensions: If the error is due to a PHP extension stopping the file upload, disable the extension or modify the configuration to allow the file upload.

Sending Mail on Form Submission in php

Overview of email and PHP mail() function:

- Email is an electronic communication method that allows individuals to send and receive messages, files, and other digital content between computers and other electronic devices.
- The PHP mail() function is a built-in function in PHP that enables developers to send emails from their PHP scripts. The basic syntax of the mail() function is as follows:

```
mail(to, subject, message, headers, parameters);
```

where:

to: The recipient's email address.

subject: The subject of the email.

message: The content of the email, which can include HTML and text.

headers: Optional additional headers that can be added to the email, such as CC, BCC, and Reply-To.

parameters: Optional parameters that can be passed to the underlying mail system, such as the path to a log file.

The PHP mail() function uses the Simple Mail Transfer Protocol (SMTP) to send emails, which is the standard protocol for sending email on the internet. The PHP mail() function is relatively simple to use, but has several

---

limitations, including limited support for HTML and attachments, lack of error reporting, and potential security risks if not used properly.

Configuring PHP for sending email:

To configure PHP for sending email, the following steps need to be followed:

1. Install a mail server: To send email from PHP, you need to have a mail server installed on your system. The most commonly used mail servers for PHP are Postfix, Sendmail, and Exim.

2. Configure the PHP.ini file: The PHP configuration file (php.ini) needs to be configured to specify the location of the mail server and other email-related settings.

3. Specify the sendmail_path setting: The sendmail_path setting in the php.ini file specifies the path to the sendmail or equivalent executable on your system. For example:

   ```
   sendmail_path = "/usr/sbin/sendmail -t -i"
   ```

4. Set the SMTP server: If you want to use an external SMTP server to send email, you need to set the SMTP server in the php.ini file. For example:

   ```
   SMTP = smtp.example.com
   ```

5. Set the sendmail_from setting: The sendmail_from setting specifies the email address that should be used as the "From" address in emails sent by PHP. For example:

   ```
   sendmail_from = "noreply@example.com"
   ```

6. Restart the web server: After making changes to the php.ini file, you need to restart your web server to ensure that the changes take effect.

Creating a PHP script to send email:

The following is a simple example of a PHP script that sends an email using the PHP **mail()** function:

```php
<?php
$to = "recipient@example.com";
$subject = "Test Email";
$message = "This is a test email sent from a PHP script.";
$headers = "From: sender@example.com" . "\r\n" .
"CC: cc_recipient@example.com";
mail($to, $subject, $message, $headers);
echo "Email sent successfully.";
```

- In this example, the PHP mail() function is used to send an email to recipient@example.com with a subject of "Test Email" and a message of "This is a test email sent from a PHP script." The $headers variable is used to specify the "From" and "CC" headers for the email.

- After the email is sent, the script outputs "Email sent successfully." to indicate that the email was sent successfully.

- It is important to note that when sending emails from PHP scripts, it is essential to validate the data entered by the user before sending the email to prevent security vulnerabilities and other issues.