Module 3

1. **What is RIA? Explain Characteristics of RIA.**

   Rich Internet applications (RIA) are Web-based applications that have some characteristics of graphical desktop applications.
   Built with powerful development tools, RIAs can run faster and be more engaging. They can offer users a better visual experience and more interactivity than traditional browser applications that use only HTML and HTTP.
   The term "rich Internet application" was introduced in a white paper of March 2002 by Macromedia (now merged into Adobe), though the concept had existed for a number of years earlier under names such as:
   Remote Scripting, X Internet, Rich (Web) clients,Rich Web application
   Rich Internet Applications use a Rich Client deployment model rather than a thin-client-server model.

   Characteristics:

   - Rich user experience (user interface and user interaction)
   - Have features and functionality of traditional desktop applications (i.e., highly interactive GUI).
   - Improved responsiveness
   - Broad reach
   - Platform independence
   - Low deployment costs
   - Run in a web browser or run locally in a secure environment called a sandbox
   - Do not require software installation
   - (The RIA runtime environment still needs to be installed once per system).

   RIA: Pros (In General)

   - Installation is not required
   - Easy to upgrade
   - Easily made available over internet/intranet
   - Richer UI
   - More responsive UI
   - Client/Server Balance
   - Asynchronous communication
   - Network efficiency

   RIA: Cons (In General)

   - Loss of visibility to search engines

- Proprietary (as opposed to Open Standard)
- Loss of integrity (RIAs typically don't mix well with HTML)
- Software development complications (What to cache or not to cache at client's computer?)
- RIA architecture breaks the Web page paradigm

2. **Differentiate between traditional Model and Ajax Model of Web Applications.**

Traditional Web Applications

Figure 15.1 presents the typical interactions between the client and the server in a tradi-tional web application, such as one that uses a user registration form. First, the user fills in the form's fields, then submits the form (Fig. 15.1, *Step 1*). The browser generates a re-quest to the server, which receives the request and processes it (*Step 2*). The server generates and sends a response containing the exact page that the browser will render (*Step 3*), which causes the browser to load the new page (*Step 4*) and temporarily makes the browser win-dow blank. Note that the client *waits* for the server to respond and *reloads the entire page* with the data from the response (*Step 4*). While such a **synchronous request** is being pro-

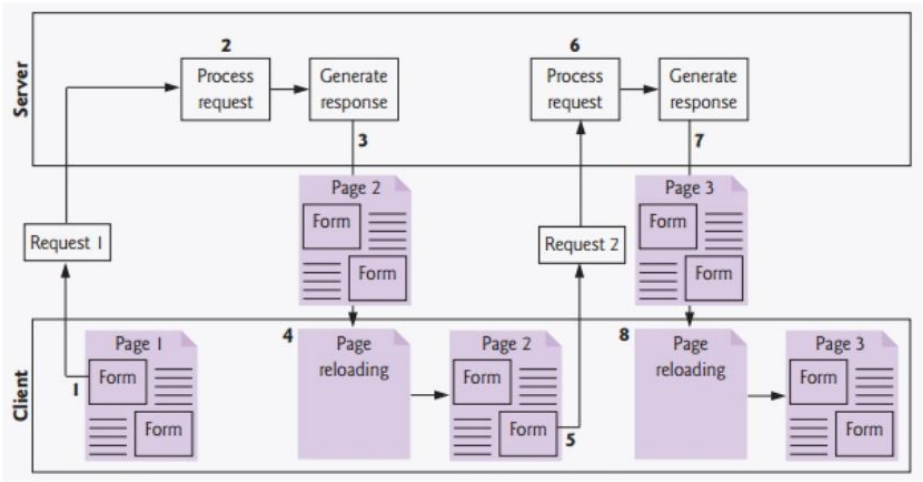


Fig. 15.1 | Classic web application reloading the page for every user interaction.

cessed on the server, the user cannot interact with the client web page. Frequent long pe-riods of waiting, due perhaps to Internet congestion, have led some users to refer to the World Wide Web as the "World Wide Wait." If the user interacts with and submits an-other form, the process begins again (*Steps 5–8*).

This model was originally designed for a web of hypertext documents—what some people call the "brochure web." As the web evolved into a full-scale applications platform, the model shown in Fig. 15.1 yielded "choppy" application performance. Every full-page refresh required users to re-establish their understanding of the full-page contents. Users began to demand a model that would yield the responsive feel of desktop applications.

Ajax Web Applications

Ajax applications add a layer between the client and the server to manage communication between the two (Fig. 15.2). When the user interacts with the page, the client creates an `XMLHttpRequest` object to manage a request (*Step 1*). The `XMLHttpRequest` object sends

the request to the server (*Step 2*) and awaits the response. The requests are **asynchronous**, so the user can continue interacting with the application on the client-side while the server processes the earlier request concurrently. Other user interactions could result in addition-al requests to the server (*Steps 3* and *4*). Once the server responds to the original request (*Step 5*), the `XMLHttpRequest` object that issued the request calls a client-side function to process the data returned by the server. This function—known as a **callback function**— uses **partial page updates** (*Step 6*) to display the data in the existing web page *without re-loading the entire page*. At the same time, the server may be responding to the second re-quest (*Step 7*) and the client-side may be starting to do another partial page update (*Step 8*). The callback function updates only a designated part of the page. Such partial page up-dates help make web applications more responsive, making them feel more like desktop applications. The web application does not load a new page while the user interacts with it.

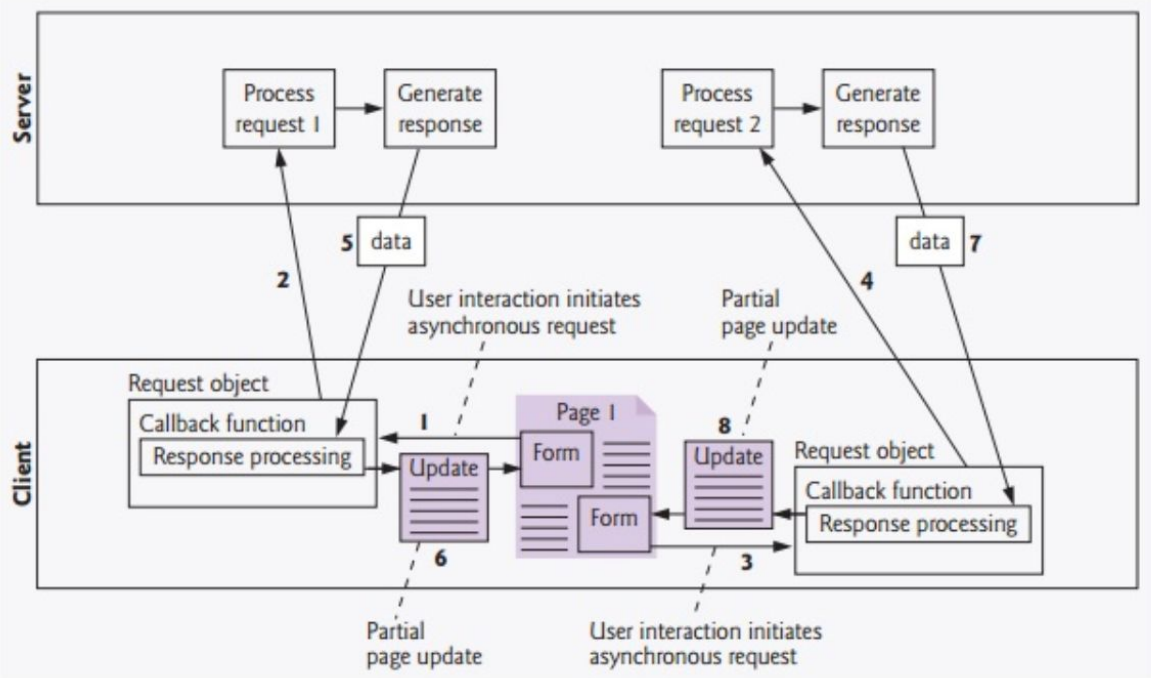


Fig. 15.2 | Ajax-enabled web application interacting with the server asynchronously.

3. **What is the XMLHttpRequest Object? Explain the common XMLHttpRequest Object properties and methods**

The XMLHttpRequest object is the key to AJAX. It has been available ever since Internet Explorer 5.5 was released in July 2000, but was not fully discovered until AJAX and Web 2.0 in 2005 became popular.

XMLHttpRequest (XHR) is an API that can be used by JavaScript, JScript, VBScript, and other web browser scripting languages to transfer and manipulate XML data to and from a webserver using HTTP, establishing an independent connection channel between a webpage's Client-Side and Server-Side.

The data returned from XMLHttpRequest calls will often be provided by back-end databases. Besides XML, XMLHttpRequest can be used to fetch data in other formats, e.g. JSON or even plain text.

You already have seen a couple of examples on how to create an XMLHttpRequest object.

Listed below are some of the methods and properties that you have to get familiar with.

XMLHttpRequest Object Methods

Method	Description
new XMLHttpRequest()	Creates a new XMLHttpRequest object
abort()	Cancels the current request
getAllResponseHeaders()	Returns header information
getResponseHeader()	Returns specific header information
open(*method,url,async,user,psw*)	Specifies the request *method*: the request type GET or POST *url*: the file location *async*: true (asynchronous) or false (synchronous) *user*: optional user name *psw*: optional password
send()	Sends the request to the server Used for GET requests
send(*string*)	Sends the request to the server. Used for POST requests

XMLHttpRequest Object Properties

Property	Description
onreadystatechange	Defines a function to be called when the readyState property changes
readyState	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
responseText	Returns the response data as a string
responseXML	Returns the response data as XML data
status	Returns the status-number of a request 200: "OK" 403: "Forbidden" 404: "Not Found" For a complete list go to the Http Messages Reference
statusText	Returns the status-text (e.g. "OK" or "Not Found")

4. **What is JSON? Compare JSON with XML.**
   https://www.tutorialspoint.com/json/
   https://www.geeksforgeeks.org/difference-between-json-and-xml/
   https://freefeast.info/tutorials-for-beginners/dotnet-tutorials/difference-between-json-and-xml-json-vs-xml/

JSON or JavaScript Object Notation is a lightweight text-based open standard designed for human-readable data interchange. Conventions used by JSON are known to programmers, which include C, C++, Java, Python, Perl, etc.

- JSON stands for JavaScript Object Notation.
- The format was specified by Douglas Crockford.
- It was designed for human-readable data interchange.
- It has been extended from the JavaScript scripting language.
- The filename extension is **.json**.
- JSON Internet Media type is **application/json**.
- The Uniform Type Identifier is public.json.

Uses of JSON

- It is used while writing JavaScript based applications that includes browser extensions and websites.
- JSON format is used for serializing and transmitting structured data over network connection.
- It is primarily used to transmit data between a server and web applications.
- Web services and APIs use JSON format to provide public data.
- It can be used with modern programming languages.

Characteristics of JSON

- JSON is easy to read and write.
- It is a lightweight text-based interchange format.
- JSON is language independent.

Simple Example in JSON

The following example shows how to use JSON to store information related to books based on their topic and edition.

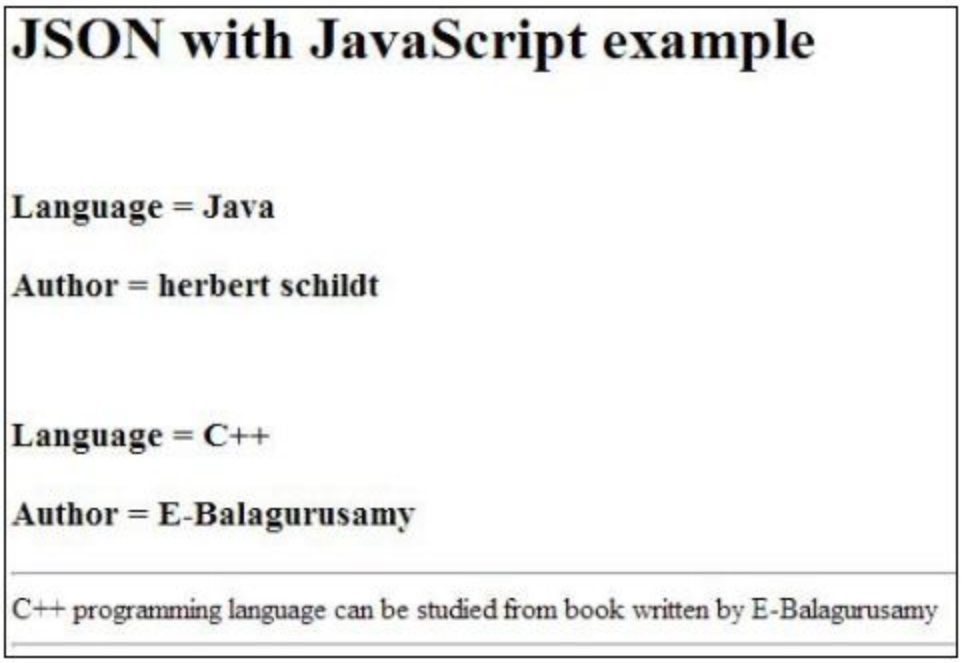
```
{
   "book": [

      {
         "id":"01",
         "language": "Java",
         "edition": "third",
         "author": "Herbert Schildt"
      },

      {
         "id":"07",
         "language": "C++",
         "edition": "second",
         "author": "E.Balagurusamy"
      }
   ]
}
```

After understanding the above program, we will try another example. Let's save the below code as **json.htm** –

save the below code as **json.htm** –

```
<html>
   <head>
      <title>JSON example</title>

      <script language = "javascript" >

         var object1 = { "language" : "Java", "author"  : "herbert schildt" };
         document.write("<h1>JSON with JavaScript example</h1>");
         document.write("<br>");
         document.write("<h3>Language = " + object1.language+"</h3>");
         document.write("<h3>Author = " + object1.author+"</h3>");

         var object2 = { "language" : "C++", "author"  : "E-Balagurusamy" };
         document.write("<br>");
         document.write("<h3>Language = " + object2.language+"</h3>");
         document.write("<h3>Author = " + object2.author+"</h3>");

         document.write("<hr />");
         document.write(object2.language + " programming language can be studied " +
         document.write("<hr />");

      </script>

   </head>

   <body>
   </body>

</html>
```

Now let's try to open json.htm using IE or any other javascript enabled browser that produces the following result –



You can refer to JSON Objects chapter for more information on JSON objects.

Difference between JSON and XML

	JSON	XML
Stands For	**JSON : "JavaScript Object Notation".**	**XML: "Extensible Markup Language".**
Extended From	JSON is extended from **JavaScript.**	XML is extended from **SGML** : "**Standard Generalized Markup Language**".
Purpose	JSON is one type of text-based format or standard for interchanging data i.e. **human readable**. JSON is developed by "**Douglas Crockford**".	XML is a **Markup Language** having format that contains set of rules for the encoding the documents which is readable for both **human & machine**. XML is developed by **W3C** :"**World Wide Web Consortium** ".
Syntax	JSON syntax is lighter than XML as JSON has serialized format of data having less redundancy. JSON does not contain start and end tags.	XML is not so lighter as JSON as having start and end tags and it takes more character than JSON to represent same data.
Speed	JSON is light – weighted in compare to XML as it has serialized format and so faster also.	XML is not so light weighted as JSON.

Support of Data Type & Array	JSON supports **datatype** including integer and strings, JSON also supports array.	XML does not provide any data type so needs to be **parsed** into particular datatype. No direct support for array also.
Object Support	JSON has support of native **objects**.	XML can get support of objects through mixed use of attributes & elements.
Comments	JSON does not support **Comments**	XML supports comments.
Namespace	JSON does not have support for Namespaces.	XML supports **Namespaces**.
Mapping	JSON is **data oriented** and can be mapped more easily.	XML is **document oriented** and needs more effort for **mapping**.
Parsing	JSON uses only **evel()** for **parsing** i.e. for interpreting the JavaScript code & returns the result. It does not need any additional code for parsing.	XML needs XML **Document Object Model** (DOM) implementation & with that additional code for mapping text back to the JavaScript objects.
Application	For W**eb services**, JSON is better.	For **configuration**, XML is better.
Changing Format	You can not change JSON data to other **format**.	In XML, using **XSLT** you can change XML data into another format like **comma –delimited, plain text**, JSON, etc.
Access	In JSON no such is interface for getting direct access to a part in JSON data-structure.	In XML, Using **XPath**, it is possible to get the direct access to a particular part of multiple parts of XML data- structure.

JSON	XML
It is JavaScript Object Notation	It is Extensible markup language
It is based on JavaScript language.	It is derived from SGML.
It is a way of representing objects.	It is a markup language and uses tag structure to represent data items.
It does not provides any support for namespaces.	It supports namespaces.
It supports array.	It doesn't supports array.
Its files are very easy to read as compared to XML.	Its documents are comparatively difficult to read and interpret.
It doesn't use end tag.	It has start and end tags.
It is less secured.	It is more secured than JSON.
It doesn't supports comments.	It supports comments.
It supports only UTF-8 encoding.	It supports various encoding.

5. **What Is A Json Parser?**

When exchanging data between a browser and a server, the data can only be text.
JSON is text, and we can convert any JavaScript object into JSON, and send JSON to the server.
We can also convert any JSON received from the server into JavaScript objects.
This way we can work with the data as JavaScript objects, with no complicated parsing and translations.
A common use of JSON is to exchange data to/from a web server.
When receiving data from a web server, the data is always a string.
Parse the data with JSON.parse(), and the data becomes a JavaScript object.
Imagine we received this text from a web server:

```
'{ "name":"John", "age":30, "city":"New York"}'
```

Use the JavaScript function JSON.parse() to convert text into a JavaScript object:

```
var obj = JSON.parse('{ "name":"John", "age":30, "city":"New York"}');
```

Storing JSON Data

As a simple example, information about me might be written in JSON as follows:

```
var jason = {

        "age" : "24",

        "hometown" : "Missoula, MT",

        "gender" : "male"

};
```

This creates an object that we access using the variable jason. By enclosing the variable's value in curly braces, we're indicating that the value is an object. Inside the object, we can declare any number of properties using a "name": "value" pairing, separated by commas. To access the information stored in jason, we can simply refer to the name of the property we need.

For instance, to access information

```
document.write('Jason is ' jason.age); // Output: Jason is 24
document.write('Jason is a ' jason.gender); // Output: Jason is a male
```

Date objects are not allowed in JSON.
If you need to include a date, write it as a string.
You can convert it back into a date object later:

Example

Convert a string into a date:

```
var text = '{ "name":"John", "birth":"1986-12-14", "city":"New York"}';

var obj = JSON.parse(text);

obj.birth = new Date(obj.birth);

document.getElementById("demo").innerHTML = obj.name + ", " + obj.birth;
```

Parsing Functions

Functions are not allowed in JSON.

If you need to include a function, write it as a string.

You can convert it back into a function later:

Example

Convert a string into a function:

```
var text = '{ "name":"John", "age":"function () {return 30;}", "city":"New York"}';

var obj = JSON.parse(text);

obj.age = eval("(" + obj.age + ")");


document.getElementById("demo").innerHTML = obj.name + ", " + obj.age();
```

6. **Explain the the different Mashup techniques.**
   http://www.informit.com/articles/article.aspx?p=1339556

Module 4

1. **What are the common usage of PHP?**
   PHP is a recursive acronym for "PHP: Hypertext Preprocessor".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. The most common uses of PHP are as follows :
   - PHP performs system functions, i.e. from files on a system it can create, open, read, write, and close them.
   - PHP can handle forms, i.e. gather data from files, save data to a file, through email you can send data, return data to the user.
   - You add, delete, modify elements within your database through PHP.
   - Access cookies variables and set cookies.
   - Using PHP, you can restrict users to access some pages of your website.
   - It can encrypt data.

2. **What are the different types of PHP variables?**
   A variable in PHP is a name of memory location that holds data. A variable is a temporary storage that is used to store data temporarily.In PHP, a variable is declared using $ sign followed by variable name that is $variablename=value;
   PHP supports the following data types:
   a. String : A string is a sequence of characters, like "Hello world!" and is declared as $x = "Hello world!";
   b. Integer : An integer data type is a non-decimal number between -2,147,483,648 and 2,147,483,647 and is declared as $x = 5985;
   c. Float (floating point numbers - also called double) : A float (floating point number) is a number with a decimal point or a number in exponential form and is declared as $x = 10.98.
   d. Boolean : A Boolean represents two possible states: TRUE or FALSE and is often used in conditional testing. For example : $y = false;
   e. Array : An array stores multiple values in one single variable. For example $cars = array("Maruti","BMW","Ferrari");

f. Object : An object is a data type which stores data and information on how to process that data. In PHP, an object must be explicitly declared. The method of declaring and using an object is ,

```
class Car {
        function Car() {
        $this->model = "VW";
        }
}
// create an object
$herbie = new Car();
// show object properties
echo $herbie->model;
```

g. NULL : Null is a special data type which can have only one value: NULL. A variable of data type NULL is a variable that has no value assigned to it. For example $x = null;

h. Resource : The special resource type is not an actual data type. It is the storing of a reference to functions and resources external to PHP.A common example of using the resource data type is a database call.

3. Explain control structure in PHP.

A PHP code may comprise of some or all or none of the following blocks of statements. Since these statements can control, modify and interrupt the normal flow of the program they are known as Control Structures. The following are the control structures in PHP :

- **If Statement**
  - An if statement is a way of controlling the execution of a statement that follows it (that is, a single statement or a block of code inside braces). The if statement evaluates an expression between parentheses. If this expression results in a true value, the statement is executed. Otherwise, the statement is skipped entirely.
  - Syntax:-

    ```
    if ( expression ) {
            // code to execute if the expression evaluates to true }
    ```

- **Else Statement**
  - When working with the if statement, you will often want to define an alternative block of code that should be executed if the expression you are testing evaluates to false.
  - Syntax :

    ```
    if ( expression ) {
    // code to execute if the expression evaluates to true
    ```

```
} else {
// code to execute in all other cases
}
```

- **Elseif Statement**
  - It is a combination of if and else. Like else, it extends an if statement to execute a different statement in case the original if expression evaluates to **FALSE**. However, unlike else, it will execute that alternative expression only if the elseif conditional expression evaluates to **TRUE.**
  - Syntax :

```
if ( expression ) {
// code to execute if the expression evaluates to true
} elseif (another expression) {
// code to execute if the previous expression failed
// and this one evaluates to true
} else { // code to execute in all other cases
}
```

- **Switch Statement**
  - The switch statement is similar to a series of IF statements on the same expression. In many occasions, you may want to compare the same variable (or expression) with many different values, and execute a different piece of code depending on which value it equals to.
  - Syntax :

```
switch ( expression ){
case result1:
// execute this if expression results in result1
break;
case result2:
// execute this if expression results in result2
break;
default:
// execute this if no break statement
// has been encountered hitherto
}
```

4. **What are Superglobals in php?**

   Several predefined variables in PHP are "superglobals", which means that they are always accessible, regardless of scope - and you can access them from any function, class or file without having to do anything special.

   The PHP superglobal variables are:

a. $GLOBALS : $GLOBALS is a PHP super global variable which is used to access global variables from anywhere in the PHP script (also from within functions or methods). PHP stores all global variables in an array called $GLOBALS[index]. The index holds the name of the variable.
b. $_SERVER : $_SERVER is a PHP super global variable which holds information about headers, paths, and script locations.
   For example : echo $_SERVER['PHP_SELF']; will print the filename that is currently being executed.
c. $_REQUEST : PHP $_REQUEST is used to collect data after submitting an HTML form.
d. $_POST : PHP $_POST is widely used to collect form data after submitting an HTML form with method="post". $_POST is also widely used to pass variables.
e. $_GET :PHP $_GET can also be used to collect form data after submitting an HTML form with method="get".$_GET can also collect data sent in the URL.
f. $_FILES
g. $_ENV
h. $_COOKIE:A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the cookie too. With PHP, you can both create and retrieve cookie values.
i. $_SESSION : Session variables hold information about one single user, and are available to all pages in one application.

5. **How can we connect to a MySQL database from a PHP script?**

To connect to MySQL using the legacy PHP MySQL functions, follow these steps:
Use the following PHP code to connect to MySQL and select a database. Replace username with your username, password with your password, and dbname with the database name:

```
<?php
mysqli_connect('localhost','username','password');
mysql_select_db("dbname");
?>
```

After the code connects to MySQL and selects the database, you can run SQL queries and perform other operations. For example, the following PHP code runs a SQL query that extracts the last names from the employees table, and stores the result in the $result variable:

```
<?php
$result = mysqli_query('SELECT lastname FROM employees');
?>
```

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

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

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

6. **How to do CRUD operations a MySQL database from a PHP script?**

/* Delete Function is left , It is very easy refer to create and update */
https://github.com/milu234/phpcrud.git

7. **How can we access the data sent through the URL with the POST / GET method?**

```php
<?php
   if( $_GET["name"] || $_GET["age"] ) {
      echo "Welcome ". $_GET['name']. "<br />";
      echo "You are ". $_GET['age']. " years old.";

      exit();
   }
?>
<html>
   <body>

      <form action = "<?php $_PHP_SELF ?>" method = "GET">
         Name: <input type = "text" name = "name" />
```

```
         Age: <input type = "text" name = "age" />
         <input type = "submit" />
      </form>

   </body>
</html>
```

8. **Create a form using html and php and ready data using GET/ POST.**

Index.php

```
<html>
<body>

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

</body>
</html>
```

Milan.php

```
<html>
<body>

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

</body>
</html>
```

9. **Explain feature of php framework**

1. There are many frameworks like Kohana ,Pear ,CakePHP but one of the important and trending framework of php is Laravel .
2. It is a MVC architecture.

3. Laravel possesses a well-fabricated toolbox that allows writing fewer codes that result in less risk of error **(LOL?!).**
4. By using this popular PHP best framework,web app developers can build applications with greater productivity and value, such as managing guiding principles, side effects, etc.
5. **Template Engine**
   Best PHP framework is highly acknowledged for its inbuilt lightweight templates that help you create amazing layouts using dynamic content seeding.

   Artisan
   Laravel offers a built-in tool for command line known as Artisan which allows performing the majority of those tedious and repetitive programming tasks that many PHP developers avoid performing manually.

   Eloquent ORM (object-relational mapping)

Laravel framework offers the Eloquent ORM that includes a simple PHP Active Record implementation. It lets the web app developers issue database queries with PHP syntax rather than writing SQL code.

MVC(Model View Controller) Architecture Support:

The MVC pattern of laravel ensures clarity of logic and presentation. This architecture support helps in improving the performance, allowing better documentation, and has multiple built-in functions.

6. Migration system for databases

Laravel migration system helps to expand the structure of the database of the web application without re-creating every time developers make a change.

Module 5

1. **Difference between**

   a) HTML and XML

   HTML tags have a fixed meaning and browsers know what it is.

   XML tags are different for different applications, and users know what they mean.

   HTML tags are used for display.

   XML tags are used to describe documents and data.

HTML	XML
HTML is an abbreviation for HyperText Markup Language.	XML stands for eXtensible Markup Language.
HTML was designed to display data with focus on how data looks.	XML was designed to be a software and hardware independent tool used to transport and store data, with focus on what data is.
HTML is a markup language itself.	XML provides a framework for defining markup languages.
HTML is a presentation language.	XML is neither a programming language nor a presentation language
HTML is case insensitive.	XML is case sensitive.
HTML is used for designing a web-page to be rendered on the client side.	XML is used basically to transport data between the application and the database.
HTML has it own predefined tags.	While what makes XML flexible is that custom tags can be defined an the tags are invented by the author of the XML document.
HTML is not strict if the user does not use the closing tags.	XML makes it mandatory for the user the close each tag that has bee used.
HTML does not preserve white space.	XML preserves white space.
HTML is about displaying data,hence static.	XML is about carrying information,hence dynamic.

b) XML DTD and XML Schema

https://www.slideshare.net/umarali1981/difference-between-dtd-and-xsd

Schema	DTD
Schema document is an XML document i.e., the structure of an XML document is specified by another XML document	DTDs follow SGML syntax
supports variety of data types similar to programming language	In DTD everything is treated as text
creating relationship among elements is possible	This is not possible in DTD without invalidating existing documents
Grouping of elements and attributes are possible to form a single logical unit	Grouping of elements and attributes is not possible in DTD
it is possible to specify an upper limit for the number of occurrences of an element	It is not possible to specify an upper limit of an element in DTDs

c) XML and XSLT-eXtensible Stylesheet Language

S.No	XML	XSLT
1	XML is used for storing data in a structured format.	XSLT is used for transforming and also for formatting XML file to display the content or to process the content.
2	XML does not perform transformation of data.	XSLT is a specialized language that performs transformation of one XML document into a different XML document. XSLT can also perform transformation of XML document into HTML document and XHTML document.
3	XPath is a specialized language that is used to address portions of the XML file.	XSLT will be using XPath for transformation and formatting of XML file.

2. What is a valid XML document? Design i)DTD ii)Schema for address book XML document.

- An XML document with correct syntax is called "Well Formed".
- An XML document validated against a DTD is both "Well Formed" and "Valid".
- A "Valid" XML document is a "Well Formed" XML document, which also conforms to the rules of a DTD.
- A DTD is a Document Type Definition. It defines the structure and the legal elements and attributes of an XML document.
- Eg of Design of DTD (External eg. note.dtd file, which can be linked to any file) :

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

```
<!ELEMENT body (#PCDATA)>
]>
```

- The DTD above is interpreted like this (#PCDATA means parse-able text data):
  - !DOCTYPE note defines that the root element of the document is note
  - !ELEMENT note defines that the note element must contain the 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"
- Ex. of a XML data parsed using the design given above (Internal).

```
<?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>
```

- Ex. of a XML data parsed using the design given above (External).

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE note SYSTEM "Note.dtd">
<note>
<to>Tove</to>
<from>Tom</from>
<heading>Reminder</heading>
<body>Don't forget!</body>
</note>
```

- DTD for Address book (Internal).

```
< ?xml version=’1.0’ encoding=’utf-8’? >
< !- DTD for a AddressBook.xml – >
< !DOCTYPE AddressBook [
< !ELEMENT AddressBook (Address+) >
< !ELEMENT Address (Name, Street, City) >
< !ELEMENT Name (#PCDATA) >
< !ELEMENT Street (#PCDATA) >
< !ELEMENT City (#PCDATA) >
] >
< AddressBook >
< Address >
< Name >Jeniffer< /Name >
< Street >Wall Street < /Street >
< City >New York< /City >
< /Address >
< /AddressBook >
```

➢

➢ DTD for Address book (External).

```
< ?xml version="1.0" encoding="UTF-8"? >
< !DOCTYPE AddressBook SYSTEM "file:/// c:/XML/AddressBook.dtd " > " >
< AddressBook >
< Address >
< Name >Jeniffer< /Name >
< Street >Wall Street< /Street >
< City >New York< /City >
< /Address >
< /AddressBook >
```

➢