

UNIT - 1 Dynamic Websites Basics

World Wide Web (WWW)

The World Wide Web (WWW) is a system of a large collection of interlinked documents accessible over the internet. These documents are hosted on web servers and identified by uniform Resource Locator (URL). The web documents are interlinked using hyperlinks creating a web of information. The web resources can be accessed by the users using a special software application called a web browser. Hyper Text Transfer Protocol (HTTP) is used for communication between web browser and web server.

English Scientist Sir Timothy Berners-Lee invented the world wide web in 1989.

He wrote the first web browser in 1990 while employed at CERN near Geneva, Switzerland.

Difference between the Internet and WWW

INTERNET	World Wide Web (WWW)
(i) The internet is a global public network formed by interconnecting millions of private, public, academic, business, and government networks together.	WWW is a service provided by the Internet and is a system of a large collection of interlinked documents accessible over the Internet.
(ii) Internet is a packet switching network on Transmission Control Protocol/Internet Protocol (TCP/IP) protocol stack.	WWW uses HTTP protocol to request and transfer web documents over the Internet.
(iii) Vinton Cerf is known as the 'father of the Internet'.	Tim Berners-Lee is known as the 'father of WWW'.
(iv) IETF develops and promotes Internet standards.	W3C develops and promotes WWW standards.

Web Browser :

The Web Browser is a software application for accessing information on the World Wide Web. When a user requests a web page from a particular website, the web browser retrieves the necessary content from a web server and then displays the page on the screen of the user's device.

A web browser interprets HTML, CSS and JavaScript code in a web page for displaying (rendering) it in the browser window.

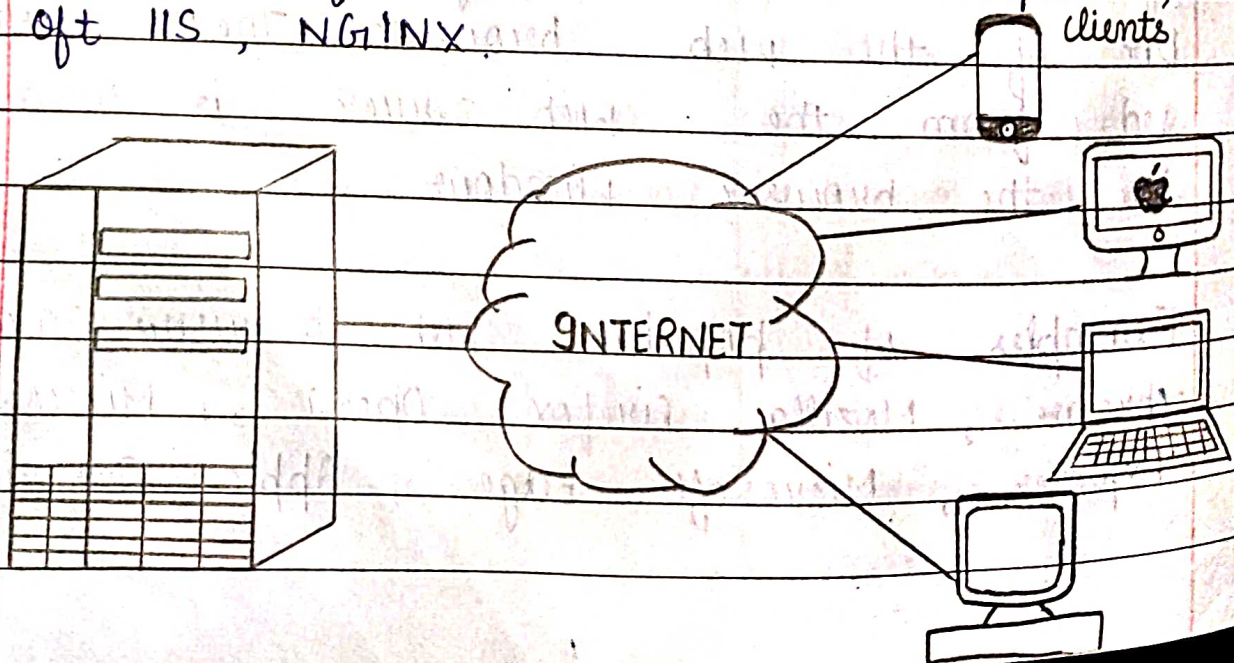
The web browser sends a HTTP request to the web server when the user enters the URL of the desired web page in the address bar of the web browser. The response received from the web server is then rendered in the browser window.

Examples of popular web browsers are: Google Chrome, Mozilla Firefox, Opera, Microsoft Internet Explorer, Microsoft Edge, Apple Safari etc.

Web Server :

WWW is based on the client - Server working model in which some nodes in the network are designated as servers that provide some services to the client nodes in the network. A web server is a software that is hosted on a computer and provides the desired web resources to the client on request. The web resources may be in the form of HTML documents, images, and other files. The client of a WWW service is generally a web browser running on the client computer.

Examples of web servers are: Apache, Microsoft IIS, NGINX.



Website :

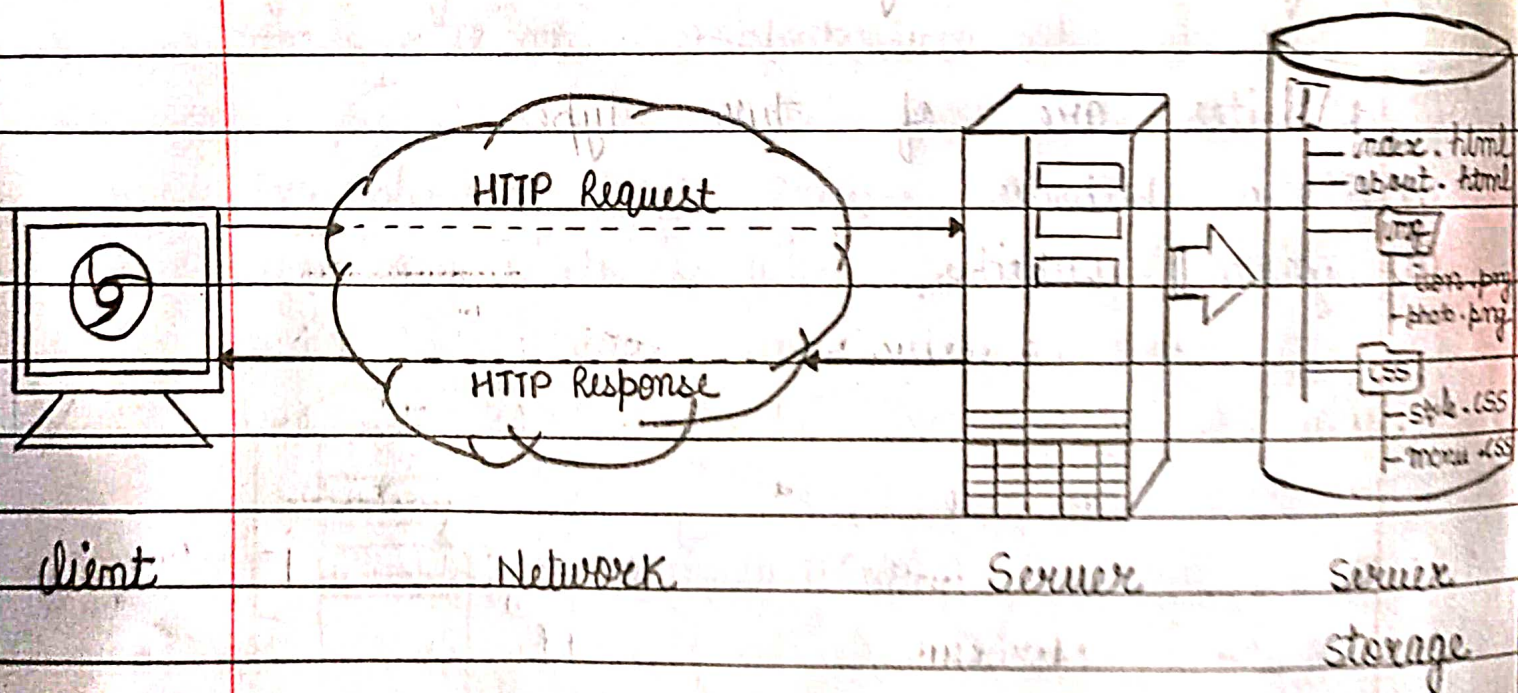
A website is a collection of interlinked web pages and other web resources with a common domain name. Websites are used by the organizations to share information with the world. When we visit a website, the first page that is displayed in the browser window is called the home page of the website. All other pages of the website are linked either directly or indirectly to the home page using hyperlinks. Examples of websites are google.com, gpham.org, himachal.gov.in etc.

Websites are of two types :

- (i) Static Websites
- (ii) Dynamic Websites

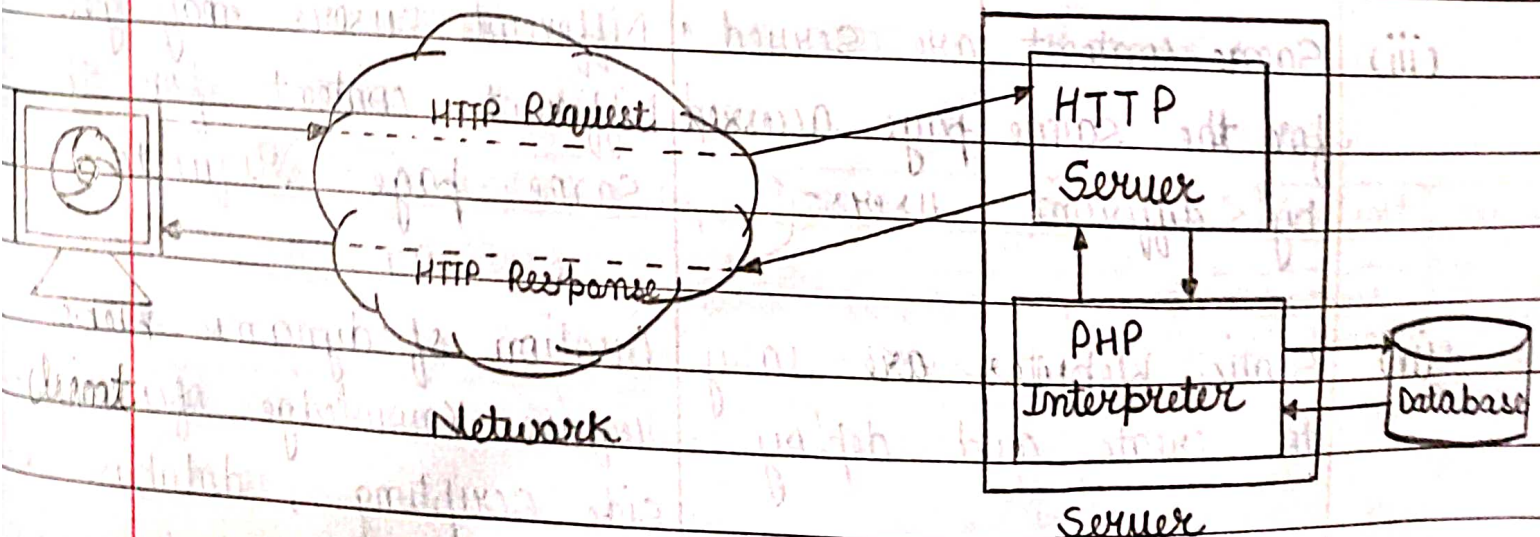
Static Websites :

A static website consists of web pages with fixed content created using HTML, CSS and Java Script. These web pages are stored as files and are directly delivered to the requesting client through HTTP response. The web server does not process the web page before serving them to the client. Therefore, same content are displayed for same page accessed by different users.



Dynamic Websites :

A dynamic website constructs the content of a web page when it is requested by the client. A dynamic web page also contains server-side scripts in addition to HTML, CSS and JavaScript code. Dynamic web pages are processed by a server side scripting language Interpreter and dynamic content are added in the page before serving to the client. Therefore, different users may get different content for the same page requests. Dynamic websites use a server-side scripting language like PHP, ASP and Python and are generally powered by a database.



Date 8 / 04 / 21
Page No. 10

Difference between Static and Dynamic Websites :

Static Website

(i) A static website consists of web pages with fixed content created using HTML, CSS and JavaScript

(ii) The web pages are directly served to the client without any processing at the server side.

(iii) Same content are served for the same page accessed by different users.

(iv) Static websites are easy to create and deploy

Dynamic Website

A dynamic website consists of web pages which includes server-side scripting in addition to HTML, CSS and JavaScript code.

The web pages are processed by a server side scripting language interpreter and dynamic content are added to the page before serving.

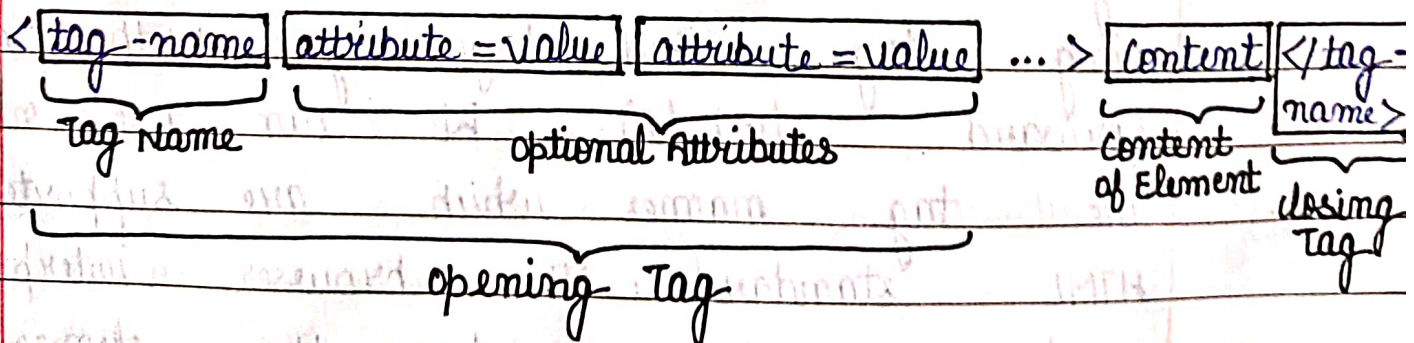
Different users may get different content for the same page requests.

Creation of dynamic websites require knowledge of server-side scripting, database design, security provisions in addition to HTML, CSS.

Web Page:

A web page is a document created for viewing in a web browser. A web page is written in Hypertext Markup Language (HTML) and may also make use of cascading Style Sheets (CSS) and Java Script. A web page consists of many HTML elements organized in a tree-like structure. HTML elements are created in a web page using HTML tags. An HTML tag consists of a tag name enclosed inside angle brackets. Tags are usually paired so that we have a tag at the beginning and the end of an HTML element. The closing tag has the same name as its opening tag, but beginning with a forward slash.

The structure of an HTML element is as :



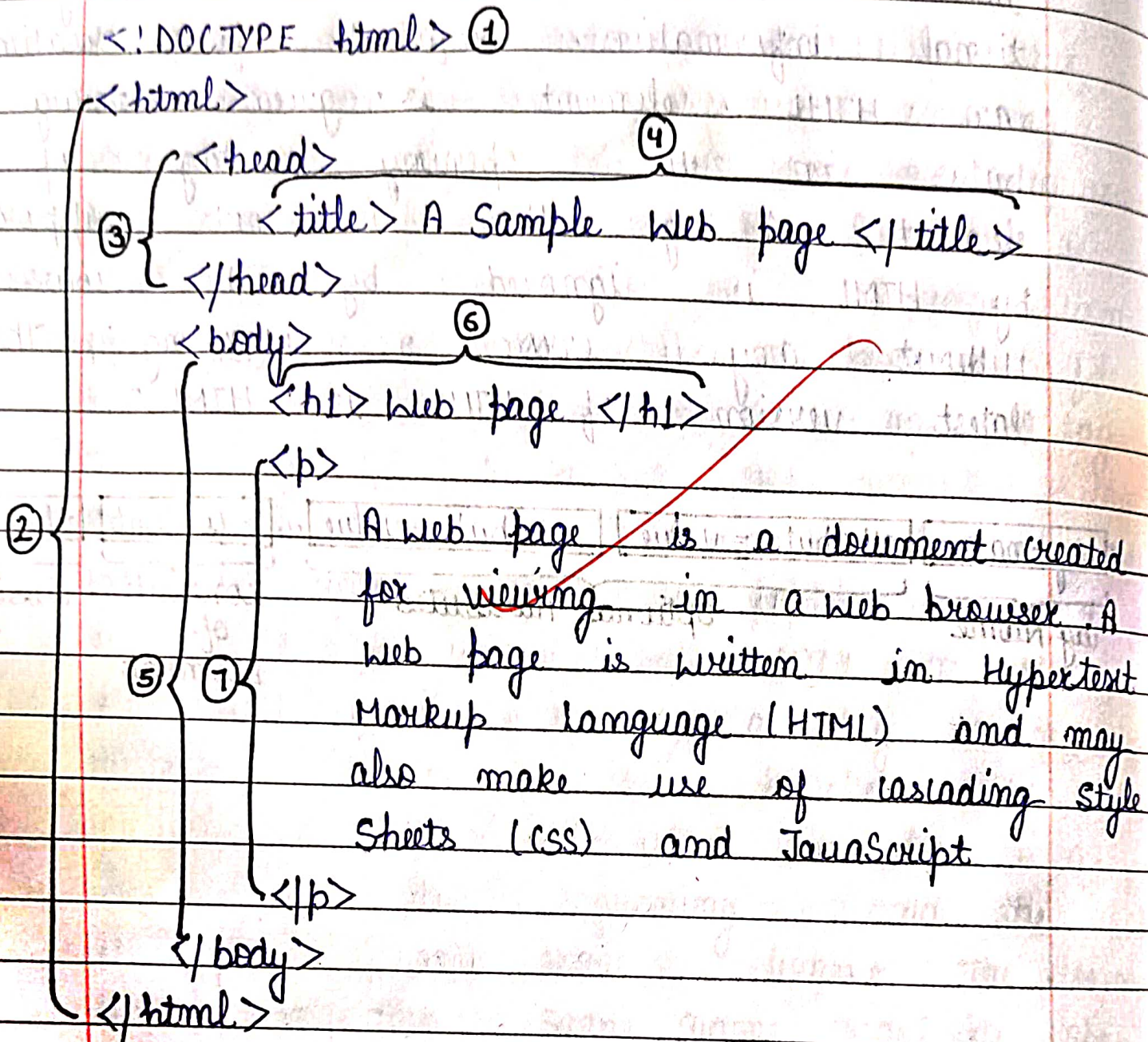
Hypertext Markup Language (HTML)

Hypertext Markup Language (HTML) is the standard markup language for creating documents to be displayed in web browser. HTML was developed by Tim - Berners - Lee. A markup language is a language used to define elements in a document file using special codes called tags. A tag is an instruction to the markup language processor to interpret the element correctly. Some of the popular markup languages are XML and HTML. In HTML, a tag consists of a tag name enclosed inside angle brackets. Tags are usually paired so that we have a tag at the beginning and the end of an HTML element. The closing tag has the same name as its opening tag, but beginning with a forward slash (/). We can use only the tag names which are supported by HTML standard. The browser interprets and correctly renders the elements.

of a web page depending upon the tags used to create the elements. Any additional information required for creating an HTML element is given using attributes in the opening `<tag-name>` tag. A tag that is not supported by HTML is ignored by the browser without any error or warning. The latest version of HTML is HTML 5.

`<tag-name` `attribute = value` `attribute = value` ... `> content` `</tag-name`
 tag name optional attributes content of element closing tag

Structure of an HTML Document:



① Tell the browser that this is an HTML5 document.

② Every HTML document has one <HTML> root.

element that contains all other elements.

③ The `<head>` element is always the child of `<html>` element and contains metadata of document.

④ `<title>` element is inside the `<head>` element and its text content is shown in the browser's title bar.

⑤ `<body>` element contains all other elements to be displayed in the browser window.

⑥ `<h1>` is a level-1 heading and we can add as many of such elements as required in the page.

⑦ `<p>` is used to create a paragraph element and we can add many such elements in the page.

Subdomain (optional): A domain name can optionally have a subdomain (e.g. www)

Port (optional): TCP port number on which the server is listening.

Path to the Resource: Path to the requested resource.

Query string (optional): A query passed to the page as a collection of parameters.

Fragment (optional): A section within a web page.

Hypertext Transfer Protocol (HTTP):

The Hypertext Transfer Protocol (HTTP) is an application - layer protocol designed for communication between web browser and web server. HTTP has following characteristics.

Client - Server:

HTTP is a client - server protocol based on request - response mechanism. The client (browser) sends a request to the web server to fetch some web resource (web page, image, file etc). The server locates the requested web resource from its local storage and sends back the response to the client.

Simple:

HTTP messages are human readable and therefore are easy to understand.

Stateless:

HTTP is a stateless protocol which means that each request is treated as a new

request without any linkage to previous requests.

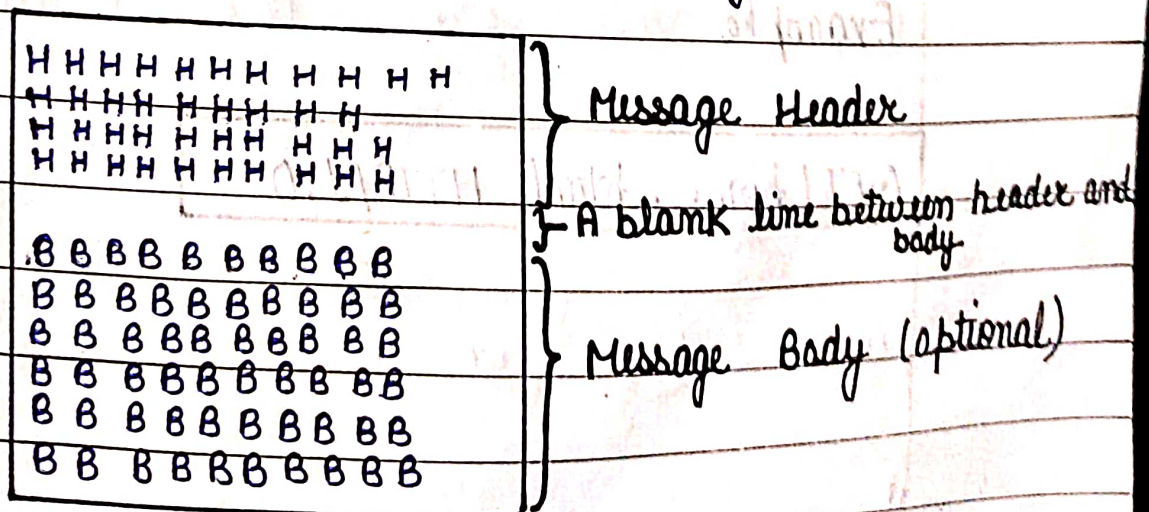
Extensible:

HTTP protocol can be easily extended by creating new headers using a simple agreement between a client and a server.

HTTP Messages:

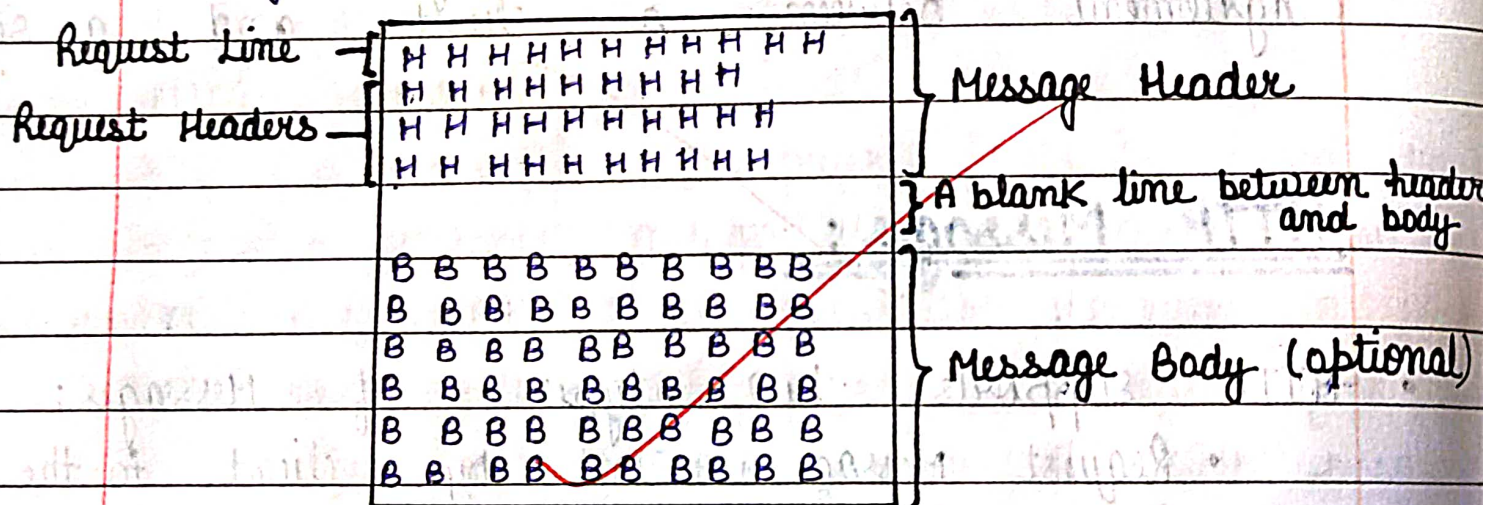
- HTTP supports two types of Messages:
 - Request Message: Sent by client to the server
 - Response Message: Sent by the server to the client

- An HTTP message consists of a message header and an optional message body as shown below.



HTTP Request Message :

- An HTTP request Message is sent by the client to the server to fetch some resource. The structure of an HTTP request message is as shown below :



- The request line has the format :

request method request uri http version

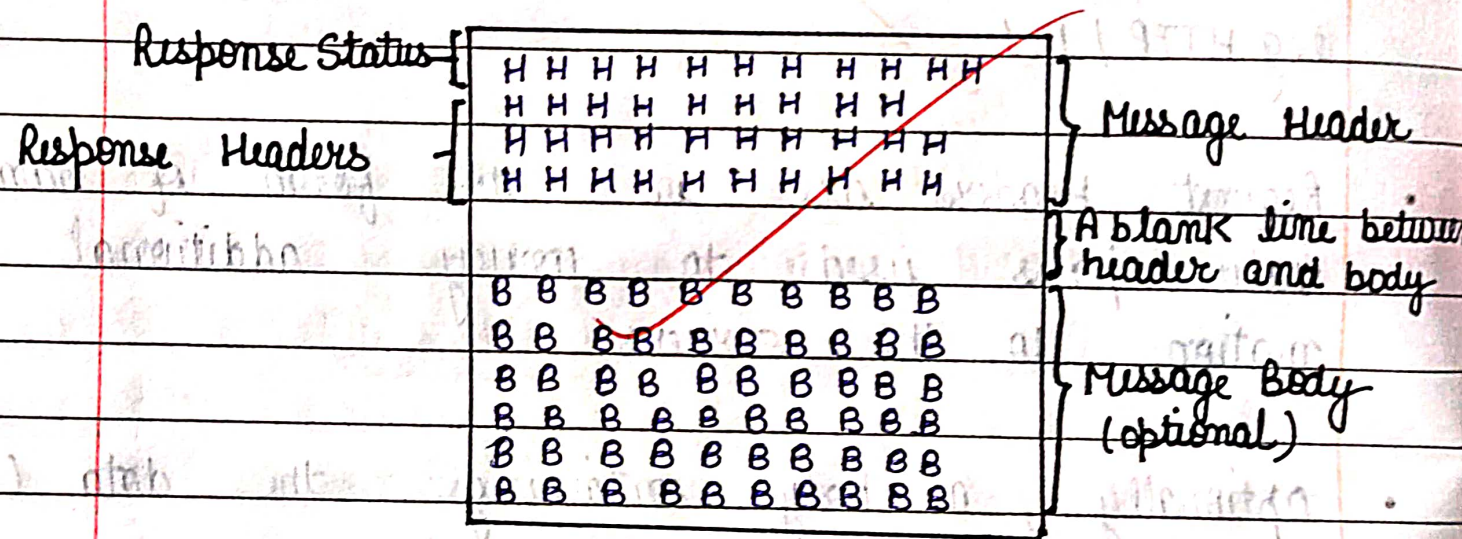
Example

GET /home.html HTTP/1.0

- The request-method can be GET, POST, PUT, HEAD, and OPTIONS. The client can use one of these methods to send a request to the server.
- The request-uri is the identifier of the requested resource.
- The http-version is the version of HTTP used e.g HTTP / 1.1
- Request Headers are in the form of name: value pairs used to convey additional information to the servers.
- optionally, a body containing the data (POST) being sent.

HTTP Response Message:

- An HTTP response message is sent by the server to the client in response to the request by the client to fetch some resource. The structure of an HTTP response message is as shown below:



- The response status has the format:

http-version status-code reason-phrase

Example

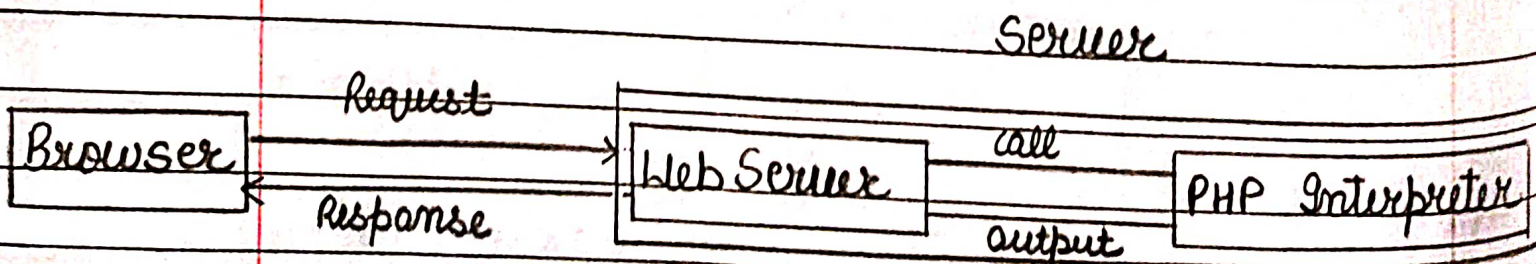
HTTP/1.1 200 OK

- The http version is the version of HTTP used
e.g. HTTP / 1.1
- Status code is a 3-digit number generated by the server to reflect the outcome of the request.
- Reason phrase gives a short explanation to the status code.
- Response Headers are in the form of name : value pairs used to provide additional information to the browser.
- optionally, a body containing the fetched resource.

UNIT - 2 Introduction to PHP

PHP:

PHP is a popular general-purpose scripting language that is especially suited for creating dynamic websites. PHP was developed by Rasmus Lerdorf in 1994. The full form of PHP is PHP: Hypertext Preprocessor. PHP code can be embedded inside HTML document. A PHP enabled Web-Server passes the requested PHP file to the PHP interpreter. PHP interpreter executes the PHP code embedded inside the web page and replaces it with its output. Web Server then serves the processed file to the client (web browser).



Features of PHP:

PHP is especially suited for server-side scripting language for creating dynamic websites. Some features of PHP are:

- Free and Open Source:

PHP is an open-source scripting language and is also free of any cost.

- Platform Independent:

Code written in PHP is platform independent and can run on any OS like Linux and Windows.

- Simple:

The syntax of PHP is like C language. Hence, PHP can be learned easily if you already know C or Java language.

- Database Support:

Dynamic websites require database support. PHP has inbuilt support for connecting to databases.

- Interpreted :

PHP is an interpreted language making it efficient, portable and secure.

- Fast :

PHP is lightweight and is very fast and efficient.

- Object Oriented :

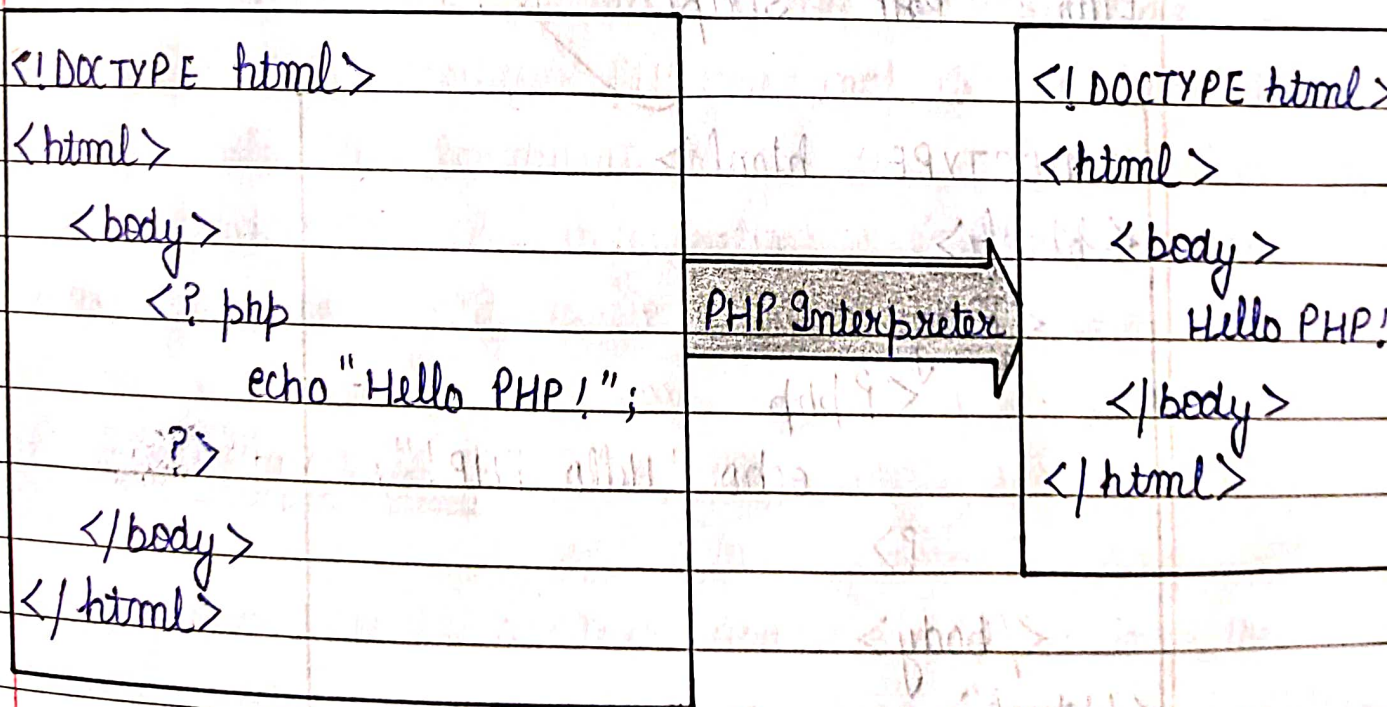
PHP supports object oriented programming (OOP) paradigm.

- Well Supported :

PHP has a rich set of built in functions, very big collection of third party libraries and has a large community of developers.

Embedding PHP in HTML:

PHP code can be embedded in a web page using `<? php ... ?>` tag. All the PHP code is written inside the tag beginning with `<? php` and ending with `?>`. PHP interpreter executes everything inside `<? php` and `?>` and substitutes the output produced by the code in its place. For example:



PHP Script:

A PHP Script is a sequence of PHP statements to carry out some task. A PHP script can be embedded anywhere inside an HTML document using `<?php ...?>` tag. A single HTML document can have several `<?php ...?>` tags. A web page that contains PHP script has a file extension of .php. Example of an HTML document with PHP script is:

```
<!DOCTYPE html>
<html>
  <body>
    <?php
      echo "Hello PHP!";
    ?>
  </body>
</html>
```


PHP Syntax :

The syntax of PHP is very similar to C and Java programming languages. A PHP script is a sequence of PHP statements that perform a specific task. A PHP statement can be an assignment, a function call, a control structure or any other statement. PHP is a case-sensitive language i.e. it differentiates between upper-case and lower-case letters. PHP statements are terminated with a semicolon. However, you do not need to have a semicolon terminating the last line of a PHP block. PHP interpreter ignores everything outside the `<?php ... ?>` tag.

PHP Comments :

A comment is a text note added in the source code of a program / script to explain the purpose or logic of the code. Comments are ignored by the compiler / interpreter while translating the source code. There are two types of comments in PHP :

• Single line comments :

A single line comment begins with // or # and continues until the end of line.

• Multiline comments :

A multiline comment begins with /* and ends with */ and may continue for several lines.

Examples of single line and multiline comments

```
<?php
// This is a single line comment
# This is also a single line comment
$a = 10; # It can be inserted in the end
of a statement also.
?>
```

```
<?php
/* This is a multiline comment
and it can continue for many lines */
echo "This line is executed by PHP";
?>
```


PHP Variable :

A variable is a named memory location used to store some data value in a program. The value of a variable can be changed during execution of the program. When a new value is assigned to a variable then its old value is overwritten. PHP is the loosely-typed programming language which means that it does not require a variable to be declared before using it. PHP automatically declares a variable with proper data type when a value is assigned to it.

PHP Variable Name :

A variable in PHP begins with a dollar sign (\$) followed by the name of the variable. Rules to form variable names in PHP are :

- A variable name starts with an alphabetic letter (a-z, A-Z) or underscore (_).

- The first character is followed by any number of letters (a-z, A-Z), digits (0-9), or underscore (_).

- The predefined variables like \$GLOBALS, \$argc, \$this etc. can not be used as user-defined variables in PHP.

- Variable names are case-sensitive in PHP.

Examples of variable names in PHP:

Some examples of variable names in PHP are :

\$123	X	\$1_a	X
\$abc	✓	\$a_1	✓
\$_12	✓	\$a b :	X
\$this	X	\$a-b	X

PHP Variable Scope :

The Scope of a variable is the portion of script within which it is visible. Variables in PHP may have following two scopes:

- Global Scope :

A variable defined outside any function has a global scope. A global variable can be accessed from anywhere within the script. To access global variables inside a function, we must first declare them as global within that function.

- Local Scope :

A variable defined inside a function has a local scope and the variable is visible only inside that function.

Examples of global and local variables:

```
$name = "Alka"; // global variable  
function my_info () {  
    global $name;  
    $branch = "IT"; // local variable
```



```
echo "My name is ", $name;  
echo "I am a student of "; $branch  
}  
my_info();  
echo "My name is ", $name;  
echo "I am a student of ", $branch
```

Output :

```
My name is Alka  
I am a student of IT  
My name is Alka  
I am a student of
```


PHP Constant :

A constant is a named memory location used to store some data value in a program. The value of a constant can not be changed during execution of the program. All constants defined in a PHP script are global. A constant can be defined in a PHP script using `define()` function. The rules for naming a constant in PHP are same as the rules for naming variables only difference is that a constant name does not begin with a \$ symbol.

```
define("PI", 3.143);
```

```
$radius = 10;
```

```
$area = PI * $radius * $radius;
```


echo and print Statements :

echo : echo Statement is used to output the values of one or more expressions separated by commas. echo does not return any value after execution.

print : print Statement is used to output the value of an expression. print returns 1 after execution.

Difference between print and echo is that print accepts a single argument and echo may accept any number of arguments.

- print returns 1 after execution and echo does not return any value.

Examples of echo and print :

```
$ name = "Alka";  
$ branch = "IT";  
echo "My name is", $name;  
print "I am a student of";  
print $branch;
```

```
My name is Alka  
I am a student of  
IT
```


Date 26/4/21
Page No. 36

echo and print Statements :

echo : echo statement is used to output the values of one or more expressions separated by commas. echo does not return any value after execution.

print : print statement is used to output the value of an expression. print returns 1 after execution.

Difference between print and echo is that print accepts a single argument and echo may accept any number of arguments.

- print returns 1 after execution and echo does not return any value.

Examples of echo and print :

```
$ name = "Alka";  
$ branch = "IT";  
echo "My name is", $name;  
print "I am a student of";  
print $branch;
```

```
My name is Alka  
I am a student of  
IT
```


PHP Data Types:

Each data item (variable, constant, expression) in a PHP script has an associated data type. The data type of a data item determines the set of operations that can be performed on the data item.

We can use PHP `gettype()` function to get information about the data type of a data item. The data types in PHP can be classified into three categories:

- **Scalar Data Types:** Integer, float, string and boolean.
- **Compound Data Types:** arrays and objects.
- **Special Data Types:** NULL.

- **String:** A string is a sequence of characters used to represent text data. Examples of string data are names and addresses of some person. In PHP, a string literal is enclosed inside

double or single quotations. We can apply various string operators like concatenation (.) etc to string data. Examples of string data type in PHP are:

```
$ first_name = "Alka";  
$ last_name = 'Sharma';  
$ full_name = $ first_name . ' ' . $ last_name;
```

- **Integer**: An integer is either a positive or negative number without any fractional part. An integer literal can either be specified in decimal, hexadecimal, octal or binary notation. We can apply arithmetic operators like +, -, *, / etc. to integer data. Examples of integers data types are:

```
$ num1 = 10; // An integer in decimal  
$ num2 = 0xa; // An integer in hexadecimal  
$ num3 = 012; // An integer in octal  
$ num4 = 0b1010; // An integer in binary
```


- **float :** A floating point number can be either positive or negative and consists of whole as well as fractional part. A floating point literal can be represented using fixed-point or scientific notations. We can apply arithmetic operators like $+$, $-$, $*$, $/$ etc. to floating point data. Examples of float data type are:

```
$ num1 = 10.0 // A floating point number in fixed-point notation.
```

```
$ num2 = 2e-5; // A floating point number in scientific notation.
```

```
$ num3 = -5.123; // Negative floating point Number.
```

- **Boolean :** A boolean data type allows only two values i.e. true and false. Various logical operators like AND, OR and NOT can be applied to boolean values. Examples of boolean data type are:

```
$ x = true; // Assigning boolean value to a variable  
$ y = $a <= 5; // Assigning the value of comparison operation.
```


`$z = $x && $y;` // Assign the value of logical operation.

PHP Data Types (Array) :

- **Array :** An array is a special variable in which we can store many data elements sequentially. An element of an array can be accessed by specifying the index of that element in the square brackets after the name of the array. Index of the first element of an array in PHP is 0 by default. There are two ways to define arrays in PHP:

using array () syntax :

An array can be created by: passing comma-separated values to the array () language construct.

Using short array syntax :

An array can be created by enclosing comma-separated values in square brackets.

- PHP also supports associative arrays in which the elements are indexed by user-defined keys. The elements in associative array are specified using Key => value pairs. Examples of arrays in PHP are:

```
$a = array (3, 2, 7, 8); // using array () construct  
$b = [2, 6, 4, 5]; // using short array syntax  
$c = array ("alika" => 30, "amil" => 20);  
// Associative array  
$a [1] = 20; // Access array element  
echo $c ["amil"];
```

PHP Data Types (Object):

- PHP also supports object-oriented programming style. We can define classes and create the objects of these classes in PHP. An object data type is associated with the object variables created from classes. Examples of objects in PHP are:


```

class MyClass { // Define a class
    private $name; // Add class members
    function setName ($n) {
        $this->name = $n;
    }
    function getName () {
        return $this->name;
    }
}

$x = new MyClass (); // Create object variable
    
```

PHP Data Types (NULL):

NULL is a special data type which can have only one value i.e. null. A variable to which no value has been assigned has the NULL data type. Example of NULL data type is:

```

echo gettype ($x); // will print NULL if variable
                  $x is not defined
    
```


Single and Double Quoted Strings :

A String literal is a sequence of zero or more characters enclosed within single or double quotation marks. There are two ways to specify string literals in PHP:

- Using Single quotes : If a string literal is enclosed inside single quotes, PHP does not parse the content of the string, i.e. any variable name or escape sequence appearing inside the string are not expanded.
- Using double quotes : If a string literal is enclosed inside double quotes, PHP parses the content of the string and substitutes the value of any variable names or escape sequence appearing inside the string.

Example :

```
$name = 'Ankur';
```

```
$age = 22;
```

```
$greeting = 'Hello';
```

```
$message = 'I am a student';
```


Difference b/w Single and Double Quoted Strings:

Strings:

Single Quoted Strings

- (i) PHP does not parse the content of the string.
- (ii) No substitution is done for any variable name appearing inside the string.
- (iii) No substitution is done for any escape sequence appearing inside the string except `'` and `"`.

Double Quoted Strings

PHP parses the content of the string.

The value of the variable is substitution for any variable name appearing inside the string.

The escape sequences appearing inside the string are replaced by the equivalent character.

Examples:

```
$name = 'Animash';
```

```
$age = 25;
```

```
$str1 = 'Hello, I am $name and my age is $age';  
echo $str1;
```



```

$ str 2 = "Hello, I am $ name and my age is $ age";
echo $ str 2 ;

```

Output

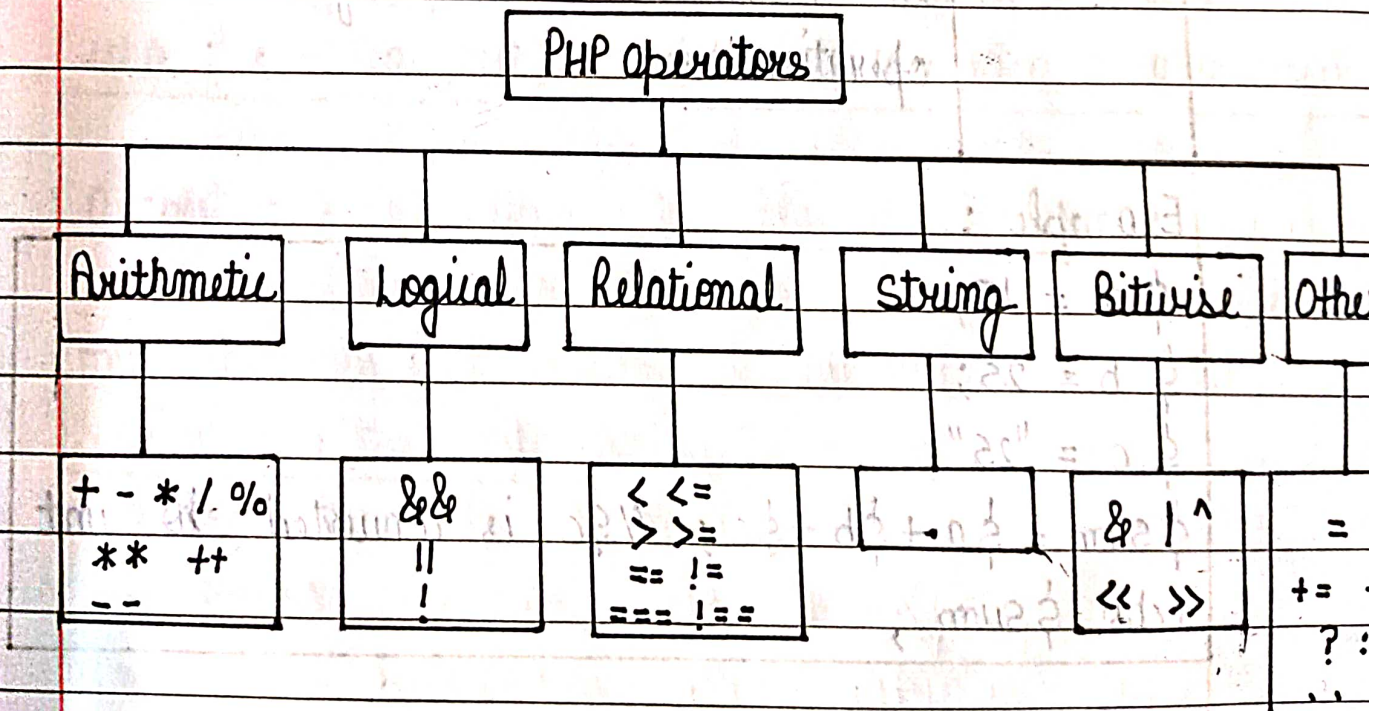
```

Hello, I am $ name and my age is $ age
Hello, I am Animash and my age is 25

```

PHP operators :

An operator is a symbol that represents some specific action to be performed on its arguments (operands). There are following types of operators in PHP:



PHP Arithmetic Operators :

- Arithmetic operators are used to perform standard operations on numbers (integers and floating point). Various arithmetic operators supported by PHP are:

Operator	Description
Addition (+)	Addition (+) operator returns the value obtained by adding the values of its two operands. Non-numeric operands are automatically converted to numeric values before the addition operation.

Example :

```
$a = 15;
$b = 25;
$c = "25";
$sum = $a + $b + $c; // $c is converted to int
echo $sum;
```

Output :

65

PHP Arithmetic Operators :

- Arithmetic operators are used to perform standard operations on numbers (integers and floating point). Various arithmetic operators supported by PHP are:

Operator	Description
Addition (+)	Addition (+) operator returns the value obtained by adding the values of its two operands. Non-numeric operands are automatically converted to numeric values before the addition operation.

Example :

```
$a = 15;
$b = 25;
$c = "25";
$sum = $a + $b + $c; // $c is converted to int
echo $sum;
```

Output :

65

Operator	Description
Subtraction (-)	Subtraction (-) operator returns the value obtained by subtracting the value of its second operand from the value of the first operand. Non-numeric operands are automatically converted to numeric values before the subtraction operation.

Example :

```

$a = 15;
$b = 25;
$c = "30";
echo $a - $b, "\n"; // output will be -10
echo $c - $a; // $c will be converted to number.

```

Output :

-10

15

Operator	Description
Multiplication (*)	Multiplication (*) operator returns the product obtained by multiplying the values of its two operands. Non-numeric operands are automatically converted to numeric values before the multiplication operation.

Example:

\$a = 15;	
\$b = 25;	
\$c = "30";	
echo \$a * \$b, "\n";	Output: 375
echo \$c * \$a;	450
// \$c will be converted to number	

Operator	Description
Division (/)	Division (/) operator returns the value obtained by dividing the value of its first operand by the value of its second operand. Non-numeric operands are automatically converted to numeric values before the division operation. The value of second operand must not be zero.

Example

Output

\$a = 15 ;

0.6

\$b = 25 ;

2

\$c = "30" ;

INF

echo \$a / \$b ; "\n" ; // Output will be 0.6

echo \$c / \$a ; "\n" ; // \$c will be converted to number

echo \$c / 0 , "\n" ; // Output will be INF

Operator

Description

Modulus
(%)

Modulus (%) operator returns the remainder obtained by dividing the value of its first operand by the value of its second operand. Non-numeric operands are automatically converted to numeric values before the modulo operation. The value of the second operand must not be zero. Both operands are converted to integers before applying the modulo operation.

Example :

Output:

\$a = 15 ;

15

\$b = 25 ;

5

\$c = 5.8 ;

0

echo \$a % \$b , "\n" ; // Output will be 15

increments the value of its operand by 1.

There are two forms of increment operator: pre-increment and post-increment.

In pre-increment form, the ++ operator appears before the variable name. The value returned by the pre-increment is the new value of the variable after applying the increment operation.

In post-increment form, the ++ operator appears after the variable name. The value returned by the post-increment is the old value of the variable before applying the increment operation.

Operator	Description
Decrement (--)	Decrement (--) operator is a unary operator i.e. it requires only one operand that must be a variable. The -- operator decreases the value of its operand by 1. There are two forms of decrement operator: pre-decrement and post-decrement.

In pre-decrement form, the -- operator appears before the variable name. The value returned by the pre-decrement is the new value of the variable after applying the decrement operation.

In post-decrement form, the -- operator appears after the variable name. The value returned by the post-decrement is the old value of the variable ~~after~~ before applying the decrement operation.

Examples of increment and decrement operators:

```
$ a = 2;
echo $ a ++,; // post-increment, output will be 2
echo $ a; // output will be 3
echo ++ $ a; // pre-increment, output will be 4
echo $ a --; // post-decrement, output will be 4
echo $ a; // output will be 3
echo -- $ a; // pre-decrement, output will be 2
```

output

2

3

4

4

3

2

PHP Relational (Comparison) Operators :

• Relational operators are used to compare the values of two operands. A relational operator returns true if the comparison holds, otherwise false is returned. Various relational operators supported by PHP are:

operator	Description
Less Than (<)	Less than (<) operator returns true if the value of its first operand is less than the value of its second operand, otherwise it returns false.
Less Than or Equal (<=)	Less than or equal to (<=) operator returns true, if the value of its first operand is less than or equal to the value of its second operand, otherwise it returns false.
Greater Than (>)	Greater than (>) operator returns true if the value of its first operand is greater than the value of its second operand, otherwise it returns false.

Greater Than
or Equal
($\geq$)

Greater Than or Equal to ($\geq$) operator returns the value of its first operand is either greater than or equal to the value of its second operand, otherwise it returns false.

Equal
==

Equal (==) operator returns true if the value of its first operand is equal to the value of its second operand, otherwise it returns false. The == operator ignores the datatypes of the operands while comparing their values.

Identical
===

Identical (===) operator returns true if the values and datatypes of its both operands are same, otherwise it returns false.

Not Equal
<> or !=

Not Equal (!=) operator returns true if the value of its first operand is not equal to the value of its second operand, otherwise it returns false. The != operator ignores the datatypes of the operands while comparing their values.

Not identical !=	Not identical (!=) operator returns true if the values or the data types of its both operands are different, otherwise it returns false.
---------------------	--

Example of Relational Operators

```

$a = 10;
$b = 20;
$c = "10";
var_dump ($a == $b); // output will be bool (false)
var_dump ($a <= $b); // output will be bool (true)
var_dump ($a > $b); // output will be bool (false)
var_dump ($a == $c); // output will be bool (true)
var_dump ($a === $c); // output will be bool (false)
var_dump ($a !== $c); // output will be bool (true)

```


PHP Logical Operators :

- Logical operators are used to manipulate boolean values. Various logical operators supported by PHP are :

Operator	Description
Logical AND (&&)	Logical AND (&&) operator returns true if the values of both of its operands is true, otherwise it returns false.
Logical OR ()	Logical OR () operator returns false if the values of both of its operands is false, otherwise it returns true.
Logical NOT (!)	Logical NOT (!) operator is a unary operator and it returns true if the value of its operand is false, otherwise it returns true.

PHP Assignment Operator :

- Assignment operators are used to assign value to a variable. The first operand of the assignment operator must be a variable. PHP provides two types of assignments: simple assignment and compound assignment. Simple assignment assigns a given value to a variable. Compound assignment combines a binary operator with the assignment. In compound assignment, the binary operator is applied to the operands and the result is then assigned to the variable. Various assignment operators supported by PHP are:

Operator	Description
Simple Assignment (=)	= operator assigns the value of the second operand to the variable specified as first operand.
Arithmetic Assignment Operators	Arithmetic assignment operators combine binary arithmetic operators with assignment operator. The arithmetic opera-

(+=, -=, *=, /=, %=, **=) This operation is carried out with the operands and the result is assigned to the variable on the left side of the operator. For example: $\$a += 5$ is equivalent to $\$a = \$a + 5$

String Concatenation Assignment (.=)

String concatenation assignment (.=) operators combine string concatenation operator (.) with assignment operator. String concatenation is carried out with the operands and the result is assigned to the variable on the left side of the operator. For example: $\$a .= "xyz"$ is equivalent to $\$a = \$a . "xyz"$

PHP String Concatenation operator :

• String concatenation operator is denoted by dot (.) symbol. String concatenation operator returns a string obtained after appending the string at second operand to the string at first operand. The operands are converted to strings before applying the concatenation operation. For example:

Example of String Concatenation :

```
$a = 10;
```

```
$b = "10";
```

```
echo $a . $b; // output will be 1010
```

```
echo $a . "cm"; // Output will be 10cm
```

PHP Bitwise operators :

- Bitwise operators are used to manipulate bits within integers. The bitwise operators require integers operands. If the operands are not integers, PHP automatically converts the operands to their equivalent integer values before applying the bitwise operators. Various bitwise operators supported by PHP are:

Operator	Description
Bitwise AND (&)	Returns an integer obtained by setting a bit to 1 if the corresponding bits in both the operands are 1, otherwise setting the bit to 0.

Bitwise OR ()	Returns an integer obtained by setting a bit to 1 if the corresponding bits in both the operands are 1, otherwise setting the bit to 0.
Bitwise XOR (^)	Returns an integer obtained by setting a bit to 1 if the corresponding bits in both the operands are different, otherwise setting the bit to 0.
Bitwise NOT (~)	Returns an integer obtained by complementing each bit of the operands.
Bitwise Left Shift (<<)	Returns an integer obtained by shifting the bits of its operand towards left by a specified number of positions.
Bitwise Right Shift (>>)	Returns an integer obtained by shifting the bits of its operands towards right by a specified number of positions.

Bitwise OR ()	Returns an integer obtained by setting a bit to 1 if the corresponding bits in both the operands are 1, otherwise setting the bit to 0.
Bitwise XOR (^)	Returns an integer obtained by setting a bit to 1 if the corresponding bits in both the operands are different, otherwise setting the bit to 0.
Bitwise NOT (~)	Returns an integer obtained by complementing each bit of the operands.
Bitwise Left Shift (<<)	Returns an integer obtained by shifting the bits of its operand towards left by a specified number of positions.
Bitwise Right Shift (>>)	Returns an integer obtained by shifting the bits of its operands towards right by a specified number of positions.

PHP Control Structures:

- A PHP Script consists of a sequence of PHP statements. The statements in a script are executed sequentially. Control Structures (control statements) are used to change the normal flow of execution of statements in a script. PHP supports several control statements which can be categorized into three types as shown below.

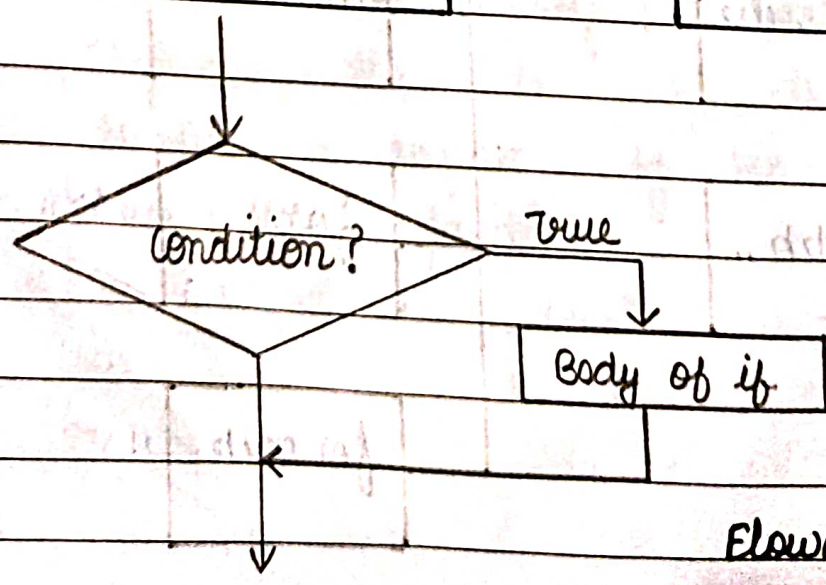
Selection/ Branching	Iterations/ Loops	Jump
if	while	break
if...else	do...while	continue
switch... case	for	return
	for each	

Selection Control Structures :

• Selection control structures : Selection control statements are used to select a statement (or statement block) for execution depending on the value of a given condition. Various selection statements in PHP are :

• Simple if statement : if statement executes the statements in its body only if a given condition is true otherwise the statement is not executed. Syntax of if statement is :

<pre>if (<condition>){ Body of if statement }</pre>	OR	<pre>if (<condition>): Body of if statement endif;</pre>
---	----	--



Flowchart of if

Example of if :

```
if ($marks >= 40) {
    echo "You have passed the test";
}
```

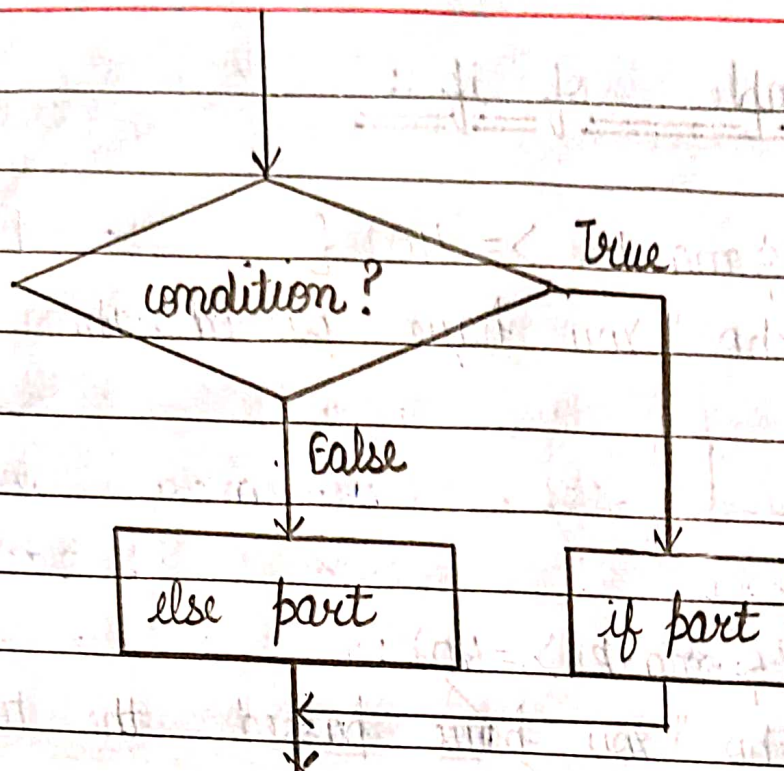
```
if ($marks >= 40) :
    echo "You have passed the test";
endif;
```

- if... else Statement : If... else control structures has two alternate blocks of statements called if part and else part. The if part is executed if the given condition is true otherwise the else part is executed. Syntax of if... else statement is

```
if (<condition>) {
    Statements in if part
} else {
    Statements in else part
}
```

OR

```
if (<condition>):
    Statements in if part
else:
    statements in else part
endif;
```

Flowchart of if...else

Example of if...else :

```

if ( $ marks >= 40 ) {
    echo "You have passed the test";
} else {
    echo "You have failed in the test";
}
  
```

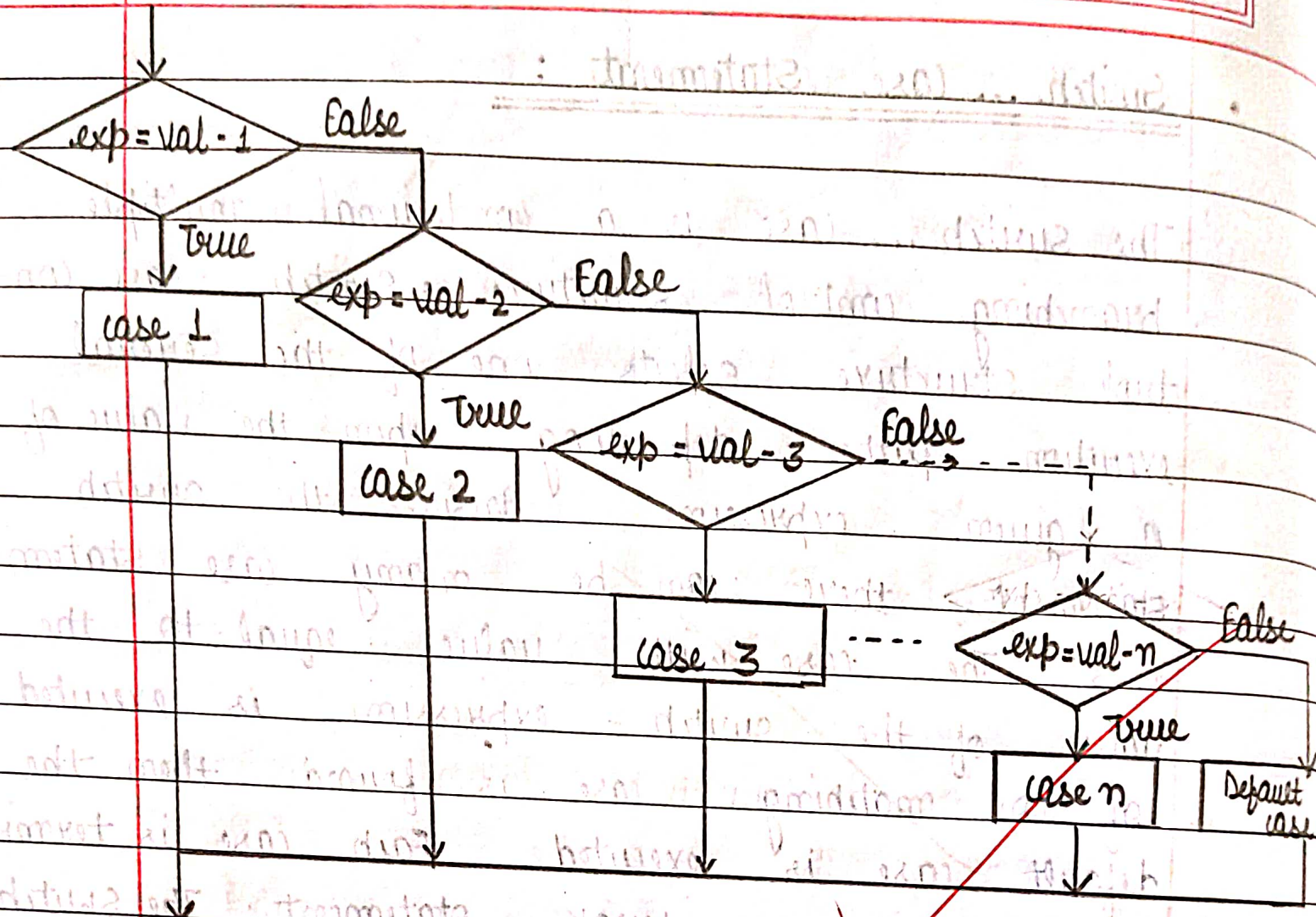
```

if ( $ marks >= 40 ) :
    echo "You have passed the test";
else :
    echo "You have failed in the test";
endif;
  
```


Switch ... case Statement :

The switch ... case is a conditional multiple branching control structure. Switch ... case control structure selects one of the several execution paths depending upon the value of a given expression. Inside the switch statement, there can be many case statements. The case with value equal to the value of the switch expression is executed. If no matching case is found then the default case is executed. Each case is terminated with a break statement. The switch expression can be an integer or a string.

```
Switch (<expression>) {
    case <value1>:
        Statements in case 1
        break;
    case <value2>:
        Statements in case 2
        break;
    .....
    default:
        Statement if no case matches
}
```

Flowchart of switch...case

Looping Control Structures :

Looping control structures are used to execute a group of statements repeatedly until a given condition ~~because~~ becomes false. PHP provides four looping statements :

- while statement
- do... while statement : do... while... ah
- for statement
- for each statement

while Statement :

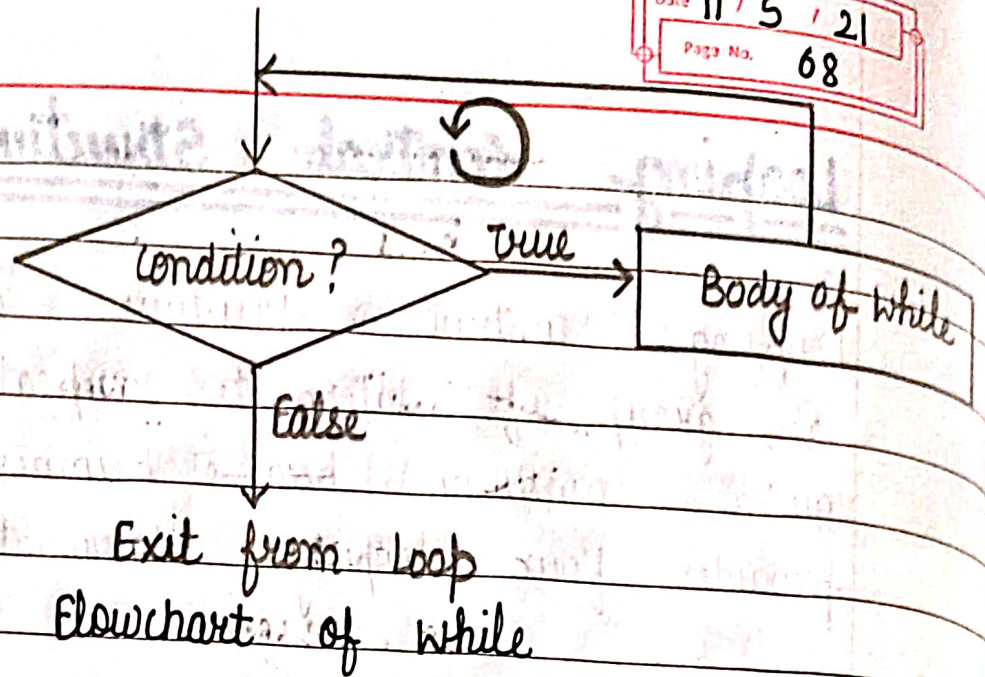
while control structure is used to execute a group of statements repeatedly until a given condition becomes false. The condition is checked before each iteration of the loop and the body of the loop is executed only if the condition is true.

```
while (<condition>) {
    Body of the loop
}
```

OR

```
while (<condition>) :
    Body of the loop
endwhile ;
```

syntax of while



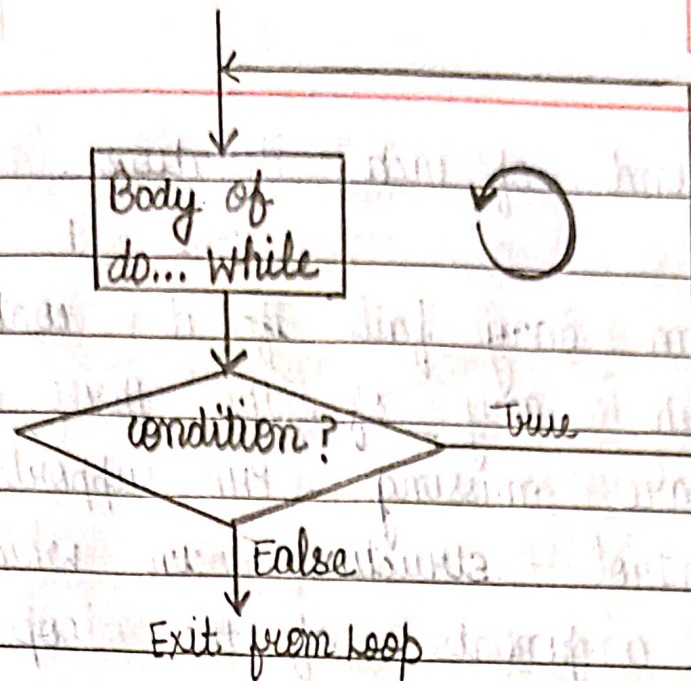
do....while Statement :

do... while control structure is used to execute a group of statements repeatedly, until a given condition becomes false. The condition is checked after each iteration of the loop and the next iteration is executed only if the loop condition is true. In do... while, body of loop is guaranteed to be executed at least once even if the loop condition is false.

```

do {
    Body of the loop
} while (<condition>);
  
```

Syntax of do... while



Flowchart of do...while

for statement :

Most of the loops used in programs have three essential parts in addition to the body of the loop :

→ Loop Initialization : Some variables are required to be initialized before entering the loop for the first time.

→ Loop condition : The condition that is checked before executing the next iteration of the loop.

→ Loop parameter updating : The variables used in the condition must be updated.

at the end of each iteration.

A program may fail if it contains a loop in which any of the three essential loop parts are missing. PHP supports the for loop control structure in which all the three essential components of the loop are written together in the for statement.

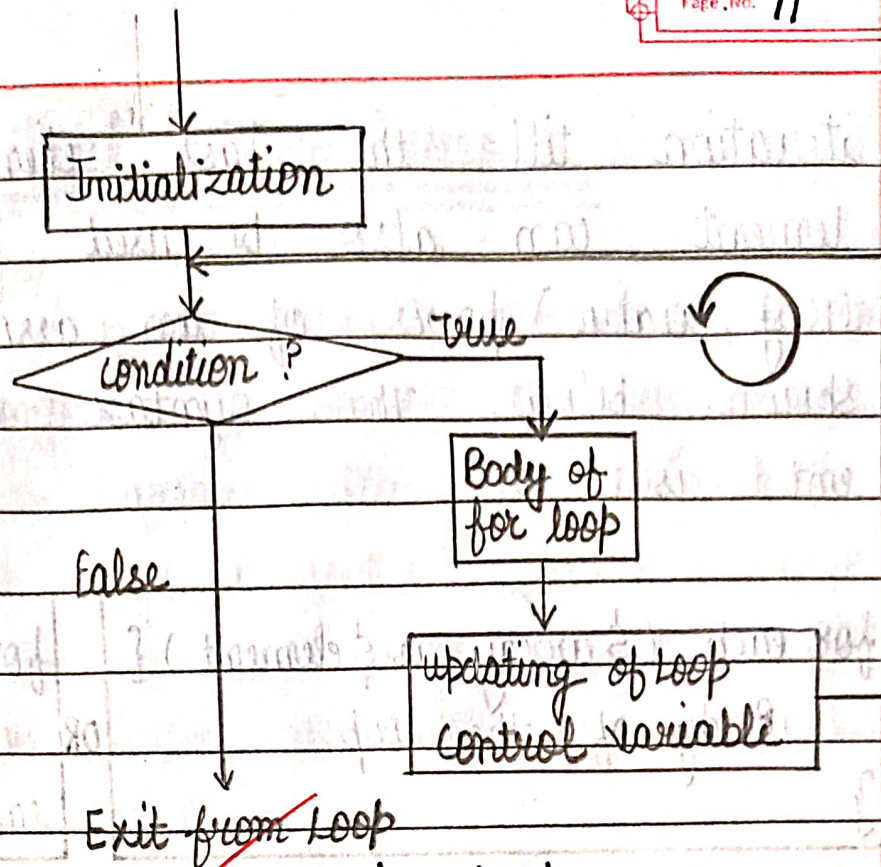
Syntax of for loop is :

```
for (< initialization > ; < condition > ; < updating > ) {
    Body of the loop
}
```

OR

```
for (< initialization > ; < condition > ; < updating > ) :
    Body of the loop
endfor ;
```

The initialization is executed before testing the loop condition for the first time. Condition is checked before each iteration and the body of the loop is executed only if the condition is true. The updating of loop control variable is carried out at the end of every iteration.



~~Flowchart of for loop~~

Example : ~~Multiplication Table of 5~~

```

for ($i = 1; $i <= 10; $i++):
    $b = 5 * $i;
    echo "5x $i = $b\n";
endfor;
  
```

foreach statement :

PHP provides a special looping control statement called foreach to process the array elements one by one. The loop begins by accessing the first element of the array and continues "accessing" the next element in each

iteration till the last element, foreach statement can also be used to access the (key, value) pairs of an associative array as shown below. The syntax of foreach statement is:

```
for each ($array as $element) {
    Body of the loop
}
```

OR

```
foreach ($array as $element) {
    Body of the loop
end for each;
```

```
foreach ($array as $key => $value) {
    Body of the loop
}
```

OR

```
foreach ($array as $key => $value):
    Body of the loop
endforeach;
```

Example: Sum of all elements of an array.

```
$arr = [5, 3, 9, 2, 4, 7, 2];
foreach ($arr as $element):
    $sum = $sum + $element;
endforeach;
echo "Sum of all elements : $sum";
```


Jump Control Structures :

Jump control structures are used to transfer the flow control to some other location within the script. Various Jump statements in PHP are:

- Continue :

Continue Statement can be used inside loops only. Continue Statement terminates the current iteration of the loop and initiates the next iteration.

Example: Count odd numbers in an array

```
$arr = [5, 3, 9, 2, 4, 7, 2];  
$count = 0;  
foreach ($arr as $element):  
    if ($element % 2 == 0) continue;  
    $count++;  
endfor;  
echo "Number of odd Numbers : $count";
```


• break :

break statement can be used inside loops and switch statements only. When break statement is used inside a loop, the loop is immediately terminated without testing the loop condition. break statement is also used inside switch statement to terminate the currently executing case.

Example: checking for prime Number

```
$ is_prime = true;
for ($i = 2; $i <= $num / 2; $i++);
    if ($num % $i == 0) {
        $is_prime = false;
        break;
    }
endfor;
if ($is_prime)
    echo "Number is prime";
```

• return :

return statement is used to return the control from the currently executing function back to

the statement where the function was called from. Return statement can also return a value along with the control.

- goto :

goto statement is used to transfer control to some other statement within the script. The statement to which the control is to be transferred must have a label.

UNIT - 3 PHP Function

PHP Function :

A large and complex programming problem can be easily solved by dividing it into several small sub-problems. Small sub-problems are easy to implement and test. Sub-problems are implemented as procedures and functions in programming languages. A function is a self-contained block of statements to perform some specific task. A function may accept zero or more arguments and can return a single value. Once a function is defined, it can be called for execution whenever required. PHP provides many built-in general purpose functions that can be directly used in the scripts. We can also define our own custom functions as per the requirement.

PHP Builtin Functions :

Some important built-in PHP functions are :

Function	Description
<code>abs()</code>	<code>abs()</code> function returns the absolute value of its numeric argument. Absolute value is the value of a number obtained by ignoring its sign. For example <code>abs(-3)</code> will return 3 and <code>abs(3)</code> will return 3.
<code>ceil()</code>	<code>ceil()</code> function returns the smallest integer which is greater than or equal to its argument. For example <code>ceil(2.1)</code> will return 3 and <code>ceil(-2.1)</code> will return -2.
<code>floor()</code>	<code>floor()</code> function returns the largest integer which is smaller than or equal to its argument. For example <code>floor(2.7)</code> will return 2 and <code>floor(-2.7)</code> will return -3.
<code>sqrt()</code>	<code>Sqrt()</code> function returns the square root of its argument. If the argument is negative

then `sqrt()` returns `NAN`. For example `sqrt(4)` will return 2 and `sqrt(-4)` will return `NAN`.

log 10() `log 10()` function returns the log to the base 10 of its argument number. If the argument is negative then `log 10()` returns `NAN`. For example `log 10(100)` will return 2 and `log 10(-100)` will return `NAN`.

pow() `pow()` function returns the result obtained by raising the first argument to the power of the second argument. For example `pow(2,3)` will return 8 and `pow(4, 0.5)` will return 2.

sin() `sin()` function returns the Trigonometric sine ratio of a given angle. The angle must be specified in radians. For example `sin(0)` will return 0 and `sin(pi()/2)` will return 1.

cos() `cos()` function returns the Trigonometric cosine ratio of a given angle. The angle must be specified in radians. For example `cos(0)` will return 1 and `cos(pi()/2)` will return 0.

tan() **tan()** function returns the trigonometric tangent ratio of a given angle. The angle must be specified in radians. For example **tan(0)** will return 0 and **tan(pi()/4)** will return 1.

PHP Builtin String Functions :

Function	Description
strtoupper()	strtoupper() function returns a string obtained by converting all lower case letters of its argument to the uppercase. For example strtoupper("hello") will return "HELLO".
strtolower()	strtolower() function returns a string obtained by converting all uppercase letters of its argument to the lowercase. For example strtolower("Hello") will return "hello".
strlen()	strlen() function returns the number of characters in a given string. For exam

ple `strlen("Hello World!")` will return 12.

`ltrim()` `ltrim()` function returns a string obtained by removing all the space characters from the left side of a given string. For example `ltrim(" Hello ")` will return "Hello".

`rtrim()` `rtrim()` function returns a string obtained by removing all the space characters from the right side of a given string. For example `rtrim(" Hello ")` will return "Hello".

`trim()` `trim()` function returns a string obtained by removing all the space characters from both sides of a given string. For example `trim(" Hello ")` will return "Hello".

`explode()` `explode()` function returns an array consisting of substrings of a given string which are separated with a given separator.

For example `explode(" ", "This is a sample string")` will return array:

This	is	a	sample	string
------	----	---	--------	--------

implode() implode() function returns a string obtained by concatenating all the elements of a given array separated with a given separator. For example if an array $\$arr = ["one", "two", "three"]$ then `implode("-", $arr)` will return "one-two-three".

strcmp() strcmp() function compares two strings and returns:

- a negative integer if first string is smaller than second string.
- Zero if first string is equal to the second string.
- A positive integer if first string is greater than second string.

For example `strcmp("a", "d")` will return -3
`strcmp("a", "a")` will return 0 and `strcmp("a", "A")` will return 3.

strpos() strpos() function returns the position of the 1st occurrence of a substring in a given string. It returns false if the substring is not found in the given string.

For example strpos("hello world", "world") will return 6 and strpos("hello world", "xyz") will return false.

htmlspecialchars() Returns a string after converting all the HTML special characters in a given string to the equivalent HTML entities. For example htmlspecialchars("10 < 15 & 12 > 15") shall return "10 & lt ; 15 & amp ; 12 & gt ; 15". By using htmlspecialchars() function we can make a string suitable to display in a web browser.

User - Defined Functions :

A function is a self-contained block of statements to perform a specific task. A function has a name and may accept arguments and return a single value to its calling statement. A function is not executed automatically, but it has to be called when we want to execute it. We can define our own custom functions in PHP using following syntax.


```
function function_name ($arg1, $arg2, ...)
{
    // Statements in the function
    return value;
}
```

Once a function is defined, it can be called using following syntax:

```
function_name (val1, val2, ...);
```

- Some important properties of functions in PHP are:

- Function names are case-insensitive.
- A function can be defined inside another function.
- A function can also call itself to solve problems using recursion.
- A function may accept zero or more arguments.
- A function may return a single value.
- A function must be called when we need to execute it.

- The scope of a function is always global
- A function can not be overloaded and redefined

Examples:

```
function hello () { // function definition
    echo "Hello";
}
```

```
HELLO(); // function call
```

```
function big ($num1, $num2) { // function definition
    if ($num1 > $num2)
        return $num1;
    else
        return $num2;
}
```

```
echo big (10, 5); // function call
```


UNIT- 4

PHP Form Processing

HTML Forms:

- A form is a component of a web page that is used to collect the user input data. A form consists of different form controls, such as text fields, buttons, checkboxes, radio buttons, drop-down menus etc. A user can enter data in these form controls and submit the data to the server for further processing. Some examples where a form is required in a web page are: online application form for admission, online survey, online quiz, online booking etc. HTML `<form>` tag is used to create a form element in a web page. The `<form>` element may contain several form control elements for different type of inputs. Different types of form controls that can be used inside a form element are:
 - `<input>`
 - `<label>`

- `<button>`
- `<textarea>`
- `<Select>`
- `<fieldset>`

• `<input>` :

The HTML `<input>` element is used to create different types of controls in web-based forms for user input. The `<input>` element is the most common form control and provides a number of combinations of input types. We can create different types of input control by specifying different value to the "type" attribute of the `<input>` element. Various types of `<input>` element are:

Type	Description	Example
text	A single-line text field. "text" is the default value if the type attribute is omitted. The "size" attribute can be used to set the width of the input field.	<code><input type="text"></code> John

- `<button>`
- `<textarea>`
- `<Select>`
- `<fieldset>`

• `<input>` :

The HTML `<input>` element is used to create different types of controls in web-based forms for user input. The `<input>` element is the most common form control and provides a number of combinations of input types. We can create different types of input control by specifying different value to the "type" attribute of the `<input>` element. Various types of `<input>` element are:

Type	Description	Example
Text	A single-line text field. "text" is the default value if the type attribute is omitted. The "size" attribute can be used to set the width of the input field.	<code><input type="text"></code> John

	The 'maxlength' attribute can be used to set the maximum number of characters allowed in the field.	
password	A single-line text field for entering password. The text typed by the user is not visible.	****
button	A push button displaying the text of the value attribute. The action that we want to carry out when the user clicks the button is specified using JavaScript.	Save
checkbox	checkboxes let a user select ZERO or MORE options from a number of choice. checkboxes are generally used in groups created	<pre> <fieldset> <legend> Hobbies </legend> <input type="checkbox" id="ckt"> <label for="ckt"> Cricket </label> </pre>

	The 'maxlength' attribute can be used to set the maximum number of characters allowed in the field.	
password	A single-line text field for entering password. The text typed by the user is not visible.	<input type = "password"> ****
button	A push button displaying the text of the value attribute. The action that we want to carry out when the user clicks the button is specified using JavaScript.	<input type = "button" value = "Save"> Save
checkbox	checkboxes let a user select ZERO or MORE options from a number of choice. checkboxes are generally used in groups created	<fieldset> <legend> Hobbies </legend> <input type = "checkbox" id = "ckt"> <label for = "ckt"> Cricket </label>

using `<fieldset>` with `<legend>` elements. A `<label>` element is also used with a checkbox to provide a description of the option. The `id` attribute of `<checkbox>` and `for` attribute of its `<label>` must be same.

```
<input type="checkbox" id="rd">
<label for="rd">
  Reading </label>
</fieldset>
```

Hobbies

☐ Cricket ☐ Reading

radio

Radio button let a user select ONLY one option from a number of choices. Radio buttons are generally used in groups created using `<fieldset>` with `<legend>` elements. A `<label>` element is also used with a radiobutton to provide a description of the option. The `id` attribute of `<radio>` and `for` attribute of its `<label>` must be same. The `name` attribute of all the radio buttons of the group must be same.

```
<fieldset>
<legend> Gender </legend>
<input type="radio" id="male" name="gender">
```

```
<label for="male">
  Male </label>
```

```
<input type="radio" id="female" name="gender">
  Female </label>
```

```
</fieldset>
```

Gender

☐ Male ☐ Female

Type	Description
date	The date as type of <input> form control creates a date field. A calendar is shown when the user clicks on the date field. The format of the calendar may vary from browser to browser. The "min" and "max" attributes of <input> tag can be used to enter the valid range of dates.

Example :

```
<label for="dob">Date of Birth:</label>
<input id="dob" type="date">
```

Date of Birth:

November 2020
 ↑
↓

S	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	1	2	3	4	5

Today

Type

Description

Submit

The `<input>` submit type creates a special button in the form. When the user clicks the submit button, the data entered by the user in the form is sent to the URL specified in the action attribute of the `<form>` element. The data will be submitted only for those form controls in which there is a name attribute defined. A form must have at least one submit button.

Example:

```
<form action = "login.php">
  <label for = "fname"> User ID: </label>
  <input type = "text" id = "uid" name = "uid"> <br>
  <label for = "pwd"> Password: </label>
  <input type = "password" id = "pwd" name = "pwd"> <br>
  <input type = "Submit" value = "Login">
```

</form>

User ID:

Password:

Login

Type	Description
reset	A reset button in a <form> element is used to clear the values entered by the user in the form controls and reset them to their initial values.

Example:

```

<form action = "login.php">
  <label for = "uid"> User ID: </label>
  <input type = "text" id = "uid" value = "admin"><br>
  <fieldset>
    <legend> Gender </legend>
    <input type = "radio" id = "male" name = "gender">
    <label for = "male"> Male </label>
    <input type = "radio" id = "female" name = "gender" checked>
    <label for = "female"> Female </label>
  </fieldset>
  <label for = "pwd"> Password: </label>
  <input type = "password" id = "pwd" name = "pwd"><br>
  <input type = "reset" value = "Reset">
</form>

```


User ID: ☐ Male ☒ FemalePassword:

Type

Description

Number

The number type input is used to create an input field for numeric data. User can enter only valid numbers in this field.

Example:

`<label for="marks"> Your Marks: </label>``<input type="number" id="marks">`Your Marks:

Type	Description
email	The email type input is used to create an input field for entering valid email addresses in the format of user_name @ domain_name. If an invalid email address is entered then the form data cannot be submitted.

Example:

```
<label for = "email"> Email ID: </label>
<input type = "email" id = "email"> <br>
<input type = "submit" value = "Submit">
```

Email ID:

Please include an '@' in the email address. 'xxx' is missing an '@'.

Type	Description
color	The color type input provides the feature to select the color from the color picker. The value returned by the color field is the RGB components of the chosen color.

Example :

```
<label for = "color"> choose color : </label>
<input type = "color" id = "color"><br>
```

Type	Description
File	The file type input provides the feature to select a file for uploading on the server. A "Browse" or "choose file" button is provided for selecting file from the computer.

Example :

```
<label for = "photo"> Select Photograph : </label>
<input type = "file" id = "photo">
Select Photograph : Chosen File No file chosen
```


HTML Form Controls (<Select>):

- <Select>: The `<Select>` element is used to create a drop-down menu. An option in the drop-down list is created using an `<option>` element inside the `<Select>` element. There can be many `<option>` elements inside a `<Select>` element. The user can select one of these options for the input data. Each `<option>` element must define the `value` attribute containing the data value to submit to the server when that option is selected. We also create logical groups of options by using `<optgroup>` element. The `<optgroup>` must have a `label` attribute to define the text that appears in the drop-down list. We can also use the `"multiple"` attribute in the `<Select>` tag to create a drop-down list in which we can choose more than one option at a time.


```
< label for = "distt" > District : </ label >
```

```
< Select name = "distt" id = "distt" >
```

```
< option value = "BLP" > Bilaspur </ option >
```

```
< option value = "CHA" > Chamba </ option >
```

```
< option value = "HMR" > Hamirpur </ option >
```

```
< option value = "KGR" > Kangra </ option >
```

```
< option value = "KNR" > Kinnaur </ option >
```

```
< option value = "KUL" > Kullu </ option >
```

```
< option value = "L&G" > Lahul & Spiti </ option >
```

```
< option value = "HAN" > Mandi </ option >
```

```
< option value = "SHL" > Shimla </ option >
```

```
< option value = "SRM" > Sirmour </ option >
```

```
< option value = "SOL" > Solan </ option >
```

```
< option value = "UNA" > Una </ option >
```

```
</ Select >
```

District :	Bilaspur
	Bilaspur
	Chamba
	Hamirpur
	Kangra
	Kinnaur
	Kullu
	Lahul & Spiti
	Mandi
	Shimla

<label for="inst"> Institute : </label>

<Select name="inst" id="inst">

<option value=""> -- Select an Institute -- </option>

<optgroup label="Polytechnics">

<option value="GIPH"> Govt. Polytechnic Hamirpur
</option>

<option value="GPK"> Govt. Polytechnic Kangra
</option>

<option value="GPS"> Govt. Polytechnic Sundernagar
</option>

<option value="GPA"> Govt. Polytechnic Ambala
</option>

</optgroup>

<optgroup label="Engineering Colleges">

<option value="GHEC"> Govt. Hydro-Engin-
eering college Bilaspur </option>

<option value="JNGEC"> Jawahar Lal Nehru
Engg. college Sundernagar </option>

<option value="RGECE"> Rajiv Gandhi Engg.
college Kangra </option>

</optgroup>

</Select>

Institute :

-- Select an Institute --

-- Select an Institute --

Polytechnics

Govt. Polytechnic Hamirpur

Govt. Polytechnic Kangra

Govt. Polytechnic Sundernagar

Govt. Polytechnic Ambala

Engineering Colleges

Govt. Hyd. Engineering College Bilaspur

Jawahar Lal Nehru Engg. College Sundernagar

Rajiv Gandhi Engg. College Kangra

HTML Form Controls (<textarea>):

• <textarea> : The <textarea> tag defines a multi-line text input control. The <textarea> element is often used in a form, to collect user inputs like comments and addresses. A text area can hold an unlimited number of characters, and the text renders in a fixed-width font (usually Courier). The size of a text area is specified by the "cols" and "rows" attributes.


```
<label for = "address"> Address: </label> <br>
<textarea id = "address" rows = "4" cols = "40"></text-
area>
```

Address:

- **<button>** : The `<button>` tag defines a clickable button. We can create a general button or special buttons like "Submit" and "Reset" using the `<button>` tag. The behaviour of a general button is specified using Java Script code. Inside a `<button>` element we can use text and images which is not possible with a button created with the `<input>` element.

`<img src = "image" />`

```
<button type = "button" value = "Submit">Submit</button>
<button type = "button" value = "Reset">Reset</button>
<button type = "button" value = "Submit">Submit</button>
```


HTML Forms:

Some important attributes of a `<form>` element are:

- **action**: The URI of the server-side script that processes the form data on submitting the form. If no action attribute is defined for a `<form>` element, the data is sent to the same page on which the form is present. Form data can be accessed through `$_REQUEST`, `$_GET` or `$_POST` superglobal PHP arrays in the server-side script.

Example: Form without action attribute:

```
<form>
  <input name = "emp_name">
  <input type = "submit" value = "Submit">
</form>
```

Example: Form with action attribute:

```
<form action = "process.php">
  <input name = "emp_name">
  <input type = "submit" value = "Submit">
</form>
```


Some important attributes of a `<form>` element are:

- **method**: The `method` `<form>` attribute defines how data is sent to the server. Form data can be transmitted to the server using a number of methods. GET and POST methods are the most common form submission methods.

GET Method: If a form is sent using GET method, the form data is appended to the URL of the server-side script using a query string. The query string begins with a question mark (?) and consists of name = value pairs separated with & symbols. GET is the default form submission method.

POST Method: If a form is sent using POST method, the form data is included as HTTP header in the body of the HTTP request. POST method is more secure way of transmitting form data to the server. Using POST method, we can submit a large amount of form data (e.g. file uploads) as compared to GET method.

Difference between GET and POST Methods:

GET	POST
- Form data is appended to the URL of the server-side script. - Only a limited form data can be submitted using GET method. - GET is not a good method to submit sensitive data. - Requests made using GET method are cached and can be found in browser history.	- Form data is sent in the body of the HTTP request. - Large data can be submitted using POST method. - POST can be used to send sensitive and confidential form data. - Requests made using POST method are not cached and cannot be found in browser history.

HTML Forms "action" Attribute :

```
<form action = "login.php" method = "GET">
  <label> User ID / Password </label>
  <input name = "User_name">
  <input type = "password" name = "pwd">
  <input type = "submit" value = "Login">
</form>
```

User ID / Password

```
<form action = "login.php" method = "POST">
  <label> User ID / Password </label>
  <input name = "User_name">
  <input type = "password" name = "pwd">
  <input type = "submit" value = "Login">
</form>
```

User ID / Password

Forms Controls "name" Attribute:

- The form elements included in a `<form>` element must have a name attribute defined. The name of a form element is used to identify the element at client as well as server end. When a form is submitted, the data of only those elements for which the name attributes are defined, is submitted to the server-side script. If the name attribute is not defined for some element, its data will not be included in the form data for submission.

```
<form action = "login.php" method = "GET">
  <label> User ID / Password </label>
  <input name = "User_name">
  <input type = "password" name = "pwd">
  <input type = "Submit" value = "login">
</form>
```

User ID / Password

admin

....

login

HTML Elements "id" Attribute:

- The id attribute can be used for any HTML element. There should be only one element with a given id in an HTML document. Various roles of the id attributes are:

→ CSS Selector: The element with a given id can be styled by using #id as CSS Selector.

→ Form Elements: A label can be associated with a form element by setting "for" attribute of the label to the "id" attribute of the form element.

→ To access element in Java Script code.

→ To create a fragment identifier inside a web page. The browser will scroll to the element with a given id if a # fragment appears at the end of the URL.

PHP Superglobals:

• PHP provides several predefined variables called Superglobals. Superglobal variables are available in all scopes within a script. All superglobals are associative arrays. Various superglobals in PHP are:

• `$GLOBALS`

• `$_SERVER`

• `$_GET`

• `$_POST`

• `$_FILES`

• `$_COOKIE`

• `$_SESSION`

• `$_REQUEST`

• `$_ENV`

• `$GLOBALS`:

`$GLOBALS` is an associative array containing references to all variables defined in the global scope. The variable names are the keys of the array.


```
<?php
```

```
$x = "Text - 1";
```

```
function test () {
```

```
    $x = "Text - 2";
```

```
    echo 'Global $x: ', $GLOBALS["x"];
```

```
    echo 'Local $x: ', $x, "\n";
```

```
}
```

```
test ();
```

```
?>
```

Output :

```
Global $x: Text - 1
```

```
Local $x: Text - 2
```

• \$_GET :

\$_GET is an associative array that contains the form data submitted to the script using HTTP GET method. \$_GET also contains the data passed to the script in the query string of the URL. The elements of the array are indexed using the names of the data elements.

```
<form method = "GET">
```

```
    <input name = "x">
```

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

```
</form>
```


<?php

```
echo 'Data in $_GET["x"] : ' . $_GET["x"] ;  
?>
```

localhost:8080/xyz.php?x=Hello

Data in \$_GET["x"] : Hello

• \$_POST :

\$_POST is an associative array that contains the form data submitted to the script using HTTP POST method. The elements of the array are indexed using the names of the data elements.

```
<form method="POST">
```

```
<input name="x">
```

```
<input type="Submit">
```

```
</form>
```

```
<?php
```

```
echo 'Data in $_POST["x"] : ' . $_POST["x"] ;  
?>
```

localhost:8080/xyz.php

Data in \$_POST["x"] : Hello

- **\$_REQUEST :**
 \$_REQUEST is an associative array that contains the data passed to the script using HTTP GET method, HTTP POST method and HTTP COOKIES. The elements of the array are indexed using the names of the data elements.

```

<form method="POST">
    <input name="x">
    <input type="submit">
</form>
<?php
    echo 'Data in $_REQUEST["x"]:' . $_REQUEST["x"];
    echo 'Data in $_REQUEST["y"]:' . $_REQUEST["y"];
?>

```

← → ⓘ localhost : 8080 / xyz.php ?y = World

Data in \$_REQUEST["x"] : Hello

Data in \$_REQUEST["y"] : World

- **\$_SERVER :**
 \$_SERVER is an associative array containing information such as headers, paths, and script locati-

ons. The entries in this array are created by the web server. Some of the important data stored in the `$_SERVER` array is as:

Index Key	Description
PHP_SELF	<code>\$_SERVER['PHP_SELF']</code> contains the filename of the currently executing script.
SERVER_ADDR	<code>\$_SERVER['SERVER_ADDR']</code> contains the IP address of the web server.
SERVER_NAME	<code>\$_SERVER['SERVER_NAME']</code> contains the name of the server host on which the current script is executing.
SERVER_SOFTWARE	<code>\$_SERVER['SERVER_SOFTWARE']</code> contains the information about the web server software.
REQUEST_METHOD	<code>\$_SERVER['REQUEST_METHOD']</code> contains the request method was used to access the page e.g. 'GET', 'HEAD', 'POST', 'PUT'.

QUERY_STRING	<code>\$_SERVER['QUERY_STRING']</code> contains the query string of the URL using which page is accessed.
DOCUMENT_ROOT	<code>\$_SERVER['DOCUMENT_ROOT']</code> contains the path of the root directory for documents served by the web server.
REMOTE_ADDR	<code>\$_SERVER['REMOTE_ADDR']</code> contains IP address of the client computer from which user is accessing the page.
HTTP_USER_AGENT	<code>\$_SERVER['HTTP_USER_AGENT']</code> contains the information about the client software (browser) using which the page is accessed.
SERVER_PORT	<code>\$_SERVER['SERVER_PORT']</code> contains the TCP port on which the web server is running.
SCRIPT_NAME	<code>\$_SERVER['SCRIPT_NAME']</code> contains the current script's filename with path.

REMOTE_HOST	<code>\$_SERVER['REMOTE_HOST']</code> contains the name of the client computer using which the page is accessed.
-------------	--

• `$_COOKIE` :

An HTTP cookie is a small piece of data that a server sends to the user's web browser. The browser stores it on client machine and sends it back with next requests to the same server. `$_COOKIE` is an associative array containing the cookies data passed to the script by the browser.

• `$_SESSION` :

A session is a way to store data to be used across multiple pages and requests. The session data is stored on the server. `$_SESSION` is an associative array containing session variables available to the current script. We can set any session variable by directly setting its value in `$_SESSION` array.

• \$ _FILES :

\$ _FILES is an associative array containing the information about the files uploaded by the client using HTTP POST method. Each element of \$ _FILES is again an associative array having following keys.

Key	Description
name	\$ _FILES['name']['name'] contains the name of the uploaded file.
type	\$ _FILES['name']['type'] contains the type of uploaded file.
tmp_name	\$ _FILES['name']['tmp_name'] is the temporary file name in which the server stores the file.
error	\$ _FILES['name']['error'] contains file upload error information.
size	\$ _FILES['name']['size'] contains the size of uploaded file in bytes.

- **\$ENV:**
\$ENV is an associative array of variables containing environment variables available to current script. These variables are imported by the server from the environment under which the PHP parser is running. The variables included in \$ENV may be different for different server environments.

Data Validation:

The authenticity and accuracy of data submitted by the users is very important in correct working of a web applications. Unacceptable data submitted by a user may cause an application to crash. To solve such problems, the concept of data validation is used. Data validation is the process of checking the input data for accuracy before processing it. Various reasons for data validation are:

- To get the right data, in the right format. The user may be forced to enter the data satisfying given constraints and in a specific format.

To authenticate users by secure passwords and CAPTCHA.

To protect data and application by denying the malicious user from injecting code through unprotected forms to damage the application.

There are two types of methods for form data validation:

• client - side validation - validation done in the browser.

• Server - side validation - validation done on the server.

Client - side validation:

Client - side validation is carried out in the web browser. The data is submitted to the server only after it satisfies all the client - side validation. There are two different types of client - side validation methods:

(i) Built-in / HTML form validation: This type of validation is directly supported by

the HTML and does not require the use of Java Script. Built-in form validation is very easy to use and has better performance than other types of validations but, it supports only limited features.

(ii) JavaScript Validation: This type of validation is coded using Java Script and is very flexible.

Server-side Validation: Server-side validation is carried out at the server by the server-side script to which the data is submitted. Once the form data is received at the server, we can apply many validation checks on it. It is a good practice to never trust the data submitted by the client and always perform the validation on the server also.

Some of the data submitted by the user cannot be validated at the client. For e.g. user authentication is not possible at the client side. For validating such data, server-side validation is required.

Client - Side Validation:

Client - side validation is carried out in the web browser. The data is submitted to the server only after it satisfies all the client - side validation. There are two different types of client - side validation methods:

Built-in / HTML Form Validation:

This type of validation is directly supported by the HTML and does not require the use of Java Script. HTML form validation make use of different HTML element attributes to check the input entered by the user. Some of these attributes are:

- **required:** the required attribute in a form element ensures that the input field is not be left blank by the user.

- **minlength and maxlength:** These two attributes are used to specify the minimum and maximum length of text data.

- min and max : These two attributes are used to specify the minimum and maximum values of numeric and date input types.

- type : The type attribute of <input> element specifies the type of data that can be entered in the input field. The type can be text, number, email etc.

- pattern : The pattern attribute specifies a regular expression that must match with the data entered by the user.

<form>

<label for = "user-name"> User Name : </label>

<input type = "text" required minlength = "30" id = "user-name" name = "user">

<label for = "pwd"> Password : </label>

<input type = "password" name = "pwd" id = "pwd" required minlength = "8">

<label for = "email"> Email ID : </label>


```
<input type="email" required <br>
<label for="mobile">Mobile No.: </label>
<input type="text" pattern="^[789]\d{9}$" required
<br>
<input type="Submit" value="Submit">
</form>
```

Java Script Validation :

Java Script Validation provides more flexibility as compared to HTML validation. The form data can be accessed in the JavaScript code and validation can be applied. JavaScript validation is applied by using "onsubmit" attribute in the <form> element. The JavaScript code assigned to the onsubmit attribute is automatically executed when the form data is submitted. Form data is not submitted by the browser if the code in the onsubmit attribute returns false. document.forms[i]["control name"] can be used to access the form elements in the JavaScript code.


```
<Script>  
function check() {  
    let val = document.forms["myform"]["age"].value;  
    if (!parseInt(val) < 0) {  
        alert("Age cannot be negative");  
        return false;  
    }  
    return true;  
}
```

```
</Script>  
<form name = "myform" action = "" method = "POST" onsubmit = "return check()">  
    <label for = "User">Your Age : </label>  
    <input type = "text" required size = "5" name =  
        "Age"> <br>  
    <input type = "submit" value = "Submit">  
</form>
```


Server - Side validation :

Server - side validation is the process to check the validity of the data submitted by the user in the server - side script. Various reasons for performing server - side validation are :

- In some cases, the form data submitted by a user can only be validated from the data stored in the database at the server. In such cases, client - side validation is not feasible. For example, password authentication, duplicate user registration etc. can be checked at the server side only.
- A malicious user may try to inject malicious code through the form input controls. Server - side validation is the only way to prevent such attacks.
- The data submitted by a genuine user may be altered by an attacker on the network. Server - side validation is the only solution to this problem.


```
<form>
```

```
  <label for = "User"> User ID : </label>
```

```
  <input type = "text" id = "User" name = "User">
```

```
  <br>
```

```
  <input type = "Submit" value = "Submit">
```

```
</form>
```

```
<?php
```

```
  if (!isset ($_REQUEST ['User'])) {
```

```
    echo "User ID cannot be blank";
```

```
  } else if (htmlspecialchars ($_REQUEST ['User']) !=
```

```
    $_REQUEST ['User']) {
```

```
    echo "User ID contains invalid characters";
```

```
  }
```

```
?>
```


Handling Uploaded Files :

The steps to upload files from a client computer to the web server are as:

- (i) To upload files, the form data submission method must be **POST**. Files can not be uploaded with **GET** method.
- (ii) The **enctype** attribute of **<form>** element must be set to **"multipart/form-data"**. Files can not be uploaded if we do not define **enctype** attribute.
- (iii) The **type** attribute for the **<input>** element must be set to **"file"**, this will make the input field as a file select control.
- (iv) The **name** attribute of **<input>** element for uploading files must be defined. If name attribute is not defined then the file will not be uploaded to the server.

(v) The **MAX_FILE_SIZE** **hidden** `<input>` field must be created before the file input field, and its value is the maximum file size supported by PHP.

(vi) The **action** attribute of the `<form>` element must be set to the server-side script that handles the uploaded file.

When a user uploads a file, the web server stores it in a temporary file. This temporary file must be copied to a file with a suitable name in appropriate directory using a server-side script. PHP `$FILES` superglobal contains all the required information about the uploaded files. Various elements in the `$FILES` are:

Key	Description
'name'	<code>\$FILES ['name of input'] ['name']</code> contains the original name of the file uploaded by the user.

'size'	<code>\$-FILES ['name of input'] ['size']</code> contains the size of the uploaded file in bytes.
'tmp_name'	<code>\$-FILES ['name of input'] ['tmp_name']</code> contains the temporary name of the file at the server.
'error'	<code>\$-FILES ['name of input'] ['error']</code> contains the error code of the upload process. The error code can be: UPLOAD_ERR_OK: Successful upload. UPLOAD_ERR_INI_SIZE: File size is more than allowed size. UPLOAD_ERR_PARTIAL: File is partially uploaded. UPLOAD_ERR_NO_FILE: No file is uploaded. UPLOAD_ERR_CANT_WRITE: File can not be written.
'type'	<code>\$-FILES ['name of input'] ['type']</code> contains the mime type of the file, for example "image/jpeg".

15/06/21
135

Date 15/06/21
Page No. 135

move_uploaded_file() PHP function is used to move an uploaded file to the suitable location with proper file name.

UNIT - 5

PHP Advanced Features

Dealing With Multiple PHP Files :

- PHP provides a feature to substitute the content of an external PHP file in the PHP file being executed.
- This feature allows us to create separate PHP files for the code that we wish to include in several web pages of a website.
- Any changes made in a PHP file will be reflected in all the pages in which this file is included.
- The extension of a PHP file created for inclusion in other PHP files is .inc.php by convention.
- PHP provides two statements i.e. **include** and **require** to substitute the content of a file in the PHP file being executed.

UNIT - 5

PHP Advanced Features

Dealing With Multiple PHP Files :

- PHP provides a feature to substitute the content of an external PHP file in the PHP file being executed.
- This feature allows us to create separate PHP files for the code that we wish to include in several web pages of a website.
- Any changes made in a PHP file will be reflected in all the pages in which this file is included.
- The extension of a PHP file created for inclusion in other PHP files is .inc.php by convention.
- PHP provides two statements i.e. **include** and **require** to substitute the content of a file in the PHP file being executed.

- The **include** and **require** statements perform the identical functions with only difference as:

include statement will produce a warning if the target file does not exist and the script continues to execute.

require statement will cause a fatal error if the target file does not exist and the script terminates.

- The **include** and **require** statements have two other variants as **include once** and **require once**.

include once statement behaves like **includes** statement with only difference that it will not include a file that has already been included.

require once statement behaves like **require** statement with only difference that it will not include a file that has already been included.

Examples of PHP File Inclusion :

file 1. inc. php

```
<?php
    echo "File 1 included";
?>
```

main.php

```
<?php
    include ("file 1. inc. php");
    include ("file 2. inc. php");
    echo "Main script executing";
?>
```

Output

File 1 included

Warning : include(): Failed opening 'file 2. inc. php' for inclusion

(include_path = '.:/usr/share/php') in /usr/www/public/main.php on line 3

Main script executing

file 1. inc. php

```
<?php
```

```
    echo "File 1 included";
```

```
?>
```

main.php

```
<?php
```

```
    require ("file 1. inc. php");
```

```
    require ("file 2. inc. php");
```

```
    echo "Script executing";
```

```
?>
```

Output

File 1 included

Fatal error: require(): Failed opening required "file 2. inc. php".

include_path = '.:/usr/share/php') in /var/www/public/main.php on line 3

- In the examples, a file "file 1. inc. php" is created that contains PHP code to display the message "File 1 included".
- The file "file 1. inc. php" is then included in the PHP file "main. php" using **include** and

require statements

- The PHP file "main.php" also include another file "file 2.inc.php" which does not exist.
- The output of include and require statements shows that :

include statement only produces a warning and the main script continues to execute.

require statement gives an error and the main script terminates.

Examples of PHP File Inclusion :

main.php

```
<?php
```

```
include ("file 1.inc.php");
```

```
include ("file 1.inc.php");
```

```
?>
```

main.php

```
<?php
```

```
include_once ("file 1.inc.php");
```

```
include_once ("file 1.inc.php");
```

```
?>
```

Output

File 1 included . File 1 included

Output

File 1 included .

- In this examples, the "file 1. inc. php" is included in the PHP file "main. php" twice using `include` and `include - once` statements.
- The output of `include` and `include - once` statements shows that :

`include` statement includes the content of the files twice.

`include - once` statement includes the content of the file in the first statement but ignores the second inclusion of the same file.

Difference Between include and require:

Both include and require statements substitute and execute the content of a given PHP file in the currently executing PHP script. The main difference between these two is:

Include

require

- | | |
|---|--|
| <p>(i) include Statement gives a warning if the file being included does not exist.</p> | <p>require statement gives an error if the file being included does not exist.</p> |
| <p>(ii) The script continues to execute if the include statement fails to include the given file.</p> | <p>The script terminates if the require statement fails to include the given file.</p> |

UNIT-7

Using XML and AJAX with PHP

Extensible Markup Language (XML) :

- XML is a text-based markup language derived from SGML (Standard Generalized Markup Language) and standardized by W3C.
- XML is a markup language used to describe the structure of data to make it suitable for storage and exchange between applications.
- The data in an XML document is organized in hierarchical elements created using tags, attributes and content.
- The tag names and attributes in an XML document are not fixed. Users may create tags / attributes as per the application requirement.

UNIT-7

Using XML and AJAX with PHP

Extensible Markup Language (XML) :

- XML is a text-based markup language derived from SGML (Standard Generalized Markup Language) and standardized by W3C.
- XML is a markup language used to describe the structure of data to make it suitable for storage and exchange between applications.
- The data in an XML document is organized in hierarchical elements created using tags, attributes and content.
- The tag names and attributes in an XML document are not fixed. Users may create tags / attributes as per the application requirement.

Role of XML :

- The Internet is a global public network that connects a huge number of computers all over the world.
- The computers connected to the Internet (or some other network) may have different hardware and software configurations.
- The objective of connecting a computer to some network is to enable sharing of data and resources with other computers on the network.
- Hence, there is a need of open standards to describe data in such a format that can be easily exchanged between different applications.
- XML is designed to create documents containing structured data suitable for storage and exchange between different applications.

Advantages of XML :

- XML is both human and machine friendly language.
- XML is Self-documenting, i.e. it contains data as well as description about the data within the same document.
- XML is an open standard recommended by W3C.
- XML format is suitable for most of the electronic documents.
- XML is platform-independent.
- XML documents are created using plain text.

Advantages of XML :

- XML is both human and machine friendly language.
- XML is Self-documenting, i.e. it contains data as well as description about the data within the same document.
- XML is an open standard recommended by W3C.
- XML format is suitable for most of the electronic documents.
- XML is platform-independent.
- XML documents are created using plain text.

Difference Between XML and HTML :

XML	HTML
(i) Full form of XML is extensible Markup language.	Full form of HTML is Hypertext Markup language.
(ii) XML is designed to create documents for storing and transferring structured data.	HTML is designed to create web pages to be displayed in Web browser.
(iii) XML tags are not fixed. User may define own tags as per the application need.	HTML tags are fixed. A tag not supported by HTML is ignored by the browser.
(iv) Tag names and attribute names are case-sensitive in XML.	Tag names and attribute names are not case-sensitive in HTML.
(v) Every opening XML tag must have a matching closing tag. A slash must be appended to	There is no strict rule in HTML to add a slash at the end of unpaired tags.

to the tag if its is an unpaired tag.

Syntax of XML :

<?xml version = "1.0" encoding "utf-8"?>	XML Declaration
<!DOCTYPE address SYSTEM "address.dtd">	DTD Declaration
<!-- Addresses of Students -->	Comment
<Student id = "501">	Root element
<name> Tammay Patel </name>	Nested element
<branch> Information Technology </branch>	
<address>	
<village> Barui </village>	Document Body
<post-office> Mohini </post-office>	
<tehsil> Hamirpur </tehsil>	
</address>	
</Student>	Closing tag of root element

- An XML document can optionally begin with XML prolog. XML prolog consists of two optional parts: XML declaration and Document Type Definition (DTD). The format of XML declaration is:


```
<?xml version = "1.0" encoding = "utf-8"?>
```

- An XML document consists of a number of elements organized as a tree. The XML document has a single root element and all other elements must be organized hierarchically under the root element.

```
<opening tag of root element>
```

```
<opening tag for child-1>child-1 content </closing tag for child-1>
```

```
<opening tag for child-n>having other elements as a children>
```

```
<opening tag for child> content </closing tag for child>
```

```
</closing tag for child-n>
```

```
</closing tag of root element>
```

- An XML element has following structure :

```
<tag name = "value" ... > content </tag>
```

TagName Attributes (optional) content closing Tag

opening Tag

- Each opening tag must have the matching closing tag with same name beginning with / character.
- An element may have zero or more attributes containing additional information about the element. The attributes are defined in the opening tag of the element in the form of `name = "value"`.
- An element may contain the actual data or other element within its opening and the closing tags.
- The elements must be nested properly i.e. the element with most recent opening tag must be closed before its parent elements.
- The names of tags and attributes are case-sensitive and must begin with a letter (a-z, A-Z), underscore (`_`) or colon (`:`) and continue with letters, digits, hyphens, underscores, colons, or full stops [periods].

- values of the attributes must be enclosed inside quotations.
- The XML comments are written inside `<!--` and `-->`.

Manipulating XML in PHP:

- PHP provides many extensions/libraries for manipulating XML.
- Simple XML extension provides a very simple way to convert XML to an object that can be processed easily in PHP.
- Using Simple XML, we can convert XML files or XML strings into objects, manipulate its contents and write XML files.
- `Simplexml_load_string()` and `Simplexml_load_file()` functions are used to create objects of type Simple XML Element consisting of all the elements defined in the XML file or XML string.

- values of the attributes must be enclosed inside quotations.
- The XML comments are written inside `<!--` and `-->`.

Manipulating XML in PHP:

- PHP provides many extensions/libraries for manipulating XML.
- Simple XML extension provides a very simple way to convert XML to an object that can be processed easily in PHP.
- Using Simple XML, we can convert XML files or XML strings into objects, manipulate its contents and write XML files.
- `simplexml_load_string()` and `simplexml_load_file()` functions are used to create objects of type Simple XML Element consisting of all the elements defined in the XML file or XML string.

Students.xml

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<!DOCTYPE address SYSTEM "address.dtd">
```

```
<!-- Address of Students -->
```

```
<Students>
```

```
<Student>
```

```
<name> Tammay Patel </name>
```

```
< address>
```

```
<village> Baru </village>
```

```
<post-office> Mohin </post-office>
```

```
</ address>
```

```
</ Student>
```

```
< Student>
```

```
<name> Amam Chaim </name>
```

```
< address>
```

```
<village> Swahal </village>
```

```
<post-office> Mohin </post-office>
```

```
</ address>
```

```
</ Student>
```

```
</ Students>
```


Students.xml

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<!DOCTYPE address SYSTEM "address.dtd">
```

```
<!-- Address of Students -->
```

```
<Students>
```

```
<Student>
```

```
<name> Tammay Patel </name>
```

```
< address>
```

```
<village> Baru </village>
```

```
<post-office> Mohim </post-office>
```

```
</ address>
```

```
</ Student>
```

```
< Student>
```

```
<name> Aman Chaim </name>
```

```
< address>
```

```
<village> Surahal </village>
```

```
<post-office> Mohim </post-office>
```

```
</ address>
```

```
</ Student>
```

```
</ Students>
```


students.php

<?php

\$f = "students.xml";

\$students = SimpleXml::load_file(\$f);

print_r(\$students);

??

SimpleXMLElement Object

(

[student] => Array

(

[0] => SimpleXMLElement Object

(

[name] => Tammay Patel

[address] => SimpleXMLElement Object

(

[village] => Barui

[post-office] => Mahin

)

)

[1] => SimpleXMLElement Object

(

[name] => Amar Chaim

[address] => SimpleXMLElement Object

(

Date 23 06 / 21

Page No. 153

[village] => Swahal

[post-office] => Mohin

UNIT- 5

Error Handling in PHP :

- An error is a condition that causes a difference between the actual and the expected output of a program / script.
- Errors must be handled properly to make reliable software.
- PHP provides four levels of programming errors:
 - E_ERROR (Fatal Errors) : Critical errors.
 - E_PARSE (Parse Errors) : Syntax errors.
 - E_WARNING (Warnings) : Serious but non-critical errors.
 - E_NOTICE (Notices) : Non-critical error that may lead to a bug.

PHP Fatal Errors:

- Errors at E_ERROR level are fatal errors which are not recoverable. The execution of the script is terminated and an error message is displayed when a fatal error occurs. Examples of E_ERROR are "call to an undefined function", "memory allocation error" etc. The code given below demonstrate the PHP fatal error.

err.php

```
<?php
```

```
error_reporting (E_ALL)
```

```
myfunc ();
```

```
echo "This message will not display";
```

```
?>
```

Output:

```
Fatal error: Uncaught Error: Call to undefined
function myfunc () in /var/www/public/err.php:3
Stack trace: #0 {main} throw in
/var/www/public/err.php on line 3
```


PHP Parse Errors:

- Errors at E_PARSE level are fatal errors arising due to invalid PHP syntax. The execution of the script is terminated and an error message is displayed when a fatal error occurs. Examples of E_PARSE are "missing semicolon", "invalid variable names" etc. The code given below demonstrates the PHP parse error:

err.php

```
<?php
```

```
error_reporting(E_ALL)
```

```
$my_var = 0
```

```
echo "This message will not display";
```

```
?>
```

Output:

Parse error: syntax error, unexpected 'echo'
(T_ECHO) in /var/www/public/err.php on line 4

PHP Warnings :

- Errors at E_WARNING level are non-critical but serious errors. The execution of the script continues and a warning message is displayed when a warning occurs. Example of E_WARNING is include a non-existent PHP file, division by zero etc. The code given below demonstrate the PHP Warning.

ex.php

```
<?php
```

```
error_reporting(E_ALL);
```

```
$x = 10;
```

```
$y = 0;
```

```
$z = $x / $y;
```

```
echo "This message will be display";
```

```
?>
```

Output :

Warning : Division by zero in
/var/www/public/ex.php on line 5

This message will be display

PHP notice :

- Errors at E_NOTICE level are non-critical errors which may cause a bug in the program. The execution of the script continues and a notice message is displayed when a notice error occurs. Examples of E_NOTICE is using uninitialized variables in the program. The code given below demonstrates the PHP notice.

```
err.php
<?php
error_reporting (E_ALL);
echo $x;
echo "This message will be display";
?>
```

Output :

Notice: Undefined variable: x in
/var/www/public/err.php on line 3
This message will be display

User - defined Error Handler :

- Programmers can define custom error handlers in PHP. The steps to create a custom error handler are as:
 - (i) Disable default error handling by setting error_reporting to 0.
 - (ii) Define a function with parameters error level, error message, error file, error line, error context | first two parameters are mandatory and remaining are optional. Write the code to handle errors in this function.
 - (iii) Set the PHP error handler to the function created in step - 2.
 - (iv) Programmers can also raise custom errors by triggering the error with trigger_error() function.

User - defined Error Handler Example :

err.php

```
<? php
```

```
error_reporting(0);
```

```
function myErrorHandler($err_level, $err_msg) {
```

```
    echo "<br>" . "Error : [" . $err_level . "] " . $err_msg;
```

```
    if ($err_level == E_USER_ERROR) {
```

```
        die(1);
```

```
    }
```

```
}
```

```
Set error handler ("myErrorHandler");
```

```
echo $x;
```

```
$x = 10;
```

```
$y = 0;
```

```
if ($y == 0) {
```

```
    trigger_error("Division by 0", E_USER_ERROR);
```

```
}
```

```
$z = $x / $y;
```

```
echo "This message will not be displayed";
```

```
?>
```

Output :

Error : [8] undefined variable : x

Error : [256] Division by 0

Exception Handling in PHP:

- An Exception is an abnormal condition arising during program/script execution that may lead to unexpected results (e.g. division by zero).
- An Exception must be handled properly otherwise the application may behave abnormally.
- PHP has the inbuilt feature of exception handling similar to other languages like Java and C++.
- When an abnormal condition is detected, create a new object of **Exception** class and raise the exception by using **throw** statement.
- The code that throws an exception must be enclosed inside a **try {}** block.
- The **try {}** block must be followed by one or more **catch {}** blocks. A **catch {}** block is

used to handle one exception raised in the `try {}` block.

- There can be one optional `finally {}` block after `catch {}` blocks that contains the clean up code. The code written inside `finally {}` block is "guaranteed to execute".
- When an exception is raised inside the `try {}` block, the control is transferred to the matching `catch {}` block of the exception.
- The statements in the matching `catch {}` block are executed and the control is transferred to the `finally` block.
- The execution of the script then resumes at the first statement following `try... catch... finally` blocks.

exception : php

<? php

```
function divide ($a , $b) {
```

```
    if ($b == 0) {
```

```
        throw new Exception ("Division by zero <br>");
```

```
    }
```

```
    echo "$a / $b = " . $a / $b . " <br>";
```

```
}
```

```
try {
```

```
    divide (10, 0);
```

```
}
```

```
catch (Exception $e) {
```

```
    echo $e -> getMessage ();
```

```
}
```

```
finally {
```

```
    echo "This section will always execute <br>";
```

```
}
```

```
echo "Script will resume from here";
```

```
?>
```

Output 1: divide (10, 0)

Division by zero

This section will always execute

Script will resume from here.

Output 2: divide (10, 2)

10/2 = 5

This section will always execute.
Script will resume from here.

HTTP Headers :

- HTTP headers let the client and the server pass additional information with an HTTP request or response. An HTTP header consists of its case-insensitive name followed by a colon (:), then by its value. There are four types of HTTP headers.
- Request Headers : contain information about the resource to be fetched, or about the client requesting the resource.
- Response Headers : hold additional information about the response, like its location or about the server providing it.
- Representation Headers : contain information about the body of the resource, like its MIME

- \$http_response_code is the response code sent with the header.

PHP header () Example :

header.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title> PHP header </title>
</head>
<body>
  <a href="getfile.php"> Click Here to Download File </a>
</body>
</html>
```

getfile.php

```
<?php
header('Content-Type: application/pdf');
header('Content-Disposition: attachment; filename=
"downloaded.pdf"');
readfile('original.pdf');
```


- In this Example, the file save dialog will appear when user presses the link. The name of the file in the dialog will be different from the original file name.

~~Answer~~
~~22/07/21~~

Connecting PHP to MySQL Database

- MySQL is a Relational Database Management System (RDBMS) that stores data in the form of tables. MySQL make use of Structured Query Language (SQL) to perform various database operations. MySQL is open-source and free RDBMS.
- PHP provides native (built-in) support for handling MySQL databases.
- MySQLi Extension is a driver used in the PHP to provide an interface for MySQL databases.
- MySQL can be used either in object-oriented or procedural style in a PHP script.

Step 1: Connecting PHP to MySQL Database

- To connect PHP to MySQL, we can use either of the following approaches.

- Procedural Approach :
`$con = mysqli_connect (host, user, password, database);`

- Object - oriented Approach :
`$con = new mysqli (host, user, password, database);`

Step 2: Checking Connection Error

- Before using the connection created in Step 1 it is important to check for any connection error. The connection error can be checked in either of the following two approaches:

- Procedural Approach :
`if (mysqli_connect_error()) {
 die ("Failed to connect : " . mysqli_connect_error());
}`

- Object Oriented Approach :
`if ($con->connect_error) {
 die ("Failed to connect : " . $con->connect_error);
}`

Step 3: Executing Simple SQL Statements

→ We can execute SQL statements using \$con using object as:

- Procedural Approach:

```
if (mysqli_query($con, $sql)) {  
    PHP code for successful query execution  
} else {  
    PHP code for failed query execution  
}
```

- Object - Oriented Approach

```
if ($con->query($sql) === TRUE) {  
    PHP code for successful query execution  
} else {  
    PHP code for failed query execution  
}
```

Step 4: Handling Query Execution Errors

→ The errors occurred while executing an SQL statement are handled as:

- Procedural Approach

```
$res = mysqli_query($con, $sql);  
if (!$res) {
```



```
echo "Error : " . mySqli -> error ($con);
```

Object Oriented Approach

```
$res = $con -> query ($sql);  
if (! $res) {  
    echo "Error : " . $con -> error;  
}
```

Step 5 : Handling the Result of SELECT SQL Statements

→ The records returned by the SELECT statement are handled in PHP as:

Procedural Approach :

```
$res = mySqli -> query ($con, $query);  
if (mySqli -> num_rows ($res) > 0) {  
    while ( $row = mySqli -> fetch_row ($res) )  
        PHP code to process each record  
}  
else {  
    PHP code when no record is returned  
}
```


Object Oriented Approach

```

$ res = $ con -> query ($ sql);
if ($ res -> num_rows > 0) {
    while ($ row = $ res -> fetch_array()) {
        Print each row as process is required
    }
} else {
    Print note when no record is returned
}

```

Step 6: Close the database connection

→ The open database connection must be closed after all the database operations have been carried out as:

Procedural Approach

```
close ($ res con);
```

Object Oriented Approach

```
$ con -> close();
```

Summary :

1. Create a database connection
2. Query and database operations
3. Close the connection

Object Oriented Approach

```

$ res = $ con -> query ($ sql);
if ($ res -> num_rows > 0) {
    while ($ row = $ res -> fetch_array()) {
        PHP code to process each record
    }
} else {
    PHP code when no record is returned
}

```

Step 6: Close the Database Connection

→ The open database connection must be closed after all the database operations have been carried out as:

Procedural Approach

```
close ($ con);
```

Object Oriented Approach

```
$ con -> close();
```

Summary :

1. Create a database connection
2. Carry out database operations
3. Handle errors

4. Close database connection.

Handling Database in PHP:

Database: A database is an organized collection of related data stored on a computer system and managed by a special software called database Management System (DBMS). A database Management System makes it easy to store, process and access data from the database. A relational database management system (RDBMS) organizes the data in the form of tables. Various popular RDBMS are Oracle, MySQL etc. Structured Query Language (SQL) is the standard language used to perform various operations on databases, managed using RDBMS.

is a grid like structure.
Database Table: A table[↑] that consists of rows and columns. In RDBMS, data is organized in the form of tables. It is very easy to understand data.

stored in tables. The columns of a database table have headings (names). The records in ~~a~~ A row of a table contains data about a specific entity. We can store data of a particular type under a column of the table. Examples of database table is:

Column Name / Attribute

R.No	Name	Subject	Marks
1.	Alka	WP	40
2.	Anil	WP	35
3.	Ankit	WP	39
4.	Ankita	WP	40

Field

The number of columns in a table are generally fixed and number of rows are variable.

My SQL : My SQL is an open source and free relational database management System (RDBMS). My SQL uses Structured Query Language (SQL) as its standard database language.

My SQL Data Types :

My SQL provides several data types that can be used for various types of data to be stored under a table column.

Various ^{data} types in My SQL are :

(i) Numeric

