Web Technology II

Course Contents:

Theory

Unit 1. Web Server Concept **[5 Hrs.]**

1.1. Introduction to Web Server
1.2. Architecture of web server
1.3. Concept of Dynamic Content
1.4. Using control flow to control dynamic content generation
1.5. Concept of Architecting Web Application

Unit 2. Review of Database: MySQL **[4 Hrs.]**

2.1. Introduction to MySQL
2.2. MySQL queries
    2.2.1. Create
    2.2.2. Insert
    2.2.3. Select
    2.2.4. Update
    2.2.5. Delete
    2.2.6. Alter
2.3. Database Normalization

Unit 3. Server-Side Script: PHP **[12 Hrs.]**

3.1. Introduction of PHP
3.2. Advantage of using PHP for web development
3.3. PHP Installation
3.4. PHP Syntax
3.5. Comments, Variable, Operators, Datatype, Strings, Keywords
3.6. Conditional Statements
3.7. Loop
3.8. Arrays
3.9. Functions
3.10. Passing variables with data between pages
    3.10.1. Get & Post Method
    3.10.2. Cookies
    3.10.3. Sessions
3.11. File Upload: Date, Include, File, File Upload
3.12. Accessing Form Elements, Form Validation
3.13. Exception and Error Handling

Unit 4. Object oriented concept and Database Connectivity **[8 Hrs.]**

4.1. Classes and Objects
4.2. Access Modifiers
4.3. Constructors and Destructors
4.4. Inheritance and Scope
4.5. Overwriting Methods
4.6. Database Connectivity
    4.6.1. Creating database with Server-Side Script
    4.6.2. Connecting Server-Side Script to Database
    4.6.3. Multiple Connections
    4.6.4. Making queries
    4.6.5. Building in Error Checking

4.6.6. Fetching Data sets
4.6.7. Displaying Queries in tables
4.6.8. Building Forms and control form data using queries

Unit 5. AJAX and eXtensible Markup Language (XML) **[8 Hrs.]**

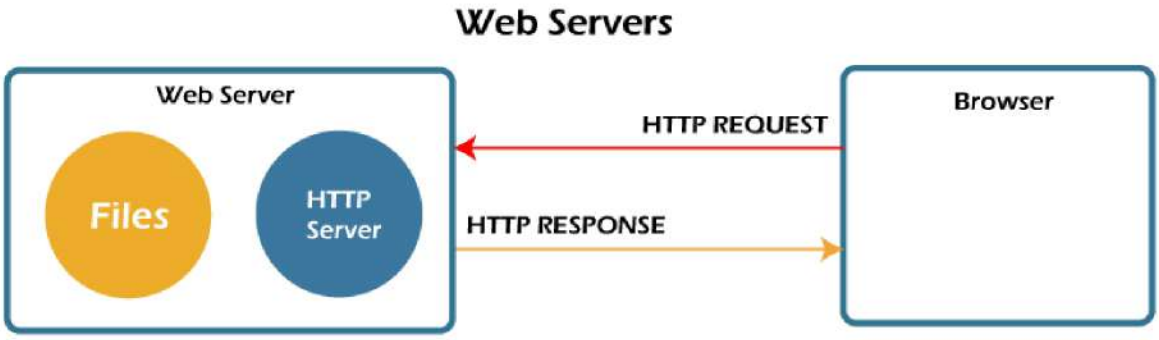
5.1. Basic concept of AJAX
5.2. Features of XML
5.3. Structure of XML: Logical Structure, Physical Structure
5.4. Naming Rules
5.5. XML Elements
5.6. XML Attributes
5.7. Element Content Models: Element Sequences i.e., <!ELEMENT counting (first, second, third, fourth)>, Element Choices <!ELEMENT choose (this.one | that.one)>, Combined Sequences and Choices
5.8. Element Occurrence Indicators: -Discussion of Three Occurrence Indicators? (Question Mark) * (Asterisk Sign) + (Plus Sign)
5.9. XML schema languages: Document Type Definition (DTD), XML Schema Definition (XSD)
5.10. XML Style Sheets (XSLT)

Unit 6. PHP Framework **[8 Hrs.]**

6.1. Introduction
6.2. Features
6.3. Basic DB & Client-Side Validation
6.4. Session & Email System
6.5. Framework with method, Classes and Cookies

Unit 1. Web Server Concept

Introduction to Web Server



A web server is dedicated software that runs on the server-side. When any user requests their web browser to run any web page, the webserver places all the data materials together into an organized web page and forwards them back to the web browser with the help of the Internet.

This intercommunication of a web server with a web browser is done with the help of a protocol named HTTP (Hypertext Transfer Protocol). These stored web pages mostly use static content, containing HTML documents, images, style sheets, text files, etc. However, web servers can serve static as well as dynamic contents.

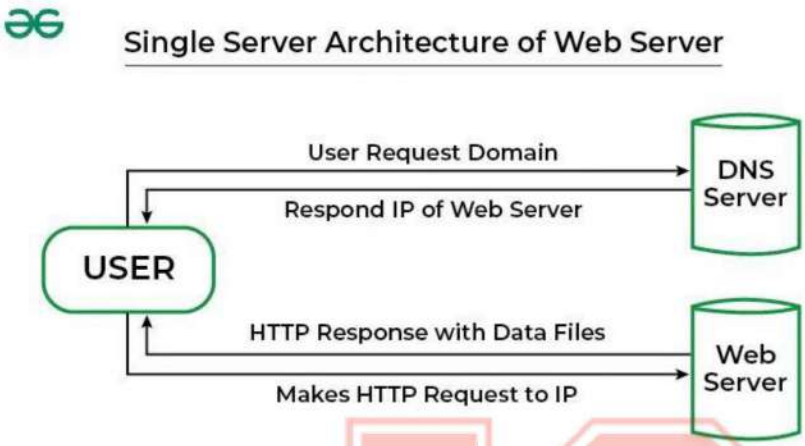
S.NO.	Static Web Servers	Dynamic Web Servers
1	Static web servers refer to the servers, which serve only the static content i.e., the content is fixed and being shown as it is.	Dynamic web servers refer to the servers where the content of the page can be updated and altered.
2	A static web server includes a computer and the HTTP (Hyper Text Transfer Protocol) software.	A dynamic web server also includes a computer with plenty of other software, unlike an application server and database model.
3	It is called static; the web pages content won't change unless the user manually changes it, and the server will deliver web files as is to the web browser.	It is called dynamic because the application server is used to update the web pages files at the server-side, and due to which, it can change on every call requested by the web browser.
4	Static web servers take less time to load the data.	The Dynamic web server can only produce the data when it is requested from the database. Therefore, it is time consuming and more complicated when compared to static web servers.

Architecture of web server:

Web server architecture refers to the structure and design of web servers, outlining how they handle incoming requests and deliver web content. There are two main approaches to web server architecture:

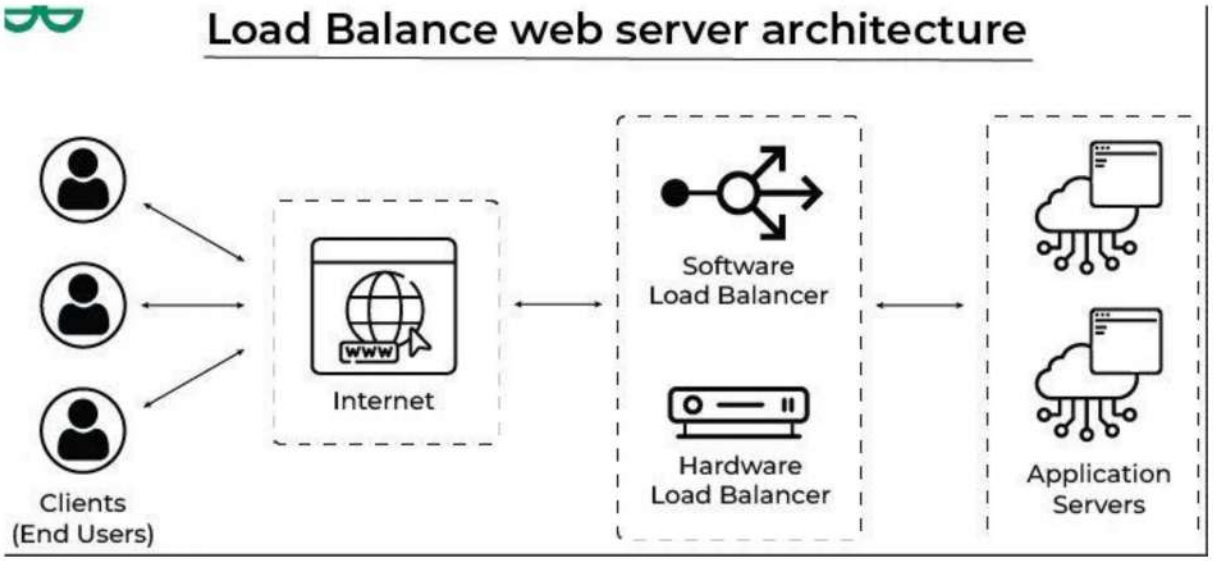
1. Single-Tier (Single Server) Architecture:

In a single-tier architecture, a single server is responsible for both processing requests and serving web content. This is suitable for small websites or applications with low traffic. However, it has limitations in terms of scalability and fault tolerance. If the server goes down, the entire service becomes unavailable.



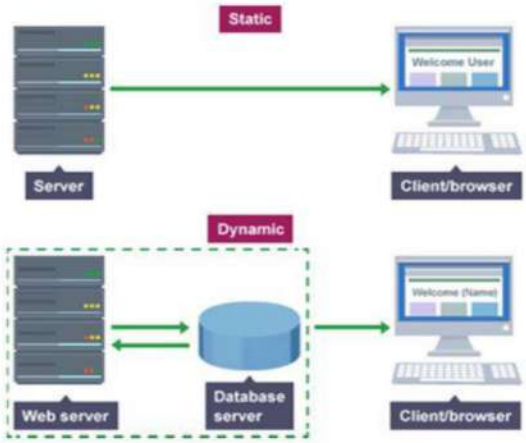
2. Multi-Tier (Load-Balanced) Architecture:

In a multi-tier architecture, multiple servers are used to distribute the workload and ensure high availability. This approach often involves load balancers that evenly distribute incoming requests across a cluster of web servers. Each server can serve web content independently, and if one server fails, the load balancer redirects traffic to healthy servers, ensuring uninterrupted service.



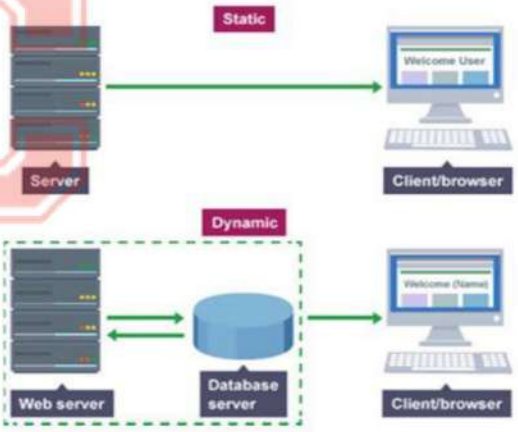
Static Website

Static Web Pages are very simple. It is written in languages such as HTML, JavaScript, CSS, etc. For static web pages when a server receives a request for a web page, then the server sends the response to the client without doing any additional process. And these web pages are seen through a web browser.



Dynamic website

Dynamic Web Pages are written in languages such as PHP, AJAX, ASP, ASP.NET, etc. In dynamic web pages, the Content of pages is different for different visitors. It takes more time to load than the static web page



Using control flow to control dynamic content generation:

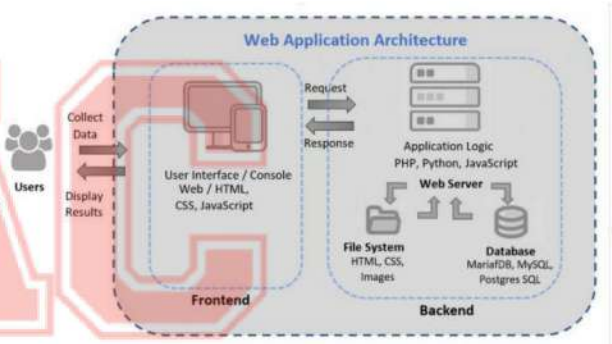
Generating Controls Using Control Flow

- Generating different types of control dynamically during runtime using control flows (if else, for loop, etc.).
- Advantages:
  - Less code
  - Easy manipulation of Controls

Concept of Architecting Web Application:

Web Application Architecture

Briefly, the web application architecture is a "skeleton" or layout that displays the interactions between application components, middleware systems, user interfaces, and databases. This kind of interaction allows a number of applications to work together simultaneously.



Unit 2. Review of Database: MySQL

Structured Query Language (SQL), which is a computer language for storing, manipulating, and retrieving data stored in relational database management systems (RDBMS). SQL was developed at IBM by Donald Chamberlin , Donald C. Messerli , and Raymond F. Boyce in the year 1970s.

MySQL is an open-source Relational Database Management System that stores data in a structured format using rows and columns. MYSQL language is easy to use as compared to other programming language like C, C++, Java, etc. By learning some basic commands we can work, create and interact with the Database.

How Does MySQL Work?

MySQL is open-source and user-friendly. It creates a database to store and manipulate the data. To perform various operations users make requests by typing specific statements. The server responds to the information from the user and Displays it on the user side.

Application of MySQL :

1. MySQL used in E-Commerce websites.
2. MySQL used in Data Warehousing.
3. MySQL is used in the Login Application.

Characteristics of MySQL:

1. MYSQL is free to use under the Community version of it. So we can download it from the MYSQL website and work on it freely.
2. MYSQL use multithreading which makes it Scalable. It can handle any amount of data. The default file size limit is 4 GB, but we can increase it according to our needs.
3. MYSQL is considered one of the fast databases. Its fastness is determined on the basis of a large number of benchmark tests.
4. MYSQL is very flexible because it supports a large number of embedded systems.
5. MYSQL is compatible to run on various operating systems such as Windows, macOS, Linux, etc.
6. MYSQL allows transactions to be rolled back, commit, and cash recovered.
7. It has a low memory leakage problem which increases its memory efficiency.
8. MYSQL version 8.0 provide dual password support, one is a current password and another is a secondary password. With the help of this we can create new passwords.
9. MYSQL provides the feature of Partitioning which improve the performance of large databases.
10. MYSQL consists of a Data Security layer that protects the data from the violator. Also, passwords are encrypted in MYSQL.
11. MYSQL follows Client-Server Architecture where the Client requests Commands and instructions and the Server will produce output as soon as the instruction is matched.

MySQL queries:

MySQL queries are commands written in the SQL (Structured Query Language) that are used to interact with a MySQL database. SQL is a standard language for interacting with relational databases, and MySQL is a popular relational database management system.

2.2.1. Create

The CREATE statement is used to create a new database, table, index, or view in MySQL.

Example:

```
CREATE DATABASE mydatabase;
CREATE TABLE mytable (
  id INT PRIMARY KEY,
  name VARCHAR(255),
  age INT
);
```

2.2.2. Insert

The INSERT statement is used to insert new records into a table.

Example:

```
INSERT INTO mytable (id, name, age) VALUES (1, 'John Doe', 25);
INSERT INTO mytable (id, name, age) VALUES (2, 'Jane Smith', 30);
```

2.2.3. Select

The SELECT statement is used to retrieve data from one or more tables.

Example:

```
SELECT * FROM mytable;
SELECT name, age FROM mytable WHERE age > 25;
```

2.2.4. Update

The UPDATE statement is used to modify existing records in a table.

Example:

```
UPDATE mytable SET age = 26 WHERE name = 'John Doe';
```

2.2.5. Delete

The DELETE statement is used to remove records from a table.

Example:

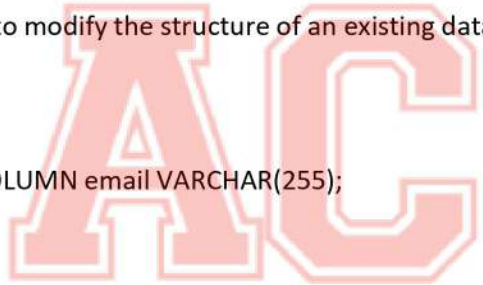
```
DELETE FROM mytable WHERE id = 2;
```

2.2.6. Alter

The ALTER statement is used to modify the structure of an existing database, table, or index.

Example:

```
ALTER TABLE mytable ADD COLUMN email VARCHAR(255);
```



Database normalization:

Database normalization is a process used in designing and organizing relational databases to reduce redundancy and improve data integrity. The normalization process involves breaking down large tables into smaller, more manageable tables and establishing relationships between them. The goal is to eliminate data anomalies, such as update, insertion, and deletion anomalies, and to ensure that the database structure is efficient and easy to maintain.

The normalization process is typically divided into different normal forms, each building on the requirements of the previous one. The commonly used normal forms are First Normal Form (1NF), Second Normal Form (2NF), Third Normal Form (3NF), Boyce-Codd Normal Form (BCNF), and Fourth Normal Form (4NF).

1. First Normal Form (1NF): A relation will be in 1NF if it contains an atomic values. It states that an attribute of a table cannot hold multiple values. It must hold only single valued attribute. First normal form disallows the multi-valued attributes.

Example: Relation "Student" is not in 1NF because of multivalued attribute "Language_known".

Student Table:

Name	Roll No	Branch	Language_known
Sita	21	IT	English, Nepali
Ram	22	Ag	English, Magar
Shyam	24	Vet	Nepali, Newari

Fig: Table1

After converting to 1NF

Name	Roll No	Branch	Language_known
Sita	21	IT	English
Sita	21	IT	Nepali
Ram	22	Ag	English
Ram	22	Ag	Magar
Shyam	24	Vet	Nepali
Shyam	24	Vet	Newari

Fig: Table2

2. Second Normal Form (2NF): In addition to meeting 1NF, a table in 2NF must have all non-prime attributes (attributes not part of the primary key) fully functionally dependent on the entire primary key. This eliminates partial dependencies.

A relation will be in 2NF if:

a. It is in 1NF.
b. No partial dependency.(It means primary key should not contain duplicate value.)

Example: Table2 is in 1NF but not in 2NF so lets convert it to 2NF.

Name	Roll No	Branch
Sita	21	IT
Ram	22	Ag
Shyam	24	Vet

Roll No	Language_known
21	English
21	Nepali
22	English
22	Magar
24	Nepali
24	Newari

3. **Third Normal Form (3NF):** In addition to meeting 2NF, a table in 3NF must have no transitive dependencies. This means that non-prime attributes should not depend on other non-prime attributes.

A table will be in 3NF:

a. If it is in 2NF.
b. There is no transitive dependency for non-prime attributes. **(Transitive dependency: If one nonprime key attributes depend on another non-primary key attributes then there is transitive dependency.)**

Example:

Name	Roll No	Branch	Fee
Sita	21	IT	30,000
Ram	22	Ag	20,000
Shyam	23	Vet	25,000
Hari	24	Foresty	18,000

There is transitive dependency between Branch and Fee.

After converting it to 3NF:

Name	Roll No	Branch
Sita	21	IT
Ram	22	Ag
Shyam	23	Vet
Hari	24	Foresty

Branch	Fee
IT	30,000
Ag	20,000
Vet	25,000
Foresty	18,000

4. **Boyce-Codd Normal Form (BCNF):** A stricter form of 3NF, ensuring that the database is free of certain types of anomalies related to functional dependencies. It is also known as 3.5 NF.

The table is in BCNF if:

a. It it is in 3NF.
b. And for any dependency A→ B, A should be a super key. (Super key must be unique)

Example:

Roll No	Std_name	Branch_id	Branch_name
1	Sita	121	IT
2	Ram	122	Ag
3	Shyam	123	Vet
4	Hari	121	IT

{Roll No} → {Std_name} {Branch_id} → {Branch_name}

Here, Roll No is super key but Branch_id is not a super key.

After converting it to BCNF:

Roll No	Std_name
1	Sita
2	Ram
3	Shyam
4	Hari

Branch_id	Branch_name
121	IT
122	Ag
123	Vet

5. **Fourth Normal Form (4NF):** Deals with multi-valued dependencies, ensuring that the table is free from certain types of redundancy involving multiple values.

A relation will be in 4NF if:

a. It is in BCNF.
b. Has no Multi valued dependency. (If two non-primary key depends on single primary key column than there is multi valued dependency.)

Example:

Std_id	Course	Hobby
1	IT	Cricket
2	Ag	Dancing
3	Vet	Singing

After converting it to 4NF:

Std_id	Course
1	IT
2	Ag
3	Vet

Std_id	Hobby
1	Cricket
2	Dancing
3	Singing

Unit 3. Server-Side Script: PHP

[12 Hrs.]

3.1. Introduction of PHP
3.2. Advantage of using PHP for web development
3.3. PHP Installation
3.4. PHP Syntax
3.5. Comments, Variable, Operators, Datatype, Strings, Keywords
3.6. Conditional Statements
3.7. Loop
3.8. Arrays
3.9. Functions
3.10. Passing variables with data between pages
    3.10.1. Get & Post Method
    3.10.2. Cookies
    3.10.3. Sessions
3.11. File Upload: Date, Include, File, File Upload
3.12. Accessing Form Elements, Form Validation
3.13. Exception and Error Handling

Introduction To PHP

➢The PHP Hypertext Preprocessor (PHP) is a programming language that allows web developers to create dynamic content that interacts with databases. PHP is basically used for developing web based software applications.

➢PHP started out as a small open source project that evolved as more and more people found out how useful it was. Rasmus Lerdorf created the first version of PHP way back in 1994.

Advantages of PHP:

1. PHP is a recursive acronym for "PHP: Hypertext Preprocessor".
2. PHP is a server side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites.
3. PHP is open source.
4. PHP is easy to learn.
5. PHP is platform independent(It supports all major OS like windows, Linux and Mac OS.)
6. It support different database systems like MySQL and Oracle.

Characteristics/features of PHP:

1. Performance: PHP script is executed much faster than those scripts which are written in other languages such as JSP and ASP.

2. Open Source: PHP source code and software are freely available on the web. You can develop all the versions of PHP according to your requirement without paying any cost.

3. Familiarity with syntax: PHP has easily understandable syntax. Programmers are comfortable coding with it.

4. Embedded: PHP code can be easily embedded within HTML tags and script.

5. **Platform Independent:** PHP is available for WINDOWS, MAC, LINUX & UNIX operating system.
6. **Database Support**: It supports different types of DB systems like MySQL and ORACLE.
7. **Web servers Support:** PHP is compatible with almost all local servers used today like Apache, Netscape, Microsoft IIS, etc.
8. **Security:** PHP is a secure language to develop the website. It consists of multiple layers of security to prevent threads and malicious attacks.

PHP installation:

In order to develop and run PHP Web pages three vital components need to be installed on your computer system.

1. Web Server: PHP will work with virtually all Web Server software, including Microsoft's Internet Information Server (IIS) but then most often used is freely available Apache Server.

2. Database: PHP will work with virtually all database software, including Oracle and Sybase but most commonly used is freely available MySQL database.

3. PHP Parser: In order to process PHP script instructions a parser must be installed to generate HTML output that can be sent to the Web Browser.

What is XAMPP?

XAMPP is one of the widely used cross-platform web servers, which helps developers to create and test their programs written like in php on their local webserver or computer. XAMPP is an abbreviation where X stands for Cross-Platform, A stands for Apache, M stands for MYSQL and the Ps stand for PHP and Perl, respectively. WAMP(for windows), LAMP(for Linux) and MAMP(for Mac OS) are alternative to XAMPP.

Basic Syntax of PHP:

```
<!DOCTYPE html>
<html>
   <body>
      <h1>Basic PHP Syntax</h1>
      <?php
         echo "Hello World!";
      ?>
   </body>
</html>
```

It gives the output:- Hello World.

PHP comments:

PHP comments can be used to describe any line of code so that other developer can understand the code easily.

Syntax for single-line comments:-

```
<!DOCTYPE html>
<html>
<body>
   <?php
      // This is a single-line comment
      # This is also a single-line comment
```

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

PHP Multi Line Comments:-

In PHP, we can comments multiple lines also. To do so, we need to enclose all lines within /* */. Let's see a simple example of PHP multiple line comment.

```
<!DOCTYPE html>
<html>
<body>
    <?php
        /*
            This is a multiple-lines comment block
            that spans over multiple
            lines
        */
    ?>
</body>
</html>
```

Creating (Declaring) PHP Variables:

In PHP, a variable starts with the $ sign, followed by the name of the variable:

```
<?php
$txt = "Hello world!"; //string
$x = 5; //integer
$y = 10.5; //float
?>
```

Rules to follow while creating variable:-

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

PHP Variable example: Declaring string, integer, and float

```
<?php
$str="hello string"; //string
$x=200; //integer
$y=44.6; //float
echo “String is: $str <br/>";
echo “Integer is: $x <br/>";
echo “Float is: $y <br/>";
?>
```

OUTPUT:-

String is hello world
Integer is 200
Float is 44.6

Php Operators:

PHP Operator is a symbol i.e used to perform operations on operands. In simple words, operators are used to perform operations on variables or values. For example:

➢$num=10+20; // here + is the operator and 10,20 are operands $num is variable.
➢PHP Operators can be categorized in following forms:
❑Arithmetic Operators(+,-,*,/,%,**)
❑Assignment Operators(=)
❑Bitwise Operators(&,|,^,~,<<,>>)
❑Comparison Operators(==, ===, !==, !=, <>, <, >, <=,>=,<=>)
❑Incrementing/Decrementing Operators(++ , --)
❑Logical Operators(and, or, xor, !, &&, ||)
❑String Operators(. , .=)
❑Array Operators(+, ==
, != ,
===
, !== , <>)
❑Type Operators(instanceof)
❑Execution Operators (``)

Data Types in PHP
Variables in PHP store values of different data types. Now let's discuss some data types that work with PHP:

1. String
2. Integer
3. Float
4. Boolean

1. String data type
A string is a data type which is represented with some text inside double quotes " ". A string can also hold numbers and special characters but they should be enclosed in the quotes. For example:

```
<?php
                $name = "Derek Emmmanel";
                echo "$name";

  echo "<br>";

                $price = “1234567”;
                Echo “$price”;
?>
```

From the code above, the variable values “Derek Emmanuel”; and "1234567" are of the string data type because they are enclosed in quotes.

2. Integer data type

Integers are whole numbers that have no decimal point. Integers can either be negative numbers (-34567) or positive numbers (34567). For example:

```
<?php
        $ad = 12345;
        var_dump($ad);
?>
```

The code above is an example of the integer data type.

3. Float data type

Floats are not whole numbers, but rather they are numbers with decimal points. Floats can also be negative decimal numbers (-34.567) or positive decimal numbers (34.567). For example:

```
<?php
        $fl = 34.567;
        var_dump($fl);
?>
```

4. Boolean data type

Boolean is a data type that represents two possible outcome, true or false. Booleans are used mostly when we are working with conditional statements like if, else, elseif, and swtich. For example:

```
$house = true;
$city = false;
```

PHP Strings:

PHP string is a sequence of characters i.e., used to store and manipulate text. PHP supports only 256-character set and so that it does not offer native Unicode support. There are 4 ways to specify a string literal in PHP.

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

Single Quoted

We can create a string in PHP by enclosing the text in a single-quote. It is the easiest way to specify string in PHP.

Example:

```
<?php
    $str='Hello text within single quote';
    echo $str;
?>
```

Output:

Hello text within single quote

Double Quoted:

In PHP, we can specify string through enclosing text within double quote also. But escape sequences and variables will be interpreted using double quote PHP strings.

Example 1

```
<?php
$str="Hello text within double quote";
echo $str;
?>
```

Output:

Hello text within double quote

Heredoc:

Heredoc syntax (<<<) is the third way to delimit strings. In Heredoc syntax, an identifier is provided after this heredoc <<< operator, and immediately a new line is started to write any text. To close the quotation, the string follows itself and then again that same identifier is provided. That closing identifier must begin from the new line without any whitespace or tab.

Naming Rules:

The identifier should follow the naming rule that it must contain only alphanumeric characters and underscores, and must start with an underscore or a non-digit character.

For Example

Valid Example

```
<?php
   $str = <<<Demo
It is a valid example
Demo;    //Valid code as whitespace or tab is not valid before closing identifier
echo $str;
?>
```

Output:

It is a valid example

Invalid Example

We cannot use any whitespace or tab before and after the identifier and semicolon, which means identifier must not be indented. The identifier must begin from the new line.

```
<?php
   $str = <<<Demo
It is Invalid example
     Demo;    //Invalid code as whitespace or tab is not valid before closing identifier
echo $str;
```

```
?>
```
This code will generate an error.

Output:

Parse error: syntax error, unexpected end of file in C:\xampp\htdocs\xampp\PMA\heredoc.php on line 7

Heredoc is similar to the double-quoted string, without the double quote, means that quote in a heredoc are not required. It can also print the variable's value.

Newdoc:

Newdoc is similar to the heredoc, but in newdoc parsing is not done. It is also identified with three less than symbols <<< followed by an identifier. But here identifier is enclosed in single-quote, e.g. <<<'EXP'. Newdoc follows the same rule as heredocs.

The difference between newdoc and heredoc is that - Newdoc is a single-quoted string whereas heredoc is a double-quoted string.

Note: Newdoc works as single quotes.

Example:

```
<?php
   $str = <<<'DEMO'
   Welcome to javaTpoint.
        Learn with newdoc example.
DEMO;
echo $str;
echo '</br>';

echo <<< 'Demo'    // Here we are not storing string content in variable str.
   Welcome to javaTpoint.
        Learn with newdoc example.
Demo;
?>
```

Output:

Welcome to javaTpoint. Learn with newdoc example.
Welcome to javaTpoint. Learn with newdoc example.

What are PHP Keywords?

PHP keywords are predefined, reserved words in PHP that are used to perform specific functions. These keywords are reserved by PHP and cannot be used as variable names, function names, or class names. PHP keywords are case-insensitive, meaning that they can be written in either upper or lower case letters.

Conditional Statements:

Conditional statements are used to perform different actions based on different conditions.
In PHP we have the following conditional statements:
1. IF STATEMENT: executes some code if one condition is true
2. IF...ELSE STATEMENT: executes some code if a condition is true and another code if that condition is false
3. IF... ELSE IF... ELSE statement: executes different codes for more than two conditions
4. Switch Statement: selects one of many blocks of code to be executed

1. If statement in PHP

If statement executes a statement or a group of statements if a specific condition is satisfied as per the user expectation.

Syntax:

```
if( condition )
{
                Execute statement(s) if condition is true;
}
```

Example:

```
<?php
$Pass_Mark=35;
$Student_Mark=70;
if ($Student_Mark>=$Pass_Mark)
{
echo “The Student is Eligible for the promotion”;
}
?>
```

OUTPUT: The Student is Eligible for the promotion

If else statement in PHP:

If statement executes a statement or a group of statement if a specific condition is satisfied by the user expectation. When the condition gets false (fail) the else block is executed.

Syntax:

```
If (condition)
{
                Execute statement (s)if condition is true;
}
Else
{
                Execute statement(s) if condition is false;
}
```

Example:

```
<?php
$Pass_Mark=35;
$Student_Mark=70;
```

```
if($Student_Mark>=$Pass_Mark){
echo "The Student is Eligible for the promotion"; // it will be printed
}
else{
echo "The Student is not Eligible for the promotion";
}?>
```

If elseif else statement in PHP:

If.. Elseif .. else statement is a combination of if-else statement. More than one statement can execute the condition based on user needs.

SYNTAX:

```
If( 1
st condition)
{
  Execute statement(s )if condition is true;
}
elseif( 2nd condition )
{
Execute statement(s) if 2nd condition is true;
}
Else
{
Execute statement(s) if both conditions are false;
}
```

Example:

```
<?php
$Pass_Mark=35;
$first_class=60;
$Student_Mark=70;
If($Student_Mark>=$first_class){
echo "The Student is eligible for the promotion with first class";
}
elseif ($Student_Mark>=$Pass_Mark){
echo "The Student is eligible for the promotion";
}
else{
echo "The Student is not eligible for the promotion";
}?>
```

Switch case in PHP:

The switch statement is used to perform different actions based on different conditions.

Syntax:

```
switch (n) {
  case label 1:
    code to be executed if n=label 1;
```

```
  break;
 case label 2:
  code to be executed if n=label 2;
 break;
 ...
 default:
  code to be executed if n is different from
all labels;
```

Example:

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

PHP loops:

Loops in PHP are used to execute the same block of code a specified number of times. PHP supports following four loop types.

➢for – loops through a block of code a specified number of times.

➢while – loops through a block of code if and as long as a specified condition is true.

➢do...while – loops through a block of code once, and then repeats the loop as long as a special condition is true.

➢foreach – loops through a block of code for each element in an array.

The for loop statement:

The for statement is used when you know how many times you want to execute a statement or a block of statements.

Syntax:

```
for(initialization; condition; increment){
// Code to be executed
}
```

for loop Example:

```
<?php
for ($x = 0; $x <= 4; $x++)
```

```
{
                echo "The number is: $x <br>";
}
?>
```

Output:

The number is: 0
The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5

PHP while Loop:

The while statement will loop through a block of code as long as the condition specified in the while statement evaluate to true.

Syntax:

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

Example of while loop:

```
<?php
$i = 1;
while($i <= 3)
{
$i++;
echo "The number is " . $i . "<br>";
}
?>
```

OUTPUT:

The number is 2
The number is 3
The number is 4

PHP do...while Loop:

The do..while loop is a variant of while loop, which evaluates the condition at the end of each loop iteration. With a do while loop the block of code executed once, and then the condition is evaluated, if the condition is true, the statement is repeated as long as the specified condition evaluated to is true.

Syntax:

```
do
{
                // Code to be executed
```

```
}
while(condition);
```

Example of do..while loop

```
<?php
$i = 1;
do{
$i++;
echo "The number is " . $i . "<br>";
}
while($i <= 3);
?>
```

OUTPUT:

The number is 2
The number is 3
The number is 4

PHP foreach Loop

The foreach loop is used to iterate over arrays.

Syntax:

```
foreach($array as $value)
{
                    // Code to be executed
}
```

And for associative array syntax is:

```
foreach($array as $key => $value){
                    // Code to be executed
}
```

Example of foreach loop

```
<?php $colors = array("Red", "Green", "Blue");
// Loop through colors array
foreach($colors as $value)
{
echo $value . "<br>";
}
?>
```

OUTPUT:

Red
Green
Blue

Foreach with associative array:

```
<?php
$person = array( "name" => “jaction", "email" =>
“jaction@mail.com", "age" => 23 );
// Loop through above array
foreach($person as $key => $value)
{
echo $key . " : " . $value . "<br>";
}
?>
```

Output:

```
name : jaction
email : jaction@mail.com
age : 23
```

PHP OOP: Classes and Objects in PHP

- OOP stands for Object-Oriented Programming.
- Procedural programming is about writing procedures or functions that perform operations on the data, while **object-oriented programming** is about creating objects that contain both data and functions.
- OOP is faster and easier to execute
- OOP provides a clear structure for the programs
- OOP helps to keep the PHP code DRY "Don't Repeat Yourself", and makes the code easier to maintain, modify and debug
- OOP makes it possible to create full reusable applications with less code and shorter development time

OOP Properties/features

- **Class** − This is a programmer-defined data type, which includes local functions as well as local data. You can think of a class as a template for making many instances of the same kind (or class) of object.
- **Object** − An individual instance of the data structure defined by a class. You define a class once and then make many objects that belong to it. Objects are also known as instance.
- **Member Variable** − These are the variables defined inside a class. This data will be invisible to the outside of the class and can be accessed via member functions. These variables are called attribute of the object once an object is created.
- **Member function** − These are the function defined inside a class and are used to access object data.
- **Inheritance** − When a class is defined by inheriting existing function of a parent class then it is called inheritance. Here child class will inherit all or few member functions and variables of a parent class.
- **Parent class** − A class that is inherited from by another class. This is also called a base class or super class.

- **Child Class** − A class that inherits from another class. This is also called a subclass or derived class.
- **Polymorphism** − This is an object oriented concept where same function can be used for different purposes. For example function name will remain same but it take different number of arguments and can do different task.
- **Overloading** − a type of polymorphism in which some or all of operators have different implementations depending on the types of their arguments. Similarly functions can also be overloaded with different implementation.
- **Data Abstraction** − Any representation of data in which the implementation details are hidden (abstracted).
- **Encapsulation** − refers to a concept where we encapsulate all the data and member functions together to form an object.
- **Constructor** − refers to a special type of function which will be called automatically whenever there is an object formation from a class.
- **Destructor** − refers to a special type of function which will be called automatically whenever an object is deleted or goes out of scope

What is PHP classes and Objects?

A **class** is a template or blueprint for objects, and an **object** is an instance of class. A class contains properties and methods inside it. A class is defined by using the **class** keyword, followed by the name of the class and a pair of curly braces ({}). All its properties and methods go inside the braces.

Syntax of class:

```
<?php
class Person
{
  // set of properties and methods...
}

?>
```

Example of Class

```
<?php
class Fruit
{
  // Properties
  public $name;
  public $color;

  // Methods
  function set_name($name) {
    $this->name = $name;
  }

  function get_name() {
    return $this->name;
  }
}
?>
```

we declared a class named Fruit consisting of two **properties ($name and $color)** and two **methods set_name() and get_name()** for setting and getting the $name property:

How to create objects from class?

Classes are nothing without objects! We can create multiple objects from a class. Each object has all the properties and methods defined in the class, but they will have different property values. Objects of a class is created using the **new** keyword. In the example below, $apple and $banana are instances of the class Fruit:

```php
<?php
class Fruit
{
 // Properties
 public $name;
 public $color;
 // Methods
 function set_name($name) {
  $this->name = $name;
 }
 function get_name() {
  return $this->name;
 }
}
$apple = new Fruit();
$banana = new Fruit();
$apple->set_name('Apple');
$banana->set_name('Banana');
echo $apple->get_name();
echo $banana->get_name();
?>
```

Here we created two objects: $apple and $banana from same class Fruit

OUTPUT:
Apple
Banana

Constructors in PHP

PHP constructor is a special method that is called automatically when an object is created. Normally constructors are used to initialize the properties of the Objects. PHP allows us to declare a constructor method for a class with the name **__construct()** as follows:

```php
<?php
class ClassName
{
        function __construct()
        {
                // implementation
        }
}
?>
```

__construct() is the constructor method.

Constructor types:

- **Default Constructor:**It has no parameters, but the values to the default constructor can be passed dynamically.
- **Parameterized Constructor:** It takes the parameters, and also you can pass different values to the data members.
- **Copy Constructor:** It accepts the address of the other objects as a parameter.

Example of Constructor

```
<?php
class Fruit {
  public $name;
  public $color;

  function __construct($name) {
    $this->name = $name;
  }
  function get_name() {
    return $this->name;
  }
}

$apple = new Fruit("Orange");
echo $apple->get_name();
?>
```

OUTPUT: Orange

In the example we created a constructor which accepts a value and stores in **$name**. We have passed a value "Orange" to the constructor when we created the object($apple) of the Fruit class.

Destructors in PHP

A destructor is called when the object is destructed or the script is stopped or exited. If we create a **__destruct()** function, PHP will automatically call this function at the end of the script.

constructor

Destructor

```
<?php
class Fruit {
  public $name;
  public $color;
  function __construct($name) {
    $this->name = $name;
  }
  function __destruct() {
    echo "The fruit is".$this.name;
  }
}
$apple = new Fruit("Apple");
?>
```

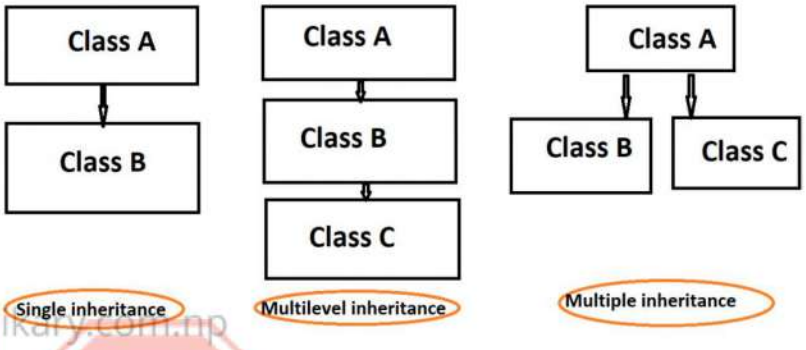
OUTPUT: The fruit is Apple.

Inheritance

In object oriented programming, Inheritance enables a class to use properties and methods of an existing class. The class which is inherited is called **Parent** class(or **super** class or **base** class) while the class which is inheriting other class is called as **Child** class(or **sub** class or **derived** class).

We use **extends** keyword to inherit class.

The child class will inherit all the public and protected properties and methods from the parent class. In addition, it can have its own properties and methods.



Types of inheritance:

- Single inheritance.
- Multi-level inheritance.
- Multiple inheritance.

Example:

```
<?php
class A                                   Parent class
  {
    function fun1()
    {
      echo “Hello World";
    }
  }
  class B extends A                       Child class
  {
    function fun2()
    {
      echo "SSSIT";
    }
  }
  $obj= new B();
  $obj->fun1();
?>
```

Output:
Hello world

In the example we created a parent class A and a child class B. class B is inheriting class A, so fun1() of class A is copied to class B. And we can call fun1() from object of class A.

More Example of inheritance:

```
<?php
class Fruit {
  public $name;
  public $color;
  public function __construct($name, $color) {
    $this->name = $name;
    $this->color = $color;
  }
  public function intro() {
    echo "The fruit is {$this->name} and the color is {$this->color}.";
  }
}

// Strawberry inherits Fruit
class Strawberry extends Fruit {
  public function message() {
    echo "Am I a fruit or a berry? ";
  }
}

$strawberry = new Strawberry("Strawberry", "red");
$strawberry->message();
$strawberry->intro();
?>
```

The Strawberry class inherits Fruit class.
This means that the Strawberry class can use the public $name and $color properties as well as the public __construct() and intro() methods from the Fruit class because of inheritance.
The Strawberry class also has its own method: message().

OUTPUT:
Am I a fruit or a berry? The fruit is Strawberry and the color is red.

PHP - Access Modifiers

Properties and methods can have access modifiers which control where they can be accessed.
There are three access modifiers:

- **public** - the property or method can be accessed from everywhere. This is default
- **protected** - the property or method can be accessed within the class and by classes derived from that class
- **private** - the property or method can ONLY be accessed within the class

Note:
The child class will inherit only the **public** and **protected** properties and methods from the parent class not **private** properties or method.

Example:

```
<?php
class Fruit {
  public $name;
  protected $color;
  private $weight;
}

$mango = new Fruit();
$mango->name = 'Mango'; // OK
$mango->color = 'Yellow'; // ERROR
$mango->weight = '300'; // ERROR
?>
```

In the above example we have added three different access modifiers to three properties (name, color, and weight). Here, if you try to set the name property it will work fine (because the name property is public, and can be accessed from everywhere). However, if you try to set the color or weight property it will result in a fatal error (because the color and weight property are protected and private):

Another Example:

```
<?php
class Fruit {
  public $name;
  public $color;
  public $weight;

  function set_name($n) {  // a public function (default)
    $this->name = $n;
  }
  protected function set_color($n) { // a protected function
    $this->color = $n;
  }
  private function set_weight($n) { // a private function
    $this->weight = $n;
  }
}

$mango = new Fruit();
$mango->set_name('Mango'); // OK
$mango->set_color('Yellow'); // ERROR
$mango->set_weight('300'); // ERROR
?>
```

In this example we have added access modifiers to two functions. Here, if you try to call the set_color() or the set_weight() function it will result in a fatal error (because the two functions are considered protected and private), even if all the properties are public:

What is Database?

- A **database** is an organized collection of data, so that it can be easily accessed and managed. There are many **databases management systems available** like MySQL, Sybase, Oracle, MongoDB, Informix, PostgreSQL, SQL Server, etc. With PHP, you can connect to and manipulate databases.
- MySQL is the most popular database system used with PHP.
- With PHP, you can connect to and manipulate databases.

Database Queries:
A query is a question or a request. We can query a database for specific information and have a record set returned.
Look at the following query (using standard SQL):
SELECT LastName FROM Employees
The query above selects all the data in the "LastName" column from the "Employees" table.

MySQL Queries:

A list of commonly used MySQL queries to create database, use database, create table, insert record, update record, delete record, select record, truncate table and drop table are given below.

1) MySQL Create Database
MySQL create database is used to create database. For example

```
create database db1;
```

2) MySQL Select/Use Database
MySQL use database is used to select database. For example

```
use db1;
```

3) MySQL Create Query
MySQL create query is used to create a table, view, procedure and function. For example:

```
CREATE TABLE customers
(id int(10),
 name varchar(50),
 city varchar(50),
 PRIMARY KEY (id )
);
```

4) MySQL Alter Query

MySQL alter query is used to add, modify, delete or drop columns of a table. Let's see a query to add column in customers table:

```
ALTER TABLE customers
ADD age varchar(50);
```

5) MySQL Insert Query

MySQL insert query is used to insert records into table. For example:

```
insert into customers values(10,'jaction',nepal');
```

6) MySQL Update Query

MySQL update query is used to update records of a table. For example:

```
update customers set name='bob', city='london' where id=101;
```

7) MySQL Delete Query

MySQL update query is used to delete records of a table from database. For example:

```
delete from customers where id=101;
```

8) MySQL Select Query

Select query is used to fetch records from database. For example:

```
SELECT * from customers;
```


9) MySQL Truncate Table Query

MySQL update query is used to truncate or remove records of a table. It doesn't remove structure. For example:

```
truncate table customers;
```

10) MySQL Drop Query

MySQL drop query is used to drop a table, view or database. It removes structure and data of a table if you drop table. For example:

```
drop table customers;
```

Connect to Database using PHP

Before we can access data in the MySQL database, we need to first connect to the MySQL server. There are three different ways to connect to MySQL server using PHP.

1) MySQLi (object-oriented)
2) MySQLi (procedural)
3) PDO(PHP Data Objects)

Connect to Database:MySQLi(Object Oriented)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname="mydb";  // database name

// Create connection
$conn = new mysqli($servername, $username, $password,$dbname);

// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}
echo "Connected successfully";
?>
```

Connect to Database:(MySQLi Procedural)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname="mydb";  // database name

// Create connection
$conn = mysqli_connect($servername, $username, $password,$dbname);

// Check connection
if (!$conn) {
  die("Connection failed: " . mysqli_connect_error());
}
echo "Connected successfully";
?>
```


Connect to Database:(PDO)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";

try {
  $conn = new PDO("mysql:host=$servername;dbname=myDB",$username, $password);
  // set the PDO error mode to exception
  $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  echo "Connected successfully";
}
catch(PDOException $e) {
echo "Connection failed: " . $e->getMessage();
}
?>
```

Close the connection

MySQLi Object-Oriented:

$conn->close();

MySQLi Procedural:

mysqli_close($conn);

PDO:

$conn = null;

PHP program to Create a MySQL Database

- The CREATE DATABASE statement is used to create a database in MySQL.
- The following examples create a database named "myDB":

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// Create connection
$conn = new mysqli($servername, $username, $password);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

// Create database
$sql = "CREATE DATABASE myDB";
if ($conn->query($sql) === TRUE)

{
  echo "Database created successfully";
} else {
  echo "Error creating database: " . $conn->error;
}

$conn->close();
?>
```

How to create a table in MySQL

The **CREATE TABLE** statement is used to create a table in MySQL.

We will create a table named "MyGuests", with five columns: "id", "firstname", "lastname", "email"

```
CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50)
)
```

NOT NULL - Each row must contain a value for that column, null values are not allowed

DEFAULT value - Set a default value that is added when no other value is passed

UNSIGNED - Used for number types, limits the stored data to positive numbers and zero

AUTO INCREMENT - MySQL automatically increases the value of the field by 1 each time a new record is added

PRIMARY KEY - Used to uniquely identify the rows in a table. The column with PRIMARY KEY setting is often an ID number, and is often used with AUTO_INCREMENT

PHP program to create a table

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

$sql = "CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
)";

if ($conn->query($sql) === TRUE) {
  echo "Table MyGuests created successfully";
} else {
  echo "Error creating table: " . $conn->error;
}

$conn->close();
?>
```

SQL query to create table

Connecting To multiple databases

```
<?php

$host = "localhost"; /* Host name */
$user = "root"; /* User */
$password = ""; /* Password */
$dbname1 = "tutorial_db1"; /* Database name 1 */
$dbname2 = "tutorial_db2"; /* Database name 2 */

$conn_1 = mysqli_connect($host, $user, $password,$dbname1);
// Check connection
if (!$conn_1) {
 die("Connection failed: " . mysqli_connect_error());
}

$conn_2 = mysqli_connect($host, $user, $password,$dbname2);
// Check connection
if (!$conn_2) {
 die("Connection failed: " . mysqli_connect_error());
}
```

In this program we have connected to multiple databases viz tutorial_db1 and tutorial._db2

Building in Error Checking

To check whether error occurred or not during connection or query execution there are different ways in php, the main error-checking function is die(). For Example in following query:

*mysqli_query($conn,"SELECT * FROM mutual_funds WHERE code = '$searchstring'") or **die**("Please check your query and try again.");*

If any error occurs in the execution of above query then die function will be executed with the message. There are other methods other than die() as below,

The **errno / mysqli_errno()** function returns the last error code for the most recent function call, if any. Also
The **error / mysqli_error()** function returns the last error description for the most recent function call, if any.

Syntax

Object oriented style:
$conn -> errno
Procedural style:
mysqli_errno(connection)

Object oriented style:
$conn -> error
Procedural style:
mysqli_error(*connection*)

Object oriented style:
$conn -> connect_error
Procedural style:
mysqli_connect_error(*connection*)

Fetch/Get Data From MySQL Database table

- Data can be fetched from MySQL tables by executing SQL **SELECT** statement through PHP function **mysqli_query**. We have several options to fetch data from MySQL.
- The most frequently used option is to use function **mysqli_fetch_array()**. This function returns row as an associative array, a numeric array, or both. This function returns FALSE if there are no more rows. PHP provides another function called **mysqli_fetch_assoc()** which also returns the row as an associative array.

Let's make a SQL query using the SELECT statement, after that we will execute this SQL query through passing it to the PHP mysqli_query() function to retrieve the table data.

Consider our **persons** table has the following records:

```
+----+------------+-----------+----------------------+
| id |first_name| last_name | email                |
+----+------------+-----------+----------------------+
| 1 | Peter     | Parker      | peterparker@mail.com
| 2 | John      | Rambo       | johnrambo@mail.com
| 3 | Clark     | Kent        | clarkkent@mail.com
| 4 | John      | Carter      | johncarter@mail.com
| 5 | Harry     | Potter      | harrypotter@mail.com
+----+------------+-----------+----------------------+
```

The PHP code in the following example selects all the data stored in the persons table (using the asterisk character (*) in place of column name selects all the data in the table) After getting data from database table we will display it in html table as below.

```
<?php
/* Attempt MySQL server connection. Assuming you are running MySQL
server with default setting (user 'root' with no password) */
$link = mysqli_connect("localhost", "root", "", "demo");

// Check connection
if($link === false){
   die("ERROR: Could not connect. " . mysqli_connect_error());
}

// Attempt select query execution
$sql = "SELECT * FROM persons";
if($result = mysqli_query($link, $sql)){
   if(mysqli_num_rows($result) > 0){
      echo "<table>";
         echo "<tr>";
            echo "<th>id</th>";
            echo "<th>first_name</th>";
            echo "<th>last_name</th>";
            echo "<th>email</th>";
         echo "</tr>";
```


```
while($row = mysqli_fetch_array($result)){
        echo "<tr>";
           echo "<td>" . $row['id'] . "</td>";
           echo "<td>" . $row['first_name'] . "</td>";
           echo "<td>" . $row['last_name'] . "</td>";
           echo "<td>" . $row['email'] . "</td>";
        echo "</tr>";
    }
    echo "</table>";
    // Free result set
    mysqli_free_result($result);
  } else{
    echo "No records matching your query were found.";
  }
} else{
  echo "ERROR: Could not able to execute $sql. " .
mysqli_error($link);
}

// Close connection
mysqli_close($link);
?>
```

In the example above, the data returned by the mysqli_query() function is stored in the $result variable. Each time mysqli_fetch_array() is invoked, it returns the next row from the result set as an array. The while loop is used to loops through all the rows in the result set. Finally the value of individual field can be accessed from the row either by passing the field index or field name to the $row variable like $row['id'] or $row[0], $row['first_name'] or $row[1], $row['last_name'] or $row[2], and $row['email'] or $row[3].

If you want to use the for loop you can obtain the loop counter value or the number of rows returned by the query by passing the $result variable to the mysqli_num_rows() function. This loop counter value determines how many times the loop should run.

How to get data from HTML form and insert it into database table After that Fetch data from database table and show result in HTML table?

Very often we will need to use a MySQL table to store data inside it and then output that data by using a PHP script. To display the table data it is best to use HTML, which upon filling in some data on the page invokes a PHP script which will update the MySQL table.To populate a new database table with data you will first need an HTML page which will collect that data from the user. The following HTML code that and passes the information to a PHP script:

```
<form action="insert.php" method="post">
Value1: <input type="text" name = "field1" /><br/>
Value2: <input type="text" name = "field2" /><br/>
Value3: <input type="text" name = "field3" /><br/>
Value4: <input type="text" name = "field4" /><br/>
Value5: <input type="text" name = "field5" /><br/>
<input type="submit" />
</form>
```

Creating html form

The above HTML code will show the user **5 text fields**, in which the user can input data and a **Submit** button. Upon clicking the **Submit** button the data submitted by the user will be passed to a script named **insert.php**.That script can have a syntax similar to the following:

```
<?php
$username = "your_username";
$password = "your_pass";
$database = "your_db";

$mysqli = new mysqli("localhost", $username, $password, $database);

$field1 = $_POST['field1'];
$field2 = $_POST['field2'];
$field3 = $_POST['field3'];
$field4 = $_POST['field4'];
$field5 = $_POST['field5'];

$query = "INSERT INTO table_name (col1, col2, col3, col4, col5)
        VALUES ('$field1','$field2','$field3','$field4','$field5')";

$mysqli->query($query);
$mysqli->close();
```

Inserting form data to database table

After the user submits the information, the **insert.php** script will save it in the database table. Then you may want to output that information, so that the user can see it on the page. Here we will display data in HTML table.

```
<html>
<body>
<?php
$username = "username";
$password = "password";
$database = "your_database";
$mysqli = new mysqli("localhost", $username, $password,
$database);
$query = "SELECT * FROM table_name";


echo '<table>
    <tr>
      <td> Value1</td>
      <td> Value2</td>
      <td> Value3</td>
      <td> Value4</td>
      <td> Value5</td>
    </tr>';
```


```
if ($result = $mysqli->query($query)) {
   while ($row = $result->fetch_assoc()) {
      $field1name = $row["col1"];
      $field2name = $row["col2"];
      $field3name = $row["col3"];
      $field4name = $row["col4"];
      $field5name = $row["col5"];

      echo '<tr>
            <td>'.$field1name.'</td>
            <td>'.$field2name.'</td>
            <td>'.$field3name.'</td>
            <td>'.$field4name.'</td>
            <td>'.$field5name.'</td>
         </tr>';
   }
   $result->free();
}
?>
</body>
</html>
```

Unit 5. AJAX and eXtensible Markup Language (XML)

5.1. Basic concept of AJAX

5.2. Features of XML

5.3. Structure of XML: Logical Structure, Physical Structure

5.4. Naming Rules

5.5. XML Elements

5.6. XML Attributes

5.7. Element Content Models: Element Sequences i.e., <!ELEMENT counting (first, second, third, fourth)>, Element Choices <!ELEMENT choose (this.one | that.one)>, Combined Sequences and Choices

5.8. Element Occurrence Indicators: -Discussion of Three Occurrence Indicators? (Question Mark) * (Asterisk Sign) + (Plus Sign)

5.9. XML schema languages: Document Type Definition (DTD), XML Schema Definition (XSD)

5.10. XML Style Sheets (XSLT)

Ajax:

Ajax is an acronym for Asynchronous Javascript and XML. It is used to communicate with the server without refreshing the web page and thus increasing the user experience and better performance.

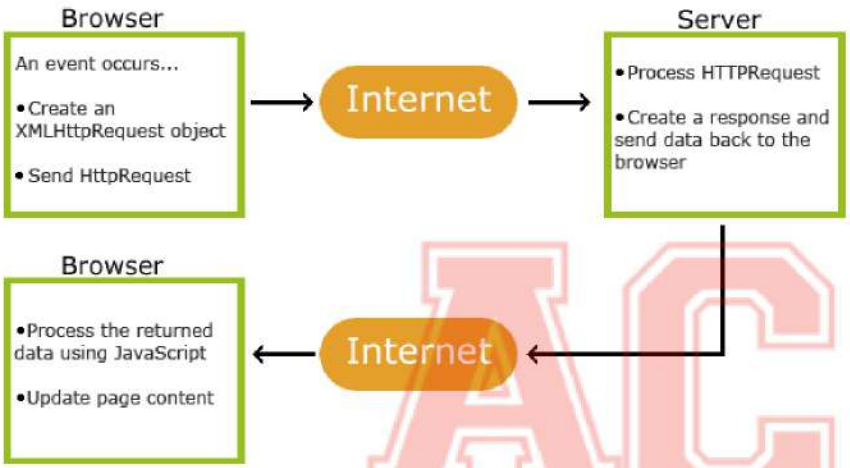
AJAX is not a programming language.

AJAX just uses a combination of:

- A browser built-in XMLHttpRequest object (to request data from a web server)
- JavaScript and HTML DOM (to display or use the data)

Examples of applications using AJAX: Google Maps, Gmail, Youtube, and Facebook tabs.

How AJAX Works



1. An event occurs in a web page (the page is loaded, a button is clicked)
2. An XMLHttpRequest object is created by JavaScript
3. The XMLHttpRequest object sends a request to a web server
4. The server processes the request
5. The server sends a response back to the web page
6. The response is read by JavaScript
7. Proper action (like page update) is performed by JavaScript

Important Features of Ajax:

- User Frendly
- It make web page faster.
- Independent of server technology.
- Increase the Performance of web page.

- Support for Live data binding
- Support for the Data View control
- Support for Client-side template rendering
- Rich and, responsive user interfaces
- Reduced consumption of server resources
- No need to pushing on a submit button and reloading the complete website.
- No need to reload the whole page only some part of page is reloaded which is required to reload.
- Apart from obtaining the XMLHTTP object, all processing is same for all browser types, because JavaScript is used.
- Using ajax develop faster and more interactive web applications.
- Not require to completely reload page due to this server use less bandwidth.

XMLHttpRequest Object:

It is an API in the form an object whose methods help in transfer of data between a web browser and a web server.

Syntax to Create a XMLHttpRequest Object :

```
var xhttp = new XMLHttpRequest();
```

What's the meaning of Asynchronous in AJAX?

Asynchronous means that the the Web Application could send and receive data from the Web Server without refreshing the page. This background process of sending and receiving data from the server along with updating different sections of a web page defines Asynchronous property/feature of AJAX.

The two major features of AJAX allow you to do the following:
•Make requests to the server without reloading the page
•Receive and work with data from the server

Properties of XMLHttpRequest object :
Readystate is a property of the XMLHttpRequest Object which holds the status of the XMLHttpRequest.
0: request not initialized
1: server connection established
2: request received
3: processing request
4: request finished and response is ready

```**onreadystatechange**``` is a property of the XMLHttpRequest Object which defines a function to be called when the readyState property changes.

```status``` is a property of the XMLHttpRequest Object which returns the status-number of a request

200: "OK"
403: "Forbidden"
404: "Not Found"

XMLHttpRequest Object Methods : To send a request to a Web Server, we use the open() and send() methods of the XMLHttpRequest object.

```
xhttp.open("GET", "content.txt", true);
xhttp.send();
```

AJAX example to change content of a web page :

```
<!DOCTYPE html>
<html>
<body>
	<div id="foo"> </div>
	<h2>The XMLHttpRequest Object</h2>
	<button type="button" onclick="changeContent()">Change Content</button>

	<script>
		function changeContent() {
			var xhttp = new XMLHttpRequest();
			xhttp.onreadystatechange = function() {
			if (this.readyState == 4 && this.status == 200) {
			document.getElementById("foo").innerHTML =
			this.responseText;
		}
		};
		xhttp.open("GET", "content.txt", true);
		xhttp.send();
		}
	</script>
</body>
</html>
```

XML:

Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. The design goals of XML focus on simplicity, generality, and usability across the Internet. It is a textual data format with strong support via Unicode for different human languages. Although the design of XML focuses on documents, the language is widely used for the representation of arbitrary data structures such as those used in web services.

1. XML stands for extensible Markup Language.
2. XML is a markup language like HTML.
3. XML is designed to store and transport data.
4. XML is designed to be self-descriptive.

Features of XML

Numerous aspects distinguish XML from other languages. The list of key XML features is shown below.

1. **Extensible and Human Readable:**
   Even if additional data is added, most XML applications will continue to function as intended.
2. **Overall Simplicity:**
   Data availability, platform modifications, data transit, and sharing are all made simpler by XML. Without losing data, XML makes it simpler to upgrade or extend to new operating systems, apps, or browsers. A wide range of "reading devices," including people, computers, voice assistants, news feeds, and more, can have access to data.
3. **Separates Data from HTML:**
   Using XML, data can be saved in a variety of XML files. Thus, you won't need to update HTML to make changes to the underlying data, allowing you to focus on using HTML/CSS for display and style.
4. **Allows XML Validation:**
   An XML document can be validated using a DTD or XML schema. By doing this, the XML document is guaranteed to be syntactically valid and any issues brought on by flawed XML are avoided.
5. **XML Supports Unicode:**
   Given that XML is compatible with Unicode, it may transmit virtually any information in any written human language.
6. **Used to Create New Languages:**
   Many new Internet languages have emerged as a result of XML. WSDL can be used to describe available web services. WAP and WML are utilized as portable device markup languages. The RSS languages are used for news feeds.

Difference between HTML and XML:

HTML	XML
It was written in 1993.	It was released in 1996.
HTML stands for Hyper Text Markup Language.	XML stands for Extensible Markup Language.
HTML is static in nature.	XML is dynamic in nature.
It is termed as a presentation language.	It is neither termed as a presentation nor a programming language.
HTML is a markup language.	XML provides a framework to define markup languages.
HTML can ignore small errors.	XML does not allow errors.
It has an extension of .html and .htm	It has an extension of .xml
HTML is not Case sensitive.	XML is Case sensitive.
HTML tags are predefined tags.	XML tags are user-defined tags.
There are limited number of tags in HTML.	XML tags are extensible.
HTML does not preserve white spaces.	White space can be preserved in XML.
HTML tags are used for displaying the data.	XML tags are used for describing the data not for displaying.
In HTML, closing tags are not necessary.	In XML, closing tags are necessary.
HTML is used to display the data.	XML is used to store data.
HTML does not carry data it just displays it.	XML carries the data to and from the database.

XML document structure:
XML document structure we distinguish between:
1. physical and
2. logical structure

Logical structure:
A document is divided into elements (one of them is the root), their attributes, and text nodes in elements, processing instructions, notations, and comments.

Physical structure:
Physically, documents are composed of a set of "entities" that are identified by unique names (except the document entity).

Physically, documents are composed of a set of "entities" that are identified by unique names (except the document entity). All documents begin with a "root" or document entity. All other entities are optional. What is important for the moment is that you understand that entities can be thought of as aliases for more complex functions. That is, a single entity name can take the place of a whole lot of text. As in any computer aliasing scheme, entity references cut down the amount of typing you have to do because anytime you need to reference that bunch of text, you simply use the alias name and the processor will expand out the contents of the alias for you.

As opposed to physical structure, XML documents have a logical structure as well. Logically, documents are composed of declarations, elements, comments, character references, and processing instructions, all of which are indicated in the document by explicit markup.

XML Syntax:
XML Documents Must Have a Root Element.
XML documents must contain one root element that is the parent of all other elements:

```
<root>
        <child>
                <subchild>.....</subchild>
        </child>
</root>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
        <to>Tove</to>
        <from>Jani</from>
        <heading>Reminder</heading>
        <body>Don't forget me this weekend!</body>
</note>
```

In this example <note> is the root element.

Xml Rules:

An XML document is called well-formed if it satisfies certain rules, specified by the W3C.

➢ **All XML Elements Must Have a Closing Tag:**

In XML, it is illegal to omit the closing tag. All elements must have a closing tag:

This is a paragraph.

➢ **XML Tags are Case Sensitive:**

XML tags are case sensitive. The tag <Letter> is different from the tag <letter>.

Opening and closing tags must be written with the same case:

<message>This is correct</message>

"Opening and closing tags" are often referred to as "Start and end tags". Use whatever you prefer. It is exactly the same thing.

➢ **XML Elements Must be Properly Nested:**

In HTML, you might see improperly nested elements like:

This text is bold and italic

In XML, all elements must be properly nested within each other:

This text is bold and italic

In the example above, "Properly nested" simply means that since the element is opened inside the element, it must be closed inside the element.

XML Attribute Values Must Always be Quoted.XML elements can have attributes in name/value pairs just like in HTML.

In XML, the attribute values must always be quoted:

```
<note date="12/11/2007">
        <to>Tove</to>
        <from>Jani</from>
</note>
```

Comments in XML:

The syntax for writing comments in XML is similar to that of HTML:

<!-- This is a comment -->

Two dashes in the middle of a comment are not allowed:

<!-- This is an invalid -- comment -->

XML Element:
An XML element is everything from (including) the element's start tag to (including) the element's end tag.
<price>29.99</price>

An element can contain:

- text
- attributes
- other elements
- or a mix of the above

Example:

```
<bookstore>
    <book category="children">
        <title>Harry Potter</title>
        <author>J K. Rowling</author>
        <year>2005</year>
        <price>29.99</price>
    </book>
    <book category="web">
        <title>Learning XML</title>
        <author>Erik T. Ray</author>
        <year>2003</year>
        <price>39.95</price>
    </book>
</bookstore>
```

In the example above:
<title>, <author>, <year>, and <price> have text content because they contain text (like 29.99).
<bookstore> and <book> have element contents, because they contain elements.

<book> has an attribute (category="children").

Empty XML Elements:
An element with no content is said to be empty.

In XML, you can indicate an empty element like this:
<element></element>

You can also use a so called self-closing tag:
<element />

XML Naming Rules:
XML elements must follow these naming rules:

- ➢ Element names are case-sensitive
- ➢ Element names must start with a letter or underscore
- ➢ Element names cannot start with the letters xml (or XML, or Xml, etc)

➢ Element names can contain letters, digits, hyphens, underscores, and periods
➢ Element names cannot contain spaces
➢ Any name can be used, no words are reserved (except xml).

XML Attributes:
XML elements can have attributes, just like HTML. Attributes are designed to contain data related to a specific element. Attributes provide extra information about elements. Attribute values must always be quoted. Either single or double quotes can be used.

For a person's gender, the <person> element can be written like this:
<person gender="female">

or like this:
<person gender='female'>

If the attribute value itself contains double quotes you can use single quotes, like in this example:
<gangster name='George "Shotgun" Ziegler'>

or you can use character entities:
<gangster name="George "Shotgun" Ziegler">

metadata (data about data) should be stored as attributes, and the data itself should be stored as elements.

Element Content Models: Element Sequences i.e., <!ELEMENT counting (first, second, third, fourth)>, Element Choices <!ELEMENT choose (this.one | that.one)>, Combined Sequences and Choices

Element content models in XML define the structure and relationships of elements within an XML document. They specify how different elements can be nested and combined. There are several types of content models, including element sequences, element choices, and mixed content.

Element Sequences:
In XML, you can define an element to contain a sequence of other elements using the <!ELEMENT> declaration. For instance:

<!ELEMENT counting (first, second, third, fourth)>
This declaration indicates that the <counting> element must contain child elements in the order of <first>, <second>, <third>, and <fourth>.

Element Choices:
Element choices allow you to specify that an element can contain one of several elements. This is done using the pipe (|) symbol. For example:

<!ELEMENT choose (this.one | that.one)>
This means that the <choose> element can contain either <this.one> or <that.one>.

Combined Sequences and Choices:
You can combine sequences and choices to create more complex content models. For example:
<!ELEMENT complex ((first, second) | (third, fourth))>

This means that the <complex> element can contain either a sequence of <first> and <second> or a sequence of <third> and <fourth>.

Element Occurrence Indicators: -Discussion of Three Occurrence Indicators?
(Question Mark) * (Asterisk Sign) + (Plus Sign)

Element occurrence:
Element occurrence indicators in XML define how many times an element can occur within its parent element. These indicators help specify the cardinality or occurrence constraints for elements.

1. Question Mark (?):
The question mark indicates that the element is optional and can occur zero or one time within its parent element.
Example:
<!ELEMENT optionalElement (#PCDATA)?>
In this example, optionalElement can occur zero or one time within its parent.

2. Asterisk Sign (*):
The asterisk sign indicates that the element can occur zero or more times within its parent element.
Example:
<!ELEMENT zeroOrMore (childElement*)>
In this example, zeroOrMore can contain any number (including zero) of childElement elements.

3. Plus Sign (+):
The plus sign indicates that the element must occur one or more times within its parent element.
Example:
<!ELEMENT oneOrMore (childElement+)>
In this example, oneOrMore must contain at least one childElement, but it can have more.

These occurrence indicators are used in conjunction with the <!ELEMENT> declaration in a Document Type Definition (DTD) or an XML Schema Definition (XSD) to define the structure and constraints of XML documents.

Example of Occurrence Indicators in a Sequence:
Consider the following example with a sequence of elements:

<!ELEMENT sequence (first, second, third, fourth)>
If you want to specify occurrence indicators for each element:

first: Occurs exactly once.
second: Can occur zero or more times.
third: Is optional (can occur zero or one time).
fourth: Must occur at least once.

```
<!ELEMENT sequence (
   first,
   second*,
   third?,
   fourth+
)>
```

This example illustrates how occurrence indicators are applied to each element in a sequence.

What is a DTD?
A DTD is a Document Type Definaition. A DTD defines the structure and the legal elements and attributes of an XML document.DTD can be of two types(internal and external).

Why to use a DTD?
With a DTD, independent groups of people can agree on a standard DTD for interchanging data.
An application can use a DTD to verify that XML data is valid.

An Internal DTD Declaration:
If the DTD is declared inside the XML file, it must be wrapped inside the <!DOCTYPE> definition.

An External DTD Declaration:
If the DTD is declared in an external file, the <!DOCTYPE> definition must contain a reference to the DTD file.

An Internal DTD Declaration:

XML document with an internal DTD

```
<?xml version="1.0"?>
        <!DOCTYPE note [
        <!ELEMENT note (to,from,heading,body)>
        <!ELEMENT to (#PCDATA)>
        <!ELEMENT from (#PCDATA)>
        <!ELEMENT heading (#PCDATA)>
        <!ELEMENT body (#PCDATA)>
]>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend</body>
</note>
```

➢ !DOCTYPE note defines that the root element of this document is note
➢ !ELEMENT note defines that the note element must contain four elements:
"to, from, heading, body"
➢ !ELEMENT to defines the to element to be of type "#PCDATA"

➢ !ELEMENT from defines the from element to be of type "#PCDATA"
➢ !ELEMENT heading defines the heading element to be of type "#PCDATA"
➢ !ELEMENT body defines the body element to be of type "#PCDATA"

An External DTD Declaration:
If the DTD is declared in an external file, the <!DOCTYPE> definition must contain a reference to the DTD file:
XML document with a reference to an external DTD

```
<?xml version="1.0"?>
<!DOCTYPE note SYSTEM "note.dtd">
<note>
        <to>Tove</to>
        <from>Jani</from>
        <heading>Reminder</heading>
        <body>Don't forget me this weekend!</body>
</note>
```

And here is the file "note.dtd", which contains the DTD:

```
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
```

XSL (Extensible Style Sheet Language):
In computing, the term Extensible Stylesheet Language (XSL) is used to refer to a family of languages used to transform and render XML documents. XSL (eXtensible Stylesheet Language) is a styling language for XML. XSL is Style Sheets for XML
➢XML does not use predefined tags, and therefore the meaning of each tag is not well understood.
➢A <table> element could indicate an HTML table, a piece of furniture, or something else - and browsers do not know how to display it!
➢So, XSL describes how the XML elements should be displayed.

XSL consists of four parts:
•XSLT - a language for transforming XML documents
•XPath - a language for navigating in XML documents
•XSL-FO - a language for formatting XML documents (discontinued in 2013)
•XQuery - a language for querying XML documents

What is XSLT?
XSLT is used to transform an XML document into another XML document, or another type of document that is recognized by a browser, like HTML and XHTML. Normally XSLT does this by transforming each XML element into an (X)HTML element.
With XSLT you can add/remove elements and attributes to or from the output file. You can also rearrange and sort elements, perform tests and make decisions about which elements

to hide and display, and a lot more.
A common way to describe the transformation process is to say that XSLT transforms an **XML source-tree into an XML result-tree.**

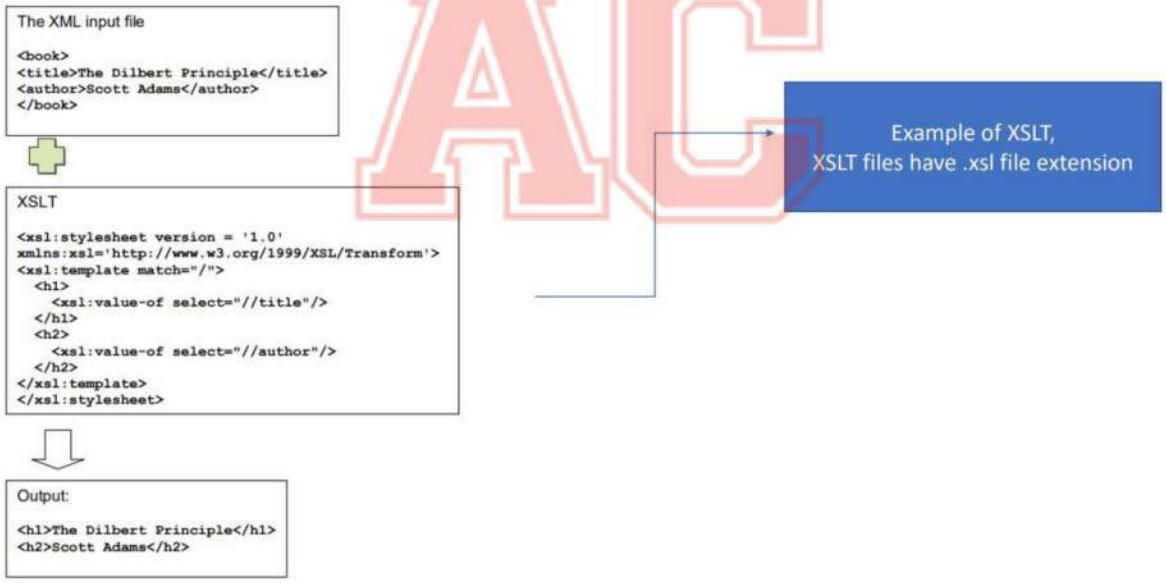
What is XPath?
XPath is a major element in the XSLT standard.
XPath can be used to navigate through elements and attributes in an XML document. Eg: /bookstore/book[1]

What is XQuery?
XQuery is to XML what SQL is to databases.
XQuery is designed to query XML data.

XQuery Example:

```
for $x in doc("books.xml")/bookstore/book
where $x/price>30
order by $x/title
return $x/title
```

The basic thing about XSLT is that a XML document can be processed with an XSLT (Extensible Stylesheet Language Transformations) as transforming instructions. The Output can be any text based format such as HTML, SGML, rich text, LaTeX, plain text, and of course XML.



XML Schema Definition (XSD):
The XML Schema is used to formally describe the grammar of XML documents, just like a DTD. In this way, a XML document can be a valid XML document.
An **XML Schema Definition** is an XML-based alternative to DTD.
It defines the elements, attributes and data types of an XML document.
It supports Namespaces.
The XML Schema aims to define the structure of XML documents in order to validate them.

Well-formed and Valid XML Documents:

An XML document is called well-formed if its syntax is correct.

An XML document is called valid if it is well-formed and it is validated against a XML Schema.

Example of XML document with XML Schema:

```
<xs:element name="book">
<xs:complexType>
  <xs:sequence>
    <xs:element name="title" type="xs:string"/>
    <xs:element name="author" type="xs:string"/>
    <xs:element name="publisher" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
</xs:element>
```

The Schema above is interpreted in this way:

- <xs:element name="book"> defines the element called "note"
- <xs:complexType> the "note" element is a complex type
- <xs:sequence> the complex type is a sequence of elements
- <xs:element name="title" type="xs:string"> the element to is of type string
- <xs:element name="author" type="xs:string"> the element from is of type string
- <xs:element name="publisher" type="xs:string"> the element heading is of type string

Introduction: PHP Framework

PHP (Hypertext Preprocessor) is a server-side scripting language widely used for web development. One of the key factors contributing to PHP's popularity is its ability to seamlessly integrate with various web frameworks. These frameworks provide developers with a structured and efficient way to build robust, scalable, and maintainable web applications. In this article, we'll delve into the world of PHP frameworks, exploring their significance, benefits, and some popular choices.

Understanding PHP Frameworks: A PHP framework is essentially a collection of pre-built modules, libraries, and tools that streamline the development process. It enforces a structured approach to coding, promoting code organization, reusability, and maintainability. Frameworks handle common tasks such as database interactions, URL routing, and input validation, allowing developers to focus on building unique features rather than reinventing the wheel.

Key Benefits of Using PHP Frameworks:

1. **Rapid Development:** PHP frameworks come with built-in functionalities and conventions that accelerate the development process. Developers can leverage ready-made components and follow established coding patterns, reducing the time required for project completion.
2. **Code Organization:** Frameworks enforce a modular and organized code structure. The separation of concerns, such as Model-View-Controller (MVC) architecture, makes it easier to manage code, debug issues, and collaborate with team members.
3. **Security Features:** Many PHP frameworks incorporate security measures by default, helping developers guard against common web vulnerabilities. Features like input validation, CSRF protection, and secure session management contribute to a more secure development environment.
4. **Scalability:** Frameworks provide a foundation for scalable applications. As projects grow, the modular structure and architectural patterns enable developers to expand functionality without sacrificing performance or introducing unnecessary complexity.
5. **Community Support:** Popular PHP frameworks have large and active communities. This means a wealth of documentation, tutorials, and community-driven plugins or extensions that can enhance development efficiency and resolve issues quickly.

Popular PHP Frameworks:

1. **Laravel:** Laravel has gained immense popularity for its elegant syntax, expressive features, and a vibrant ecosystem. It follows the MVC pattern, making it easy for developers to create modern and scalable web applications.

2. **Symfony:** Symfony is a robust and highly customizable framework that provides reusable PHP components. It's suitable for both small and large projects, offering flexibility and scalability. Symfony's emphasis on best practices and modularity has made it a favorite among enterprise-level applications.

3. **CodeIgniter:** CodeIgniter is known for its simplicity and lightweight nature. It's an excellent choice for developers who prefer a framework with minimal configuration but still want the benefits of a solid MVC architecture.

4. **Zend Framework (Laminas Project):** Formerly known as Zend Framework, the Laminas Project is a collection of professional, loosely coupled components. It allows developers to use individual components without necessarily adopting the full framework, providing a high level of flexibility.

PHP frameworks play a pivotal role in modern web development by offering a structured and efficient approach to building applications. Whether you're a seasoned developer or a newcomer to PHP, exploring and utilizing frameworks can significantly enhance your productivity, code quality, and overall development experience. As technology evolves, so too will PHP frameworks, ensuring that developers have the tools needed to create powerful and scalable web applications.

Laravel

Laravel, a robust and elegant PHP web framework, has emerged as a game-changer in the world of web development. Developed by Taylor Otwell, Laravel provides developers with a powerful and expressive syntax, along with a plethora of features and tools that simplify the often-complex process of building modern web applications. In this article, we will delve into the key aspects of Laravel, exploring its features, advantages, and why it has become the framework of choice for many developers.

MVC, or Model-View-Controller, is a software architectural pattern that separates an application into three interconnected components: the Model, the View, and the Controller. Laravel, a PHP web framework, is built upon the MVC pattern, providing developers with a structured and organized approach to building web applications. Let's explore how MVC is implemented in Laravel:

1. **Model:**

In the MVC pattern, the Model represents the application's data and business logic. In Laravel, models are used to interact with the database. Each model typically corresponds to a table in the database, and Eloquent, Laravel's ORM (Object-Relational Mapping) system, makes working with databases and models straightforward.

2. **View:**

The View is responsible for presenting the application's data to the user. In Laravel, views are typically created using the Blade templating engine. Blade provides a clean and expressive syntax for writing templates and allows for the inclusion of PHP code within the templates.

3. **Controller:**

The Controller acts as an intermediary between the Model and the View. It receives user input, processes it (interacting with the Model if necessary), and returns the appropriate output (rendering the View). In Laravel, controllers are responsible for handling HTTP requests and orchestrating the flow of the application.

Authentication and Authorization

Authentication and authorization are essential components of web application security, ensuring that users can access the right resources and perform appropriate actions. Laravel, being a robust PHP web framework, provides a comprehensive system for handling authentication and authorization.

Authentication in Laravel:

Authentication is the process of verifying the identity of a user. Laravel simplifies user authentication through its built-in authentication system, which includes user registration, login, and password reset functionalities.

1. **User Model:**

In Laravel, a User model typically represents users in the application. The framework includes a pre-built User model, and developers can customize it to fit their needs or create a new model if necessary.

2. **Registration:**

Laravel provides ready-made controllers and views for user registration. Developers can enable user registration with a single Artisan command:

3. **Login and Logout:**

Logging in users is simplified with the **Auth** facade. Developers can use the **attempt** method to validate user credentials and log in the user.

4. **Middleware:**

Laravel provides middleware for protecting routes from unauthenticated access. Developers can apply the **auth** middleware to routes, ensuring that only authenticated users can access the associated resources.

Authorization in Laravel:

Authorization is the process of determining what actions a user is allowed to perform. Laravel uses Gates and Policies to handle authorization.

1. **Gates:**

Gates are closures that determine if a user is authorized to perform a given action. Developers can define gates in the **AuthServiceProvider**:

2. **Policies:**

Policies are classes that organize authorization logic for a particular model. Laravel provides an Artisan command to generate a policy.

3. **Middleware for Authorization:**

Similar to authentication middleware, Laravel offers middleware for authorization. Developers can use the **can** middleware to check if a user is authorized to access a resource.

Laravel provides a robust and flexible system for authentication and authorization. The built-in features, combined with middleware, gates, and policies, empower developers to create secure web applications with fine-grained control over user access and actions. Whether it's protecting routes, handling user registration, or defining complex authorization logic, Laravel's authentication and authorization components contribute to the framework's reputation for building secure and scalable web applications.

Email system in PHP

Creating a basic email system in PHP involves using the mail function, which is a built-in function in PHP. However, keep in mind that the mail function relies on the server's mail configuration. Here's a simple example to send an email using PHP:

PHP mail() function is used to send email in php. We can send text message, html message and attachment with message using PHP mail() function.

Synatax,

bool mail(String $to ,string $subject, $string subject, $string message…)

```
<?php
// Set the recipient email address
$to = "recipient@gmail.com";
// Set the subject of the email
$subject = "Test Email";
// Set the message body
$message = "This is a test email sent from PHP.";
// Set additional headers if needed
$headers = "From: sender@example.com\r\n";
// Use the mail() function to send the email
$mailSent = mail($to, $subject, $message, $headers);

// Check if the mail was sent successfully
if ($mailSent = = true) {
    echo "Email sent successfully!";
} else {
    echo "Failed to send email.";
}
?>
```

Client-side validation in PHP

PHP is a server-side scripting language, and client-side validation is typically performed using JavaScript. JavaScript allows you to validate user input on the client side before

submitting the form data to the server. This can help improve the user experience by providing instant feedback and reducing the load on the server.

Here's a simple example of client-side validation using JavaScript in conjunction with a PHP form:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Form with Client-Side Validation</title>
    <script>
        function validateForm() {
            var name = document.forms["myForm"]["name"].value;
            var email = document.forms["myForm"]["email"].value;

            if (name === "") {
                alert("Name must be filled out");
                return false;
            }
            // Simple email validation (you might want to use a more robust validation)
            if (email === "" || !email.includes("@")) {
                alert("Invalid email address");
                return false;
            }

            // If all validations pass, the form will be submitted
            return true;
```

```
    }
  </script>
</head>
<body>
  <form name="myForm" action="process_form.php" onsubmit="return validateForm()" method="post">
    <label for="name">Name:</label>
    <input type="text" id="name" name="name">
    <br>
    <label for="email">Email:</label>
    <input type="text" id="email" name="email">
    <br>
    <input type="submit" value="Submit">
  </form>
</body>
</html>
```

==========

Framework with method, class in PHP

PHP has several frameworks that facilitate the organization of code through the use of classes and methods. Here are three popular PHP frameworks that extensively use classes and methods:

1. Laravel:
    - Description: Laravel is a powerful and elegant PHP framework that follows the Model-View-Controller (MVC) architectural pattern. It provides a clean and expressive syntax and includes an ORM (Eloquent), routing, middleware, and more.

Example Class and Method:

```
// Example Controller in Laravel

namespace App\Http\Controllers;
```

```
use App\Http\Controllers\Controller;

class UserController extends Controller {
    public function index () {
        // Your code here
    }
}
```

==========

```
<?php
// Define a simple class
class MyClass {
    // Property (attribute) of the class
    public $name;

    // Constructor method (automatically called when an object is created)
    public function __construct($name) {
        $this->name = $name;
    }

    // Method to display a greeting
    public function greet() {
        echo "Hello, " . $this->name . "!";
    }
}
// Create an instance (object) of the class
$obj = new MyClass("John");
```

```
// Access the property and call the method
echo $obj->name;  // Output: John
$obj->greet();    // Output: Hello, John!
?>
```

==========

Framework cookies in php

When dealing with cookies in PHP, you might be working within a web development framework that provides abstractions or utilities to make handling cookies more convenient. However, even without a specific framework, PHP itself provides functions to work with cookies.

Here's a basic example of how to set and retrieve cookies in PHP without any specific framework:

Setting a Cookie:

```
<?php
// Set a cookie named 'user' with the value 'John' that expires in 1 hour
setcookie('user', 'John', time() + 3600, '/');
?>
```

Retrieving a Cookie:

```
<?php
// Check if the 'user' cookie is set
if (isset($_COOKIE['user'])) {
    // Retrieve and use the value of the 'user' cookie
    $username = $_COOKIE['user'];
    echo "Welcome back, $username!";
} else {
    echo "Welcome, guest!";
}
?>
```