CS8651 INTERNET PROGRAMMING	
UNIT I WEBSITE BASICS, HTML 5, CSS 3, WEB 2.0	**9**
Web Essentials: Clients, Servers and Communication – The Internet – Basic Internet protocols – World wide web – HTTP Request Message – HTTP Response Message – Web Clients – Web Servers – HTML5 – Tables – Lists – Image – HTML5 control elements – Semantic elements – Drag and Drop – Audio – Video controls - CSS3 – Inline, embedded and external style sheets – Rule cascading – Inheritance – Backgrounds – Border Images – Colors – Shadows – Text – Transformations – Transitions – Animations.	

WEB ESSENTIALS

1.1 Web Essentials:

Server:

The software that distributes the information and the machine where the information and software reside is called the server.

• provides requested service to client

• e.g., Web server sends requested Web page

Client:

The software that resides on the remote machine, communicates with the server, fetches the information, processes it, and then displays it on the remote machine is called the client.

• initiates contact with server (—speaks first‖)

• typically requests service from server

• Web: client implemented in browser

Web server:

Software that delivers Web pages and other documents to browsers using the HTTP protocol

Web Page:

A web page is a document or resource of information that is suitable for the World Wide Web and can be accessed through a web browser.

Website:

A collection of pages on the World Wide Web that are accessible from the same URL and typically residing on the same server.

URL:

Uniform Resource Locator, the unique address which identifies a resource on the Internet for routing purposes.

1.2 Client-server paradigm:

IThe Client-Server paradigm is the most prevalent model for distributed computing protocols. It is the basis of all distributed computing paradigms at a higher level of abstraction. It is service-oriented, and employs a request-response protocol.

A server process, running on a server host, provides access to a service. A client process, running on a client host, accesses the service via the server process.The interaction of the process proceeds according to a protocol.

The primary idea of a client/server system is that you have a central repository of information—some kind of data, often in a database—that you want to distribute on demand to some set of people or machines.

1.3 The Internet:

• Medium for communication and interaction in inter connected network.

• Makes information constantly and instantly available to anyone with a connection.

Web Browsers:

• User agent for Web is called a browser:

o Internet Explorer

o Firefox

Web Server:

• Server for Web is called Web server:

o Apache (public domain)

o MS Internet Information Server

Protocol:

Protocols are agreed formats for transmitting data between devices.

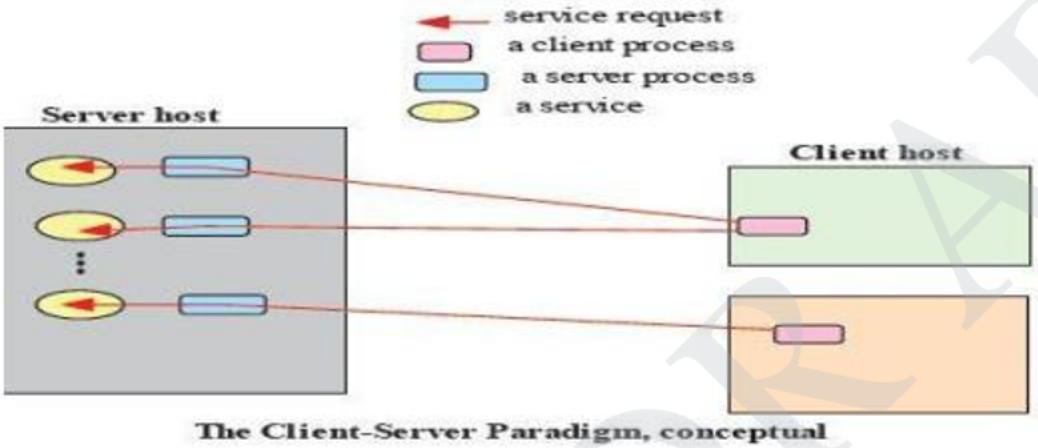
The protocol determines:

i. The error checking required

ii. Data compression method used

iii. The way the end of a message is signalled

iv. The way the device indicates that it has received the message



The Client-Server Paradigm, conceptual

1.4 Internet Protocol:

There are many protocols used by the Internet and the WWW:

o TCP/IP

o HTTP

o FTP

o Electronic mail protocols IMAP

o POP

TCP/IP

The Internet uses two main protocols (developed by Vincent Cerf and Robert Kahn)

Transmission control protocol (TCP):Controls disassembly of message into packets at the origin reassembles at the destination

Internet protocol (IP):Specifies the addressing details for each packet Each packet is labelled with its origin and destination.

1.5 Hypertext Transfer Protocol (HTTP)

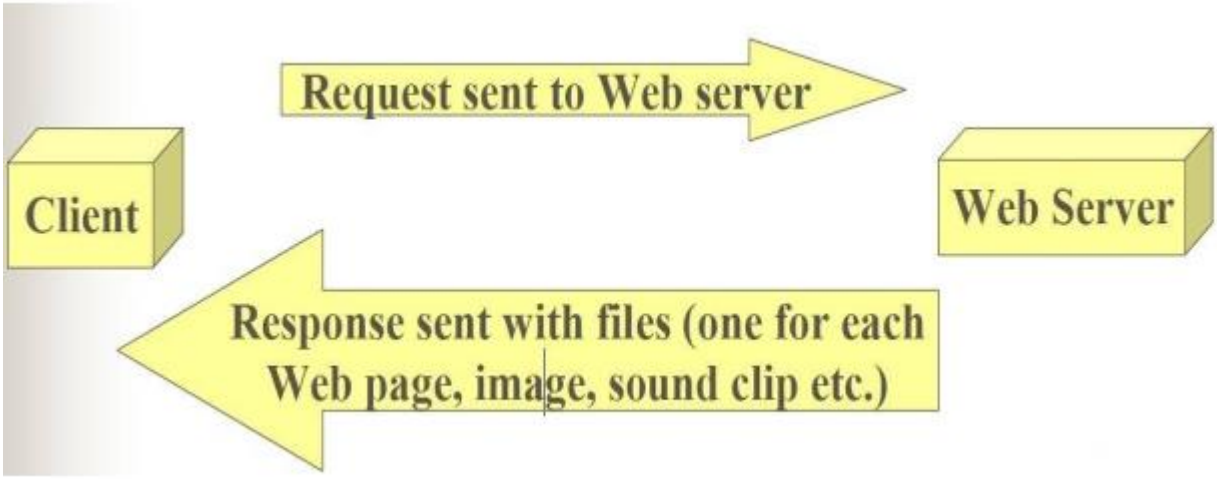
• The hypertext transfer protocol (HTTP) was developed by Tim Berners-Lee in 1991

• HTTP was designed to transfer pages between machines

• The client (or Web browser) makes a request for a given page and the Server is responsible for finding it and returning it to the client

• The browser connects and requests a page from the server

• The server reads the page from the file system, sends it to the client and terminates the connection.



Electronic Mail Protocols:
• Electronic mail uses the client/server model
• The organisation has an email server devoted to handling email
o Stores and forwards email messages
• Individuals use email client software to read and send email
o (e.g. Microsoft Outlook, or Netscape Messenger)
• Simple Mail Transfer Protocol (SMTP)

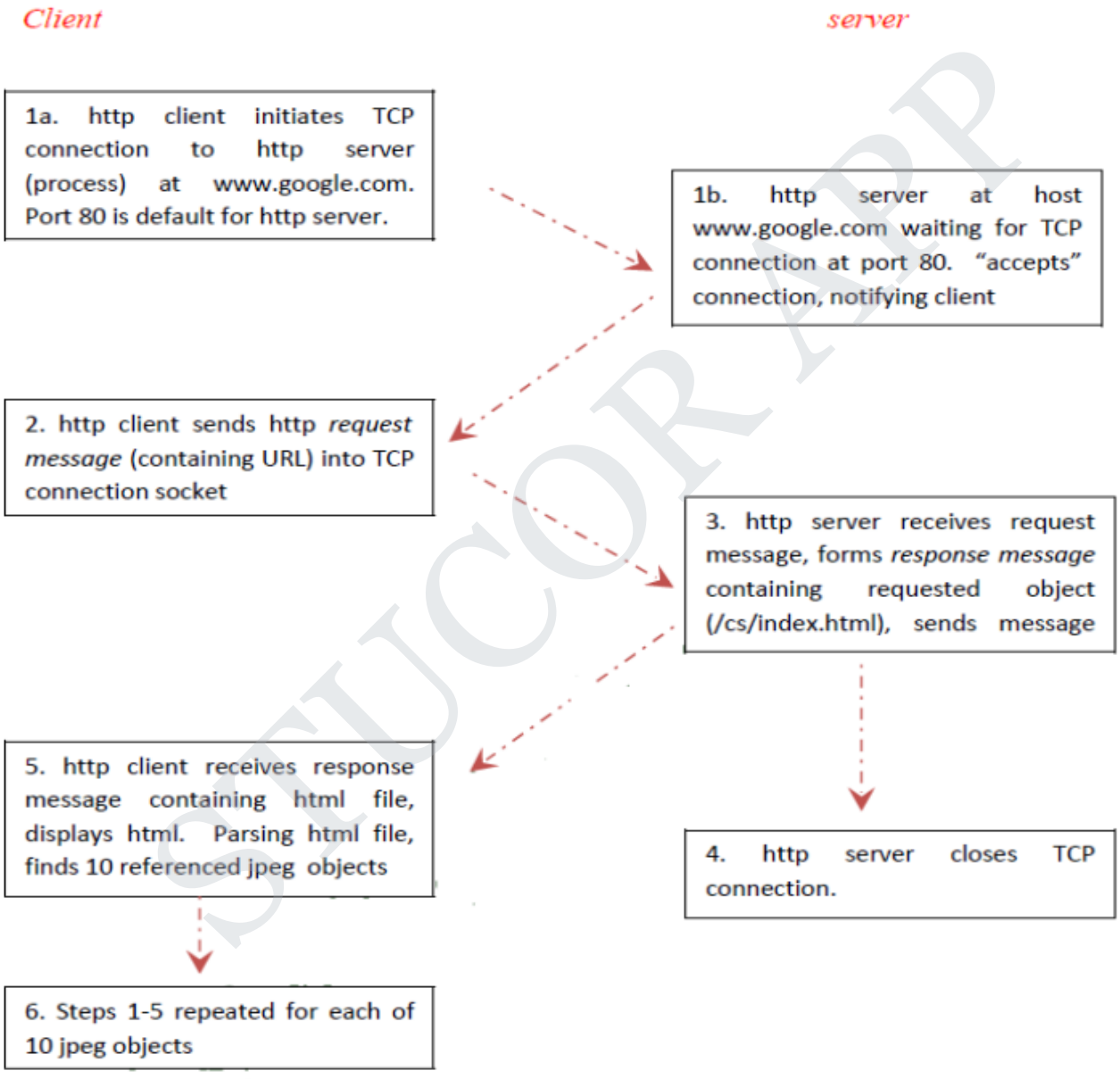
o Specifies format of mail messages
• Post Office Protocol (POP) tells the email server to:
o Send mail to the user's computer and delete it from the server
o Send mail to the user's computer and do not delete it from the server
o Ask whether new mail has arrived.

1.6 Interactive Mail Access Protocol (IMAP)
Newer than POP, provides similar functions with additional features.
o e.g. can send specific messages to the client rather than all the messages.
A user can view email message headers and the sender's name before downloading the entire message.
Allows users to delete and search mailboxes held on the email server.
The disadvantage of POP: You can only access messages from one PC.
The disadvantage of IMAP : Since email is stored on the email server, there is a need for more and more expensive (high speed) storage space.
1.7 World Wide Web: comprises software (Web server and browser) and data (Web sites).
Internet Protocol (IP) Addresses:
- Every node has a unique numeric address
- Form: 32-bit binary number
- New standard, IPv6, has 128 bits (1998)
- Organizations are assigned groups of IPs for their computers
- **Domain names**
- Form: host-name. domain-names
- First domain is the smallest (Google)
- Last domain specifies the type of organization (.com)
- Fully qualified domain name - the host name and all of the domain names
- DNS servers - convert fully qualified domain names to IPs

1.8 HTTP:
•□Hypertext Transfer Protocol (HTTP) is the communication protocol used by the Internet to transfer hypertext documents.
•□A protocol to transfer hypertext requests and information between servers and browsers

- Hypertext is text, displayed on a computer, with references (hyperlinks) to other text that the reader can immediately follow, usually by a mouse HTTP is behind every request for a web document or graph, every click of a hypertext link, and every submission of a form.
- HTTP specifies how clients **request** data, and how servers **respond** to these requests.
- The client makes a request for a given page and the server is responsible for finding it and returning it to the client.
- The browser connects and requests a page from the server.
- The server reads the page from the file system and sends it to the client and then terminates the connection

HTTP Transactions



1.9 HTTP Message:

HTTP message is the information transaction between the client and server.

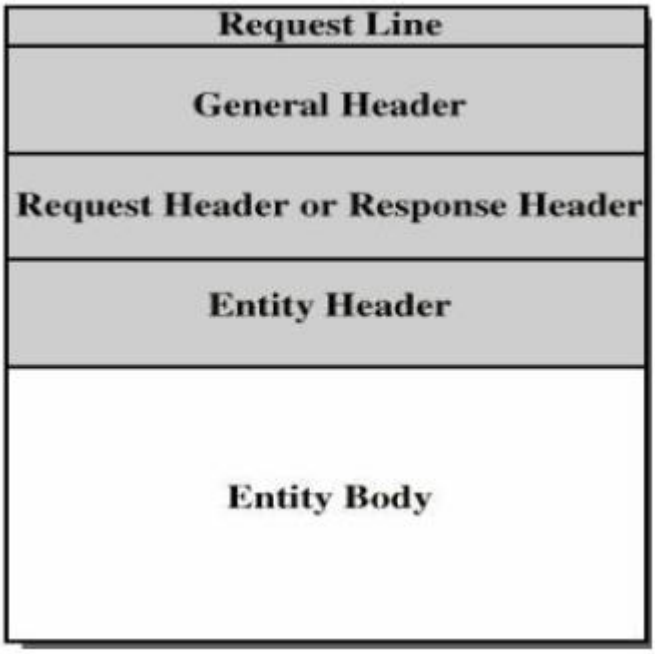
Two types of HTTP Message:

1. Requests

a. Client to server

2. Responses

a. Server to client



Fields

· Request line or Response line
· General header
· Request header or Response header
· Entity header
· Entity body

.10 Request Message:

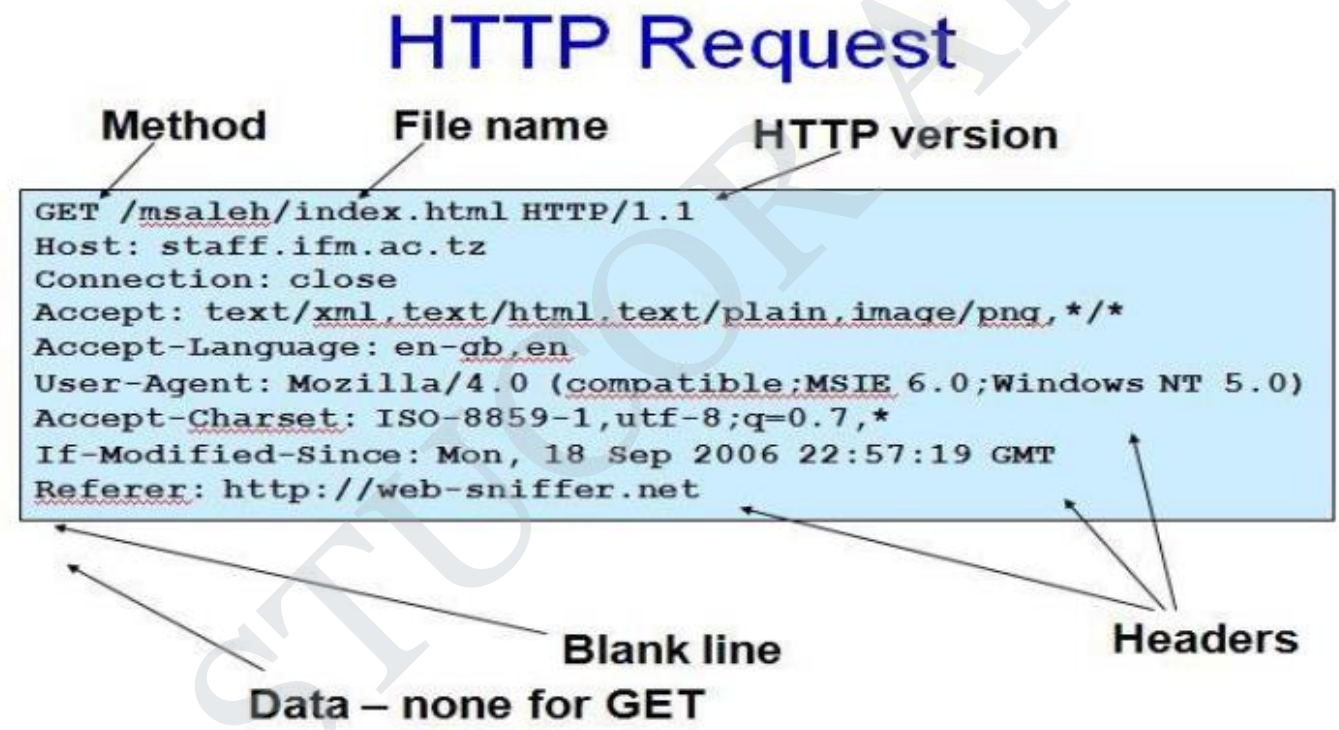
Request Line:

• A request line has three parts, separated by spaces
o a *method* name
o the local path of the requested resource
o the version of HTTP being used
• A typical request line is:
o GET /path/to/file/index.html HTTP/1.1
• Notes:
o **GET** is the most common HTTP method; it says "give me this resource". Other methods include **POST** and **HEAD.** Method names are always uppercase
o The path is the part of the URL after the host name, also called the *request URI*
o The HTTP version always takes the form "**HTTP/x.x**", uppercase.

Request Header:

HTTP Request Headers

Header	Description
From	Email address of user
User-Agent	Client s/w
Accept File	File types that client will accept
Accept-encoding	Compression methods
Accept-Language	Languages
Referrer	URL of the last document the client displayed
If-Modified-Since	Return document only if modified since specified
Content-length	Length (in bytes) of data to follow



.11 Response Message:

Response Line:

• A request line has three parts, separated by spaces
o the HTTP version,
o a *response status code* that gives the result of the request, and
o an English *reason phrase* describing the status code
• Typical status lines are:
o HTTP/1.0 200 OK or
o HTTP/1.0 404 Not Found
• Notes:
o The HTTP version is in the same format as in the request line, "**HTTP/x.x**".

o The status code is meant to be computer-readable; the reason phrase is meant to be human-readable, and may vary.

HTTP Request Header:

HTTP Response Headers

Header	Description
Server	Server software
Date	Current Date
Last-Modified	Modification date of document
Expires	Date at which document expires
Location	The location of the document in redirection responses
Pragma	A hint, e.g., no cache
MIME-version	
Link	URL of document's parent
Content-Length	Length in bytes
Allowed	Requests that user can issue, e.g., GET

EXAMPLE

HTTP Method:

• HTTP method is supplied in the request line and specifies the operation that the client has requested.

Some common methods:

• Options
• Get
• Head
• Post
• Put
• Move
• Delete

Two methods that are mostly used are the GET and POST:

o **GET** for queries that can be safely repeated

o **POST** for operations that may have side effects (e.g. ordering a book from an on-line store).

The GET Method

• It is used to retrieve information from a specified URI and is assumed to be a safe, repeatable operation by browsers, caches and other HTTP aware components

• Operations have no side effects and GET requests can be re-issued.

• For example, displaying the balance of a bank account has no effect on the account and can be safely repeated.

• Most browsers will allow a user to refresh a page that resulted from a **GET**, without displaying any kind of warning

• Proxies may automatically retry **GET** requests if they encounter a temporary network connection problem.

• GET requests is that they can only supply data in the form of parameters encoded in the URI (known as a **Query String**) – [downside]

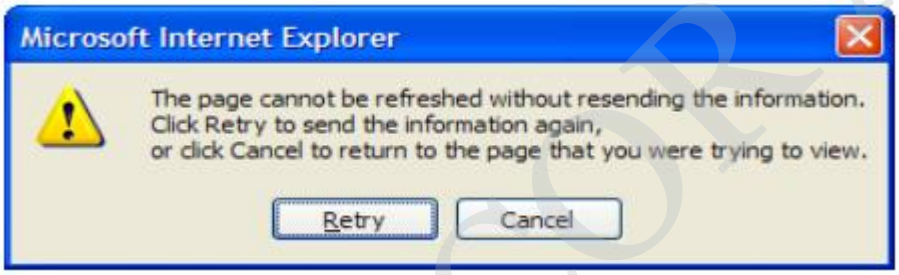
• Cannot be unused for uploading files or other operations that require large amounts of data to be sent to the server.

The POST Method

• Used for operations that have side effects and cannot be safely repeated.

• For example, transferring money from one bank account to another has side effects and should not be repeated without explicit approval by the user.

• If you try to refresh a page in Internet Explorer that resulted from a **POST**, it displays the following message to warn you that there may



The **POST** request message has a content body that is normally used to send parameters and data

• The IIS server returns two status codes in its response for a **POST** request

o The first is **100 Continue** to indicate that it has successfully received the **POST** request

o The second is **200 OK** after the request has been processed.

HTTP response status codes

• Informational (1xx)

• Successful (2xx)

• Redirection (3xx)

o 301: moved permanently

• Client error (4xx)

o 403 : forbidden

o 404: Not found

• Server error (5xx)

o 503: Service unavailable

o 505: HTTP version not supported

1.12 HTTP

❖ **Features**

• Persistent TCP Connections: Remain open for multiple requests

• Partial Document Transfers: Clients can specify start and stop positions

• Conditional Fetch: Several additional conditions

• Better content negotiation
• More flexible authentication.

❖ **Web Browsers :**

• Mosaic - NCSA (Univ. of Illinois), in early 1993 First to use a GUI, led to Explosion of Web use Initially for X-Windows, under UNIX, but was ported to other platforms by late 1993
• Browsers are clients - always initiate, servers react (although sometimes servers require responses)
• Most requests are for existing documents, using Hypertext Transfer Protocol
• (HTTP) But some requests are for program execution, with the output being returned as a document.

Browser: A web browser is a software application for retrieving, presenting, and traversing information resources on the World Wide Web.

❖ **Web Servers:**

- Provide responses to browser requests, either existing documents or dynamicallyBuilt documents.
- Browser-server connection is now maintained through more than one request- Response cycle
- All communications between browsers and servers use Hypertext Transfer Protocol
- Web servers run as background processes in the operating system.
- Monitor a communications port on the host, accepting HTTP messages when they appear

All current Web servers came from either

1. The original from CERN
2. The second one, from NCSA

- Web servers have two main directories:

1. Document root (servable documents)
2. Server root (server system software)

- Document root is accessed indirectly by clients
- Its actual location is set by the server Configuration file
- Requests are mapped to the actual location
- Virtual document trees
- Virtual hosts
- Proxy servers
- Web servers now support other Internet protocols

- Apache (open source, fast, reliable)
- IIS
- Maintained through a program with a GUI interface.

HTML 5

HTML is the main markup language for describing the structure of web pages.

HTML stands for HyperText Markup Language. HTML is the basic building block of World Wide Web.

Hypertext is text displayed on a computer or other electronic device with references to other text that the user can immediately access, usually by a mouse click or key press.

Apart from text, hypertext may contain tables, lists, forms, images, and other presentational elements. It is an easy-to-use and flexible format to share information over the Internet.

Markup languages use sets of markup tags to characterize text elements within a document, which gives instructions to the web browsers on how the document should appear.

HTML was originally developed by Tim Berners-Lee in 1990. He is also known as the father of the web. In 1996, the World Wide Web Consortium (W3C) became the authority to maintain the HTML specifications. HTML also became an international standard (ISO) in 2000. HTML5 is the latest version of HTML. HTML5 provides a faster and more robust approach to web development.

HTML Tags and Elements

HTML is written in the form of HTML elements consisting of markup tags. These markup tags are the fundamental characteristic of HTML. Every markup tag is composed of a keyword, surrounded by angle brackets, such as <html>, <head>, <body>, <title>, <p>, and so on.

HTML tags normally come in pairs like <html> and </html>. The first tag in a pair is often called the opening tag (or start tag), and the second tag is called the closing tag (or end tag).

An opening tag and a closing tag are identical, except a slash (/) after the opening angle bracket of the closing tag, to tell the browser that the command has been completed.

Inserting Images into Web Pages

Images enhance visual appearance of the web pages by making them more interesting and colorful.

The <img> tag is used to insert images in the HTML documents. It is an empty element and contains attributes only. The syntax of the <img> tag can be given with:

```
<img src="url" alt="some_text">
```

The following example inserts three images on the web page:

Example
Try this code »

```
<img src="kites.jpg" alt="Flying Kites">
<img src="sky.jpg" alt="Cloudy Sky">
<img src="balloons.jpg" alt="Balloons">
```

Each image must carry at least two attributes: the src attribute, and an alt attribute.

The src attribute tells the browser where to find the image. Its value is the URL of the image file.

Whereas, the alt attribute provides an alternative text for the image, if it is unavailable or cannot be displayed for some reason. Its value should be a meaningful substitute for the image.

HTML Tables

Creating Tables in HTML

HTML table allows you to arrange data into rows and columns. They are commonly used to display tabular data like product listings, customer's details, financial reports, and so on.

You can create a table using the <table> element. Inside the <table> element, you can use the <tr> elements to create rows, and to create columns inside a row you can use the <td> elements. You can also define a cell as a header for a group of table cells using the <th> element.

The following example demonstrates the most basic structure of a table.

Example

```
<table>
   <tr>
      <th>No.</th>
      <th>Name</th>
      <th>Age</th>
   </tr>
   <tr>
      <td>1</td>
      <td>Peter Parker</td>
      <td>16</td>
   </tr>
   <tr>
      <td>2</td>
      <td>Clark Kent</td>
      <td>34</td>
   </tr>
</table>
```

Tables do not have any borders by default. You can use the CSS border property to add borders to the tables. Also, table cells are sized just large enough to fit the contents by default. To add more space around the content in the table cells you can use the CSS padding property.

Defining a Table Header, Body, and Footer

HTML provides a series of tags <thead>, <tbody>, and <tfoot> that helps you to create more structured table, by defining header, body and footer regions, respectively.

The following example demonstrates the use of these elements.

Example

```
<table>
   <thead>
      <tr>
         <th>Items</th>
         <th>Expenditure</th>
      </tr>
   </thead>
   <tbody>
      <tr>
         <td>Stationary</td>
         <td>2,000</td>
      </tr>
      <tr>
         <td>Furniture</td>
         <td>10,000</td>
      </tr>
   </tbody>
   <tfoot>
      <tr>
         <th>Total</th>
         <td>12,000</td>
```

```
        </tr>
    </tfoot>
</table>
```

HTML Lists

HTML lists are used to present list of information in well formed and semantic way. There are three different types of list in HTML and each one has a specific purpose and meaning.

- **Unordered list** — Used to create a list of related items, in no particular order.
- **Ordered list** — Used to create a list of related items, in a specific order.
- **Description list** — Used to create a list of terms and their descriptions.

HTML Unordered Lists

An unordered list created using the `<ul>` element, and each list item starts with the `<li>` element.

The list items in unordered lists are marked with bullets. Here's an example:

Example

```
<ul>
    <li>Chocolate Cake</li>
    <li>Black Forest Cake</li>
    <li>Pineapple Cake</li>
</ul>
```

— The output of the above example will look something like this:

- Chocolate Cake
- Black Forest Cake
- Pineapple Cake

You can also change the bullet type in your unordered list using the CSS list-style-type property. The following style rule changes the type of bullet from the default *disc* to *square*:

Example

```
ul {
    list-style-type: square;
}
```

Please check out the tutorial on CSS lists to learn about styling HTML lists in details.

HTML Ordered Lists

An ordered list created using the `<ol>` element, and each list item starts with the `<li>` element. Ordered lists are used when the order of the list's items is important.

The list items in an ordered list are marked with numbers. Here's an example:

Example

```
<ol>
  <li>Fasten your seatbelt</li>
  <li>Starts the car's engine</li>
  <li>Look around and go</li>
</ol>
```

— The output of the above example will look something like this:

1. Fasten your seatbelt
2. Starts the car's engine
3. Look around and go

HTML5 Image

HTML Images Syntax

In HTML, images are defined with the `<img>` tag.

The `<img>` tag is empty, it contains attributes only, and does not have a closing tag.

The `src` attribute specifies the URL (web address) of the image:

```
<img src="url">
```

EXAMPLE

```
<!DOCTYPE html>

<html>

<body>



<h2>HTML Image</h2>

<img src="html5.gif" alt="HTML5 Icon" style="width:128px;height:128px;">



</body>

</body>

</html>
```

OUTPUT

HTML Image

HTML Form

HTML Forms are required to collect different kinds of user inputs, such as contact details like name, email address, phone numbers, or details like credit card information, etc.

Forms contain special elements called **controls** like ***inputbox, checkboxes, radio-buttons, submit buttons***, etc. Users generally complete a form by modifying its controls e.g. entering text, selecting items, etc. and submitting this form to a web server for further processing.

The <form> tag is used to create an HTML form. Here's a simple example of a login form:

Example

```
<form>
   <label>Username: <input type="text"></label>
   <label>Password: <input type="password"></label>
   <input type="submit" value="Submit">
</form>
```

The following section describes different types of controls that you can use in your form.

Input Element

This is the most commonly used element within HTML forms.

It allows you to specify various types of user input fields, depending on the type attribute. An input element can be of type *text field*, *password field*, *checkbox*, *radio button*, *submit button*, *reset button*, *file select box*, as well as several new input types introduced in HTML5.

The most frequently used input types are described below.

Text Fields

Text fields are one line areas that allow the user to input text.

Single-line text input controls are created using an <input> element, whose type attribute has a value of text. Here's an example of a single-line text input used to take username:

Example

```
<form>
   <label for="username">Username:</label>
   <input type="text" name="username" id="username">
</form>
```

— The output of the above example will look something like this:

Username:

Password Field

Password fields are similar to text fields. The only difference is; characters in a password field are masked, i.e. they are shown as asterisks or dots. This is to prevent someone else from reading the password on the screen. This is also a single-line text input controls created using an `<input>` element whose `type` attribute has a value of `password`.

Example

```
<form>
  <label for="user-pwd">Password:</label>
  <input type="password" name="user-password" id="user-pwd">
</form>
```

— The output of the above example will look something like this:

Password:

Radio Buttons

Radio buttons are used to let the user select exactly one option from a pre-defined set of options. It is created using an `<input>` element whose `type` attribute has a value of `radio`.

Example

Try this code »

```
<form>
  <input type="radio" name="gender" id="male">
  <label for="male">Male</label>
  <input type="radio" name="gender" id="female">
  <label for="female">Female</label>
</form>
```

— The output of the above example will look something like this:

Male Female

Checkboxes

Checkboxes allows the user to select one or more option from a pre-defined set of options. It is created using an `<input>` element whose `type` attribute has a value of `checkbox`.

Example

```
<form>
  <input type="checkbox" name="sports" id="soccer">
  <label for="soccer">Soccer</label>
  <input type="checkbox" name="sports" id="cricket">
```

```
    <label for="cricket">Cricket</label>
    <input type="checkbox" name="sports" id="baseball">
    <label for="baseball">Baseball</label>
</form>
```

— The output of the above example will look something like this:

Soccer Cricket Baseball

File Select box

The file fields allow a user to browse for a local file and send it as an attachment with the form data. Web browsers such as Google Chrome and Firefox render a file select input field with a Browse button that enables the user to navigate the local hard drive and select a file.

This is also created using an `<input>` element, whose `type` attribute value is set to file.

Example

```
<form>
    <label for="file-select">Upload:</label>
    <input type="file" name="upload" id="file-select">
</form>
```

— The output of the above example will look something like this:

Upload: Choose File No file chosen

Textarea

Textarea is a multiple-line text input control that allows a user to enter more than one line of text. Multi-line text input controls are created using an `<textarea>` element.

Example

```
<form>
    <label for="address">Address:</label>
    <textarea rows="3" cols="30" name="address" id="address"></textarea>
</form>
```

— The output of the above example will look something like this:

Address:

Select Boxes

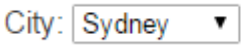
A select box is a dropdown list of options that allows user to select one or more option from a pull-down list of options. Select box is created using the `<select>` element and `<option>` element.

The <option> elements within the <select> element define each list item.

Example

```
<form>
  <label for="city">City:</label>
  <select name="city" id="city">
    <option value="sydney">Sydney</option>
    <option value="melbourne">Melbourne</option>
    <option value="cromwell">Cromwell</option>
  </select>
</form>
```

— The output of the above example will look something like this:



Submit and Reset Buttons

A submit button is used to send the form data to a web server. When submit button is clicked the form data is sent to the file specified in the form's action attribute to process the submitted data.

A reset button resets all the forms control to default values. Try out the following example by typing your name in the text field, and click on submit button to see it in action.

Example

```
<form action="action.php" method="post">
  <label for="first-name">First Name:</label>
  <input type="text" name="first-name" id="first-name">
  <input type="submit" value="Submit">
  <input type="reset" value="Reset">
</form>
```

HTML5 Colors

```
<!DOCTYPE html>

<html>

<body>


<h1 style="background-color:Tomato;">Tomato</h1>

<h1 style="background-color:Orange;">Orange</h1>

<h1 style="background-color:DodgerBlue;">DodgerBlue</h1>

<h1 style="background-color:MediumSeaGreen;">MediumSeaGreen</h1>
```

```
<h1 style="background-color:Gray;">Gray</h1>
<h1 style="background-color:SlateBlue;">SlateBlue</h1>
<h1 style="background-color:Violet;">Violet</h1>
<h1 style="background-color:LightGray;">LightGray</h1>

</body>
</html>
```

OUTPUT

Tomato

Orange

DodgerBlue

MediumSeaGreen

Gray

SlateBlue

Violet

LightGray

HTML5 Audio

Embedding Audio in HTML Document

Inserting audio onto a web page was not easy before, because web browsers did not have a uniform standard for defining embedded media files like audio.

Using the HTML5 audio Element

The newly introduced HTML5 `<audio>` element provides a standard way to embed audio in web pages. However, the audio element is relatively new but it works in most of the modern web browsers.

The following example simply inserts an audio into the HTML5 document, using the browser default set of controls, with one source defined by the `src` attribute.

Example

```
<audio controls="controls" src="media/birds.mp3">
   Your browser does not support the HTML5 Audio element.
</audio>
```

An audio, using the browser default set of controls, with alternative sources.

Example

```
<audio controls="controls">
   <source src="media/birds.mp3" type="audio/mpeg">
   <source src="media/birds.ogg" type="audio/ogg">
   Your browser does not support the HTML5 Audio element.
</audio>
```

HTML5 Video

Embedding Video in HTML Document

Inserting video onto a web page was not relatively easy, because web browsers did not have a uniform standard for defining embedded media files like video.

Using the HTML5 video Element

The newly introduced HTML5 <video> element provides a standard way to embed video in web pages. However, the video element is relatively new, but it works in most of the modern web browsers.

The following example simply inserts a video into the HTML document, using the browser default set of controls, with one source defined by the src attribute.

Example

```
<video controls="controls" src="media/shuttle.mp4">
   Your browser does not support the HTML5 Video element.
</video>
```

A video, using the browser default set of controls, with alternative sources.

Example

```
<video controls="controls">
   <source src="media/shuttle.mp4" type="video/mp4">
   <source src="media/shuttle.ogv" type="video/ogg">
   Your browser does not support the HTML5 Video element.
</video>
```

New HTML5 Elements

The most interesting new HTML5 elements are:

New **semantic elements** like <header>, <footer>, <article>, and <section>.

New **attributes of form elements** like number, date, time, calendar, and range.

New **graphic elements**: <svg> and <canvas>.

New **multimedia elements**: <audio> and <video>.

What are Semantic Elements?

A semantic element clearly describes its meaning to both the browser and the developer.

Examples of **non-semantic** elements: and - Tells nothing about its content.

Examples of **semantic** elements: <form>, <table>, and <article> - Clearly defines its content.

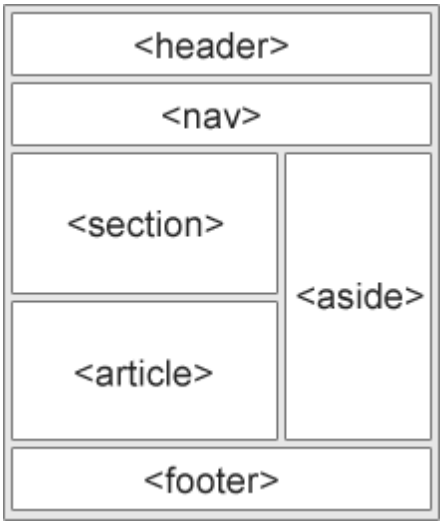
New Semantic Elements in HTML5

Many web sites contain HTML code like:

to indicate navigation, header, and footer.

HTML5 offers new semantic elements to define different parts of a web page:

- <article>
- <aside>
- <details>
- <figcaption>
- <figure>
- <footer>
- <header>
- <main>
- <mark>
- <nav>
- <section>
- <summary>
- <time>



HTML5 <section> Element

The <section> element defines a section in a document.

According to W3C's HTML5 documentation: "A section is a thematic grouping of content, typically with a heading."

A home page could normally be split into sections for introduction, content, and contact information.

Example

```
<section>
  <h1>WWF</h1>
  <p>The World Wide Fund for Nature (WWF) is....</p>
</section>
```

HTML5 <article> Element

The <article> element specifies independent, self-contained content.

An article should make sense on its own, and it should be possible to read it independently from the rest of the web site.

Examples of where an <article> element can be used:

- Forum post
- Blog post
- Newspaper article

Example

```
<article>
  <h1>What Does WWF Do?</h1>
  <p>WWF's mission is to stop the degradation of our planet's natural environment,
  and build a future in which humans live in harmony with nature.</p>
</article>
```

HTML5 <header> Element

The <header> element specifies a header for a document or section.

The <header> element should be used as a container for introductory content.

You can have several <header> elements in one document.

The following example defines a header for an article:

Example

```
<article>
  <header>
    <h1>What Does WWF Do?</h1>
    <p>WWF's mission:</p>
  </header>
  <p>WWF's mission is to stop the degradation of our planet's natural environment,
  and build a future in which humans live in harmony with nature.</p>
</article>
```

HTML5 <footer> Element

The <footer> element specifies a footer for a document or section.

A <footer> element should contain information about its containing element.

A footer typically contains the author of the document, copyright information, links to terms of use, contact information, etc.

You may have several <footer> elements in one document.

Example

```
<footer>
  <p>Posted by: Hege Refsnes</p>
  <p>Contact information: <a href="mailto:someone@example.com">
  someone@example.com</a>.</p>
</footer>
```

HTML5 <figure> and <figcaption> Elements

The purpose of a figure caption is to add a visual explanation to an image.

In HTML5, an image and a caption can be grouped together in a <figure> element:

Example

```
<figure>
  <img src="pic_trulli.jpg" alt="Trulli">
  <figcaption>Fig1. - Trulli, Puglia, Italy.</figcaption>
</figure>
```

OUTPUT

Places to Visit

Puglia's most famous sight is the unique conical houses (Trulli) found in the area around Alberobello, a declared UNESCO World Heritage Site.

Fig.1 - Trulli, Puglia, Italy.

Semantic Elements in HTML5

Below is an alphabetical list of the new semantic elements in HTML5.

The links go to our complete HTML5 Reference.

Tag	Description
<article>	Defines an article
<aside>	Defines content aside from the page content
<details>	Defines additional details that the user can view or hide
<figcaption>	Defines a caption for a <figure> element

<figure>	Specifies self-contained content, like illustrations, diagrams, photos, code listings, etc.
<footer>	Defines a footer for a document or section
<header>	Specifies a header for a document or section
<main>	Specifies the main content of a document
<mark>	Defines marked/highlighted text
<nav>	Defines navigation links
<section>	Defines a section in a document
<summary>	Defines a visible heading for a <details> element
<time>	Defines a date/time

HTML5 Drag and Drop

Drag the W3Schools image into the rectangle.

Drag and Drop

Drag and drop is a very common feature. It is when you "grab" an object and drag it to a different location.

In HTML5, drag and drop is part of the standard: Any element can be draggable.

HTML Drag and Drop Example

The example below is a simple drag and drop example:

Example

```
<!DOCTYPE HTML>
<html>
<head>
<script>
function allowDrop(ev) {
  ev.preventDefault();
}

function drag(ev) {
  ev.dataTransfer.setData("text", ev.target.id);
}

function drop(ev) {
  ev.preventDefault();
  var data = ev.dataTransfer.getData("text");
  ev.target.appendChild(document.getElementById(data));
}
</script>
</head>
<body>

<div id="div1" ondrop="drop(event)" ondragover="allowDrop(event)"></div>

<img id="drag1" src="img_logo.gif" draggable="true" ondragstart="drag(event)" width="336" height="69">

</body>
</html>
```

OUTPUT

Drag the W3Schools image into the rectangle:

HTML5 <nav> Element

The <nav> element defines a set of navigation links.

Notice that NOT all links of a document should be inside a <nav> element. The <nav> element is intended only for major block of navigation links.

Example

```
<nav>
  <a href="/html/">HTML</a> |
  <a href="/css/">CSS</a> |
  <a href="/js/">JavaScript</a> |
  <a href="/jquery/">jQuery</a>
</nav>
```

What Is CSS3 And Why Is It Used?

To help build highly interactive online pages, CSS3 is invariably used due to its importance in providing greater options in the design process. When marketing products and services, web design plays a vital part; a site should be created in a manner that will draw potential customers to explore and revisit a website more often. Many **web design firm**s are developing and enhancing websites through the use of CSS3 as this is a great form of web development. This article will help define CSS3 and will point out its advantages.

Definition

The acronym CSS stands for Cascading Style Sheets which is used to augment the functionality, versatility. and efficient performance of site content. It allows for the creation of content-rich websites that do not require much weight or codes; this translates into more interactive graphics and animation, superior user-interface, and significantly more organization and rapid download time.

It is used with HTML to create content structure, with CSS3 being used to format structured content. It is responsible for font properties, colors, text alignments, graphics, background images, tables and other components. This tool provides extra capabilities such as absolute, fixed and relative positioning of various elements. The increasing popularity of CSS3 when used by **web design firm**s stimulates major browsers such as Google Chrome, Firefox, Safari, and IE9 to adopt and embrace this programming language.

Advantages

Although CSS3 is not the only web development solution, it does allow provide greater advantages for several reasons.

- **Customization** – A web page can be customized and alterations created in the design by simply changing a modular file.
- **Bandwidth Requirements** – It decreases server bandwidth requirements, giving rapid download time when a site is accessed with desktop or hand-held devices, providing an improved user experience.
- **Consistency** – It delivers consistent and accurate positioning of navigational elements on the website.
- **Appealing** – It makes the site more appealing with adding videos and graphics easier.
- **Viewing** – It allows online videos to be viewed without the use of third-party plug-ins.
- **Visibility** – It delivers the opportunity to improve brand visibility by designing effective online pages.
- **Cost Effective** – It is cost-effective, time-saving, and supported by most browsers.

Since the introduction of CSS3, there is greater control of the presentation of content and various elements on a website; however it is not really responsible for overall design as it only specifies the structure and content presentation of certain web pages.

External, internal, and inline CSS styles

Cascading Style Sheets (CSS) are files with styling rules that govern how your website is presented on screen. CSS rules can be applied to your website's HTML files in various ways. You can use an **external stylesheet**, an **internal stylesheet**, or an **inline style**. Each method has advantages that suit particular uses.

An **external stylesheet** is a standalone .css file that is linked from a web page. The advantage of external stylesheets is that it can be created once and the rules applied to multiple web pages. Should you need to make widespread changes to your site design, you can make a single change in the stylesheet and it will be applied to all linked pages, saving time and effort.

An **internal stylesheet** holds CSS rules for the page in the **head** section of the HTML file. The rules only apply to that page, but you can configure CSS classes and IDs that can be used to style multiple elements in the page code. Again, a single change to the CSS rule will apply to all tagged elements on the page.

Inline styles relate to a specific HTML tag, using a **style** attribute with a CSS rule to style a specific page element. They're useful for quick, permanent changes, but are less flexible than external and internal stylesheets as each inline style you create must be separately edited should you decide to make a design change.

Using external CSS stylesheets

An HTML page styled by an external CSS stylesheet must reference the .css file in the document head. Once created, the CSS file must be uploaded to your server and linked in the HTML file with code such as:

```
<link href="style.css" rel="stylesheet" type="text/css">
```

You can name your stylesheet whatever you wish, but it should have a .css file extension.

Using internal CSS stylesheets

Rather than linking an external .css file, HTML files using an internal stylesheet include a set of rules in their **head** section. CSS rules are wrapped in <style> tags, like this:

```
<head>
<style type="text/css">

  h1 {
        color:#fff
        margin-left: 20px;
    }

  p {
```

```
    font-family: Arial, Helvetica, Sans Serif;
   }



</style>
</head>
```

Using inline styles

Inline styles are applied directly to an element in your HTML code. They use the **style** attribute, followed by regular CSS properties.

For example:

```
<h1 style="color:red;margin-left:20px;">Today's Update</h1>
```

Rule Cascading

Cascade and inheritance

Conflicting rules

CSS stands for **Cascading Style Sheets**, and that first word *cascading* is incredibly important to understand — the way that the cascade behaves is key to understanding CSS.
At some point, we will find that the CSS have created two rules which could potentially apply to the same element. The **cascade**, and the closely-related concept of **specificity**, are mechanisms that control which rule applies when there is such a conflict. Which rule is styling your element may not be the one you expect, so you need to understand how these mechanisms work.
Also significant here is the concept of **inheritance**, which means that some CSS properties by default inherit values set on the current element's parent element, and some don't. This can also cause some behavior that you might not expect.

The cascade

Stylesheets **cascade** — at a very simple level this means that the order of CSS rules matter; when two rules apply that have equal specificity the one that comes last in the CSS is the one that will be used.

EXAMPLE
In the below example, we have two rules that could apply to the `h1`. The `h1` ends up being colored blue — these rules have an identical selector and therefore carry the same specificity, so the last one in the source order wins.

```
h1 {

   color: red;
```

```
}

h1 {

   color: blue;

}

   <h1>This is my heading.</h1>
```

OUTPUT

This is my heading.

Specificity

Specificity is how the browser decides which rule applies if multiple rules have different selectors, but could still apply to the same element. It is basically a measure of how specific a selector's selection will be:

- An element selector is less specific — it will select all elements of that type that appear on a page — so will get a lower score.
- A class selector is more specific — it will select only the elements on a page that have a specific `class` attribute value — so will get a higher score.

Example time! Below we again have two rules that could apply to the `h1`. The below `h1` ends up being colored red — the class selector gives its rule a higher specificity, and so it will be applied even though the rule with the element selector appears further down in the source order.

EXAMPLE

```
main-heading {

   color: red;

}

      h1 {

   color: blue;

}
```

```
.    <h1 class="main-heading">This is my heading.</h1>
```

OUTPUT

This is my heading.

Inheritance

Inheritance also needs to be understood in this context — some CSS property values set on parent elements are inherited by their child elements, and some aren't.

For example, if you set a color and font-family on an element, every element inside it will also be styled with that color and font, unless you've applied different color and font values directly to them.

Some properties do not inherit — for example if you set a width of 50% on an element, all of its descendants do not get a width of 50% of their parent's width. If this was the case, CSS would be very frustrating to use!

```
body {
   color: blue;
}

span {
   color: black;
}
```

```
<p>As the body has been set to have a color of blue this is inherited through the
descendants.</p>

<p>We can change the color by targetting the element with a selector,
 such as this

<span>span</span>.</p>
```

OUTPUT

As the body has been set to have a color of blue this is inherited through the descendants.

We can change the color by targetting the element with a selector, such as this **span.**

CSS3 Shadow Effects

With CSS3 you can create two types of shadows: `text-shadow` (adds shadow to text) and `box-shadow` (adds shadow to other elements).

CSS3 Text Shadow

The `text-shadow` property can take up to four values:

- the horizontal shadow
- the vertical shadow
- the blur effect
- the color

Examples:

- Normal text shadow

```
h1 {
text-shadow: 2px 2px 5px crimson;
}
```

- Glowing text effect

```
h1 {
text-shadow: 0 0 4px #00FF9C;
}
```

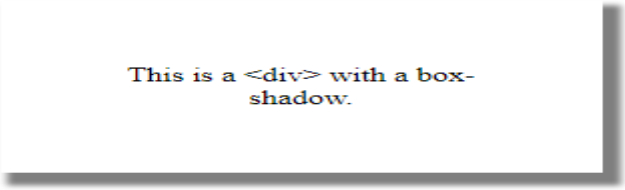
CSS3 Box Shadow

The `box-shadow` property can take up to six values:

- (optional) the inset keyword (changes the shadow to one inside the frame)
- the horizontal shadow
- the vertical shadow
- the blur effect
- the spreading
- the color

Examples:

```
.first-div {
        box-shadow: 1px 1px 5px 3px grey;
}
```



CSS Animations

CSS allows animation of HTML elements without using JavaScript or Flash!

What are CSS Animations?

An animation lets an element gradually change from one style to another.

You can change as many CSS properties you want, as many times you want.

To use CSS animation, you must first specify some keyframes for the animation.

Keyframes hold what styles the element will have at certain times.

The @keyframes Rule

When you specify CSS styles inside the @keyframes rule, the animation will gradually change from the current style to the new style at certain times.

To get an animation to work, you must bind the animation to an element.

The following example binds the "example" animation to the element. The animation will last for 4 seconds, and it will gradually change the background-color of the element from "red" to "yellow":

Example

```
/* The animation code */
@keyframes example {
```

```
  from {background-color: red;}
  to {background-color: yellow;}
}

/* The element to apply the animation to */
div {
  width: 100px;
  height: 100px;
  background-color: red;
  animation-name: example;
  animation-duration: 4s;
}
```

Note: The animation-duration property defines how long time an animation should take to complete. If the animation-duration property is not specified, no animation will occur, because the default value is 0s (0 seconds).

In the example above we have specified when the style will change by using the keywords "from" and "to" (which represents 0% (start) and 100% (complete)).

It is also possible to use percent. By using percent, you can add as many style changes as you like.

The following example will change the background-color of the element when the animation is 25% complete, 50% complete, and again when the animation is 100% complete:

Example

```
/* The animation code */
@keyframes example {
  0%   {background-color: red;}
  25%  {background-color: yellow;}
  50%  {background-color: blue;}
  100% {background-color: green;}
}

/* The element to apply the animation to */
div {
  width: 100px;
  height: 100px;
  background-color: red;
  animation-name: example;
  animation-duration: 4s;
}
```

CSS Animation Properties

The following table lists the @keyframes rule and all the CSS animation properties:

Property	Description
@keyframes	Specifies the animation code
animation	A shorthand property for setting all the animation properties

animation-delay	Specifies a delay for the start of an animation
animation-direction	Specifies whether an animation should be played forwards, backwards or in alternate cycles
animation-duration	Specifies how long time an animation should take to complete one cycle
animation-fill-mode	Specifies a style for the element when the animation is not playing (before it starts, after it ends, or both)
animation-iteration-count	Specifies the number of times an animation should be played
animation-name	Specifies the name of the @keyframes animation
animation-play-state	Specifies whether the animation is running or paused
animation-timing-function	Specifies the speed curve of the animation

CSS Transitions

CSS Transitions is a module of CSS that lets you create gradual transitions between the values of specific CSS properties. The behavior of these transitions can be controlled by specifying their timing function, duration, and other attributes.

Properties

- transition
- transition-delay
- transition-duration
- transition-property
- transition-timing-function

The **transition** CSS property is a shorthand property for transition-property, transition-duration, transition-timing-function, and transition-delay.

CSS transition Property

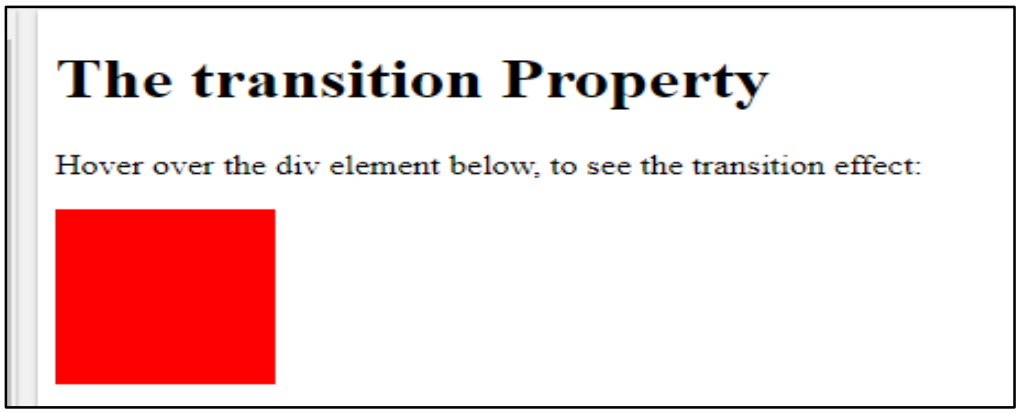
Example

Hover over a element to gradually change the width from 100px to 300px:

```
div {
  width: 100px;
  transition: width 2s;
}

div:hover {
  width: 300px;
}
```

OUTPUT



Definition and Usage

The transition property is a shorthand property for:

- transition-property
- transition-duration
- transition-timing-function

Property Values

Value	Description
transition-property	Specifies the name of the CSS property the transition effect is for
transition-duration	Specifies how many seconds or milliseconds the transition effect takes to complete
transition-timing-function	Specifies the speed curve of the transition effect
transition-delay	Defines when the transition effect will start
initial	Sets this property to its default value. Read about *initial*
inherit	Inherits this property from its parent element. Read about *inherit*

Example

When an <input type="text"> gets focus, gradually change the width from

100px to 250px:

```
input[type=text] {
  width: 100px;
  transition: width .35s ease-in-out;
}

input[type=text]:focus {
```

```
  width: 250px;
}
```

OUTPUT

The width Property

Set the width of the input field to 100 pixels. However, when the input field gets focus, make it 250 pixels wide:

Search:

CSS background-color

The `background-color` property specifies the background color of an element.

Example

The background color of a page is set like this:

```
body {
  background-color: lightblue;
}
```

CSS background-image

The `background-image` property specifies an image to use as the background of an element.

By default, the image is repeated so it covers the entire element.

Example

The background image for a page can be set like this:

```
body {
  background-image: url("paper.gif");
}
```

CSS background - Shorthand property

To shorten the code, it is also possible to specify all the background properties in one single property. This is called a shorthand property.

The shorthand property for background is `background`.

Example

Use the shorthand property to set all the background properties in one declaration:

```
body {
  background: #ffffff url("img_tree.png") no-repeat right top;
}
```

CSS Border - Shorthand Property

As you can see from the examples above, there are many properties to consider when dealing with borders.

To shorten the code, it is also possible to specify all the individual border properties in one property.

The `border` property is a shorthand property for the following individual border properties:

- `border-width`
- `border-style` (required)
- `border-color`

Example

```
p {
  border: 5px solid red;
}
```

Result:

Some text

UNIT II - CLIENT SIDE PROGRAMMING

Java Script: An introduction to JavaScript – JavaScript DOM Model – Date and Objects – Regular Expressions – Exception Handling – Validation – Built-in objects – Event Handling – DHTML with JavaScript – JSON introduction – Syntax – Function Files – Http Request – SQL.

PART - A

Q.No	Questions
1.	**Evaluate** various Java Script Object models.
2.	**Define** DOM.
3.	**Give** any four methods of Date objects.

JavaScript Get Date Methods

These methods can be used for getting information from a date object:

Method	Description
getFullYear()	Get the **year** as a four digit number (yyyy)
getMonth()	Get the **month** as a number (0-11)
getDate()	Get the **day** as a number (1-31)
getHours()	Get the **hour** (0-23)
getMinutes()	Get the **minute** (0-59)
getSeconds()	Get the **second** (0-59)
getMilliseconds()	Get the **millisecond** (0-999)
getTime()	Get the time (milliseconds since January 1, 1970)
getDay()	Get the weekday as a number (0-6)
Date.now()	Get the time. ECMAScript 5.

JavaScript Set Date Methods

Set Date methods let you set date values (years, months, days, hours, minutes, seconds, milliseconds) for a Date Object.

Set Date Methods

Set Date methods are used for setting a part of a date:

Method	Description
setDate()	Set the day as a number (1-31)
setFullYear()	Set the year (optionally month and day)
setHours()	Set the hour (0-23)
setMilliseconds()	Set the milliseconds (0-999)
setMinutes()	Set the minutes (0-59)
setMonth()	Set the month (0-11)
setSeconds()	Set the seconds (0-59)
setTime()	Set the time (milliseconds since January 1, 1970)

4. **Write** the JavaScript methods to retrieve the data and time based on the computer locale.

```
<script>

// Use of Date.now() function
var d = Date(Date.now());

// Converting the number of millisecond in date string
a = d.toString()

// Printing the current date
document.write("The current date is: " + a)

</script>
```

OUTPUT

The current date is: Thu Jan 09 2020 20:35:39 GMT+0530 (India Standard Time)

5. Can you **list** the different methods defined in document and window object of JavaScript.

Window Object

The window object represents an open window in a browser.

Window Object Methods

Method	Description
alert()	Displays an alert box with a message and an OK button
atob()	Decodes a base-64 encoded string
blur()	Removes focus from the current window
btoa()	Encodes a string in base-64
clearInterval()	Clears a timer set with setInterval()
clearTimeout()	Clears a timer set with setTimeout()
close()	Closes the current window
confirm()	Displays a dialog box with a message and an OK and a Cancel button
focus()	Sets focus to the current window
getComputedStyle()	Gets the current computed CSS styles applied to an element
getSelection()	Returns a Selection object representing the range of text selected by the user
matchMedia()	Returns a MediaQueryList object representing the specified CSS media query string
moveBy()	Moves a window relative to its current position
moveTo()	Moves a window to the specified position
open()	Opens a new browser window
print()	Prints the content of the current window
prompt()	Displays a dialog box that prompts the visitor for input

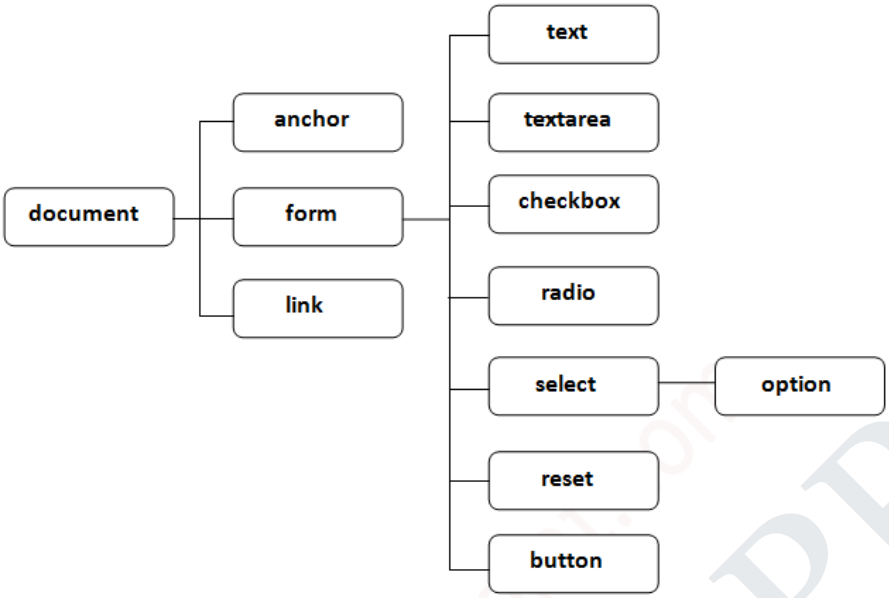
requestAnimationFrame()	Requests the browser to call a function to update an animation before the next repaint
resizeBy()	Resizes the window by the specified pixels
resizeTo()	Resizes the window to the specified width and height
scroll()	Deprecated. This method has been replaced by the scrollTo() method.
scrollBy()	Scrolls the document by the specified number of pixels
scrollTo()	Scrolls the document to the specified coordinates
setInterval()	Calls a function or evaluates an expression at specified intervals (in milliseconds)
setTimeout()	Calls a function or evaluates an expression after a specified number of milliseconds
stop()	Stops the window from loading

Document Object Model

1. Document Object
2. Properties of document object
3. Methods of document object
4. Example of document object

- The **document object** represents the whole html document.
- When html document is loaded in the browser, it becomes a document object.
- It is the **root element** that represents the html document. It has properties and methods.
- By the help of document object, we can add dynamic content to our web page

- According to W3C - *"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."*



Properties of document object

Let's see the properties of document object that can be accessed and modified by the document object.

Methods of document object

We can access and change the contents of document by its methods.

The important methods of document object are as follows:

Method	Description
write("string")	writes the given string on the doucment.
writeln("string")	writes the given string on the doucment with newline character at the end.
getElementById()	returns the element having the given id value.
getElementsByName()	returns all the elements having the given name value.
getElementsByTagName()	returns all the elements having the given tag name.
getElementsByClassName()	returns all the elements having the given class name.

6.	**Name** which parser is best in parsing in large size documents. Why?
7.	**Summarize** benefits of using JavaScript code in an HTML document. **Advantages of JavaScript** The merits of using JavaScript are − • **Less server interaction** − You can validate user input before sending the page off to the server. This saves server traffic, which means less load on your server. • **Immediate feedback to the visitors** − They don't have to wait for a page reload to see if they have forgotten to enter something. • **Increased interactivity** − You can create interfaces that react when the user hovers over them with a mouse or activates them via the keyboard. • **Richer interfaces** − You can use JavaScript to include such items as drag-and-drop components and sliders to give a Rich Interface to your site visitors.
8.	**Predict** the need for client and server side scripting. **Client-side scripting** (embedded scripts) is code that exists inside the client's HTML page. This code will be processed on the client machine and the HTML page will NOT perform a PostBack to the web-server. Traditionally, client-side scripting is used for page navigation, data validation and formatting. The language used in this scripting is JavaScript. JavaScript is compatible and is able to run on any internet browser. The two main benefits of client-side scripting are: 1. The user's actions will result in an immediate response because they don't require a trip to the server. 2. Fewer resources are used and needed on the web-server. **Server-side scripting** is a technique used in web development which involves employing **scripts** on a web **server** which produce a response customized for each user's (client's) request to the website. The alternative is for the web **server** itself to deliver a static web page. The **client-side script** executes the code to the **client side** which is visible to the users while a **server-side script** is executed **in the server** end which users cannot see.
9.	**Interpret** how exceptions are handled in Java script.

The try...catch...finally Statement

The latest versions of JavaScript added exception handling capabilities. JavaScript implements the **try...catch...finally** construct as well as the **throw** operator to handle exceptions.

You can **catch** programmer-generated and **runtime** exceptions, but you cannot **catch** JavaScript syntax errors.

Here is the **try...catch...finally** block syntax –

```
<script type = "text/javascript">
  <!--
    try {
      // Code to run
      [break;]
    }

    catch ( e ) {
      // Code to run if an exception occurs
      [break;]
    }

    [ finally {
      // Code that is always executed regardless of
      // an exception occurring
    }]
  //-->
</script>
```

The **try** block must be followed by either exactly one **catch** block or one **finally** block (or one of both). When an exception occurs in the **try** block, the exception is placed in **e** and the **catch** block is executed. The optional **finally** block executes unconditionally after try/catch.

Examples

Here is an example where we are trying to call a non-existing function which in turn is raising an exception. Let us see how it behaves without **try...catch**−

```
<html>
  <head>
    <script type = "text/javascript">
      <!--
        function myFunc() {
          var a = 100;
          alert("Value of variable a is : " + a );
        }
      //-->
```

```
    </script>
  </head>

  <body>
    <p>Click the following to see the result:</p>

    <form>
      <input type = "button" value = "Click Me" onclick = "myFunc();" />
    </form>
  </body>
</html>
```

10. **Define** JavaScript statement with an example.

JavaScript Statements

Example

```
var x, y, z;     // Statement 1
x = 5;           // Statement 2
y = 6;           // Statement 3
z = x + y;       // Statement 4
```

JavaScript statements are composed of:
Values, Operators, Expressions, Keywords, and Comments.
This statement tells the browser to write "Hello Dolly." inside an HTML element with id="demo":

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Statements</h2>

<p>In HTML, JavaScript statements are executed by the browser.</p>

<p id="demo"></p>

<script>

document.getElementById("demo").innerHTML = "Hello Dolly.";

</script>

</body>

</html>
```

OUTPUT

	JavaScript Statements In HTML, JavaScript statements are executed by the browser. Hello Dolly.

11.	**Point out** any two techniques of event programming.
	What is an Event ?
	JavaScript's interaction with HTML is handled through events that occur when the user or the browser manipulates a page.
	When the page loads, it is called an event. When the user clicks a button, that click too is an event. Other examples include events like pressing any key, closing a window, resizing a window, etc.
	Developers can use these events to execute JavaScript coded responses, which cause buttons to close windows, messages to be displayed to users, data to be validated, and virtually any other type of response imaginable.
	Events are a part of the Document Object Model (DOM) Level 3 and every HTML element contains a set of events which can trigger JavaScript Code.
	Please go through this small tutorial for a better understanding HTML Event Reference. Here we will see a few examples to understand a relation between Event and JavaScript −
	onclick Event Type
	This is the most frequently used event type which occurs when a user clicks the left button of his mouse. You can put your validation, warning etc., against this event type.
	Example
	Try the following example.
	`<html>`
	`<head>`
	`<script type = "text/javascript">`
	`<!--`
	`function sayHello() {`
	`alert("Hello World")`
	`}`
	`//-->`
	`</script>`
	`</head>`

```
  <body>
    <p>Click the following button and see result</p>
    <form>
      <input type = "button" onclick = "sayHello()" value = "Say Hello" />
    </form>
  </body>
</html>
```

Example 2

```
<!doctype html>
<html>
<head>
        <script>
        function hov() {
                var e = document.getElementById('hover');
                e.style.display = 'none';
        }
        </script>
</head>
<body>
        <div id="hover" onmouseover="hov()"
        style="background-color:green;height:200px;width:200px;">
        </div>
</body>
</html>
```

OUTPUT

Before mouse is taken over green square-

Green square gets disappear after mouse is taken over it.

12. **What** is DHTML?

DHTML

DHTML stands for Dynamic HTML, it is totally different from HTML. The DHTML make use of Dynamic object model to make changes in settings and also in properties and methods.

	DHTML allows different scripting languages in a web page to change their variables, which enhance the effects, looks and many others functions after the whole page have been fully loaded or under a view process, or otherwise static HTML pages on the same. DHTML is used to create interactive and animated web pages that are generated in real-time, also known as dynamic web pages so that when such a page is accessed, the code within the page is analyzed on the web server and the resulting HTML is sent to the client's web browser.
13.	**Differentiate** HTML and DHTML **Some differences between HTML and DHTML:** • HTML is a mark-up language, while DHTML is a collection of technology. • DHTML creates dynamic web pages, whereas HTML creates static web pages. • DHTML allows including small animations and dynamic menus in Web pages. • DHML used events, methods, properties to insulate dynamism in HTML Pages. • DHML is basically using JavaScript and style sheets in an HTML page. • HTML sites will be slow upon client-side technologies, while DHTML sites will be fast enough upon client-side technologies. • HTML creates a plain page without any styles and Scripts called as HTML. Whereas, DHTML creates a page with HTML, CSS, DOM and Scripts called as DHTML. • HTML cannot have any server side code but DHTML may contain server side code. • In HTML, there is no need for database connectivity, but DHTML may require connecting to a database as it interacts with user. • HTML files are stored with .htm or .html extension, while DHTML files are stored with .dhtm extension. • HTML does not require any processing from browser, while DHTML requires processing from browser which changes its look and feel.
14.	**Point** out the key features of DHTML. **Key Features:** Following are the some major key features of DHTML: • Tags and their properties can be changed using DHTML. • It is used for real-time positioning. • Dynamic fonts can be generated using DHTML. • It is also used for data binding. • It makes a webpage dynamic and be used to create animations, games, applications along with providing new ways of navigating through websites. • The functionality of a webpage is enhanced due to the usage of low-bandwidth effect by DHTML. • DHTML also facilitates the use of methods, events, properties, and codes.
15.	**Classify** the data types in JSON **JSON** (JavaScript **Object** Notation) is a lightweight **data**-interchange **format**. It is easy for humans to read and write. ... **JSON** is built on two **structures**: A collection of name/value pairs. In various languages, this is realized as an **object**, record, struct, dictionary, hash table, keyed list, or associative array.

	JSON Data Types. At the granular level, **JSON** consist of 6 **data types**. First four **data types** (string, number, boolean and null) can be referred as simple **data types**. Other two **data types** (**object** and array) can be referred as complex **data types**.

16.	**How** to convert text into a JavaScript object using JSON?
	Converting JSON text to JavaScript Object
	
	JSON (JavaScript Object Notation) is a lightweight data-interchange format. As its name suggests, JSON is derived from the JavaScript programming language, but it's available for use by many languages including Python, Ruby, PHP, and Java and hence, it can be said as language-independent. For humans, it is easy to read and write and for machines, it is easy to parse and generate. It is very useful for storing and exchanging data.
	
	A JSON object is a key-value data format that is typically rendered in curly braces. JSON object consist of curly braces (**{ }**) at the either ends and have key-value pairs inside the braces. Each key-value pair inside braces are separated by comma (**,**). JSON object looks something like this :
	`{`
	`"key":"value",`
	`"key":"value",`
	`"key":"value",`
	`}`
	
	Example for a JSON object :
	
	`{`
	`"rollno":101",`
	`"name":"Mayank",`
	`"age":20,`
	`}`
	
	Conversion of JSON text to Javascript Object
	
	JSON text/object can be converted into Javascript object using the function **JSON.parse()**.
	
	`var object1 = JSON.parse('{"rollno":101, "name":"Mayank", "age":20}');`
	
	For getting the value of any key from a Javascript object, we can use the values as: **object1.rollno**

17.	**Evaluate** the syntax to create arrays in JSON.

JavaScript | JSON Arrays

The JSON Arrays is similar to JavaScript Arrays.

Syntax of Arrays in JSON Objects:

```
// JSON Arrays Syntax

{
    "name":"Peter parker",
    "heroName": "Spiderman",
    "friends" : ["Deadpool", "Hulk", "Wolverine"]
}
```

Accessing Array Values:

The Array values can be accessed using the index of each element in an Array.

EXAMPLE

```
<script>
var myObj, i, x = "";
myObj = {
  "name":"John",
  "age":30,
  "cars":[ "Ford", "BMW", "Fiat" ]
};

for (i in myObj.cars) {
  x += myObj.cars[i] + "<br>";
}
document.getElementById("demo").innerHTML = x;
</script>
```

OUTPUT

Looping through an array using a for in loop:

Ford
BMW
Fiat

18. **How** will you make a request with JSON?

What is a JSON request?
JavaScript Object Notation (**JSON**) is a standard text-based format for representing structured data based on JavaScript object syntax. It is commonly used for transmitting data in web applications (e.g., sending some data from the server to the client, so it can be displayed on a web page, or vice versa).

jQuery getJSON() Method

	The jQuery getJSON() method sends asynchronous http GET request to the server and retrieves the data in JSON format by setting accepts header to application/json, text/javascript. This is same as get() method, the only difference is that getJSON() method specifically retrieves JSON data whereas get() method retrieves any type of data. It is like shortcut method to retrieve JSON data. $.getJSON(url,[data],[callback]); Parameter Description: • url: request url from which you want to retrieve the data • data: JSON data to be sent to the server as a query string • callback: function to be executed when request succeeds The following example shows how to retrieve JSON data using getJSON() method. Example: jQuery getJSON() Method $.getJSON('/jquery/getjsondata', {name:'Steve'}, function (data, textStatus, jqXHR){ $('p').append(data.firstName); }); **OUTPUT** **jQuery get() method demo** Steve
19.	**Define** DDL and DML **DDL(Data Definition Language) :** DDL or Data Definition Language actually consists of the SQL commands that can be used to define the database schema. It simply deals with descriptions of the database schema and is used to create and modify the structure of database objects in the database. **Examples of DDL commands:** • **CREATE** – is used to create the database or its objects (like table, index, function, views, store procedure and triggers). • **DROP** – is used to delete objects from the database. • **ALTER**-is used to alter the structure of the database. • **TRUNCATE**–is used to remove all records from a table, including all spaces allocated for the records are removed. • **COMMENT** –is used to add comments to the data dictionary. • **RENAME** –is used to rename an object existing in the database
20.	**Write** SQL query to find minimum and maximum marks in a table. **DML(Data Manipulation Language) :** The SQL commands that deals with the manipulation of data present in the database belong to DML or Data Manipulation Language and this includes most of the SQL statements.

	Examples of DML: • **INSERT** – is used to insert data into a table. • **UPDATE** – is used to update existing data within a table. • **DELETE** – is used to delete records from a database table.

PART - B

Q.No	Questions

1.	i) **Examine** variables and data types in JavaScript.
	Variables in JavaScript:
	Variables in JavaScript are containers which hold reusable data. It is the basic unit of storage in a program.
	• The value stored in a variable can be changed during program execution.
	• A variable is only a name given to a memory location, all the operations done on the variable effects that memory location.
	• In JavaScript, all the variables must be declared before they can be used.
	JavaScript variables are containers for storing data values.
	In this example, x, y, and z, are variables:
	Examples
	`var x = 5;`
	`var y = 6;`
	`var z = x + y;`
	`var price1 = 5;`
	`var price2 = 6;`
	`var total = price1 + price2;`
	Declaring (Creating) JavaScript Variables
	Creating a variable in JavaScript is called "declaring" a variable.
	You declare a JavaScript variable with the var keyword:
	`var carName;`
	After the declaration, the variable has no value (technically it has the value of undefined).
	To **assign** a value to the variable, use the equal sign:
	`carName = "Volvo";`
	You can also assign a value to the variable when you declare it:
	`var carName = "Volvo";`

EXAMPLE

<script> var carName = "Volvo"; document.getElementById("demo").innerHTML = carName; </script>	**OUTPUT** JavaScript Variables Create a variable, assign a value to it, and display it: Volvo

ii) **Give** various Operators in JavaScript.

JavaScript Operators

JavaScript Arithmetic Operators

Arithmetic operators are used to perform arithmetic on numbers:

Operator	Description	Example

+	Addition	var x = 5;
		var y = 2;
		var z = x + y;

-	Subtraction	var x = 5;
		var y = 2;
		var z = x - y;

*	Multiplication	var x = 5;
		var y = 2;
		var z = x * y;

**	Exponentiation (ES2016)	
/	Division	
%	Modulus (Division Remainder)	
++	Increment	
--	Decrement	

JavaScript Assignment Operators

Assignment operators assign values to JavaScript variables.

Operator	Example	Same As

=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y
**=	x **= y	x = x ** y

The **addition assignment** operator (+=) adds a value to a variable.

Assignment

```
var x = 10;
x += 5;
```

JavaScript String Operators

The + operator can also be used to add (concatenate) strings.

Example

```
var txt1 = "John";
var txt2 = "Doe";
var txt3 = txt1 + " " + txt2;
```

The result of txt3 will be:

John Doe

The += assignment operator can also be used to add (concatenate) strings:

Example

```
var txt1 = "What a very ";
txt1 += "nice day";
```

The result of txt1 will be:

What a very nice day

When used on strings, the + operator is called the concatenation operator.

JavaScript Comparison Operators

Operator	Description
==	equal to
===	equal value and equal type
!=	not equal
!==	not equal value or not equal type

>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
?	ternary operator

JavaScript Logical Operators

Operator	Description
&&	logical and
||	logical or
!	logical not

JavaScript Type Operators

Operator	Description
typeof	Returns the type of a variable
instanceof	Returns true if an object is an instance of an object type

Type operators are fully described in the **JS Type Conversion** chapter.

JavaScript Bitwise Operators

Bit operators work on 32 bits numbers.

Any numeric operand in the operation is converted into a 32 bit number. The result is converted back to a JavaScript number.

Operator	Description	Example	Same as	Result	Decimal
&	AND	5 & 1	0101 & 0001	0001	1
|	OR	5 | 1	0101 | 0001	0101	5
~	NOT	~ 5	~0101	1010	10
^	XOR	5 ^ 1	0101 ^ 0001	0100	4
<<	Zero fill left shift	5 << 1	0101 << 1	1010	10
>>	Signed right shift	5 >> 1	0101 >> 1	0010	2

	>>>	Zero fill right shift	5 >>> 1	0101 >>> 1	0010	2

The examples above uses 4 bits unsigned examples. But JavaScript uses 32-bit signed numbers. Because of this, in JavaScript, ~ 5 will not return 10. It will return -6. ~00000000000000000000000000000101 will return 11111111111111111111111111111010

2.

(i) **Summariz**e about DOM Model.

What is the DOM?

The DOM is a W3C (World Wide Web Consortium) standard.

The DOM defines a standard for accessing documents:

"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."

The W3C DOM standard is separated into 3 different parts:

- Core DOM - standard model for all document types
- XML DOM - standard model for XML documents
- HTML DOM - standard model for HTML documents

JavaScript HTML DOM

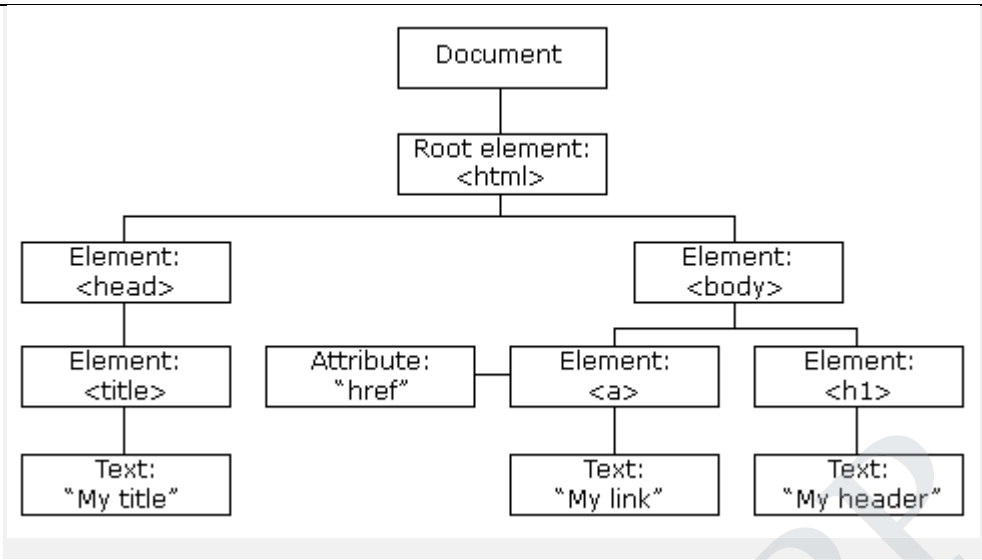
With the HTML DOM, JavaScript can access and change all the elements of an HTML document.

The HTML DOM (Document Object Model)

When a web page is loaded, the browser creates a **D**ocument **O**bject **M**odel of the page.

The **HTML DOM** model is constructed as a tree of **Objects**:

The HTML DOM Tree of Objects



With the object model, JavaScript gets all the power it needs to create dynamic HTML:

- JavaScript can change all the HTML elements in the page
- JavaScript can change all the HTML attributes in the page
- JavaScript can change all the CSS styles in the page
- JavaScript can remove existing HTML elements and attributes
- JavaScript can add new HTML elements and attributes
- JavaScript can react to all existing HTML events in the page
- JavaScript can create new HTML events in the page

(ii) **Describe** the concepts of Popup Boxes.

JavaScript Popup Boxes

JavaScript has three kind of popup boxes: Alert box, Confirm box, and Prompt box.

JavaScript Popup Boxes

JavaScript has three kind of popup boxes: Alert box, Confirm box, and Prompt box.

Alert Box	Syntax	Example
An alert box is often used if you want to make sure information comes through to the user. When an alert box pops up, the user will have to click "OK" to proceed	window.alert("*sometext*"); The window.alert() method can be written without the window prefix	alert("I am an alert box!");

	Confirm Box
	
	When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.
	
	If the user clicks "OK", the box returns **true**. If the user clicks "Cancel", the box returns **false**.
		Syntax
		
		window.confirm("*sometext*");
		
		The window.confirm() method can be written without the window prefix.
			Example
			
			`if (confirm("Press a button!")) {`
			`  txt = "You pressed OK!";`
			`} else {`
			`  txt = "You pressed Cancel!";`
			`}`

	Prompt Box
	
	A prompt box is often used if you want the user to input a value before entering a page.
	
	If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.
		Syntax
		
		window.prompt("*sometext*","*defaultText*");
		
		The window.prompt() method can be written without the window prefix.
			Example
			
			`var person = prompt("Please enter your name", "Harry Potter");`
			
			`if (person == null || person == "") {`
			`  txt = "User cancelled the prompt.";`
			`} else {`
			`  txt = "Hello " + person + "! How are you today?";`
			`}`

3. (i) **Write** short notes on Date and Objects.

The **Date Object**. The **Date object** is a built-in **object** in **JavaScript** that stores the **date** and time. It provides a number of built-in methods for formatting and managing that data. By default, a new **Date** instance without arguments provided creates an **object** corresponding to the current **date** and time

The Date Object

The `Date` object is a built-in object in JavaScript that stores the date and time. It provides a number of built-in methods for formatting and managing that data.

By default, a new `Date` instance without arguments provided creates an object corresponding to the current date and time. To demonstrate JavaScript's `Date`, let's create a variable and assign the current date to it.

EXAMPLE

now.js

```
// Set variable to current date and time
const now = new Date();

// View the output
now;
```

```
Output
Wed Oct 18 2017 12:41:34 GMT+0000 (UTC)
```

Date Creation	Output
`new Date()`	Current date and time
`new Date(timestamp)`	Creates date based on milliseconds since Epoch time
`new Date(date string)`	Creates date based on date string
`new Date(year, month, day, hours, minutes, seconds, milliseconds)`	Creates date based on specified date and time

(ii) **Explain** in detail about Regular expressions.

A JavaScript Regular Expression (or **Regex**) is a sequence of characters that we can utilize to work effectively with strings. Using this syntax, we can:

- **search** for text in a string
- **replace** substrings in a string
- **extract** information from a string

4.

(i) **Describe** the control statements in Java.

Conditional Statements

Very often when you write code, you want to perform different actions for different decisions.

You can use conditional statements in your code to do this.

In JavaScript we have the following conditional statements:

- Use `if` to specify a block of code to be executed, if a specified condition is true
- Use `else` to specify a block of code to be executed, if the same condition is false
- Use `else if` to specify a new condition to test, if the first condition is false
- Use `switch` to specify many alternative blocks of code to be executed

JavaScript Loops

Loops are handy, if you want to run the same code over and over again, each time with a different value.

Often this is the case when working with arrays:

```
<script>

var cars = ["BMW", "Volvo", "Saab", "Ford", "Fiat", "Audi"];

var text = "";

var i;
for (i = 0; i < cars.length; i++) {
  text += cars[i] + "<br>";


}

document.getElementById("demo").innerHTML = text;

</script>
```

OUTPUT

JavaScript For Loop

BMW
Volvo
Saab
Ford
Fiat
Audi

	(ii) **Discuss** any two validation functions in java script.
5.	(i) **Write** a Java script to find the Prime number between 1 and 100. `<html>` `  <head>` `  <title>JavaScript Prime</title>` `  </head>` `  <body>` `  <script>` `  for (var limit = 1; limit <= 20; limit++) {` `  var a = false;` `  for (var i = 2; i <= limit; i++) {` `  if (limit%i===0 &amp;&amp; i!==limit) {` `  a = true;` `  }` `  }` `  if (a === false) {` `  document.write(" "+limit);` `  }` `  }` `  </script>` `  </body>` `</html>` (ii) **Write** a Java Script to find factorial of a given number. `function factorial(n) {` `  return (n != 1) ? n * factorial(n - 1) : 1;` `}` `alert( factorial(5) ); // 120`

6.	(i) **Demonstrate** a java script for displaying the context menu.
	
	(ii) **Demonstrate** a java script to display the welcome message using the alert whenever button of a html form is pressed.
	<html>
	<head>
	<title>Display Alert Message on Button Click Event.</title>
	<script type="text/javascript">
	function showMessage(){
	alert("Welcome friends, You pressed the Button.");
	}
	</script>
	</head>
	<body>

```
        <center>
                <h1>Display Alert Message on Button Click Event.</h1>
                <b>Click on button to display message: </b><br><br>
                <input type="button" id="btnShowMsg" value="Click Me!"
onClick='showMessage()'/>
        </center>
</body>

</html>
```

Result

Welcome friends, You pressed the Button.

7.

(i) Evaluate how DOM Nodes and Trees can be used.

What is the purpose of HTML DOM Node Tree?

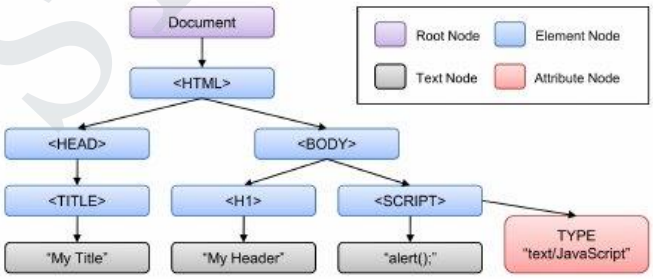
- HTML DOM view the HTML document with a tree structure format and it consists of root node and child nodes.

- The node-tree is being accessed using the tree formation and the structure in which the elements get created.

- The contents that are used being modified or removed using the new elements and it can be created within the limitations.

- The structure consists of a document that is the root and within it Root element <html> from where the tree starts.

- It consists of sub-elements like <head> and <body> and other text and attributes written in the HTML format.



(ii) Evaluate the way of Traversing and modifying a DOM Tree.

8.

(i) **Discuss** the concepts of Registering Event handlers.
As mentioned above, **events** are actions or occurrences that happen in the system you are programming — the system produces (or "fires") a signal of some kind when an event occurs,

and also provides a mechanism by which some kind of action can be automatically taken (that is, some code running) when the event occurs. For example in an airport when the runway is clear for a plane to take off, a signal is communicated to the pilot, and as a result, they commence piloting the plane.

In the case of the Web, events are fired inside the browser window, and tend to be attached to a specific item that resides in it — this might be a single element, set of elements, the HTML document loaded in the current tab, or the entire browser window. There are a lot of different types of events that can occur, for example:

The user clicking the mouse over a certain element or hovering the cursor over a certain element.

The user pressing a key on the keyboard.

The user resizing or closing the browser window.

A web page finishing loading.

A form being submitted.

A video being played, or paused, or finishing play.

An error occurring.

Each available event has an **event handler**, which is a block of code (usually a JavaScript function that you as a programmer create) that will be run when the event fires. When such a block of code is defined to be run in response to an event firing, we say we are **registering an event handler**. Note that event handlers are sometimes called **event listeners** — they are pretty much interchangeable for our purposes, although strictly speaking, they work together. The listener listens out for the event happening, and the handler is the code that is run in response to it happening.
In the following example, we have a single `<button>`, which when pressed, makes the background change to a random color:

```
<button>Change color</button>
```

The JavaScript looks like so:

```
const btn = document.querySelector('button');

function random(number) {
  return Math.floor(Math.random() * (number+1));
}

btn.onclick = function() {
  const rndCol = 'rgb(' + random(255) + ',' + random(255) + ',' +
random(255) + ')';
  document.body.style.backgroundColor = rndCol;
}
```

(ii) **Discuss** the concepts of Event Handling.

JavaScript Events

HTML events are **"things"** that happen to HTML elements.

When JavaScript is used in HTML pages, JavaScript can **"react"** on these events.

HTML Events

An HTML event can be something the browser does, or something a user does.

Here are some examples of HTML events:

- An HTML web page has finished loading
- An HTML input field was changed
- An HTML button was clicked

Often, when events happen, you may want to do something.

JavaScript lets you execute code when events are detected.

HTML allows event handler attributes, **with JavaScript code**, to be added to HTML elements.

With single quotes:

```
<element event='some JavaScript'>
```

With double quotes:

```
<element event="some JavaScript">
```

In the following example, an onclick attribute (with code), is added to a <button> element:

Example

```
<button onclick="document.getElementById('demo').innerHTML = Date()">The time is?</button>
```

OUTPUT

Sun Jan 12 2020 06:00:41 GMT+0530 (India Standard Time)

In the example above, the JavaScript code changes the content of the element with id="demo".

9. **Analyze** a web page to create a clock with timing event.

```
function showTime(){
    var date = new Date();
    var h = date.getHours(); // 0 - 23
    var m = date.getMinutes(); // 0 - 59
```

```
    var s = date.getSeconds(); // 0 - 59
    var session = "AM";

    if(h == 0){
        h = 12;
    }

    if(h > 12){
        h = h - 12;
        session = "PM";
    }

    h = (h < 10) ? "0" + h : h;
    m = (m < 10) ? "0" + m : m;
    s = (s < 10) ? "0" + s : s;

    var time = h + ":" + m + ":" + s + " " + session;
    document.getElementById("MyClockDisplay").innerText = time;
    document.getElementById("MyClockDisplay").textContent = time;

    setTimeout(showTime, 1000);

}

showTime();
```

```
<html>
<head>
<title>Digital Clock</title>
<style>
#clock{
        color:#F0F;
}

</style>
</head>
<body>
<script>
function digclock()
{
        var d = new Date()
        var t = d.toLocaleTimeString()
```

```
        document.getElementById("clock").innerHTML = t
}


setInterval(function(){digclock()},1000)


</script>
Digital Clock
<div id="clock">


</div>
</body>
</html>
```

OUTPUT

Output :

Digital Clock
5:54:26 PM

10. (i) **Write** short notes on DHTML with JavaScript.

Dynamic HyerText Markup Language (DHTML) is a combination of Web development technologies used to create dynamically changing websites. Web pages may include animation, dynamic menus and text effects. The technologies used include a combination of HTML, JavaScript or VB Script,
CSS and the document object model (DOM).

Designed to enhance a Web user's experience, DHTML includes the following features:

- Dynamic content, which allows the user to dynamically change Web page content
- Dynamic positioning of Web page elements
- Dynamic style, which allows the user to change the Web page's color, font, size or content

EXAMPLE

```
<! DOCTYPE html PUBLIC "-//abc//DTD XHTML 1.1//EN"
"http://www.abc.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.abc.org/1999/xhtml">
<head>
<title>DHTML example</title>
<script type="text/JavaScript">
    function greet_user()
    {
        var name=document.getElementById("userName").value;
```

```
    if(name==””)
        {
            alert(“Welcome”+name);
        }
        else
        {
            alert(“Please provide User Name”)
        }
    }
</script>
</head>
    <body>
        <table border=”1” cellspacing=”3”>
        <tr>
            <td colspan=”2”><h6> Please Enter Your Name </h6></td>
        </tr>
        <tr>
            <td><h4>User Name </h4></td>
            <td><input type=”text” id=”userName” ></td>
        </tr>
        <tr>
            <td colspan=”2”><input type=”button” value=”Submit”
onclick=”greet_user()”/>
        </table>
    </body>
</html>
```

(ii) **Classify** moving elements.

11. **Explain** the concept of JSON with example.

JSON is a format for storing and transporting data.

JSON is often used when data is sent from a server to a web page.

What is JSON?

- JSON stands for **J**ava**S**cript **O**bject **N**otation
- JSON is a lightweight data interchange format
- JSON is language independent *
- JSON is "self-describing" and easy to understand

* The JSON syntax is derived from JavaScript object notation syntax, but the JSON format is text only. Code for reading and generating JSON data can be written in any programming language.

JSON Example

This JSON syntax defines an employees object: an array of 3 employee records (objects):

JSON Example

```
{
"employees":[
  {"firstName":"John", "lastName":"Doe"},
  {"firstName":"Anna", "lastName":"Smith"},
  {"firstName":"Peter", "lastName":"Jones"}
]
}
```

12. **Describe** in detail about JSON Objects and Arrays.

JSON Arrays

Arrays as JSON Objects

Example

```
[ "Ford", "BMW", "Fiat" ]
```

Arrays in JSON are almost the same as arrays in JavaScript.

In JSON, array values must be of type string, number, object, array, boolean or *null*.

In JavaScript, array values can be all of the above, plus any other valid JavaScript expression, including functions, dates, and *undefined.*

Arrays in JSON Objects

Arrays can be values of an object property:

Example

```
{
"name":"John",
"age":30,
"cars":[ "Ford", "BMW", "Fiat" ]
}
```

Accessing Array Values

You access the array values by using the index number:

Example

```
x = myObj.cars[0];
```

Looping Through an Array

You can access array values by using a **for-in** loop:

Example

```
for (i in myObj.cars) {
  x += myObj.cars[i];
}
```

OUTPUT

Looping through an array using a for in loop:

Ford
BMW
Fiat

13. **Analyze** about Function files and Http Request with sample program.

```
<!doctype html>
<title>Example</title>

<script>
// Store XMLHttpRequest and the JSON file location in variables
var xhr = new XMLHttpRequest();
var url = "https://www.quackit.com/json/tutorial/artists.txt";

// Called whenever the readyState attribute changes
xhr.onreadystatechange = function() {

  // Check if fetch request is done
  if (xhr.readyState == 4 && xhr.status == 200) {

    // Parse the JSON string
    var jsonData = JSON.parse(xhr.responseText);

    // Call the showArtists(), passing in the parsed JSON string
    showArtists(jsonData);
  }
};
```

```
// Do the HTTP call using the url variable we specified above
xhr.open("GET", url, true);
xhr.send();

// Function that formats the string with HTML tags, then outputs the result
function showArtists(data) {
   var output = "<ul>"; // Open list
   var i;

   // Loop through the artists, and add them as list items
   for (var i in data.artists) {
      output += "<li>" + data.artists[i].artistname + " (Born: " + data.artists[i].born + ")</li>";
   }

   output += "</ul>"; // Close list

   // Output the data to the "artistlist" element
   document.getElementById("artistList").innerHTML = output;
}
</script>

<!-- The output appears here -->
<div id="artistList"></div>
```

OUTPUT

- Leonard Cohen (Born: 1934)
- Joe Satriani (Born: 1956)
- Snoop Dogg (Born: 1971)

14.

Summarize about

(i) SQL Data Definition Commands

Examples of Sql DDL commands are

- CREATE – Create an object. I mean, create a database, table, triggers, index, functions, stored procedures, etc.
- DROP – This SQL DDL command helps to delete objects. For example, delete tables, delete a database, etc.
- ALTER – Used to alter the existing database or its object structures.
- TRUNCATE – This SQL DDL command removes records from tables
- RENAME – Renaming the database objects

(ii) Data Manipulation Commands.

Examples of DML commands are

- SELECT – This SQL DML command select records or data from a table
- INSERT – Insert data into a database table.
- UPDATE – This SQL DML command will update existing records within a table

	• DELETE – Delete unwanted records from a table

PART – C

Q.No	Questions

1.	**Analyze** the merits and demerits of DOM parser with neat example.
	
	<!DOCTYPE html>
	<html>
	<head>
	<script type="text/javascript">
	var str = "<root><customer name='John' address='Chicago'></customer></root>";
	function CreateXMLDocument () {
	var xmlDoc = null;
	if (window.DOMParser) {
	var parser = new DOMParser ();
	xmlDoc = parser.parseFromString (str, "text/xml");
	} else if (window.ActiveXObject) {
	xmlDoc = new ActiveXObject ("Microsoft.XMLDOM");
	xmlDoc.async = false;
	xmlDoc.loadXML (str);
	}
	
	var customerNode = xmlDoc.getElementsByTagName ("customer")[0];
	var customerName = customerNode.getAttribute ("name");
	alert ("The name of the first customer is " + customerName);
	}
	</script>
	</head>
	<body>
	<button onclick="CreateXMLDocument ();">Create an XML document from a string</button>
	</body>
	</html>

2.	**Consider** a java script and HTML to create a page with two panes. The first pane (on left) should have a text area where HTML code can bet typed by the user. The pane on the left should have a text area where HTML code can be typed by the user. The pane on the right side should display the preview of the HTML code typed by the user, as it would be seen on the browser.
3.	**Develop** a DHTML program to handle the user click event. <!DOCTYPE html> <html>

	<body> Click the button to display the time. <button onclick="getElementById('demo').innerHTML=Date()">What is the time?</button> </body> </html> **OUTPUT** Click the button to display the time. What is the time? Sat Jan 11 2020 22:36:24 GMT+0530 (India Standard Time)
4.	**Formulate** a JavaScript program that work with JSON. **JSON Object Syntax** Example { "name":"John", "age":30, "car":null } ➢ JSON objects are surrounded by curly braces {}. ➢ JSON objects are written in key/value pairs. ➢ Keys must be strings, and values must be a valid JSON data type (string, number, object, array, boolean or null). ➢ Keys and values are separated by a colon. ➢ Each key/value pair is separated by a comma. **Accessing Object Values** You can access the object values by using dot (.) notation: Example myObj = { "name":"John", "age":30, "car":null }; x = myObj.name; You can also access the object values by using bracket ([]) notation: Example

```
myObj = { "name":"John", "age":30, "car":null };
x = myObj["name"];
```

Looping an Object

You can loop through object properties by using the for-in loop:

Example

In a for-in loop, use the bracket notation to access the property *values*:

Example

```
<!DOCTYPE html>
<html>
<body>

<p>How to loop through all properties in a JSON object.</p>

<p id="demo"></p>

<script>
var myObj, x;
myObj = {"name":"John", "age":30, "car":null};
for (x in myObj) {
  document.getElementById("demo").innerHTML += x + " " + myObj[x] + "<br>";
}
</script>

</body>
</html
```

OUTPUT

How to loop through all properties in a JSON object.

name John
age 30
car null

Nested JSON Objects

Values in a JSON object can be another JSON object.

Example

```
myObj = {
  "name":"John",
```

```
  "age":30,
 "cars": {
   "car1":"Ford",
   "car2":"BMW",
   "car3":"Fiat"
 }
}
```

You can access nested JSON objects by using the dot notation or bracket notation:

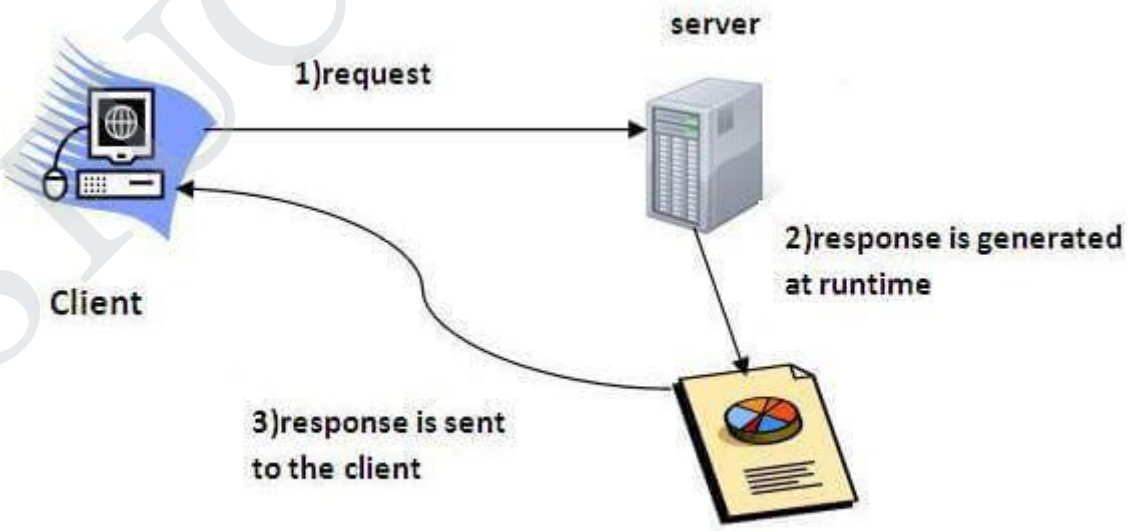
Example

```
x = myObj.cars.car2;
// or:
x = myObj.cars["car2"];
```

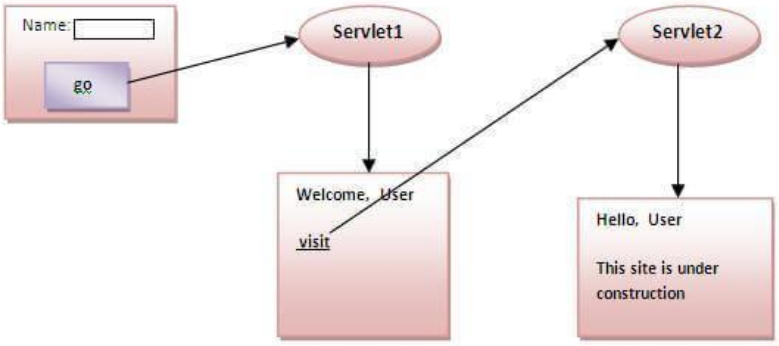
OUTPUT

How to access nested JSON objects.

BMW
BMW

CS8651 Internet Programming – 2017Reg	
UNIT III **- SERVER SIDE PROGRAMMING**	
Servlets: Java Servlet Architecture – Servlet Life Cycle – Form GET and POST actions – Session Handling – Understanding Cookies – Installing and Configuring Apache Tomcat Web Server; DATABASE CONNECTIVITY: JDBC perspectives, JDBC program example; **JSP:** Understanding Java Server Pages – JSP Standard Tag Library (JSTL) – Creating HTML forms by embedding JSP code.	
PART-A	
Q.No	**Questions**
1.	**What** are servlets? A servlet is a Java programming language class that is used to extend the capabilities of servers that host applications accessed by means of a request-response programming model. Although servlets can respond to any type of request, they are commonly used to extend the applications hosted by web servers. For such applications, Java Servlet technology defines HTTP-specific servlet classes. ➢ Servlet is a technology which is used to create a web application. ➢ Servlet is an API that provides many interfaces and classes including documentation. ➢ Servlet is an interface that must be implemented for creating any Servlet. ➢ Servlet is a class that extends the capabilities of the servers and responds to the incoming requests. It can respond to any requests. ➢ Servlet is a web component that is deployed on the server to create a dynamic web page. 
2.	**List** the application of servlets. Servlets may be used at different levels on a distributed framework. The following are some examples of servlet usage: ➢ To accept form input and generate HTML Web pages dynamically.

	➢ As part of middle tiers in enterprise networks by connecting to SQL databases via JDBC. ➢ In conjunction with applets to provide a high degree of interactivity and dynamic Web content generation. ➢ For collaborative applications such as online conferencing. ➢ A community of servlets could act as active agents which share data with each other. ➢ Servlets could be used for balancing load among servers which mirror the same content. ➢ Protocol support is one of the most viable uses for servlets. For example, a file service can start with NFS and move on to as many protocols as desired; the transfer between the protocols would be made transparent by servlets. Servlets could be used for tunneling over HTTP to provide chat, newsgroup or other file server functions.
3.	**Summarize** the advantages and disadvantages of servlets. **Servlet Advantage** ➢ Servlets provide a way to generate dynamic documents that is both easier to write and faster to run. ➢ Provide all the powerfull features of JAVA, such as Exception handling and garbage collection. ➢ Servlet enables easy portability across Web Servers. ➢ Servlet can communicate with different servlet and servers. ➢ Since all web applications are stateless protocol, servlet uses its own API to maintain session **Servlet Disadvantage** ➢ Designing in servlet is difficult and slows down the application. ➢ Writing complex business logic makes the application difficult to understand. ➢ You need a Java Runtime Environment on the server to run servlets. ➢ CGI is a completely language independent protocol, so you can write ➢ CGIs in whatever languages you have available (including Java if you want to).
4.	**Show** how is session tracking is achieved by the URL rewriting? **URL Rewriting** In URL rewriting, we append a token or identifier to the URL of the next Servlet or the next resource. We can send parameter name/value pairs using the following format: url?name1=value1&name2=value2&?? A name and a value is separated using an equal = sign, a parameter name/value pair is separated from another parameter using the ampersand(&). When the user clicks the hyperlink, the parameter name/value pairs will be passed to the server. From a Servlet, we can use getParameter() method to obtain a parameter value.



Advantage of URL Rewriting

1. It will always work whether cookie is disabled or not (browser independent).
2. Extra form submission is not required on each pages.

Disadvantage of URL Rewriting

1. It will work only with links.
2. It can send Only textual information.

URL Rewriting

URL rewriting is another way to support anonymous session tracking. With URL rewriting, every local URL the user might click on is dynamically modified, or rewritten, to include extra information. The extra information can be in the form of extra path information, added parameters, or some custom, server-specific URL change.

Example 7-2 shows a revised version of our shopping cart viewer that uses URL rewriting in the form of extra path information to anonymously track a shopping cart.

Example 7-2. Session tracking using URL rewriting import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class ShoppingCartViewerRewrite extends HttpServlet { public void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException { res.setContentType("text/html");
PrintWriter out = res.getWriter(); out.println(""); out.println("");
// Get the current session ID, or generate one if necessary
String sessionid = req.getPathInfo();
if (sessionid == null)
{ sessionid = generateSessionId();

```
}
// Cart items are associated with the session ID
String[] items = getItemsFromCart(sessionid);
 // Print the current cart items.
out.println("You currently have the following items in your cart:
");
 if (items == null) { out.println("None"); } else { out.println("
        ");
        for (int i = 0; i < items.length; i++)
        { out.println("

    • " + items[i]); } out.println("

"); }
// Ask if the user wants to add more items or check out. // Include the session
ID in the action URL.
 out.println("
");
out.println("Would you like to
");
out.println("\"); out.println("\"); out.println("
        ");

        // Offer a help page. Include the session ID in the URL. out.println("For
        help, click here");

        out.println(""); }

        private static String generateSessionId() { String uid = new
        java.rmi.server.UID().toString();

        // guaranteed unique return java.net.URLEncoder.encode(uid);

         // encode any special chars }

        private static String[] getItemsFromCart(String sessionid) {

        // Not implemented }

         }

        This servlet first tries to retrieve the current session ID using
        getPathInfo() . If a session ID is not specified, it calls
        generateSessionId() to generate a new unique session ID using an RMI
```

class designed specifically for this. The session ID is used to fetch and display the current items in the cart. The ID is then added to the form's ACTION attribute, so it can be retrieved by the ShoppingCart servlet. The session ID is also added to a new help URL that invokes the Help servlet. This wasn't possible with hidden form fields because the Help servlet isn't the target of a form submission. docstore.mik.ua/orelly/java-ent/servlet/ch07_03.htm

5. **Compare** GET and POST request type.

GET	POST
1) In case of Get request, only **limited amount of data** can be sent because data is sent in header.	In case of post request, **large amount of data** can be sent because data is sent in body.
2) Get request is **not secured** because data is exposed in URL bar.	Post request is **secured** because data is not exposed in URL bar.
3) Get request **can be bookmarked.**	Post request **cannot be bookmarked.**
4) Get request is **idempotent** . It means second request will be ignored until response of first request is delivered	Post request is **non-idempotent.**
5) Get request is **more efficient** and used more than Post.	Post request is **less efficient** and used less than get.

6. **Summarize** the servlet interface and its methods.

Servlet Interface

1. Servlet Interface
2. Methods of Servlet interface

- **Servlet interface provides** common behavior to all the servlets.
- Servlet interface defines methods that all servlets must implement.

- Servlet interface needs to be implemented for creating any servlet (either directly or indirectly).
- It provides 3 life cycle methods that are used to initialize the servlet, to service the requests, and to destroy the servlet and 2 non-life cycle methods.

Methods of Servlet interface

There are 5 methods in Servlet interface. The init, service and destroy are the life cycle methods of servlet. These are invoked by the web container.

Method	Description
public void init(ServletConfig config)	initializes the servlet. It is the life cycle method of servlet and invoked by the web container only once.
public void service(ServletRequest request,ServletResponse response)	provides response for the incoming request. It is invoked at each request by the web container.
public void destroy()	is invoked only once and indicates that servlet is being destroyed.
public ServletConfig getServletConfig()	returns the object of ServletConfig.
public String getServletInfo()	returns information about servlet such as writer, copyright, version etc.

Servlet Example by implementing Servlet interface

File: First.java

```
1.  import java.io.*;
2.  import javax.servlet.*;
3.
4.  public class First implements Servlet{
5.  ServletConfig config=null;
6.
7.  public void init(ServletConfig config){
8.  this.config=config;
9.  System.out.println("servlet is initialized");
10. }
```

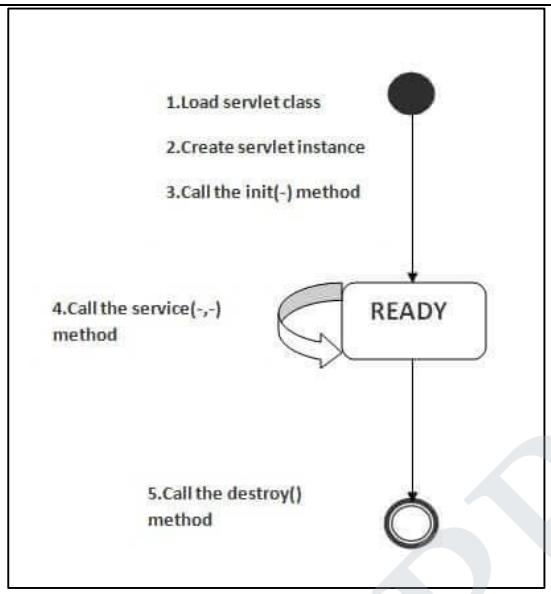
```
11.
12.  public void service(ServletRequest req,ServletResponse res)
13.  throws IOException,ServletException{
14.
15.  res.setContentType("text/html");
16.
17.  PrintWriter out=res.getWriter();
18.  out.print("<html><body>");
19.  out.print("<b>hello simple servlet</b>");
20.  out.print("</body></html>");
21.
22.  }
23.  public void destroy(){System.out.println("servlet is destroyed");}
24.  public ServletConfig getServletConfig(){return config;}
25.  public String getServletInfo(){return "copyright 2007-1010";}
26.
27.  }
```

7. **Sketch** the Servlet life cycle.

Life Cycle of a Servlet (Servlet Life Cycle)

The web container maintains the life cycle of a servlet instance. Let's see the life cycle of the servlet:

A. Life Cycle of a Servlet
   1. Servlet class is loaded
   2. Servlet instance is created
   3. init method is invoked
   4. service method is invoked
   5. destroy method is invoked



As displayed in the above diagram, there are three states of a servlet: new, ready and end. The servlet is in new state if servlet instance is created. After invoking the init() method, Servlet comes in the ready state. In the ready state, servlet performs all the tasks. When the web container invokes the destroy() method, it shifts to the end state.

8. **Show** the use of 'param' variable in JSP.

Jsp param

<jsp:param> tag is used to represent parameter value during jsp forward or include action this should be the sub tag of <jsp:forward> or <jsp:include>.

When an include or forward element is invoked, the original request object is provided to the target page. If you wish to provide additional data to that page, you can append parameters to the request object by using the jsp:param element.

Syntax

```
<jsp:param name=" " value=" "/>
```

Example

```
<jsp:include page="contact.jsp"/>
<jsp:param name="param1" value="value1"/>
</jsp:include>
```

Example

```
<jsp:forward page="home.jsp"/>
```

```
<jsp:param name="param1" value="value1"/>
</jsp:forward>
```

9.

Quote the uses of cookies.

Cookies in Servlet

A **cookie** is a small piece of information that is persisted between the multiple client requests.

A cookie has a name, a single value, and optional attributes such as a comment, path and domain qualifiers, a maximum age, and a version number.

How Cookie works

By default, each request is considered as a new request. In cookies technique, we add cookie with response from the servlet. So cookie is stored in the cache of the browser. After that if request is sent by the user, cookie is added with request by default. Thus, we recognize the user as the old user.

Types of Cookie

There are 2 types of cookies in servlets.

1. Non-persistent cookie
2. Persistent cookie

Non-persistent cookie

It is **valid for single session** only. It is removed each time when user closes the browser.

Persistent cookie

It is **valid for multiple session** . It is not removed each time when user closes the browser. It is removed only if user logout or signout.

Advantage of Cookies

1. Simplest technique of maintaining the state.
2. Cookies are maintained at client side.

Disadvantage of Cookies

1. It will not work if cookie is disabled from the browser.

	2. Only textual information can be set in Cookie object.
10.	**Analyze** about java scriplet. In JavaServer Pages (JSP) technology, a **scriptlet** is a piece of **Java**-code embedded in the HTML-like JSP code. The **scriptlet** is everything inside the <% %> tags. Between these the user can add any valid **Scriptlet** i.e. any valid **Java** Code. In AppleScript, a **scriptlet** is a small script. JSP scriptlet tag A scriptlet tag is used to execute java source code in JSP. Syntax is as follows: <% java source code %> Example of JSP scriptlet tag In this example, we are displaying a welcome message. <html> <body> <% out.print("welcome to jsp"); %> </body> </html>

11.	**Express** appropriate java script code to remove an element (current element) from a DOM.
	
	A. Removing Existing HTML Elements
	
	To remove an HTML element, use the `remove()` method:
	
	1. **Example**
	
	 This is a paragraph.
	 This is another paragraph.
	
	<script>
	var elmnt = document.getElementById("p1");
	elmnt.remove();
	</script>

12.	**Define** JSP.

	JavaServer Pages (JSP) is a technology for developing Webpages that supports dynamic content. This helps developers insert java code in HTML pages by making use of special JSP tags, most of which start with <% and end with %>. A JavaServer Pages component is a type of Java servlet that is designed to fulfill the role of a user interface for a Java web application. Web developers write JSPs as text files that combine HTML or XHTML code, XML elements, and embedded JSP actions and commands. Using JSP, you can collect input from users through Webpage forms, present records from a database or another source, and create Webpages dynamically. JSP tags can be used for a variety of purposes, such as retrieving information from a database or registering user preferences, accessing JavaBeans components, passing control between pages, and sharing information between requests, pages etc.
13.	**List** any three advantages of java servlet over JSP. 1. Being an extension to Java servlet, it can use every feature of Java Servlet. Also, custom tags can be used along with it. 2. There is no need to recompile JSP when changed. The changes automatically appear when run. 3. The tags which are used are easy to understand and write. 4. Supports Java API's which can now be easily used and integrated with the HTML code. 5. The results which are obtained are in HTML format, so can be opened on any browsers. 6. Customized JSP tags can be used. **Ex:** Tags with XML. 7. Changes can be added into the business logic page rather than changing in each and every page.
14.	**Rewrite** the code segment to store current server time in session using Java Servlet API. **Write a servlet application to print the current date and time.** **Answer:** The most important advantage of using Servlet is that we can use all the

methods available in core java. The Date class is available in java.util package.

Below program shows how to print the current date and time. We can use simple Date object with toString() to print current date and time.

DateSrv.java

```
import java.io.*;
import javax.servlet.*;

public class DateSrv extends GenericServlet
{
    //implement service()
    public void service(ServletRequest req, ServletResponse res) throws
IOException, ServletException
    {
        //set response content type
        res.setContentType("text/html");
        //get stream obj
        PrintWriter pw = res.getWriter();
        //write req processing logic
        java.util.Date date = new java.util.Date();
        pw.println("<h2>"+"Current Date & Time: "
+date.toString()+"</h2>");
        //close stream object
        pw.close();
    }
}
```

Output:

Current Date & Time: Mon Dec 12 18:01:39 IST 2016

15.	**Compare** the difference between JSP and servlet.

Difference Between Servlet and JSP

In this article we will list some of the differences between Servlets and JSP.

SERVLET	JSP
A servlet is a server-side program and written purely on Java.	JSP is an interface on top of Servlets. In another way, we can say that JSPs are extension of servlets to minimize the effort of developers to write User Interfaces using Java programming.
Servlets run faster than JSP	JSP runs slower because it has the transition phase for converting from JSP page to a Servlet file. Once it is converted to a Servlet then it will start the compilation
Executes inside a Web server, such as Tomcat	A JSP program is compiled into a Java servlet before execution. Once it is compiled into a servlet, it's life cycle will be same as of servlet. But, JSP has it's own API for the lifecycle.
Receives HTTP requests from users and provides HTTP responses	Easier to write than servlets as it is similar to HTML.
We can not build any custom tags	One of the key advantage is we can build custom tags using JSP API (there is a separate

	package available for writing the custom tags) which can be available as the re-usable components with lot of flexibility
Servlet advantages include: **1. Performance :** get loaded upon first request and remains in memory idenfinately. **2. Simplicity :** Run inside controlled server environment. No specific client software is needed:web broser is enough **3. Session Management** : overcomes HTTP's stateless nature **4. Java Technology :** network access,Database connectivity, j2ee integration	**JSP Provides an extensive infrastructure for:** 1. Tracking sessions. 2. Managing cookies. 3. Reading and sending HTML headers. 4. Parsing and decoding HTML form data. 5. **JSP is Efficient:** Every request for a JSP is handled by a simple Java thread 6. **JSP is Scalable:** Easy integration with other backend services 7. **Seperation of roles:** Developers, Content Authors/Graphic Designers/Web Masters

II. Difference between Servlet and JSP

In brief, it can be defined as Servlet are the java programs that run on a Web server and act as a middle layer between a request coming from HTTP client and databases or applications on the HTTP server.While JSP is simply a text document that contains two types of text: static text which is predefined and dynamic text which is rendered after server response is received.

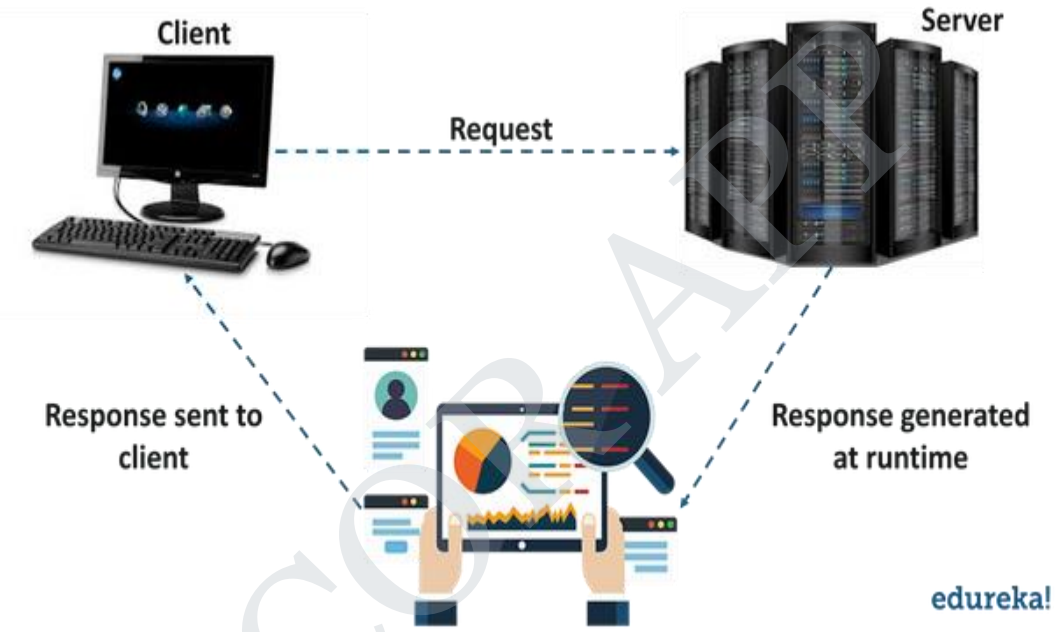
sr. No.	Key	Servlet	JSP
1	Implementation	Servlet is developed on Java language.	JSP is primarily written in HTML language although Java code could also be written on it but for it, JSTL or other language is required.
2	MVC	In contrast to MVC we can state servlet as a controller which receives the request process and send back the response.	On the other hand, JSP plays the role of view to render the response returned by the servlet.
3	Request type	Servlets can accept and process all type of protocol requests.	JSP on the other hand is compatible with HTTP request only.
4	Session Management	In Servlet by default session management is not enabled, the user has to enable it explicitly.	On the other hand in JSP session management is automatically enabled.
5	Performance	Servlet is faster than JSP.	JSP is slower than Servlet because first the translation of JSP to java code is taking place and then compiles.
6	Modification reflected	Modification in Servlet is a time-consuming task because it includes reloading, recompiling and restarting the server as we made any change in our code to get reflected.	On the other hands JSP modification is fast as just need to click the refresh button and code change would get reflected.

16.

Summarize briefly about the interaction between a webserver and a servlet.

How does a web server interact with a servlet?

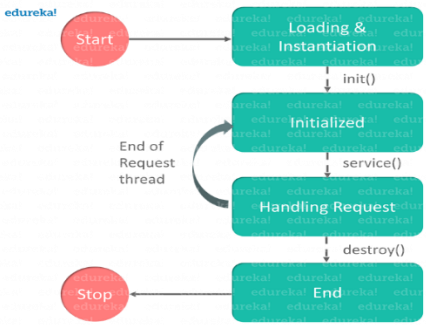
A servlet is a **Java Programming** language class that is used to extend the capabilities of servers that host applications accessed by means of a request-response programming model. Although servlets can respond to any type of request, they are commonly used to extend the applications hosted by web servers. It is also a web component that is deployed on the server to create a dynamic web page.



In this figure you can see, a client sends a request to the server and the server generates the response, analyses it and sends the response to the client.

Stages of the Servlet Life Cycle: The Servlet life cycle mainly goes through four stages,

- Loading a Servlet.
- Initializing the Servlet.
- Request handling
- Destroying the Servlet.



	Let's look at each of these stages in details: 1. **Loading a Servlet**: The first stage of the Servlet life cycle involves loading and initializing the Servlet by the Servlet container. The Web container or Servlet Container can load the Servlet at either of the following two stages :Initializing the context, on configuring the Servlet with a zero or positive integer value.If the Servlet is not preceding the stage, it may delay the loading process until the Web container determines that this Servlet is needed to service a request. 2. **Initializing a Servlet**: After the Servlet is instantiated successfully, the Servlet container initializes the instantiated Servlet object. The container initializes the Servlet object by invoking the *init(ServletConfig)* method which accepts ServletConfig object reference as a parameter. 3. **Handling request**: After initialization, the Servlet instance is ready to serve the client requests. The Servlet container performs the following operations when the Servlet instance is located to service a request :It creates the **ServletRequest** and **ServletResponse.** In this case, if this is an HTTP request then the Web container creates **HttpServletRequest** and **HttpServletResponse** objects which are subtypes of the **ServletRequest** and **ServletResponse** objects respectively. 4. **Destroying a Servlet**: When a Servlet container decides to destroy the Servlet, it performs the following operations,It allows all the threads currently running in the service method of the Servlet instance to complete their jobs and get released.After currently running threads have completed their jobs, the Servlet container calls the **destroy()** method on the Servlet instance. After the **destroy()** method is executed, the Servlet container releases all the references of this Servlet instance so that it becomes eligible for garbage collection.
17.	**Define** JDBC. What is JDBC? JDBC stands for **J**ava **D**ata**b**ase **C**onnectivity, which is a standard Java API for database-independent connectivity between the Java programming language and a wide range of databases. The JDBC library includes APIs for each of the tasks mentioned below that are commonly associated with database usage. • Making a connection to a database. • Creating SQL or MySQL statements. • Executing SQL or MySQL queries in the database. • Viewing & Modifying the resulting records.

	Fundamentally, JDBC is a specification that provides a complete set of interfaces that allows for portable access to an underlying database. Java can be used to write different types of executables, such as − • Java Applications • Java Applets • Java Servlets • Java ServerPages (JSPs) • Enterprise JavaBeans (EJBs). All of these different executables are able to use a JDBC driver to access a database, and take advantage of the stored data. JDBC provides the same capabilities as ODBC, allowing Java programs to contain database-independent code.

18.	**Formulate** the three methods that are central to the life cycle of the servlet.
	Servlets - Life Cycle
	A servlet life cycle can be defined as the entire process from its creation till the destruction. The following are the paths followed by a servlet.
	• The servlet is initialized by calling the **init()** method.
	• The servlet calls **service()** method to process a client's request.
	• The servlet is terminated by calling the **destroy()** method.
	• Finally, servlet is garbage collected by the garbage collector of the JVM.
	Now let us discuss the life cycle methods in detail.
	The init() Method
	The init method is called only once. It is called only when the servlet is created, and not called for any user requests afterwards. So, it is used for one-time initializations, just as with the init method of applets.
	The servlet is normally created when a user first invokes a URL corresponding to the servlet, but you can also specify that the servlet be loaded when the server is first started.
	When a user invokes a servlet, a single instance of each servlet gets created, with each user request resulting in a new thread that is handed off to doGet or doPost as appropriate. The init() method simply creates or loads some data that will be used throughout the life of the servlet.
	The init method definition looks like this −
	`public void init() throws ServletException {`
	`  // Initialization code...`
	`}`

The service() Method

The service() method is the main method to perform the actual task. The servlet container (i.e. web server) calls the service() method to handle requests coming from the client(browsers) and to write the formatted response back to the client.

Each time the server receives a request for a servlet, the server spawns a new thread and calls service. The service() method checks the HTTP request type (GET, POST, PUT, DELETE, etc.) and calls doGet, doPost, doPut, doDelete, etc. methods as appropriate.

A. The destroy() Method

The destroy() method is called only once at the end of the life cycle of a servlet. This method gives your servlet a chance to close database connections, halt background threads, write cookie lists or hit counts to disk, and perform other such cleanup activities.

After the destroy() method is called, the servlet object is marked for garbage collection. The destroy method definition looks like this −

```
public void destroy() {
  // Finalization code...
}
```

19. **Distinguish** between servlets and JSP.

Difference between Servlet and JSP

SERVLET	JSP
Servlet is a java code.	JSP is a html based code.
Writing code for servlet is harder than JSP as it is html in java.	JSP is easy to code as it is java in html.
Servlet plays a controller role in MVC approach.	JSP is the view in MVC approach for showing output.
Servlet is faster than JSP.	JSP is slower than Servlet because the first step in JSP lifecycle is the translation of JSP to java code and then compile.
Servlet can accept all protocol requests.	JSP only accept http requests.
In Servlet, we can override the service() method.	In JSP, we cannot override its service() method.

	In Servlet by default session management is not enabled, user have to enable it explicitly.	In JSP session management is automatically enabled.
	In Servlet we have to implement everything like business logic and presentation logic in just one servlet file.	In JSP business logic is separated from presentation logic by using javaBeans.
	Modification in Servlet is a time consuming task because it includes reloading, recompiling and restarting the server.	JSP modification is fast, just need to click the refresh button.

20. **Discuss** the need to use JSTL tags?

III. JSTL (JSP Standard Tag Library)

The JSP Standard Tag Library (JSTL) represents a set of tags to simplify the JSP development.

A. Advantage of JSTL

1. **Fast Development** JSTL provides many tags that simplify the JSP.
2. **Code Reusability** We can use the JSTL tags on various pages.
3. **No need to use scriptlet tag** It avoids the use of scriptlet tag.

B. JSTL Tags

There JSTL mainly provides five types of tags:

Tag Name	Description

	Core tags	The JSTL core tag provide variable support, URL management, flow control, etc. The URL for the core tag is **http://java.sun.com/jsp/jstl/core**. The prefix of core tag is **c**.
	Function tags	The functions tags provide support for string manipulation and string length. The URL for the functions tags is **http://java.sun.com/jsp/jstl/functions** and prefix is **fn**.
	Formatting tags	The Formatting tags provide support for message formatting, numbe date formatting, etc. The URL for the Formatting tags is **http://java.sun.com/jsp/jstl/fmt** and prefix is **fmt**.
	XML tags	The XML tags provide flow control, transformation, etc. The URL for t XML tags is **http://java.sun.com/jsp/jstl/xml** and prefix is **x**.
	SQL tags	The JSTL SQL tags provide SQL support. The URL for the SQL tags is **http://java.sun.com/jsp/jstl/sql** and prefix is **sql**.

PART-B

Q.No	Questions
1.	(i) **Integrate** how servlets work and its life cycle. (ii) Explain and **develop** the Servlet API.

2.	(i) **Analyze** a JavaScript to find factorial of a given number.
	
	`function factorial(x)`
	`{`
	
	`  if (x === 0)`
	`{`
	`  return 1;`
	`}`
	`  return x * factorial(x-1);`
	
	`}`
	`console.log(factorial(5));`
	
	`OUTPUT :  120`

(ii) **Differentiate** GET and POST method.

Get vs. Post

There are many differences between the Get and Post request. Let's see these differences:

GET	POST
1) In case of Get request, only **limited amount of data** can be sent because data is sent in header.	In case of post request, **large amount of data** can be sent because data is sent in body.
2) Get request is **not secured** because data is exposed in URL bar.	Post request is **secured** because data is not exposed in URL bar.
3) Get request **can be bookmarked.**	Post request **cannot be bookmarked.**
4) Get request is **idempotent** . It means second request will be ignored until response of first request is delivered	Post request is **non-idempotent.**
5) Get request is **more efficient** and used more than Post.	Post request is **less efficient** and used less than get.

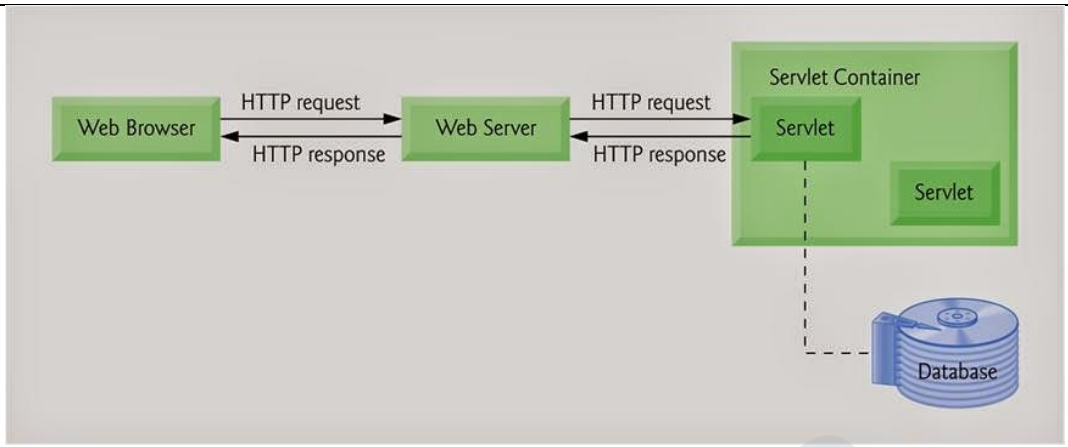
GET and POST

Two common methods for the request-response between a server and client are:

- **GET**- It requests the data from a specified resource
- **POST**- It submits the processed data to a specified resource

3. **Demonstrate** the Servlet architecture and explain its working principle.

Servlet Architecture



Servlets read the explicit data sent by the clients (browsers). This includes an HTML form on a Web page or it could also come from an applet or a custom HTTP client program.

Read the implicit HTTP request data sent by the clients (browsers). This includes cookies, media types and compression schemes the browser understands, and so forth.

Process the data and generate the results. This process may require talking to a database, executing an RMI or CORBA call, invoking a Web service, or computing the response directly.

Send the explicit data (i.e., the document) to the clients (browsers). This document can be sent in a variety of formats, including text (HTML or XML), binary (GIF images), Excel, etc.

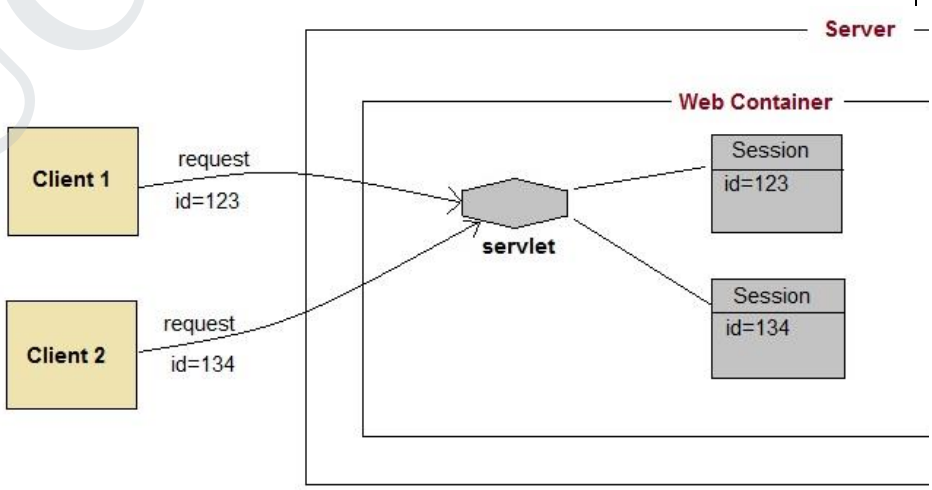
Send the implicit HTTP response to the clients (browsers). This includes telling the browsers or other clients what type of document is being returned (e.g., HTML), setting cookies and caching parameters, and other such tasks.

Servlet API:

Servelt API contains three packages

javax.servlet: Package contains a number of classes and interfaces that describe the contract
between a servlet class and the runtime environment provided for an instance of such a class a
conforming servelt container.
javax.servlet.aanotation: Package contains a number of annotations that allow users to use
annotations to declare servlets , filters, listeners and specify the metadata for the declared component
javax.servlet.http: Package contains a number of classes and interfaces that describe and define the contract between a servlet class rnning under the HTTP protocal and the runtime environment provided for an instance of such class by a confirming servlet container.

4. **Consider** a database that has a table Employee with two columns Employee Id and Name. Assume that the administrator user id and password to access to access the database table are Scott and Tiger. Write a JDBC program that can query and print all entries in the

	table employee. Make the database using type 2 driver database.driver and connection string jdbc :db.oci.
5.	**Describe** in detail the session handling in server side programming. IV. **Managing Session in Servlets** We all know that **HTTP** is a stateless protocol. All requests and responses are independent. But sometimes you need to keep track of client's activity across multiple requests. For eg. When a User logs into your website, not matter on which web page he visits after logging in, his credentials will be with the server, until he logs out. So this is managed by creating a session. **Session Management** is a mechanism used by the **Web container** to store session information for a particular user. There are four different techniques used by Servlet application for session management. They are as follows: 1. **Cookies** 2. **Hidden form field** 3. **URL Rewriting** 4. **HttpSession** Session is used to store everything that we can get from the client from all the requests the client makes. A. How Session Works  B. he basic concept behind session is, whenever a user starts using our application, we can save a unique identification information about him, in an object which is available throughout the application, until its destroyed. So wherever the user goes, we will always have his information and we can always manage which user is doing what. Whenever a user wants to exit from your application, destroy the object with his information.

6.

(i) **Discuss** about JSTL.

V. JSTL (JSP Standard Tag Library)

The JSP Standard Tag Library (JSTL) represents a set of tags to simplify the JSP development.

A. Advantage of JSTL

1. **Fast Development** JSTL provides many tags that simplify the JSP.
2. **Code Reusability** We can use the JSTL tags on various pages.
3. **No need to use scriptlet tag** It avoids the use of scriptlet tag.

B. JSTL Tags

There JSTL mainly provides five types of tags:

Tag Name	Description
Core tags	The JSTL core tag provide variable support, URL management, flow etc. The URL for the core tag is **http://java.sun.com/jsp/jstl/co** prefix of core tag is **c**.
Function tags	The functions tags provide support for string manipulation and strin The URL for the functions tags is **http://java.sun.com/jsp/jstl/functions** and prefix is **fn**.
Formatting tags	The Formatting tags provide support for message formatting, numb date formatting, etc. The URL for the Formatting tags is **http://java.sun.com/jsp/jstl/fmt** and prefix is **fmt**.
XML tags	The XML tags provide flow control, transformation, etc. The URL for tags is **http://java.sun.com/jsp/jstl/xml** and prefix is **x**.
SQL tags	The JSTL SQL tags provide SQL support. The URL for the SQL tags is **http://java.sun.com/jsp/jstl/sql** and prefix is **sql**.

(ii) **Summarize** a client server JSP program to find simple interest and display the result in client.

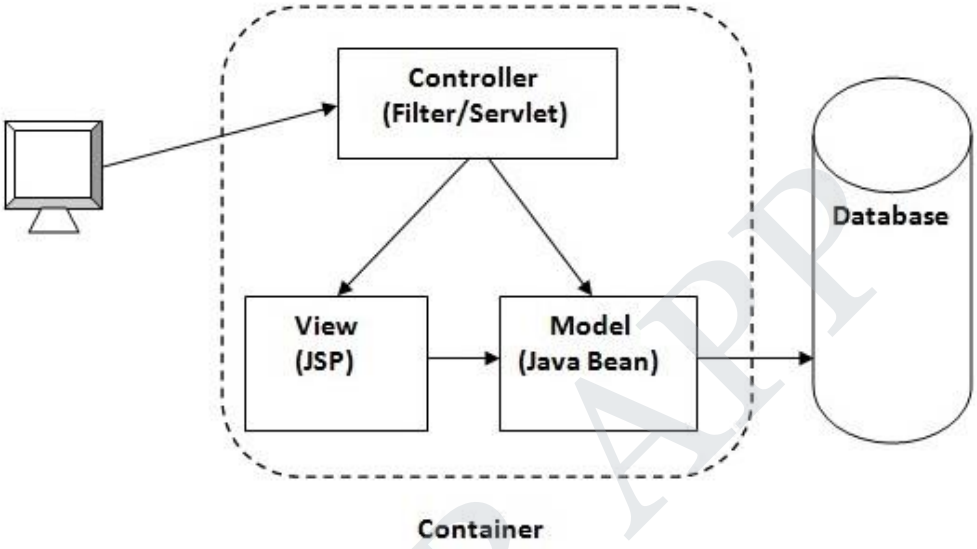
7.	**Explain** the use of cookies for tracking for tracking requests with a program. **Session Tracking in JSP** **Session Tracking :** HTTP is a "stateless" protocol which means each time a client retrieves a Web page, the client opens a new connection to the Web server and the server does not keep any record of previous client request.Session tracking is a mechanism that is used to maintain state about a series of requests from the same user(requests originating from the same browser) across some period of time. A session id is a unique token number assigned to a specific user for the duration of that user's session. **Need Of Session Tracking :** HTTP is a stateless protocol so When there is a series of continuous request and response from a same client to a server, the server cannot identify which client is sending request.If we want to maintain the conversational state,session tracking is needed. For example, in a shopping cart application a client keeps on adding items into his cart using multiple requests.When every request is made,the server should identify in which client's cart the item is to be added. So in this scenario, there is a certain need for session tracking. Solution is, when a client makes a request it should introduce itself by providing unique identifier every time.There are four ways to maintain session between web client and web server. **Methods to track session :** Cookies URL Rewriting Hidden Fields Session API **Cookies :** Cookies mostly used for session tracking. Cookie is a key value pair of information, sent by the server to the browser. This should be saved by the browser in its space in the client computer.

	Whenever the browser sends a request to that server it sends the cookie along with it. Then the server can identify the client using the cookie. This is not an effective way because many time browser does not support a cookie or users can opt to disable cookies using their browser preferences. In such case, the browser will not save the cookie at client computer and session tracking fails.
8.	(i) **Explain** about the standard actions in JSP. Actions are used for controlling the behavior of servlet engine. How many standard Action Tags are available in JSP? There are 11 types of Standard Action Tags as following: • jsp:useBean • jsp:include • jsp:setProperty • jsp:getProperty • jsp:forward • jsp:plugin • jsp:attribute • jsp:body • jsp:text • jsp:param • jsp:attribute • jsp:output (ii) **Analyze** MVC architecture of JSP. VI. **MVC in JSP** 1. MVC in JSP 2. Example of following MVC in JSP **MVC** stands for Model View and Controller. It is a **design pattern** that separates the business logic, presentation logic and data. **Controller** acts as an interface between View and Model. Controller intercepts all the incoming requests. **Model** represents the state of the application i.e. data. It can also have business logic.

View represents the presentaion i.e. UI(User Interface).

a) *Advantage of MVC (Model 2) Architecture*

1. Navigation Control is centralized
2. Easy to maintain the large application



9.

Explain in detail about Servlet database connectivity with an example of student database.

Example of Registration form in servlet

Table creation :

```
CREATE TABLE  "REGISTERUSER"
   (    "NAME" VARCHAR2(4000),
    "PASS" VARCHAR2(4000),
    "EMAIL" VARCHAR2(4000),
    "COUNTRY" VARCHAR2(4000)
    )
/
```

Example of Registration form in servlet

In this example, we have created the three pages.

- register.html
- Register.java

- web.xml

register.html

In this page, we have getting input from the user using text fields and combobox. The information entered by the user is forwarded to Register servlet, which is responsible to store the data into the database.

```
<html>
<body>
<form action="servlet/Register" method="post">

Name:<input type="text" name="userName"/><br/><br/>
Password:<input type="password" name="userPass"/><br/><br/>
Email Id:<input type="text" name="userEmail"/><br/><br/>
Country:
<select name="userCountry">
. <option>India</option>
. <option>Pakistan</option>
. <option>other</option>
. </select>
.
. <br/><br/>
. <input type="submit" value="register"/>
.
. </form>
. </body>
. </html>
```

Register.java

This servlet class receives all the data entered by user and stores it into the database. Here, we are performing the database logic. But you may separate it, which will be better for the web application.

```
import java.io.*;
import java.sql.*;
import javax.servlet.ServletException;
import javax.servlet.http.*;

public class Register extends HttpServlet {
public void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

. response.setContentType("text/html");
```

	. PrintWriter out = response.getWriter(); . . String n=request.getParameter("userName"); . String p=request.getParameter("userPass"); . String e=request.getParameter("userEmail"); . String c=request.getParameter("userCountry"); . . **try**{ . Class.forName("oracle.jdbc.driver.OracleDriver"); . Connection con=DriverManager.getConnection(. "jdbc:oracle:thin:@localhost:1521:xe","system","oracle"); . . PreparedStatement ps=con.prepareStatement(. "insert into registeruser values(?,?,?,?)"); . . ps.setString(1,n); . ps.setString(2,p); . ps.setString(3,e); . ps.setString(4,c); . . **int** i=ps.executeUpdate(); . **if**(i>0) . out.print("You are successfully registered..."); . . . }**catch** (Exception e2) {System.out.println(e2);} . . out.close(); . } . . }
10.	**Demonstrate** the procedure of installing and configuring Apache Tomcat. # How To Install Apache Tomcat 8 on Ubuntu 16.04 Apache Tomcat is a web server and servlet container that is used to serve Java applications. Tomcat is an open source implementation of the Java Servlet and JavaServer Pages technologies, released by the Apache Software Foundation. This

tutorial covers the basic installation and some configuration of the latest release of Tomcat 8 on your Ubuntu 16.04 server.

Step 1: Install Java

Tomcat requires Java to be installed on the server so that any Java web application code can be executed. We can satisfy that requirement by installing OpenJDK with apt-get.

First, update your apt-get package index:

```
sudo apt-get update
```

Then install the Java Development Kit package with apt-get:

```
sudo apt-get install default-jdk
```

Now that Java is installed, we can create a `tomcat` user, which will be used to run the Tomcat service.

Step 2: Create Tomcat User

For security purposes, Tomcat should be run as an unprivileged user (i.e. not root). We will create a new user and group that will run the Tomcat service.

First, create a new `tomcat` group:

```
sudo groupadd tomcat
```

Next, create a new `tomcat` user. We'll make this user a member of the `tomcat` group, with a home directory of `/opt/tomcat` (where we will install Tomcat), and with a shell of `/bin/false` (so nobody can log into the account):

```
sudo useradd -s /bin/false -g tomcat -d /opt/tomcat tomcat
```

Now that our `tomcat` user is set up, let's download and install Tomcat.

Step 3: Install Tomcat

The best way to install Tomcat 8 is to download the latest binary release then configure it manually.

We will install Tomcat to the `/opt/tomcat` directory. Create the directory, then extract the archive to it with these commands:

```
sudo mkdir /opt/tomcat
```

```
sudo tar xzvf apache-tomcat-8*tar.gz -C /opt/tomcat --strip-components=1
```

Next, we can set up the proper user permissions for our installation.

Step 4: Update Permissions

The `tomcat` user that we set up needs to have access to the Tomcat installation. We'll set that up now.

Change to the directory where we unpacked the Tomcat installation:

```
cd /opt/tomcat
```

Give the `tomcat` group ownership over the entire installation directory:

```
sudo chgrp -R tomcat /opt/tomcat
```

Next, give the `tomcat` group read access to the `conf` directory and all of its contents, and **execute** access to the directory itself:

```
sudo chmod -R g+r conf
sudo chmod g+x conf
```

Make the `tomcat` user the owner of the `webapps`, `work`, `temp`, and `logs` directories:

```
sudo chown -R tomcat webapps/ work/ temp/ logs/
```

Now that the proper permissions are set up, we can create a systemd service file to manage the Tomcat process.

Step 5: Create a systemd Service File

We want to be able to run Tomcat as a service, so we will set up systemd service file.

Tomcat needs to know where Java is installed. This path is commonly referred to as "JAVA_HOME". The easiest way to look up that location is by running this command:

Step 6: Adjust the Firewall and Test the Tomcat Server

Now that the Tomcat service is started, we can test to make sure the default page is available.

Before we do that, we need to adjust the firewall to allow our requests to get to the service. If you followed the prerequisites, you will have a `ufw` firewall enabled currently.

Tomcat uses port `8080` to accept conventional requests. Allow traffic to that port by typing:

```
sudo ufw allow 8080
```

Step 7: Configure Tomcat Web Management Interface

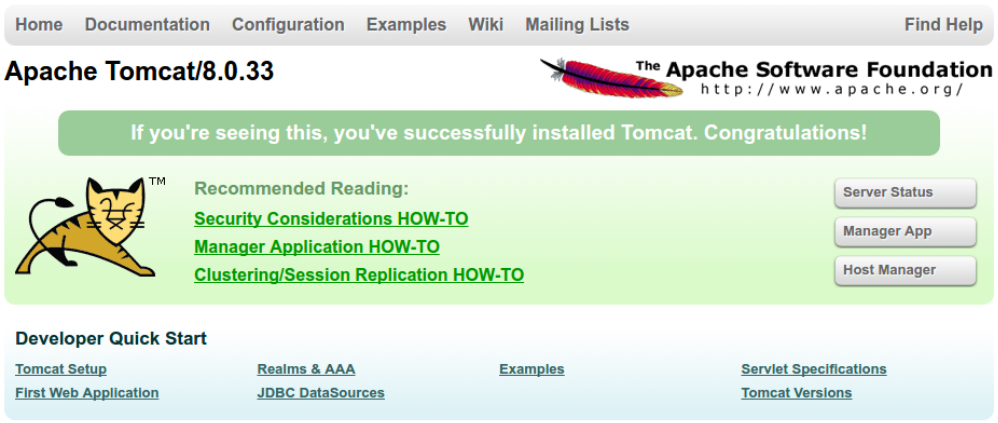
Step 8: Access the Web Interface

Now that we have create a user, we can access the web management interface again in a web browser. Once again, you can get to the correct interface by entering your server's domain name or IP address followed on port 8080 in your browser:

```
Open in web browser
```

```
http://server_domain_or_IP:8080
```

The page you see should be the same one you were given when you tested earlier:



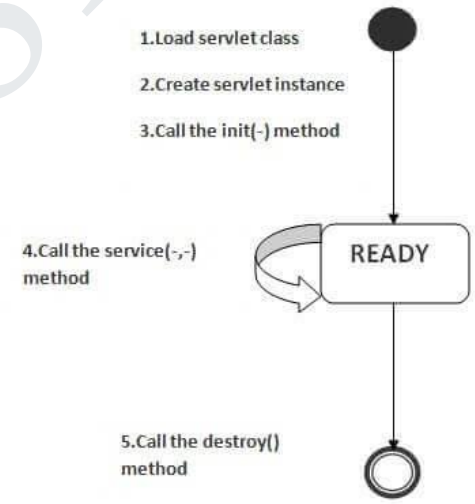
Your installation of Tomcat is complete! Your are now free to deploy your own Java web applications!

11. (i) **Discuss** about the Servlet life cycle.

VII. Life Cycle of a Servlet (Servlet Life Cycle)

The web container maintains the life cycle of a servlet instance. Let's see the life cycle of the servlet:

1. Servlet class is loaded.
2. Servlet instance is created.
3. init method is invoked.
4. service method is invoked.
5. destroy method is invoked.



	As displayed in the above diagram, there are three states of a servlet: new, ready and end. The servlet is in new state if servlet instance is created. After invoking the init() method, Servlet comes in the ready state. In the ready state, servlet performs all the tasks. When the web container invokes the destroy() method, it shifts to the end state. (ii) **List** JSP advantages. 1. **Advantages of JSP over Servlet** There are many advantages of JSP over the Servlet. They are as follows: a) ***1) Extension to Servlet*** JSP technology is the extension to Servlet technology. We can use all the features of the Servlet in JSP. In addition to, we can use implicit objects, predefined tags, expression language and Custom tags in JSP, that makes JSP development easy. b) ***2) Easy to maintain*** JSP can be easily managed because we can easily separate our business logic with presentation logic. In Servlet technology, we mix our business logic with the presentation logic. c) ***3) Fast Development: No need to recompile and redeploy*** If JSP page is modified, we don't need to recompile and redeploy the project. The Servlet code needs to be updated and recompiled if we have to change the look and feel of the application. d) ***4) Less code than Servlet*** In JSP, we can use many tags such as action tags, JSTL, custom tags, etc. that reduces the code. Moreover, we can use EL, implicit objects, etc.
12.	(i) **Explain** and write a simple JDBC program. Java Database Connectivity with MySQL To connect Java application with the MySQL database, we need to follow 5 following steps. 1. **Driver class:** The driver class for the mysql database is **com.mysql.jdbc.Driver**. 2. **Connection URL:** The connection URL for the mysql database is **jdbc:mysql://localhost:3306/sonoo** where jdbc is the API, mysql is the database, localhost is the server name on which mysql is running, we

may also use IP address, 3306 is the port number and sonoo is the database name. We may use any database, in such case, we need to replace the sonoo with our database name.

3. **Username:** The default username for the mysql database is **root**.
4. **Password:** It is the password given by the user at the time of installing the mysql database. In this example, we are going to use root as the password.

Let's first create a table in the mysql database, but before creating table, we need to create database first.

```
create database sonoo;
use sonoo;
create table emp(id int(10),name varchar(40),age int(3));
```

Example to Connect Java Application with mysql database

In this example, sonoo is the database name, root is the username and password both.

```
import java.sql.*;
class MysqlCon{
public static void main(String args[]){
try{
Class.forName("com.mysql.jdbc.Driver");
Connection con=DriverManager.getConnection(
"jdbc:mysql://localhost:3306/sonoo","root","root");
//here sonoo is database name, root is username and password
Statement stmt=con.createStatement();
. ResultSet rs=stmt.executeQuery("select * from emp");
. while(rs.next())
. System.out.println(rs.getInt(1)+"  "+rs.getString(2)+"  "+rs.getString(3));
. con.close();
. }catch(Exception e){ System.out.println(e);}
. }
. }
```

(ii) **List** various JSP scripting components.

VIII. JSP Scripting Element

JSP Scripting element are written inside <% %> tags. These code inside <% %> tags are processed by the JSP engine during translation of the JSP page. Any other text in the JSP page is considered as HTML code or plain text.

Example:

```
html>
  <head>
```

```
    <title>My First JSP Page</title>
  </head>
  <%
    int count = 0;
  %>
  <body>
    Page Count is <% out.println(++count); %>
  </body>
</html>
```

1. Types of scripting elements

Scripting Element	Example
Comment	<%-- comment --%>
Directive	<%@ directive %>
Declaration	<%! declarations %>
Scriptlet	<% scriplets %>
Expression	<%= expression %>

B. JSP Comment

JSP Comment is used when you are creating a JSP page and want to put in comments about what you are doing. JSP comments are only seen in the JSP page. These comments are not included in servlet source code during translation phase, nor they appear in the HTTP response. Syntax of JSP comment is as follows :

```
<%-- JSP  comment  --%>
```

Simple Example of JSP Comment

```
<html>
  <head>
    <title>My First JSP Page</title>
  </head>
  <%
    int count = 0;
  %>
  <body>
    <%-- Code to show page count  --%>
    Page Count is <% out.println(++count); %>
  </body>
</html>
```

13.	(i) **Demonstrate** with suitable example for core and formatting tags in JSTL.

JSTL Formatting tags

The formatting tags provide support for message formatting, number and date formatting etc. The url for the formatting tags is **http://java.sun.com/jsp/jstl/fmt** and prefix is **fmt.**

The JSTL formatting tags are used for internationalized web sites to display and format text, the time, the date and numbers. The syntax used for including JSTL formatting library in your JSP is:

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt"  %>
```

Formatting Tags	Descriptions
fmt:parseNumber	It is used to Parses the string representation of a currency, percentage or number.
fmt:timeZone	It specifies a parsing action nested in its body or the time zone for any time formatting.
fmt:formatNumber	It is used to format the numerical value with specific format or precision.
fmt:parseDate	It parses the string representation of a time and date.
fmt:bundle	It is used for creating the ResourceBundle objects which will be used by their tag body.
fmt:setTimeZone	It stores the time zone inside a time zone configuration variable.
fmt:setBundle	It loads the resource bundle and stores it in a bundle configuration variable or the named scoped variable.
fmt:message	It display an internationalized message.
fmt:formatDate	It formats the time and/or date using the supplied pattern and styles.

JSTL Core <c:choose>, <c:when>, <c:otherwise> Tag

The < c:choose > tag is a conditional tag that establish a context for mutually exclusive conditional operations. It works like a Java **switch** statement in which we choose between a numbers of alternatives.

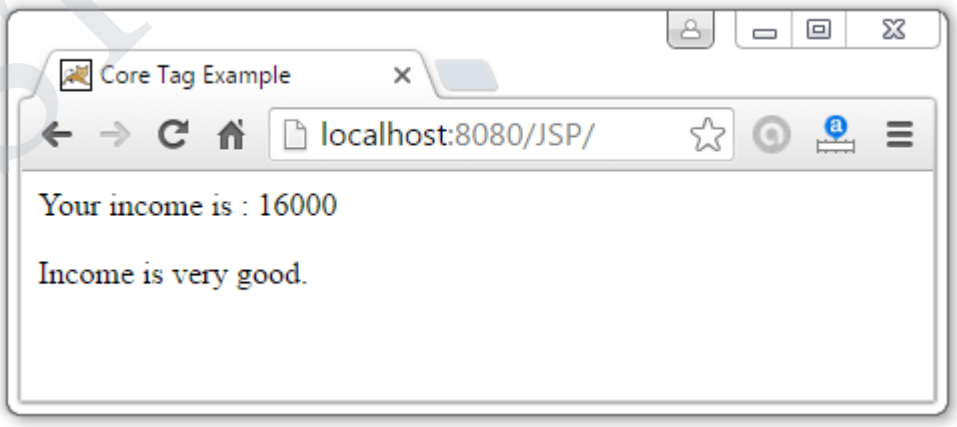
Example

Let's see the simple example of < c:choose >, < c:when > < c:otherwise > tag:

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
<title>Core Tag Example</title>
</head>
<body>
<c:set var="income" scope="session" value="${4000*4}"/>
<p>Your income is : <c:out value="${income}"/></p>
<c:choose>
    <c:when test="${income <= 1000}">
       Income is not good.
    </c:when>
    <c:when test="${income > 10000}">
        Income is very good.
    </c:when>
    <c:otherwise>
       Income is undetermined...
    </c:otherwise>
</c:choose>
</body>
</html>
```

This will produce the following result:

Your income is : 16000
Income is very good.



(ii) **Demonstrate** with suitable example for SQL and XML tags in JSTL.

JSTL SQL

The **<sql:setDataSource>** tag sets the data source configuration variable or saves the data-source information in a scoped variable that can be used as input to the other JSTL database actions.

Attribute

The **<sql:setDataSource>** tag has the following attributes −

Attribute	Description
driver	Name of the JDBC driver class to be registered
url	JDBC URL for the database connection
user	Database username
password	Database password
password	Database password
dataSource	Database prepared in advance
var	Name of the variable to represent the database
scope	Scope of the variable to represent the database

Example

Consider the following information about your MySQL database setup −

- We are using **JDBC MySQL** driver.
- We are going to connect to TEST database on local machine.
- We would use **user_id** and **mypassword** to access TEST database.

All the above parameters will vary based on your MySQL or any other database setup. Considering the above parameters, following example uses the **setDataSource** tag −

```
<%@ taglib uri = "http://java.sun.com/jsp/jstl/core" prefix = "c" %>

<%@ taglib uri = "http://java.sun.com/jsp/jstl/sql" prefix = "sql"%>
```

```
<html>
  <head>
    <title>JSTL sql:setDataSource Tag</title>
  </head>

  <body>
    <sql:setDataSource var = "snapshot" driver = "com.mysql.jdbc.Driver"
      url = "jdbc:mysql://localhost/TEST"
      user = "user_id"  password = "mypassword"/>
    <sql:query dataSource = "${snapshot}" sql = "..." var = "result" />

  </body>
</html>
```

JSTL XML

<x:parse> Tag

The <x:parse> tag is used for parse the XML data specified either in the tag body or an attribute. It is used for parse the xml content and the result will stored inside specified variable.

The syntax used for including the <x:parse> tag is:

<x:parse attributes**>** body content **</x:parse>**

EXAMPLE

Let us put the following content in **novels.xml** file:

```
<books>
<book>
  <name>Three mistakes of my life</name>
  <author>Chetan Bhagat</author>
  <price>200</price>
</book>
<book>
  <name>Tomorrow land</name>
  <author>NUHA</author>
  <price>2000</price>
```

</book>
</books>

index.jsp

(IN the same directory)

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %>

<html>
<head>
  <title>x:parse Tag</title>
</head>
<body>
<h2>Books Info:</h2>
<c:import var="bookInfo" url="novels.xml"/>

<x:parse xml="${bookInfo}" var="output"/>
<p>First Book title: <x:out select="$output/books/book[1]/name" /></p>
<p>First Book price: <x:out select="$output/books/book[1]/price" /></p>
<p>Second Book title: <x:out select="$output/books/book[2]/name" /></p>
<p>Second Book price: <x:out select="$output/books/book[2]/price" /></p>

</body>
</html>
```

Output:

Books Info:

First Book title: Three mistakes of my life

First Book price: 200

Second Book title: Tomorrow land

Second Book price: 2000

14. **Define** HTML and JSP. Use the same and design a scientific calculator.

Calculator.jsp

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
        <head>
                <title>Calculator</title>
```

```
		<style>
			h1 {
				font-family: Arial;
				font-size: 14pt;
				font-weight: normal;
			}
			td input {
				font-family: Arial;
				font-size: 10pt;
				width: 30px;
			}

			input.double {
				width: 60px;
			}

			input.doubleheight {
				height: 48px;
			}

			input.display {
				border: 1px solid black;
				readonly: true;
				width: 120px;
				padding: 2px;
			}
		</style>
	</head>

	<body>
		<h1>Calculator</h1>

		<form method="GET" action="calculate.html">
		<table border="0" cellpadding="0" cellspacing="0">
		<tr>
			<td colspan="4">
				<input class="display" type="text" id="display" value="<c:out
value="${displayAmount}"/>" readonly/>
			</td>
		</tr>
		<tr>
			<td><input type="submit" name="button" id="btn-7"
value="7"/></td>
			<td><input type="submit" name="button" id="btn-8"
value="8"/></td>
			<td><input type="submit" name="button" id="btn-9"
value="9"/></td>
			<td><input type="submit" name="button" id="btn-/"
value="/"/></td>
```

```
                        <td><input type="submit" name="button" id="btn-C"
value="C"/></td>
                    </tr>
                    <tr>
                        <td><input type="submit" name="button" id="btn-4"
value="4"/></td>
                        <td><input type="submit" name="button" id="btn-5"
value="5"/></td>
                        <td><input type="submit" name="button" id="btn-6"
value="6"/></td>
                        <td><input type="submit" name="button" id="btn-*"
value="*"/></td>
                        <td></td>
                    </tr>
                    <tr>
                        <td><input type="submit" name="button" id="btn-1"
value="1"/></td>
                        <td><input type="submit" name="button" id="btn-2"
value="2"/></td>
                        <td><input type="submit" name="button" id="btn-3"
value="3"/></td>
                        <td><input type="submit" name="button" id="btn--" value="-
"/></td>
                        <td rowspan="2"><input class="doubleheight" type="submit"
name="button" id="btn-=" value="="/></td>
                    </tr>
                    <tr>
                        <td colspan="2"><input class="double" type="submit"
name="button" id="btn-0" value="0"/></td>
                        <td><input type="button" name="button" id="btn-."
value="."/></td>
                        <td><input type="submit" name="button" id="btn-+"
value="+"/></td>
                    </tr>
                    </table>
                    </form>
            </body>
</html>
```

PART – C

Q.No	Questions
1.	**Design** a HTML forms by embedding JSP code for submission of a resume to a job portal website with appropriate database connectivity.
2.	**Evaluate** a complete query application for books database using JDBC.
3.	**Write** a program that allows the user to **select** a favourite programming language and post the choice to the server. The response is a web page in which the user can click a

	link to view a list of book recommendations. The cookies previously stored on the client are read by the servlet and form a web page containing the book recommendation. Use servlet cookies and HTML.
4.	**Develop** a JSP program to display the grade of a student by accepting the marks of five subjects.

CS8651 Internet Programming – 2017Reg	
PHP: An introduction to PHP – Using PHP – Variables – Program control – Built-in functions – Form Validation – Regular Expressions – File handling – Cookies – Connecting to Database; **XML:** Basic XML – Document Type Definition – XML Schema DOM and Presenting XML, XML Parsers and Validation, XSL and XSLT Transformation, News Feed (RSS and ATOM).	
Internet Programming – UNIT-IV	
Q.No	**Questions**
1.	**Define** PHP. List the features. ## What is PHP? • PHP is an acronym for "PHP: Hypertext Preprocessor" • PHP is a widely-used, open source scripting language • PHP scripts are executed on the server • PHP is free to download and use **PHP is an amazing and popular language!** It is powerful enough to be at the core of the biggest blogging system on the web (WordPress)! It is deep enough to run the largest social network (Facebook)! It is also easy enough to be a beginner's first server side language! ## What is a PHP File? • PHP files can contain text, HTML, CSS, JavaScript, and PHP code • PHP code is executed on the server, and the result is returned to the browser as plain HTML • PHP files have extension ".php" ## What Can PHP Do? • PHP can generate dynamic page content • PHP can create, open, read, write, delete, and close files on the server • PHP can collect form data • PHP can send and receive cookies • PHP can add, delete, modify data in your database • PHP can be used to control user-access • PHP can encrypt data With PHP you are not limited to output HTML. You can output images, PDF files, and even Flash movies. You can also output any text, such as XHTML and XML.

2.	**List** the rules for creating variables in PHP. ## PHP Variables A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume). Rules for PHP variables: • A variable starts with the $ sign, followed by the name of the variable • A variable name must start with a letter or the underscore character • A variable name cannot start with a number • A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _) • Variable names are case-sensitive ($age and $AGE are two different variables) • PHP variable names are case-sensitive
3.	**Illustrate** a PHP program to determine the type of browser that a web client is using. **Display the Browser – PHP Script** The following PHP function can be used to display the browser: **<?php** **function** get_the_browser**()** **{** **if(**strpos**(**$_SERVER**[**'HTTP_USER_AGENT'**]**, 'MSIE'**)** !== false**)** **return** 'Internet explorer'; **elseif(**strpos**(**$_SERVER**[**'HTTP_USER_AGENT'**]**, 'Trident'**)** !== false**)** **return** 'Internet explorer'; **elseif(**strpos**(**$_SERVER**[**'HTTP_USER_AGENT'**]**, 'Firefox'**)** !== false**)** **return** 'Mozilla Firefox'; **elseif(**strpos**(**$_SERVER**[**'HTTP_USER_AGENT'**]**, 'Chrome'**)** !== false**)** **return** 'Google Chrome'; **elseif(**strpos**(**$_SERVER**[**'HTTP_USER_AGENT'**]**, 'Opera Mini'**)** !== false**)** **return** "Opera Mini"; **elseif(**strpos**(**$_SERVER**[**'HTTP_USER_AGENT'**]**, 'Opera'**)** !== false**)** **return** "Opera"; **elseif(**strpos**(**$_SERVER**[**'HTTP_USER_AGENT'**]**, 'Safari'**)** !== false**)** **return** "Safari"; **else** **return** 'Other';

	} **?>** In the above code, we are checking each possible browser that may be and return the browser name. Here we haven't checked the Mozilla because of most of the browser using this as the user agent string. Below is how to display the browser name on our web page: Echo get_the_browser();
4.	**Name** any four built-in functions in PHP. PHP Reference The PHP reference contains different categories of all PHP functions and constants, along with examples. Array Calendar Date Directory Error File system Filter FTP Libxml Mail Math Misc MySQLi Network SimpleXML Stream String XML Parser Zip Timezones
5.	**Infer** when should the super global arrays in PHP be used? Superglobals were introduced in PHP 4.1.0, and are built-in variables that are always available in all scopes. PHP Global Variables - Superglobals

Some 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:

- $GLOBALS
- $_SERVER
- $_REQUEST
- $_POST
- $_GET
- $_FILES
- $_ENV
- $_COOKIE
- $_SESSION

Which super global array in PHP would contain a HTML form's POST data?

PHP Superglobal - $_POST

Super global variables are built-in variables that are always available in all scopes.

PHP $_POST

PHP $_POST is a PHP super global variable which is used to collect form data after submitting an HTML form with method="post". $_POST is also widely used to pass variables.

The example below shows a form with an input field and a submit button. When a user submits the data by clicking on "Submit", the form data is sent to the file specified in the action attribute of the <form> tag. In this example, we point to the file itself for processing form data. If you wish to use another PHP file to process form data, replace that with the filename of your choice. Then, we can use the super global variable $_POST to collect the value of the input field:

Example

```
<html>
<body>

<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
  Name: <input type="text" name="fname">
  <input type="submit">
</form>
```

```php
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // collect value of input field
    $name = $_POST['fname'];
    if (empty($name)) {
        echo "Name is empty";
    } else {
        echo $name;
    }
}
?>

</body>
</html>
```

6.

Classify the difference between echo() & print() functions.

PHP echo and print Statements

With PHP, there are two basic ways to get output: echo and print.

echo and print are more or less the same. They are both used to output data to the screen.

The differences are small:

The PHP echo Statement	**The PHP print Statement**
➢ echo has no return value ➢ echo can take multiple parameters ➢ echo is marginally faster than print.	➢ print has a return value of 1 so it can be used in expressions. ➢ print can take one argument
The echo statement can be used with or without parentheses: echo or echo().	The print statement can be used with or without parentheses: print or print().
Example-1 <?php echo " PHP is Fun! "; echo "Hello world! "; echo "I'm about to learn PHP! "; echo "This ", "string ", "was ", "made ", "with multiple parameters."; ?>	Example-1 <?php print " PHP is Fun! "; print "Hello world! "; print "I'm about to learn PHP!"; ?>
PHP is Fun! Hello world!	**PHP is Fun!** Hello world!

Example-2

```
<?php
$txt1 = "Learn PHP";
$txt2 = "W3Schools.com";
$x = 5;
$y = 4;

echo "<h2>" . $txt1 . "</h2>";
echo "Study PHP at " . $txt2 . "<br>";
echo $x + $y;
?>
```

Learn PHP
Study PHP at W3Schools.com
9

Example-2

```
<?php
$txt1 = "Learn PHP";
$txt2 = "W3Schools.com";
$x = 5;
$y = 4;

print "<h2>" . $txt1 . "</h2>";
print "Study PHP at " . $txt2 . "<br>";
print $x + $y;
?>
```

Learn PHP
Study PHP at W3Schools.com
9

7. **List** any two advantages of XML document.

Using XML to exchange information offers many benefits.

Advantages of XML include the following:

- XML uses human, not computer, language. XML is readable and understandable, even by novices, and no more difficult to code than HTML.
- XML is completely compatible with Java™ and 100% portable. Any application that can process XML can use your information, regardless of platform.
- XML is extendable. Create your own tags, or use tags created by others, that use the natural language of your domain, that have the attributes you need, and that makes sense to you and your users.

8. **Give** the difference between DTD and XML schema for defining XML document structure with appropriate examples.

DTD vs XSD

There are many differences between DTD (Document Type Definition) and XSD (XML Schema Definition). In short, DTD provides less control on XML structure whereas XSD (XML schema) provides more control.

The important differences are given below:

No.	**DTD**(Document Type Definition)	**XSD**(XML Schema Definition)
1)	DTD stands for **Document Type Definition**.	XSD stands for XML Schema Definition.
2)	DTDs are derived from **SGML** syntax.	XSDs are written in XML.

	3)	DTD **doesn't support datatypes**.	XSD **supports datatypes** for elements and attributes.
	4)	DTD **doesn't support namespace**.	XSD **supports namespace**.
	5)	DTD **doesn't define order** for child elements.	XSD **defines order** for child elements.
	6)	DTD is **not extensible**.	XSD is **extensible**.
	7)	DTD is **not simple to learn**.	XSD is **simple to learn** because you don't need to learn new language.
	8)	DTD provides **less control** on XML structure.	XSD provides **more control** on XML structure.

9. **Analyze** about Query String in PHP.

Query string

The information can be sent across the web pages. This information is called query string. This query string can be passed from one page to another by appending it to the address of the page. You can pass more than one query string by inserting the & sign between the query strings. A query string can contain two things: the query string ID and its value. The query string passed across the web pages is stored in $_REQUEST, $_GET, or $_POST variable.

Query string handling in PHP

Query strings

To access the data in a query string you can use the *$_GET* global array. Each element in this array has a key which is the name of the query string variable and a value which is the value of that variable.

```
<a href="mypage.php?variable1=value1&variable2=value2">my link</a>
```

	This link loads the page mypage.php with two variables *variable1* and *variable2* with values value1 and value2 respectively.

	echo $_GET['variable1'];
	echo $_GET['variable2'];
	// outputs:
	//value1
	//value2

	Form data The get method of forms sends the data to a page via a query string.
	<form name="form1" id="form1" method="get" action=""> <input name="textbox" id="textbox" type="text" value="value1" /> <input name="textbox2" id="textbox2" type="text" value="value2" /> <input type="submit" name="submitbutton" id="submitbutton" value="Submit" /> </form>
	This form passes the value of the two text boxes to the page myform.php.

	print_r($_GET);
	// outputs:
	// Array (
	// [textbox] => value1
	// [textbox2] => value2
	// [submitbutton] => Submit
	//)
	
	echo $_GET['textbox'];
	//outputs: value1

10.	**Show** an example for XML namespace. A **Namespace** is a set of unique names. Namespace is a mechanisms by which element and attribute name can be assigned to a group. The Namespace is identified by URI(Uniform Resource Identifiers).

Namespace Declaration

A Namespace is declared using reserved attributes. Such an attribute name must either be **xmlns** or begin with **xmlns:** shown as below −

```
<element xmlns:name = "URL">
```

Syntax

- The Namespace starts with the keyword **xmlns**.
- The word **name** is the Namespace prefix.
- The **URL** is the Namespace identifier.

Example

Namespace affects only a limited area in the document. An element containing the declaration and all of its descendants are in the scope of the Namespace. Following is a simple example of XML Namespace −

```
<?xml version = "1.0" encoding = "UTF-8"?>
<cont:contact xmlns:cont = "www.tutorialspoint.com/profile">
   <cont:name>Tanmay Patil</cont:name>
   <cont:company>TutorialsPoint</cont:company>
   <cont:phone>(011) 123-4567</cont:phone>
</cont:contact>
```

Here, the Namespace prefix is **cont**, and the Namespace identifier (URI) as *www.tutorialspoint.com/profile*. This means, the element names and attribute names with the **cont** prefix (including the contact element), all belong to the *www.tutorialspoint.com/profile* namespace.

11.

Define XML parse tree.

An XML document is always descriptive. The tree structure is often referred to as **XML Tree** and plays an important role to describe any XML document easily.

The tree structure contains root (parent) elements, child elements and so on. By using tree structure, you can get to know all succeeding branches and sub-branches starting from the root. The parsing starts at the root, then moves down the first branch to an element, take the first branch from there, and so on to the leaf nodes.

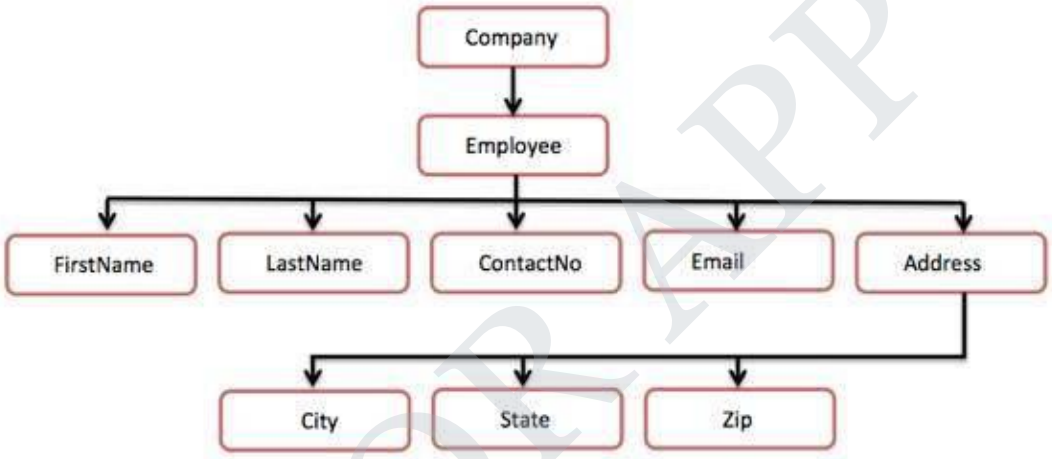
Example

Following example demonstrates simple XML tree structure −

```
<?xml version = "1.0"?>
<Company>
   <Employee>
      <FirstName>Tanmay</FirstName>
```

```
        <LastName>Patil</LastName>
        <ContactNo>1234567890</ContactNo>
        <Email>tanmaypatil@xyz.com</Email>
        <Address>
            <City>Bangalore</City>
            <State>Karnataka</State>
            <Zip>560212</Zip>
        </Address>
    </Employee>
</Company>
```

Following tree structure represents the above XML document −



In the above diagram, there is a root element named as <company>. Inside that, there is one more element <Employee>. Inside the employee element, there are five branches named <FirstName>, <LastName>, <ContactNo>, <Email>, and <Address>. Inside the <Address> element, there are three sub-branches, named <City> <State> and <Zip>.

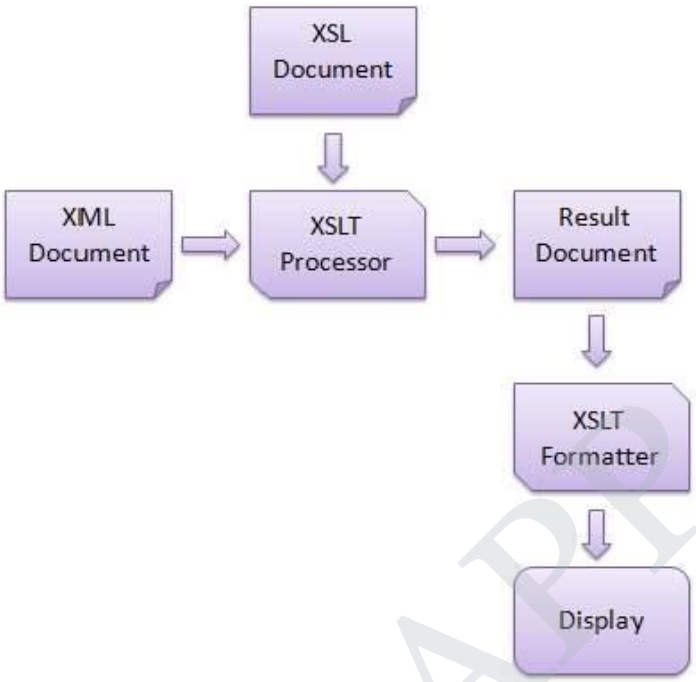
12. **Identify** why XSLT is an important tool for development of web applications.

What is XSLT

XSLT, Extensible Stylesheet Language Transformations, provides the ability to transform XML data from one format to another automatically.

How XSLT Works

An XSLT stylesheet is used to define the transformation rules to be applied on the target XML document. XSLT stylesheet is written in XML format. XSLT Processor takes the XSLT stylesheet and applies the transformation rules on the target XML document and then it generates a formatted document in the form of XML, HTML, or text format. This formatted document is then utilized by XSLT formatter to generate the actual output which is to be displayed to the end-user.



Advantages

Here are the advantages of using XSLT −

- Independent of programming. Transformations are written in a separate xsl file which is again an XML document.
- Output can be altered by simply modifying the transformations in xsl file. No need to change any code. So Web designers can edit the stylesheet and can see the change in the output quickly.

13.

Assess the data types in XML schema.

You can define XML schema elements in the following ways −

Simple Type

Simple type element is used only in the context of the text. Some of the predefined simple types are: xs:integer, xs:boolean, xs:string, xs:date. For example −

```
<xs:element name = "phone_number" type = "xs:int" />
```

Complex Type

A complex type is a container for other element definitions. This allows you to specify which child elements an element can contain and to provide some structure within your XML documents. For example −

```
<xs:element name = "Address">
   <xs:complexType>
      <xs:sequence>
         <xs:element name = "name" type = "xs:string" />
         <xs:element name = "company" type = "xs:string" />
         <xs:element name = "phone" type = "xs:int" />
      </xs:sequence>
```

```
    </xs:complexType>
</xs:element>
```

14.

Explain DTD for XML Schemas.

A **document type definition** (**DTD**) is a set of *markup declarations* that define a *document type* for a SGML-family markup language (GML, SGML, XML, HTML).

A DTD defines the valid building blocks of an XML document. It defines the document structure with a list of validated elements and attributes. A DTD can be declared inline inside an XML document, or as an external reference.

XML DTD schema example

An example of a very simple external XML DTD to describe the schema of a list of persons might consist of:

```
<!ELEMENT people_list (person)*>
<!ELEMENT person (name, birthdate?, gender?, socialsecuritynumber?)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT birthdate (#PCDATA)>
<!ELEMENT gender (#PCDATA)>
<!ELEMENT socialsecuritynumber (#PCDATA)>
```

Taking this line by line:

1. `people_list` is a valid element name, and an instance of such an element contains any number of `person` elements. The `*` denotes there can be 0 or more `person` elements within the `people_list` element.
2. `person` is a valid element name, and an instance of such an element contains one element named `name`, followed by one named `birthdate` (optional), then `gender` (also optional) and `socialsecuritynumber` (also optional). The `?` indicates that an element is optional. The reference to the `name` element name has no `?`, so a `person` element *must* contain a `name` element.
3. `name` is a valid element name, and an instance of such an element contains "parsed character data" (#PCDATA).
4. `birthdate` is a valid element name, and an instance of such an element contains parsed character data.
5. `gender` is a valid element name, and an instance of such an element contains parsed character data.
6. `socialsecuritynumber` is a valid element name, and an instance of such an element contains parsed character data.

An example of an XML file that uses and conforms to this DTD follows. The DTD is referenced here as an external subset, via the SYSTEM specifier and a URI. It assumes that we can identify the DTD with the relative URI reference "example.dtd"; the

"people_list" after "!DOCTYPE" tells us that the root tags, or the first element defined in the DTD, is called "people_list":

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE people_list SYSTEM "example.dtd">
<people_list>
  <person>
    <name>Fred Bloggs</name>
    <birthdate>2008-11-27</birthdate>
    <gender>Male</gender>
  </person>
</people_list>
```

15.

Evaluate the process of displaying XML document in browser.

Display an XML Document in a Web Browser

Displaying XML Using CSS

XML stands for **E**xtensible **M**arkup **L**anguage. It is a dynamic markup language. It is used to transform data from one form to another form.
An XML file can be displayed using two ways. These are as follows :-

1. Cascading Style Sheet
2. Extensible Stylesheet Language Transformation

Displaying XML file using CSS :
CSS can be used to display the contents of the XML document in a clear and precise manner. It gives the design and style to whole XML document.

- **Basic steps in defining a CSS style sheet for XML :**
  For defining the style rules for the XML document, the following things shoulde be done :-

  1. Define the style rules for the text elements such as font-size, color, font-weight, etc.
  2. Define each element either as a block, inline or list element, using the display property of CSS.
  3. Identify the titles and bold them.
- **Linking XML with CSS :**
  In order to display the XML file using CSS, link XML file with CSS. Below is the syntax for linking the XML file with CSS:
  <?xml-stylesheet type="text/css" href="name_of_css_file.css"?>
- **Example 1.**
  In this example, the XML file is created that contains the information about five books and displaying the XML file using CSS.
  XML file :

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="Rule.css"?>
<books>
```

```
<heading>Welcome To GeeksforGeeks </heading>
<book>
   <title>Title -: Web Programming</title>
   <author>Author -: Chrisbates</author>
   <publisher>Publisher -: Wiley</publisher>
   <edition>Edition -: 3</edition>
   <price> Price -: 300</price>
</book>
<book>
   <title>Title -: Internet world-wide-web</title>
   <author>Author -: Ditel</author>
   <publisher>Publisher -: Pearson</publisher>
   <edition>Edition -: 3</edition>
   <price>Price -: 400</price>
</book>

</books>
```

In the above example, Books.xml is linked with Rule.css which contains the corresponding style sheet rules.

CSS FILE :

```
books {
    color: white;
    background-color : gray;
    width: 100%;
}
heading {
    color: green;
    font-size : 40px;
    background-color : powderblue;
}
heading, title, author, publisher, edition, price {
    display : block;
}
title {
    font-size : 25px;
    font-weight : bold;
}
```

- **Output :**

16.	**Summarize** about the need for Namespace in XML.

XML namespaces are used for providing uniquely named elements and attributes in an XML document. They are defined in a W3C recommendation.[1][2] An XML instance may contain element or attribute names from more than one XML vocabulary. If each vocabulary is given a namespace, the ambiguity between identically named elements or attributes can be resolved.

How to get rid of name conflict?

1) By Using a Prefix

You can easily avoid the XML namespace by using a name prefix.

```
<h:table>
  <h:tr>
    <h:td>Aries</h:td>
    <h:td>Bingo</h:td>
  </h:tr>
</h:table>
<f:table>
  <f:name>Computer table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>
```

Note: In this example, you will get no conflict because both the tables have specific names.

2) By Using xmlns Attribute

You can use xmlns attribute to define namespace with the following syntax:

```
<element xmlns:name = "URL">
```

Let's see the example:

```
<root>
<h:table xmlns:h="http://www.abc.com/TR/html4/">
  <h:tr>
    <h:td>Aries</h:td>
    <h:td>Bingo</h:td>
  </h:tr>
</h:table>

<f:table xmlns:f="http://www.xyz.com/furniture">
  <f:name>Computer table</f:name>
  <f:width>80</f:width>
```

```
. <f:length>120</f:length>
. </f:table>
. </root>
```

In the above example, the <table> element defines a namespace and when a namespace is defined for an element, the child elements with the same prefixes are associated with the same namespace.

17.

Analyze on ATOM in RSS.

What is Atom 1.0 ?

Atom is the name of an XML-based Web content and metadata syndication format, and an application-level protocol for publishing and editing Web resources belonging to periodically updated websites.

Atom is a relatively recent spec and is much more robust and feature-rich than RSS. For instance, where RSS requires descriptive fields such as title and link only in item breakdowns, Atom requires these things for both items and the full Feed.

All Atom Feeds must be well-formed XML documents, and are identified with the *application/atom+xml* media type.

Structure of an Atom 1.0 Feed

A Feed consists of some metadata, followed by any number of entries. Here is a basic structure of an Atom 1.0 Feed.

```
<?xml version="1.0"?>
<feed xmlns="http://www.w3.org/2005/Atom">
   <title>...</title>
   <link>...</link>
   <updated>...</updated>

   <author>
      <name>...</name>
   </author>

   <id>...</id>

   <entry>
      <title>...</title>
      <link>...</link>
      <id>...</id>

      <updated>...</updated>
      <summary>...</summary>
   </entry>

</feed>
```

	## Atom 1.0 Feed Tags An Atom 1.0 Feed Document will be constructed of the following two elements: • <feed> Elements • <entry> Elements
18.	**Summarize** the advantage of RSS documents? RSS - Advantages RSS is taking off so quickly because people are liking it. RSS is easy to use and it has advantages for a publisher as well as for a subscriber. Here we have listed out a few advantages of RSS for subscribers as well as for publishers. ## Advantages for Subscribers RSS subscribers are the people who subscribe to read a published Feed. Here are some of the advantages of RSS Feeds for subscribers: • **All news at one place:** You can subscribe to multiple news groups and then you can customize your reader to have all the news on a single page. It will save you a lot of time. • **News when you want it:** Rather than waiting for an e-mail, you go to your RSS reader when you want to read a news. Furthermore, RSS Feeds display more quickly than information on web-sites, and you can read them offline if you prefer. • **Get the news you want:** RSS Feed comes in the form of headlines and a brief description so that you can easily scan the headlines and click only those stories that interest you. • **Freedom from e-mail overload:** You are not going to get any email for any news or blog update. You just go to your reader and you will find updated news or blog automatically whenever there is a change on the RSS server. • **Easy republishing:** You may be both a subscriber and a publisher. For example, you may have a web-site that collects news from various other sites and then republishes it. RSS allows you to easily capture that news and display it on your site. ## Advantages for Publishers RSS publishers are the people who publish their content through RSS feed. We would suggest you to use RSS: • if you want to get your message out and easily, • if you want people to see what you publish, and • if you want your news to bring people back to your site. Here are some of the advantages of RSS if you publish on the Web:

	• **Easier publishing:** RSS is really simple publishing. You don't have to maintain a database of subscribers to send your information to them, instead they will access your Feed using a reader and will get updated content automatically. • **A simpler writing process:** If you have a new content on your web site, you only need to write an RSS Feed in the form of titles and short descriptions, and link back to your site. • **An improved relationship with your subscribers:** Because people subscribe from their side, they don't feel as if you are pushing your content on them. • **The assurance of reaching your subscribers:** RSS is not subject to spam filters, your subscribers get the Feeds, which they subscribe to and nothing more. • **Links back to your site:** RSS Feeds always include links back to a website. It directs a lot of traffic towards your website. • **Relevance and timeliness:** Your subscribers always have the latest information from your site.
19.	**Rewrite** the declaration for elements in XML.

20.	How would you **prepare** the steps to get the RSS file on web?
	
	Uploading an RSS Feed
	
	Here are the simple steps to put your RSS Feed on the web.
	• First decide which version of RSS Feed you are going to use for your site. We would recommend you to use the latest version available.
	• Create your RSS Feed in a text file with extension either .xml or .rdf. Upload this file on your web server.
	• You should validate your RSS Feed before making it live. Check the next chapter on RSS Feed Validation.
	• Create a link on your Web Pages for the RSS Feed file. You will use a small yellow button for the link that says either RSS or XML.
	
	Now, your RSS Feed is online and people can start using it. But there are ways to promote your RSS Feed so that more number of people can use your RSS Feed.
	
	Promote Your RSS Feed
	
	• Submit your RSS Feed to the RSS Feed Directories. There are many directories available on the web, where you can register your Feed. Some of them are given here:
	  ○ Syndic8: Over 300,000 Feeds listed.
	  ○ Daypop: Over 50,000 feeds listed.
	  ○ Newsisfree: Over 18,000 Feeds.
	• Register your Feed with the major search engines. Similar to your web pages, you can add your Feed as well with the following major search engines.
	  ○ Yahoo - http://publisher.yahoo.com/promote.php

- Google - http://www.google.com/webmasters/add.html
- Bing - http://www.bing.com/toolbox/submit-site-url

Keeping Up-To-Date Feed

As we have explained earlier, RSS Feed makes sense for the site which are changing their content very frequently, for example, any news or blogging sites.

So now, you have got RSS Feed buttons from Google, Yahoo, and MSN. You must make sure to update your content frequently and that your RSS Feed is constantly available.

PART-B

1. (i) **Describe** about the introduction and installation of PHP.

Introduction to PHP

PHP is one of the most widely used server side scripting language for web development. Popular websites like Facebook, Yahoo, Wikipedia etc are developed using PHP.

PHP is so popular because it's very simple to learn, code and deploy on server, hence it has been the first choice for beginners since decades.

Uses of PHP

To further fortify your trust in PHP, here are a few applications of this amazing scripting language:

1. It can be used to **create Web applications** like Social Networks(Facebook, Digg), Blogs(Wordpress, Joomla), eCommerce websites(OpenCart, Magento etc.) etc.
2. **Common Line Scripting**. You can write PHP scripts to perform different operations on any machine, all you need is a PHP parser for this.
3. **Create Facebook applications** and easily integrate Facebook plugins in your website, using Facebook's PHP SDK. Check this link for more information.
4. **Sending Emails** or building email applications because PHP provides with a robust email sending function.
5. Wordpress is one of the most used blogging(CMS) platform in the World, and if you know PHP, you can try a hand in **Wordpress plugin development**.

Manual Installation

Step 1: Download the files. Download the latest **PHP** 5 ZIP package from www.**php**.net/downloads.**php**. ...

Step 2: Extract the files. ...

Step 3: Configure **php**. ...

Step 4: Add C:**php** to the path environment variable. ...

Step 5: Configure **PHP** as an Apache module. ...

Step 6: Test a **PHP** file.

(ii) **Design** simple calculator using PHP.

Calculator.php

```
<!DOCTYPE html>
<html>
	<head>
		<title>Simple Calculator In PHP | Webdevtrick.com</title>
		<meta charset="utf-8">
		<meta http-equiv="X-UA-Compatible" content="IE=edge">
		<meta name="viewport" content="width=device-width, initial-scale=1">

		<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css"
rel="stylesheet">
	</head>
	<body>

		<div class="container" style="margin-top: 50px">

		<?php

		// If the submit button has been pressed
		if(isset($_POST['submit']))
		{
		// Check number values
		if(is_numeric($_POST['number1']) && is_numeric($_POST['number2']))
		{
		// Calculate total
		if($_POST['operation'] == 'plus')
		{
		$total = $_POST['number1'] + $_POST['number2'];
		}
		if($_POST['operation'] == 'minus')
		{
		$total = $_POST['number1'] - $_POST['number2'];
		}
		if($_POST['operation'] == 'multiply')
		{
		$total = $_POST['number1'] * $_POST['number2'];
```

```
                    }
                    if($_POST['operation'] == 'divided by')
                    {
                    $total = $_POST['number1'] / $_POST['number2'];
                    }

                    // Print total to the browser
                    echo "<h1>{$_POST['number1']} {$_POST['operation']}
{$_POST['number2']} equals {$total}</h1>";

                    } else {

                    // Print error message to the browser
                    echo 'Numeric values are required';

                    }
                    }
                    // end PHP. Code by webdevtrick.com
                    ?>

            <!-- Calculator form by webdevtrick.com -->
            <form method="post" action="calculator.php">
                <input name="number1" type="text" class="form-control" style="width:
150px; display: inline" />
                <select name="operation">
                    <option value="plus">Plus</option>
                    <option value="minus">Minus</option>
                    <option value="multiply">Multiply</option>
                    <option value="divided by">Devide</option>
                </select>
                <input name="number2" type="text" class="form-control" style="width:
150px; display: inline" />
                <input name="submit" type="submit" value="Calculate" class="btn btn-
primary" />
            </form>

                    </div>

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

2. **Explain** about control statements and data types in PHP with example.

Control Statements in PHP with Examples

Like any other languages, PHP is built out of a series of control statements. The control statement can be an assignment, a function call, a loop, a conditional statement or even a statement that does nothing or an empty statement.

In PHP we have the following conditional statements:

if statement – We use this control statement to execute some code only if a specified condition is true.

if…else statement – We use this control statement to execute some code if a condition is true and another code if the condition is false.

if…elseif….else statement – We use this control statement to select one of several blocks of code to be executed

switch statement – We use this control statement to select one of many blocks of code to be executed

1. The if Statement

Use the if statement to execute some code only if a specified condition is true.
The expression is evaluated to its Boolean value. If expression evaluates to TRUE, PHP will execute statement, and if it evaluates to FALSE – it'll ignore it

Syntax

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

The following example would display " A is bigger than B" if $a is bigger than $b:

```
<?php
if ($a > $b)
echo "A is bigger than B";
?>
```

2. The if…else Statement

elseif, as its name suggests, 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.

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

For example, the following code would display a is bigger than b, a equal to b or a is smaller than b:

```
<?php
if ($a > $b) {
echo "a is bigger than b";
} elseif ($a == $b) {
echo "a is equal to b";
} else {
echo "a is smaller than b";
}
?>
```

3. The if…elseif….else Statement

4. The Switch Statement

The switch statement is similar to IF statements on the same expression. In many occasions,

Use the if….elseif…else statement to select one of several blocks of code to be executed.

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

Note: Note that elseif and else if will only be considered exactly the same when using curly brackets as in the above example. When using a colon to define your if/elseif conditions, you must not separate else if into two words, or PHP will fail with a parse error.

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. This is exactly what the switch statement is for.

```
switch ( )
{
case condition1
break;
case condition2
break;
}
```

For example, the following code would display $i matched value as 0 or 1 or 2:

```
<?php
switch ($i) {
case 0:
echo "i equals 0";
case 1:
echo "i equals 1";
case 2:
echo "i equals 2";
}
?>
```

3.

(i) **Create** an XML document that marks up various sports and their descriptions. Use XSLT to tabulate neatly the elements and attributes of the document.

```
<?xml version="1.0"?>

-<events league="WC Falun" tournament="2010/2011" template="World Cup"
sport="Cross Country Skiing" ut="2012-09-05" id="821135">


-<event id="866683" status="Finished" round="8001 - 1/1 (Final)" date="2011-03-20
13:15:00" name="10 km Freestyle Handicap Pursuit">


-<results participantname="Marit Bjoergen" participantid="43427">

<result id="9498426" value="1" type="rank"/>

<result id="9498424" value="27:58.0" type="duration"/>
```

<result id="9505038" value="200" type="points"/>

<result id="9498425" value="" type="comment"/>

<result id="9497448" value="1" type="startnumber"/>

</results>

-<results participantname="Justyna Kowalczyk" participantid="43775">

<result id="9498429" value="2" type="rank"/>

<result id="9498427" value="+1:58.0" type="duration"/>

<result id="9505039" value="160" type="points"/>

<result id="9498428" value="" type="comment"/>

<result id="9497454" value="2" type="startnumber"/>

</results>
</event></events>

(ii) **Illustrate** a JSP page that enables the user to input the first name and in response outputs the last name.

POST Method Example Using Form

Below is the **main.jsp** JSP program to handle the input given by web browser using the GET or the POST methods.

```
<html>
 <head>
    <title>Using GET and POST Method to Read Form Data</title>
 </head>

 <body>
    <center>
    <h1>Using POST Method to Read Form Data</h1>

    <ul>
      <li><p><b>First Name:</b>
        <%= request.getParameter("first_name")%>
      </p></li>
      <li><p><b>Last  Name:</b>
        <%= request.getParameter("last_name")%>
      </p></li>
    </ul>
```

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

Following is the content of the **Hello.htm** file −

```
<html>
  <body>

    <form action = "main.jsp" method = "POST">
      First Name: <input type = "text" name = "first_name">
      <br />
      Last Name: <input type = "text" name = "last_name" />
      <input type = "submit" value = "Submit" />
    </form>

  </body>
</html>
```

Let us now keep **main.jsp** and hello.htm in **<Tomcat-installationdirectory>/webapps/ROOT directory**. When you access ***http://localhost:8080/Hello.htm***, you will receive the following output.

First Name:

Last Name:

Try to enter the First and the Last Name and then click the submit button to see the result on your local machine where tomcat is running.

4. **Create** a webserver based chat application using PHP. The application should provide the following functions Login, Send message (to one or more contacts) and Receive messages (from one or more contacts)

5. (i) **Write** a PHP program that tests whether an email address is input correctly. Test your program with both valid and invalid email addresses.

PHP - Validate Name, E-mail, and URL

Example

```
<?php
// define variables and set to empty values
```

```
$nameErr = $emailErr = $genderErr = $websiteErr = "";
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  if (empty($_POST["name"])) {
    $nameErr = "Name is required";
  } else {
    $name = test_input($_POST["name"]);
    // check if name only contains letters and whitespace
    if (!preg_match("/^[a-zA-Z ]*$/",$name)) {
      $nameErr = "Only letters and white space allowed";
    }
  }

  if (empty($_POST["email"])) {
    $emailErr = "Email is required";
  } else {
    $email = test_input($_POST["email"]);
    // check if e-mail address is well-formed
    if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
      $emailErr = "Invalid email format";
    }
  }

  if (empty($_POST["website"])) {
    $website = "";
  } else {
    $website = test_input($_POST["website"]);
    // check if URL address syntax is valid (this regular expression
also allows dashes in the URL)
    if (!preg_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-
9+&@#\/%?=~_|!:,.;]*[-a-z0-9+&@#\/%=~_|]/i",$website)) {
      $websiteErr = "Invalid URL";
    }
  }

  if (empty($_POST["comment"])) {
    $comment = "";
  } else {
    $comment = test_input($_POST["comment"]);
  }

  if (empty($_POST["gender"])) {
    $genderErr = "Gender is required";
  } else {
    $gender = test_input($_POST["gender"]);
  }
```

	} ?> OUTPUT **PHP Form Validation Example** * required field Name: * E-mail: * Website: Comment: Gender: Female Male Other * Submit **Your Input:** Anand M anand@ibm.com www.ibm.com Yhis is a comment male
6.	**Identify** and explain about database connectivity illustrate PHP connectivity with any of the databases. *METHOD FOR: CONNECTING TO MYSQL USING MYSQL* The MySQL Improved extension uses the *mysqli* class, which replaces the set of legacy MySQL functions.

To connect to MySQL using the MySQL Improved extension, follow these steps:

1. 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:

```
2.   <?php
3.       $mysqli = new mysqli("localhost", "username", "password
", "dbname");
?>
```

```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";
?>
```

7. (i) **Discuss** on methods for using cookies in PHP.

Cookies in PHP

Cookies are used to store the information of a web page in a remote browser, so that when the same user comes back to that page, that information can be retrieved from the browser itself.

Uses of cookie

Cookies are often used to perform following tasks:

- **Session management**: Cookies are widely used to manage user sessions. For example, when you use an online shopping cart, you keep adding items in the cart and finally when you checkout, all of those items are added to the list of items you have purchased. This can be achieved using cookies.
- **User identification**: Once a user visits a webpage, using cookies, that user can be remembered. And later on, depending upon the search/visit pattern of the user,

content which the user likely to be visited are served. A good example of this is 'Retargetting'. A concept used in online marketing, where depending upon the user's choice of content, advertisements of the relevant product, which the user may buy, are served.

- **Tracking / Analytics**: Cookies are used to track the user. Which, in turn, is used to analyze and serve various kind of data of great value, like location, technologies (e.g. browser, OS) form where the user visited, how long (s)he stayed on various pages etc.

How to create a cookie in PHP

PHP has a setcookie() function to send a cookie. We will discuss this function in detail now.

```
setcookie(name, value, expire, path, domain, secure, httponly)
```

setcookie() returns boolean.

Example:

Following example shows how to create a cookie in PHP.

```
<?php
$cookie_value = "w3resource tutorials";
setcookie("w3resource", $cookie_value, time()+3600, "/home/your_usename/",
"example.com", 1, 1);
if (isset($_COOKIE['cookie']))
echo $_COOKIE["w3resource"];
?>
```

(ii) **Give** a note on regular expressions.

8. **Summarize** in detail the XML schema, built in and user defined data types.

What is an XML Schema?

An XML Schema describes the structure of an XML document.

The XML Schema language is also referred to as XML Schema Definition (XSD).

XSD Example

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="note">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="to" type="xs:string"/>
      <xs:element name="from" type="xs:string"/>
      <xs:element name="heading" type="xs:string"/>
      <xs:element name="body" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

</xs:schema>
```

The purpose of an XML Schema is to define the legal building blocks of an XML document:

- the elements and attributes that can appear in a document
- the number of (and order of) child elements
- data types for elements and attributes
- default and fixed values for elements and attributes

XSD Simple Elements

XML Schemas define the elements of your XML files.

A simple element is an XML element that contains only text. It cannot contain any other elements or attributes.

The text can be of many different types like boolean, string, date, etc.), or it can be a custom type that you can define yourself.

You can also add restrictions (facets) to a data type in order to limit its content, or you can require the data to match a specific pattern.

The syntax for defining a simple element is:

```
<xs:element name="xxx" type="yyy"/>
```

where xxx is the name of the element and yyy is the data type of the element.

XML Schema has a lot of built-in data types. The most common types are:

- xs:string
- xs:decimal
- xs:integer
- xs:boolean
- xs:date
- xs:time

Example

Simple element definitions:

```
<xs:element name="lastname" type="xs:string"/>
<xs:element name="age" type="xs:integer"/>
<xs:element name="dateborn" type="xs:date"/>
```

Here are some XML elements:

```
<lastname>Ronald</lastname>
<age>36</age>
<dateborn>1970-03-27</dateborn>
```

Simple elements may have a default value which is automatically assigned to the element when no other value is specified as shown below:

```
<xs:element name="color" type="xs:string" default="red"/>
```

How to Define a Complex Element

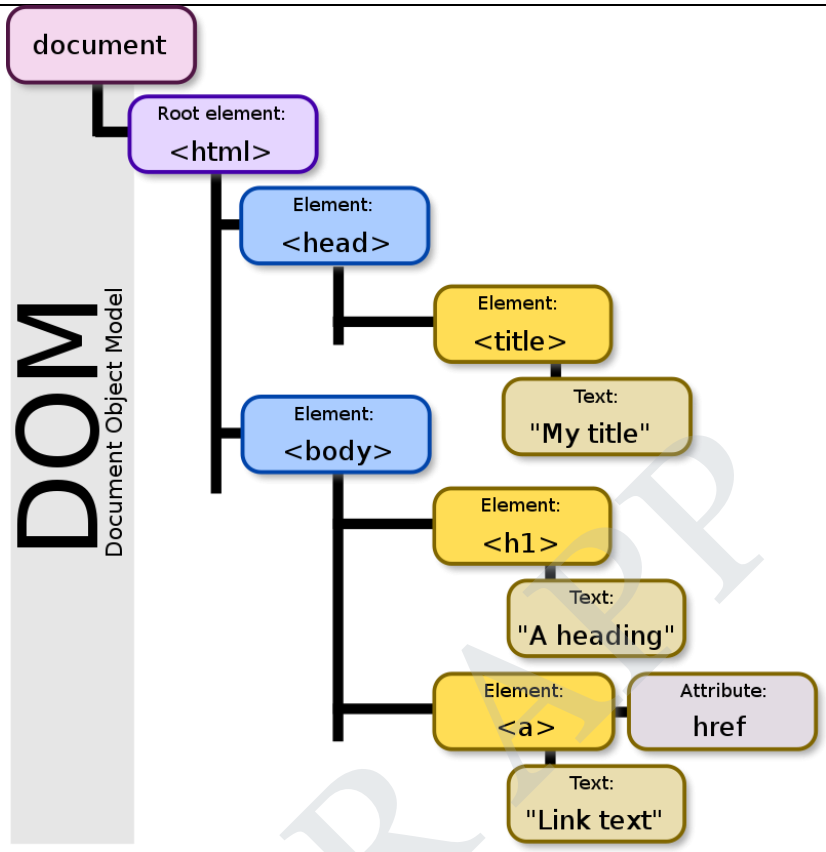
We can define a complex element in an XML Schema two different ways:

1. The "employee" element can be declared directly by naming the element, like this:

```
<xs:element name="employee">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

	2. The "employee" element can have a type attribute that refers to the name of the complex type to use:
	
	`<xs:element name="employee" type="personinfo"/>`
	
	`<xs:complexType name="personinfo">`
	`  <xs:sequence>`
	`  <xs:element name="firstname" type="xs:string"/>`
	`  <xs:element name="lastname" type="xs:string"/>`
	`  </xs:sequence>`
	`</xs:complexType>`
	
	If you use the method described above, several elements can refer to the same complex type, like this:
	
	`<xs:element name="employee" type="personinfo"/>`
	`<xs:element name="student" type="personinfo"/>`
	`<xs:element name="member" type="personinfo"/>`
	
	`<xs:complexType name="personinfo">`
	`  <xs:sequence>`
	`  <xs:element name="firstname" type="xs:string"/>`
	`  <xs:element name="lastname" type="xs:string"/>`
	`  </xs:sequence>`
	`</xs:complexType>`

9.	(i) **Demonstrate** the building blocks of DOM. The **Document Object Model** (**DOM**) is a cross-platform and language-independent interface that treats an XML or HTML document as a tree structure wherein each node is an object representing a part of the document. The DOM represents a document with a logical tree. Each branch of the tree ends in a node, and each node contains objects. DOM methods allow programmatic access to the tree; with them one can change the structure, style or content of a document. Nodes can have event handlers attached to them. Once an event is triggered, the event handlers get executed.



(ii) **Classify** the types of DTD.

XML DTD

An XML document with correct syntax is called "Well Formed".

An XML document validated against a DTD is both "Well Formed" and "Valid".

DTD stands for Document Type Definition.

A DTD defines the structure and the legal elements and attributes of an XML document.

A "Valid" XML document is "Well Formed", as well as it conforms to the rules of a DTD:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE note SYSTEM "Note.dtd">
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
```

```
<body>Don't forget me this weekend!</body>
</note>
```

The DOCTYPE declaration above contains a reference to a DTD file. The content of the DTD file is shown and explained below.

The purpose of a DTD is to define the structure and the legal elements and attributes of an XML document:

Note.dtd:

```
<!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:

- !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"

Tip: #PCDATA means parseable character data.

10. How do you **infer the significant** differences between DID and XML schema for defining XML document structures with appropriate examples.

Difference Between XML Schema and DTD

XML Schema vs. DTD

DTD, or Document Type Definition, and XML Schema, which is also known as XSD, are two ways of describing the structure and content of an XML document. DTD is the older

	of the two, and as such, it has limitations that XML Schema has tried to improve. Summary: 1. XML Schema is namespace aware, while DTD is not. 2. XML Schemas are written in XML, while DTDs are not. 3. XML Schema is strongly typed, while DTD is not. 4. XML Schema has a wealth of derived and built-in data types that are not available in DTD. 5. XML Schema does not allow inline definitions, while DTD does.
11.	(i) **List** out data types data types of XML **XML Schema Data Types** XML Schema data types can be generally categorized a "simple type" (including embedded simple type) and "complex type." The "embedded simple type" is already defined, but can be used to create a new type through restriction or extension. **Table : XML Schema Data Types** Simple Type — User can independently define. This type is used when a restriction is placed on an embedded simple type to create and use a new type. Complex Type — User can independently define. This type is used when the type has a child element or attribute. A simple type is a type that only contains text data. This type can be used with element declarations and attribute declarations. On the other hand, a complex data type is a type that has a child element or attribute structure. ●**Simple Type Example** <xs:element name="Department" type="xs:string" />

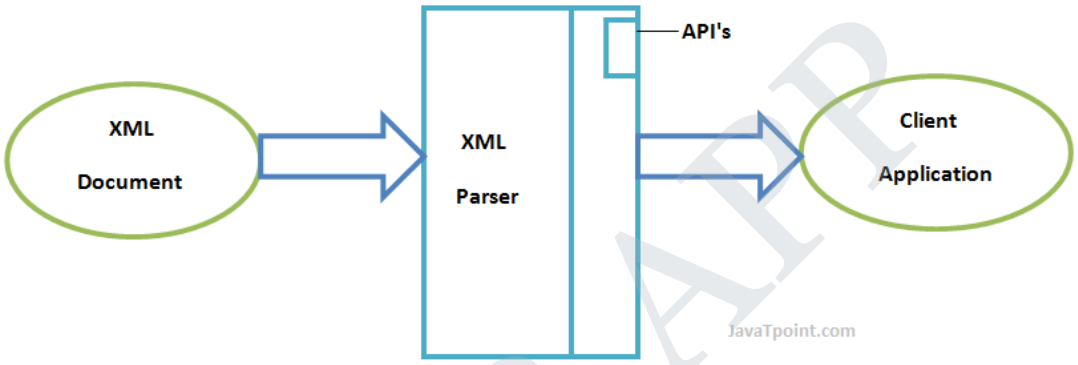
	Here, the section described together with "xs:string" is an embedded simple type according to XML Schema. In this example, we have established the definition that the data type for the element called "Department" is a text string. **●Complex Type Example** <xs:complexType name="EmployeeType"> <xs:sequence maxOccurs="unbounded"> <xs:element ref="Name" /> <xs:element ref="Department" /> </xs:sequence> </xs:complexType> <xs:element name="Name" type="xs:string" /> <xs:element name="Department" type="xs:string" /> (ii) **Explain** about the attributes of XML. XML Elements vs. Attributes Take a look at these examples: <person gender="female"> <firstname>Anna</firstname> <lastname>Smith</lastname> </person> <person> <gender>female</gender> <firstname>Anna</firstname> <lastname>Smith</lastname> </person> In the first example gender is an attribute. In the last, gender is an element. Both examples provide the same information. There are no rules about when to use attributes or when to use elements in XML.
12.	**Summarize** on the following (i) DOM based Parsing. (ii) SAX based Parsing.

XML Parsers

An XML parser is a software library or package that provides interfaces for client applications to work with an XML document. The XML Parser is designed to read the XML and create a way for programs to use XML.

XML parser validates the document and check that the document is well formatted.

Let's understand the working of XML parser by the figure given below:



Types of XML Parsers

These are the two main types of XML Parsers:

1. DOM
2. SAX

DOM (Document Object Model)

A DOM document is an object which contains all the information of an XML document. It is composed like a tree structure. The DOM Parser implements a DOM API. This API is very simple to use.

Features of DOM Parser

A DOM Parser creates an internal structure in memory which is a DOM document object and the client applications get information of the original XML document by invoking methods on this document object.

DOM Parser has a tree based structure.

Advantages

	1) It supports both read and write operations and the API is very simple to use. 2) It is preferred when random access to widely separated parts of a document is required. Disadvantages 1) It is memory inefficient. (consumes more memory because the whole XML document needs to loaded into memory). 2) It is comparatively slower than other parsers. --- SAX (Simple API for XML) A SAX Parser implements SAX API. This API is an event based API and less intuitive. Features of SAX Parser It does not create any internal structure. Clients does not know what methods to call, they just overrides the methods of the API and place his own code inside method. It is an event based parser, it works like an event handler in Java. Advantages 1) It is simple and memory efficient. 2) It is very fast and works for huge documents. Disadvantages 1) It is event-based so its API is less intuitive. 2) Clients never know the full information because the data is broken into pieces.
13.	(i) **Compare and contrast** RSS & ATOM. **Difference Between RSS and ATOM**

RSS vs ATOM

- Really Simple Syndication or RSS has been the standard for web feeds for a considerable time.
- Web feeds contains either a summary or the full text content of a web page.
- The problem with RSS is the often confusing and non standard conventions used by RSS due in part to its scattered development.
- The advent of the ATOM syndication standard was a response to the design flaws of the RSS standard.
- The primary advantage of the ATOM is its adaptation as the IETF standard.
- Being an IETF standard, each atom feed contains an explicit declaration of the format of the content along with what language is used.
- RSS feeds do not declare its content, but since it only contains plain text or escaped HTML, it is rather easy for the browser to distinguish which is which.

A major flaw of RSS is in its code. RSS code isn't really very usable in other XML vocabularies since it wasn't really intended to do so at the very beginning. ATOM code has been built from the ground with modularity in mind. Therefore, a great majority of its code is reusable even with other XML vocabularies like RSS.

Summary:

1. ATOM is an IETF standard while RSS is not
2. ATOM feeds explicitly indicates the content while the browser is left to figure out whether the RSS feed contains plain text or escaped HTML
3. ATOM code is modular and reusable while RSS code is not
4. RSS still holds dominance in the syndication format due to its head start and popularity

(ii) **Explain** in detail about XSL elements.

XSLT <xsl:element>

Definition and Usage

The <xsl:element> element is used to create an element node in the output document.

Syntax

```
<xsl:element
name="name"
namespace="URI"
use-attribute-sets="namelist">

  <!-- Content:template -->

</xsl:element>
```

Attributes

Attribute	Value	Description
name	name	Required. Specifies the name of the element to be created (the value of the name attribute can be set to an expression that is computed at run-time, like this: <xsl:element name="{$country}" />
namespace	URI	Optional. Specifies the namespace URI of the element (the value of the namespace attribute can be set to an expression that is computed at run-time, like this: <xsl:element name="{$country}" namespace="{$someuri}"/>
use-attribute-sets	namelist	Optional. A white space separated list of attribute-sets containing attributes to be added to the element

Example 1

Create a "singer" element that contains the value of each artist element:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <xsl:for-each select="catalog/cd">
    <xsl:element name="singer">
      <xsl:value-of select="artist" />
    </xsl:element>
    <br />
  </xsl:for-each>
</xsl:template>

</xsl:stylesheet>
```

EXAMPLE OUTPUT FILES

XML File

```
<catalog>
<cd>
<title>Empire Burlesque</title>
<artist>Bob Dylan</artist>
<country>USA</country>
<company>Columbia</company>
<price>10.90</price>
<year>1985</year>
</cd>
<cd>
<title>Hide your heart</title>
<artist>Bonnie Tyler</artist>
<country>UK</country>
<company>CBS Records</company>
<price>9.90</price>
<year>1988</year>
</cd>
</catalog>
```

XSL File

```
xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version=
"1.0">
<xsl:template match="/">
<html>
<body>
<h2>My CD Collection</h2>
<xsl:for-each select="catalog/cd">
```

```
<xsl:element name="singer">
<xsl:value-of select="artist"/>
</xsl:element>
<br/>
</xsl:for-each>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

RESULT

My CD Collection

Bob Dylan
Bonnie Tyler
Dolly Parton
Gary Moore

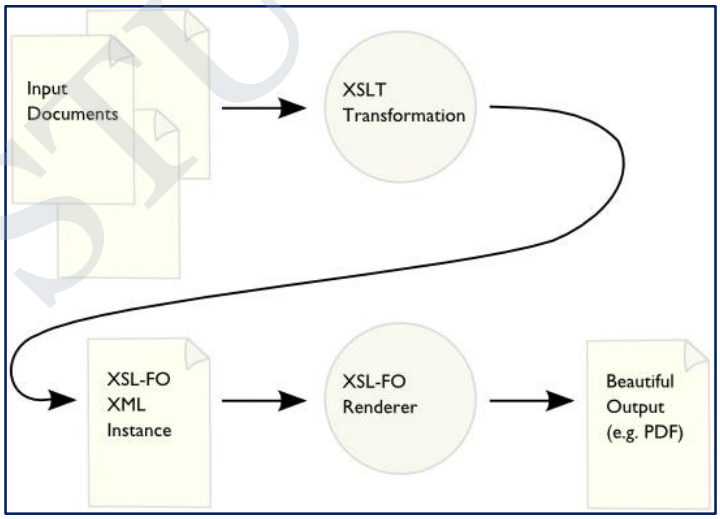
14.

Explain in detail about
(i) XSL and XSLT transformation

XSLT - Transformation

What is XSLT?

XSL Transformations (XSLT 2.0) is a language for transforming XML documents into other XML documents, text documents or HTML documents. You might want to format a chapter of a book using XSL-FO, or you might want to take a database query and format it as HTML.



Wildly Popular

XSLT has become the language of choice for a very wide range of XML applications. It is of course still used to produce XSL-FO documents for printing, but it is also used to integrate back-end software for Web sites. We can find XSLT inside most modern Web browsers, so

that XML can be transformed on the fly without the user even noticing; you will find XSLT on the desktop, in servers, in network appliances.

What is XSLT Used For?

If you make a purchase on eBay, or buy a book at Amazon, chances are that pretty much everything you see on every Web page has been processed with XSLT. Use XSLT to process multiple XML documents and to produce any combination of text, HTML and XML output. XSLT support is shipped with all major computer operating systems today, as well as being built in to all major Web browsers.

XSLT – Transformation : STEPS

1) Start with a Raw XML Document
2) Create an XSL Style Sheet
3) Link the XSL Style Sheet to the XML Document

1) Start with a Raw XML Document

We want to **transform** the following XML document ("cdcatalog.xml") into XHTML:

```
<?xml version="1.0" encoding="UTF-8"?>
<catalog>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <country>USA</country>
    <company>Columbia</company>
    <price>10.90</price>
    <year>1985</year>
  </cd>
.</catalog>
```

2) Create an XSL Style Sheet

Then you create an XSL Style Sheet ("cdcatalog.xsl") with a transformation template:

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

<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
  <body>
  <h2>My CD Collection</h2>
  <table border="1">
    <tr bgcolor="#9acd32">
      <th>Title</th>
      <th>Artist</th>
    </tr>
```

```
    <xsl:for-each select="catalog/cd">
    <tr>
      <td><xsl:value-of select="title"/></td>
      <td><xsl:value-of select="artist"/></td>
    </tr>
    </xsl:for-each>
  </table>
  </body>
  </html>
</xsl:template>

</xsl:stylesheet>
```

3) Link the XSL Style Sheet to the XML Document

Add the XSL style sheet reference to your XML document ("cdcatalog.xml"):

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="cdcatalog.xsl"?>
<catalog>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>


    <country>USA</country>
    <company>Columbia</company>
    <price>10.90</price>
    <year>1985</year>
  </cd>
.</catalog>
```

If you have an XSLT compliant browser it will nicely **transform** your XML into XHTML.

Sample OUTPUT

My CD Collection

Title	Artist
Empire Burlesque	Bob Dylan
Hide your heart	Bonnie Tyler
Greatest Hits	Dolly Parton
Still got the blues	Gary Moore

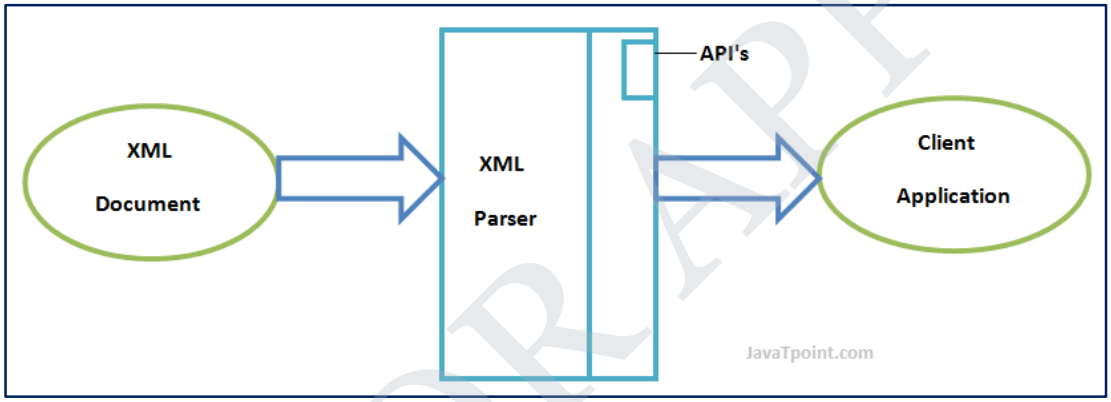
(ii) Comparison of DOM & SAX

XML Parsers

An XML parser is a software library or package that provides interfaces for client applications to work with an XML document. The XML Parser is designed to read the XML and create a way for programs to use XML.

XML parser validates the document and check that the document is well formatted.

Let's understand the working of XML parser by the figure given below:



Types of XML Parsers

These are the two main types of XML Parsers:

3. DOM
4. SAX

DOM (Document Object Model)

A DOM document is an object which contains all the information of an XML document. It is composed like a tree structure. The DOM Parser implements a DOM API. This API is very simple to use.

Features of DOM Parser

A DOM Parser creates an internal structure in memory which is a DOM document object and the client applications get information of the original XML document by invoking methods on this document object.

DOM Parser has a tree based structure.

	Advantages 1) It supports both read and write operations and the API is very simple to use. 2) It is preferred when random access to widely separated parts of a document is required. Disadvantages 1) It is memory inefficient. (consumes more memory because the whole XML document needs to loaded into memory). 2) It is comparatively slower than other parsers. --- SAX (Simple API for XML) A SAX Parser implements SAX API. This API is an event based API and less intuitive. Features of SAX Parser It does not create any internal structure. Clients does not know what methods to call, they just overrides the methods of the API and place his own code inside method. It is an event based parser, it works like an event handler in Java. Advantages 1) It is simple and memory efficient. 2) It is very fast and works for huge documents. Disadvantages 1) It is event-based so its API is less intuitive. 2) Clients never know the full information because the data is broken into pieces.
	PART-C
1.	**Explain** how you shall carry out String Manipulations using a PHP Program.

Manipulating PHP Strings

PHP provides many built-in functions for manipulating strings like calculating the length of a string, find substrings or characters, replacing part of a string with different characters, take a string apart, and many others.

Here are the examples of some of these functions.

Calculating the Length of a String

The `strlen()` function is used to calculate the number of characters inside a string. It also includes the blank spaces inside the string.

Example

Run this code »

```
<?php
$my_str = 'Welcome to Tutorial Republic';

// Outputs: 28
echo strlen($my_str);
?>
```

Counting Number of Words in a String

The `str_word_count()` function counts the number of words in a string.

Example

Run this code »

```
<?php
$my_str = 'The quick brown fox jumps over the lazy dog.';

// Outputs: 9
echo str_word_count($my_str);
?>
```

Replacing Text within Strings

The `str_replace()` replaces all occurrences of the search text within the target string.

Example

Run this code »

```
<?php
$my_str = 'If the facts do not fit the theory, change the facts.';

// Display replaced string
echo str_replace("facts", "truth", $my_str);
?>
```

The output of the above code will be:

If the truth do not fit the theory, change the truth.

You can optionally pass the fourth argument to the str_replace() function to know how many times the string replacements was performed, like this.

Example

Run this code »

```
<?php
$my_str = 'If the facts do not fit the theory, change the facts.';

// Perform string replacement
str_replace("facts", "truth", $my_str, $count);

// Display number of replacements performed
echo "The text was replaced $count times.";
?>
```

The output of the above code will be:

The text was replaced 2 times.

Reversing a String

The strrev() function reverses a string.

Example

Run this code »

```
<?php
$my_str = 'You can do anything, but not everything.';

// Display reversed string
echo strrev($my_str);
?>
```

The output of the above code will be:

.gnihtyreve ton tub ,gnihtyna od nac uoY

2. **Design** a PHP application for College Management System with appropriate built-in functions and database.

3. **Design** application to send an email using PHP.

PHP mail() Function

Example

Send a simple email:

```
<?php
// the message
$msg = "First line of text\nSecond line of text";

// use wordwrap() if lines are longer than 70 characters
$msg = wordwrap($msg,70);

// send email
mail("someone@example.com","My subject",$msg);
?>
```

Syntax

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

Parameter Values

Parameter	Description
to	Required. Specifies the receiver / receivers of the email
subject	Required. Specifies the subject of the email. **Note:** This parameter cannot contain any newline characters
message	Required. Defines the message to be sent. Each line should be separated with a LF (\n). Lines should not exceed 70 characters. **Windows note:** If a full stop is found on the beginning of a line in the message, it might be removed. To solve this problem, replace the full stop with a double dot: <?php

	$txt = str_replace("\n.", "\n..", $txt); ?>
headers	Optional. Specifies additional headers, like From, Cc, and Bcc. The additional headers should be separated with a CRLF (\r\n). **Note:** When sending an email, it must contain a From header. This can be set with this parameter or in the php.ini file.
parameters	Optional. Specifies an additional parameter to the sendmail program (the one defined in the sendmail_path configuration setting). (i.e. this can be used to set the envelope sender address when using sendmail with the -f sendmail option)

Technical Details

Return Value:	Returns the hash value of the *address* parameter, or FALSE on failure. **Note:** Keep in mind that even if the email was accepted for delivery, it does NOT mean the email is actually sent and received!
PHP Version:	4+
PHP Changelog:	PHP 7.2: The headers parameter also accepts an array PHP 5.4: Added header injection protection for the *headers* parameter. PHP 4.3.0: (Windows only) All custom headers (like From, Cc, Bcc and Date) are supported, and are not case-sensitive. PHP 4.2.3: The *parameter* parameter is disabled in safe mode PHP 4.0.5: The *parameter* parameter was added

More Examples

Send an email with extra headers:

```
<?php
$to = "somebody@example.com";
$subject = "My subject";
$txt = "Hello world!";
$headers = "From: webmaster@example.com" . "\r\n" .
"CC: somebodyelse@example.com";

mail($to,$subject,$txt,$headers);
?>
```

Send an HTML email:

```
<?php
$to = "somebody@example.com, somebodyelse@example.com";
$subject = "HTML email";

$message = "
<html>
<head>
<title>HTML email</title>
</head>
<body>
<p>This email contains HTML Tags!</p>
<table>
<tr>
<th>Firstname</th>
<th>Lastname</th>
</tr>
<tr>
<td>John</td>
<td>Doe</td>
</tr>
</table>
</body>
</html>
";

// Always set content-type when sending HTML email
$headers = "MIME-Version: 1.0" . "\r\n";
$headers .= "Content-type:text/html;charset=UTF-8" . "\r\n";

// More headers
$headers .= 'From: <webmaster@example.com>' . "\r\n";
$headers .= 'Cc: myboss@example.com' . "\r\n";

mail($to,$subject,$message,$headers);
?>
```

4.

Summarize about XML schema and XML Parsers and Validation.

XML Schema

What is an XML Schema?

An XML Schema describes the structure of an XML document.

The XML Schema language is also referred to as XML Schema Definition (XSD).

XSD Example

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="note">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="to" type="xs:string"/>
      <xs:element name="from" type="xs:string"/>
      <xs:element name="heading" type="xs:string"/>
      <xs:element name="body" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

</xs:schema>
```

The purpose of an XML Schema is to define the legal building blocks of an XML document:

- the elements and attributes that can appear in a document
- the number of (and order of) child elements
- data types for elements and attributes
- default and fixed values for elements and attributes

Why Learn XML Schema?

In the XML world, hundreds of standardized XML formats are in daily use.

Many of these XML standards are defined by XML Schemas.

XML Schema is an XML-based (and more powerful) alternative to DTD.

XML Parser

All major browsers have a built-in XML parser to access and manipulate XML.

The XML DOM (Document Object Model) defines the properties and methods for accessing and editing XML.

However, before an XML document can be accessed, it must be loaded into an XML DOM object.

All modern browsers have a built-in XML parser that can convert text into an XML DOM object.

Parsing a Text String

This example parses a text string into an XML DOM object, and extracts the info from it with JavaScript:

Example

```
<html>
<body>

<p id="demo"></p>

<script>
var text, parser, xmlDoc;

text = "<bookstore><book>" +
"<title>Everyday Italian</title>" +
"<author>Giada De Laurentiis</author>" +
"<year>2005</year>" +
"</book></bookstore>";

parser = new DOMParser();
xmlDoc = parser.parseFromString(text,"text/xml");

document.getElementById("demo").innerHTML =
xmlDoc.getElementsByTagName("title")[0].childNodes[0].nodeValue;
</script>

</body>
</html>
```

OUTPUT

Everyday Italian

XML - Validation

Validation is a process by which an XML document is validated. An XML document is said to be valid if its contents match with the elements, attributes and associated document type declaration(DTD), and if the document complies with the constraints expressed in it. Validation is dealt in two ways by the XML parser. They are −

- Well-formed XML document
- Valid XML document

Well-formed XML Document

An XML document is said to be **well-formed** if it adheres to the following rules −

- Non DTD XML files must use the predefined character entities for **amp(&)**, **apos(single quote)**, **gt(>)**, **lt(<)**, **quot(double quote)**.
- It must follow the ordering of the tag. i.e., the inner tag must be closed before closing the outer tag.
- Each of its opening tags must have a closing tag or it must be a self ending tag.(<title>....</title> or <title/>).
- It must have only one attribute in a start tag, which needs to be quoted.
- **amp(&)**, **apos(single quote)**, **gt(>)**, **lt(<)**, **quot(double quote)** entities other than these must be declared.

Example

Following is an example of a well-formed XML document −

```
<?xml version = "1.0" encoding = "UTF-8" standalone = "yes" ?>
<!DOCTYPE address
[
  <!ELEMENT address (name,company,phone)>
  <!ELEMENT name (#PCDATA)>
  <!ELEMENT company (#PCDATA)>
  <!ELEMENT phone (#PCDATA)>
]>

<address>
  <name>Tanmay Patil</name>
  <company>TutorialsPoint</company>
  <phone>(011) 123-4567</phone>
</address>
```

The above example is said to be well-formed as −

- It defines the type of document. Here, the document type is **element** type.
- It includes a root element named as **address**.
- Each of the child elements among name, company and phone is enclosed in its self explanatory tag.
- Order of the tags is maintained.

Valid XML Document

If an XML document is well-formed and has an associated Document Type Declaration (DTD), then it is said to be a valid XML document.

CS8651 -UNIT 5-NOTES

CS8651- Internet Programming 2017Reg	
UNIT V - INTRODUCTION TO AJAX and WEB SERVICES	
JAX: Ajax Client Server Architecture – XML Http Request Object – Call Back Methods; **Web Services:** Introduction – Java web services Basics – Creating, Publishing, Testing and Describing a Web services WSDL) –Consuming a web service, Database Driven web service from an application – SOAP.	
	PART-A
No	**Questions**
1.	**Describe** AJAX Control Extender Toolkit. **What is the ASP.NET AJAX Control Toolkit?** The ASP.NET AJAX Control Toolkit is an open-source project built on top of the Microsoft ASP.NET AJAX framework. It is a joint effort between Microsoft and the ASP.NET AJAX community that provides a powerful infrastructure to write reusable, customizable and extensible ASP.NET AJAX extenders and controls, as well as a rich array of controls that can be used out of the box to create an interactive Web experience. They are designed using concepts that are familiar to ASP.NET Web Forms application developers. Using the Ajax Control Toolkit, you can build Ajax-enabled ASP.NET Web Forms applications and ASP.NET MVC Web applications by dragging the controls from the Visual Studio Toolbox onto a page. The Ajax Control Toolkit is an open-source project that is part of the CodePlex Foundation The AJAX Control Toolkit contains more than 30 controls that enable you to easily create rich, interactive web pages.
2.	**Discuss** the advantages of AJAX. Advantages of AJAX • Reduce the traffic travels between the client and the server. • Response time is faster so increases performance and speed. • You can use JSON (JavaScript Object Notation) which is alternative to XML. JSON is key value pair and works like an array. • You can use Firefox browser with an add-on called as Firebug to debug all Ajax calls. • Ready Open source JavaScript libraries available for use – JQuery, Prototype, Scriptaculous, etc.. • AJAX communicates over HTTP Protocol.
3.	**Identify** the role of a callback function in performing a partial page update in an AJAX application. **Partial-page rendering with UpdatePanels**

One of the most fascinating controls in the ASP.NET AJAX framework is the *UpdatePanel*. This new control replaces the need for a page to refresh during a postback. Only portions of a page, designated by the UpdatePanel, are updated. This technique is known as *partial-page rendering* and can be highly effective in improving the user experience.

6.1.1. Evolution of the UpdatePanel

For years, programming with the XMLHttpRequest object has been the most commonly used approach for communicating with the server from client-side script. The complexities involved in coding those types of applications scared away a lot of developers. To assist, the overall scripting model in ASP.NET 2.0 was significantly enhanced to introduce the idea of *script callbacks*—a way for server controls to communicate with client-side scripts between callbacks. This model was powerful because it offered access to the state of all the controls on the page during a callback. Unfortunately, many developers found the model difficult to work with, and numerous concerns were raised. The lack of support for passing complex types as parameters to the server (only strings were allowed) made the prototype too rigid and exposed its limitations. Developers began to look elsewhere for solutions.

In an effort to address these concerns, members of the ASP.NET team began work on a communication library built on top of the callbacks. The primary objective of the library was to simplify the use of callbacks and to provide a rich set of APIs for enabling the exchange of complex and simple types between the server and client. From this library came a control called the *RefreshPanel*. The purpose of the RefreshPanel was to offer a server control that refreshed the contents of a page without a page refresh. Out of this hard work, the UpdatePanel emerged, with deeper integration into the page lifecycle and a more transparent footprint on the page.

NOTE

A *callback* is a piece of code that is passed in as a parameter or argument to other code. The other piece of code can call the callback code (usually a function) at any time, even numerous times, in response to some processing.

<table>
<tr><td></td><td></td></tr>
<tr><td>4.</td><td>Differentiate AJAX forms with HTML5 forms.

AJAX is the name of a communication architecture between web pages and server side.

JQuery is a javascript library that is written for unifying JS method calls (regarding DOM manipulation, String and Array functions, DOM queries...etc.) in all browsers.

HTML5 is the rendering specification to be implemented by all browser providers. For this rendering to work JS Engine should also be updated, so HTML5 also means a JS engine with new features like drawing on a canvas.</td></tr>
<tr><td>5.</td><td>What is XML Http Request object? List its properties.

<u>The XMLHttpRequest Object</u>

With the XMLHttpRequest object you can update parts of a web page, without reloading the whole page.

The XMLHttpRequest Object

The XMLHttpRequest object is used to exchange data with a server behind the scenes.

The XMLHttpRequest object is the developers dream, because you can:

• Update a web page without reloading the page
• Request data from a server after the page has loaded
• Receive data from a server after the page has loaded
• Send data to a server in the background

XMLHttpRequest Object Methods

<table>
<tr><th>Method</th><th>Description</th></tr>
<tr><td>abort()</td><td>Cancels the current request</td></tr>
<tr><td>getAllResponseHeaders()</td><td>Returns header information</td></tr>
<tr><td>getResponseHeader()</td><td>Returns specific header information</td></tr>
<tr><td>open(method,url,async,uname,pswd)</td><td>Specifies the type of request, the URL, if the request should be handled asynchronously or not, and other optional attributes of a request</td></tr>
</table>
</td></tr>
</table>

	method: the type of request: GET or POST url: the location of the file on the server async: true (asynchronous) or false (synchronous)
send(string)	send(string) Sends the request off to the server. string: Only used for POST requests
setRequestHeader()	Adds a label/value pair to the header to be sent

XMLHttpRequest Object Properties

Property	**Description**
onreadystatechange	Stores a function (or the name of a function) to be called automatically each time the readyState property changes
readyState	Holds the status of the XMLHttpRequest. Changes from 0 to 4: 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 (e.g. "404" for "Not Found" or "200" for "OK")
statusText	Returns the status-text (e.g. "Not Found" or "OK")

6. **Summarize** the need of SOAP and show its structure.

XML Soap

- SOAP stands for **S**imple **O**bject **A**ccess **P**rotocol
- SOAP is an application communication protocol
- SOAP is a format for sending and receiving messages
- SOAP is platform independent

	• SOAP is based on XML • SOAP is a W3C recommendation Why SOAP? It is important for web applications to be able to communicate over the Internet. The best way to communicate between applications is over HTTP, because HTTP is supported by all Internet browsers and servers. SOAP was created to accomplish this. A SOAP message is an ordinary XML document containing the following elements: • An Envelope element that identifies the XML document as a SOAP message • A Header element that contains header information • A Body element that contains call and response information • A Fault element containing errors and status information

7.	Can you **develop** the service end point interface in RPC?
	Example
	
	Sample Java Beans service endpoint implementation and interface
	
	The following example illustrates a simple explicit Java Beans service endpoint implementation and the associated service endpoint interface.
	
	`/** This is an excerpt from the service implementation file, EchoServicePortTypeImpl.java package com.ibm.was.wssample.echo;`
	`import java.io.ByteArrayInputStream;`
	`import java.io.ByteArrayOutputStream;`
	`import javax.xml.bind.JAXBContext;`
	`import javax.xml.bind.Marshaller;`
	`import javax.xml.bind.Unmarshaller;`
	`import javax.xml.transform.stream.StreamSource;`
	
	`@javax.jws.WebService(serviceName = "EchoService", endpointInterface = "com.ibm.was.wssample.echo.EchoServicePortType",`
	
	`targetNamespace="http://com/ibm/was/wssample/echo/", portName="EchoServicePort")`
	`public class EchoServicePortTypeImpl implements EchoServicePortType {`
	
	`public EchoServicePortTypeImpl() {`
	`}`
	
	`public String invoke(String obj) {`
	`String str;`
	`....`
	`str = obj;`
	`....`
	
	`return str;`

```
                }

}
/**  This is a sample EchoServicePortType.java service interface */
 import javax.jws.WebMethod;
 import javax.jws.WebParam;
 import javax.jws.WebResult;
 import javax.jws.WebService;
 import javax.xml.ws.*;


@WebService(name = "EchoServicePortType",
                    targetNamespace =
"http://com/ibm/was/wssample/echo/",
                    wsdlLocation="WEB-INF/wsdl/Echo.wsdl")
public interface EchoServicePortType {


    /** ...the method process ...*/
        @WebMethod
    @WebResult(name = "response", targetNamespace =
"http://com/ibm/was/wssample/echo/")
    @RequestWrapper(localName = "invoke", targetNamespace =
"http://com/ibm/was/wssample/echo/",  className =
"com.ibm.was.wssample.echo.Invoke")
    @ResponseWrapper(localName = "echoStringResponse",
targetNamespace = "http://com/ibm/was/wssample/echo/", className =
"com.ibm.was.wssample.echo.EchoStringResponse")
    public String invoke(
        @WebParam(name = "arg0", targetNamespace =
"http://com/ibm/was/wssample/echo/")
        String arg0);

}
```

Sample Provider endpoint implementation

The following example illustrates a simple Provider service endpoint interface for a Java class.

```
package jaxws.provider.source;
 import javax.xml.ws.Provider;
 import javax.xml.ws.WebServiceProvider;
 import javax.xml.transform.Source;

@WebServiceProvider() public class SourceProvider implements
Provider<Source> {

    public Source invoke(Source data) {
        return data;
    }
}
```

8. **List** any four examples of web services.

A Web Service Example

In the following example we will use ASP.NET to create a simple Web Service that converts the temperature from Fahrenheit to Celsius, and vice versa:

```
<%@ WebService Language="VBScript" Class="TempConvert" %>

Imports System
Imports System.Web.Services

Public Class TempConvert :Inherits WebService

<WebMethod()> Public Function FahrenheitToCelsius(ByVal
Fahrenheit As String) As String
  dim fahr
  fahr=trim(replace(Fahrenheit,",","."))
  if fahr="" or IsNumeric(fahr)=false then return "Error"
  return ((((fahr) - 32) / 9) * 5)
end function

<WebMethod()> Public Function CelsiusToFahrenheit(ByVal Celsius
As String) As String
  dim cel
  cel=trim(replace(Celsius,",","."))
  if cel="" or IsNumeric(cel)=false then return "Error"
  return ((((cel) * 9) / 5) + 32)
end function

end class
```

This document is saved as an .asmx file. This is the ASP.NET file extension for XML Web Services.

Put the Web Service on Your Web Site

Using a form and the HTTP POST method, you can put the web service on your site, like this:

Fahrenheit to Celsius: [ ] Submit

Celsius to Fahrenheit: [ ] Submit

code to add the Web Service to a web page:

	<form action='tempconvert.asmx/FahrenheitToCelsius' method="post" target="_blank"> <table> <tr> <td>Fahrenheit to Celsius:</td> <td> <input class="frmInput" type="text" size="30" name="Fahrenheit"> </td> </tr> <tr> <td></td> <td align="right"> <input type="submit" value="Submit" class="button"> </td> </tr> </table> </form> <form action='tempconvert.asmx/CelsiusToFahrenheit' method="post" target="_blank"> <table> <tr> <td>Celsius to Fahrenheit:</td> <td> <input class="frmInput" type="text" size="30" name="Celsius"> </td> </tr> <tr> <td></td> <td align="right"> <input type="submit" value="Submit" class="button"> </td> </tr> </table> </form> Substitute the "tempconvert.asmx" with the address of your web service like: http://www.example.com/xml/tempconvert.asmx
9.	**Discover** an example for web service registry along with its functions. **Web Services Discovery** provides access to software systems over the Internet using standard protocols. In the most basic scenario there is a *Web Service Provider* that publishes a service and a *Web Service Consumer* that uses this service. Web Service Discovery is the process of finding suitable web services for a given task.[1] Publishing a web service involves creating a software artifact and making it accessible to potential consumers. Web service providers augment a service endpoint

	interface with an interface description using the Web Services Description Language (WSDL) so that a consumer can use the service. **Universal Description, Discovery, and Integration** (**UDDI**) is an XML-based registry for business internet services. A provider can explicitly register a service with a *Web Services Registry* such as UDDI or publish additional documents intended to facilitate discovery such as Web Services Inspection Language (WSIL) documents. The service users or consumers can search web services manually or automatically. The implementation of UDDI servers and WSIL engines should provide simple search APIs or web-based GUI to help find Web services. Web services may also be discovered using multicast mechanisms like WS-Discovery, thus reducing the need for centralized registries in smaller networks.
10.	**Analyze** the need for web service. **We should use web services as it comes with various advantages listed below** **Re-usability** Once we develop some **business logic**,we can make it **reuse** for other applications **Example:** If **10 different applications** requires to **use** our **logic** We can expose our **logic** over a **network** as a **web service** So all the **10 different application** can **access** it from the **network**. **Interoperability** It provides the freedom for a **developers** to choose whatever the **technology** they want to use for development. Web services uses a **set of standards** and **protocols** and enable us to achieve interoperability. Hence applications developed in **Java,Mainframe,Ruby** or any other technology can call the web service and use it. **Loosely coupled** Web service exist **independent** of the other parts of the application that uses it. So any **changes** to the **application** can be made **without affecting** the web service. **Deployability** It is very **easy** to **deploy** the **web application** as they are deployed over standard internet technologies.
11.	**Give** the uses of WSDL along with its definition.

WSDL stands for Web Services Description Language. It is the standard format for describing a web service. WSDL was developed jointly by Microsoft and IBM.

Features of WSDL

- WSDL is an XML-based protocol for information exchange in decentralized and distributed environments.
- WSDL definitions describe how to access a web service and what operations it will perform.
- WSDL is a language for describing how to interface with XML-based services.
- WSDL is an integral part of Universal Description, Discovery, and Integration (UDDI), an XML-based worldwide business registry.
- WSDL is the language that UDDI uses.
- WSDL is pronounced as 'wiz-dull' and spelled out as 'W-S-D-L'.

WSDL Usage

WSDL is often used in combination with SOAP and XML Schema to provide web services over the Internet. A client program connecting to a web service can read the WSDL to determine what functions are available on the server. Any special datatypes used are embedded in the WSDL file in the form of XML Schema. The client can then use SOAP to actually call one of the functions listed in the WSDL.

12. **Compare** SOAP and HTTP.

Difference between SOAP and HTTP

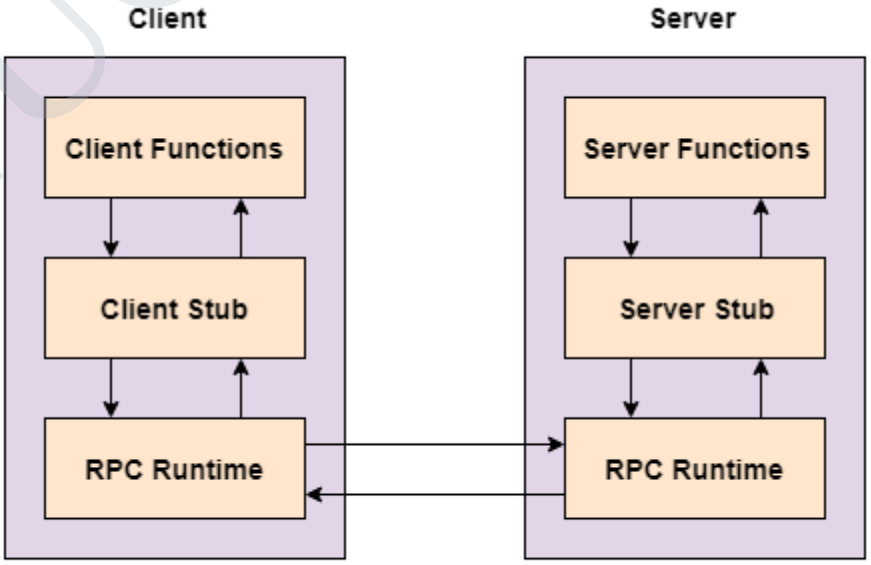
SOAP	HTTP
➢ SOAP was originally defined as S- Simple O- Object A-Access P-protocol. ➢ It is a protocol specification which is used for exchanging structured information. ➢ It is used in the implementation of web services in computer-based networks. ➢ SOAP for its message format relies on XML Information set and sometimes relies on other application layer protocols as well, such as Hypertext Transfer Protocol (HTTP) or Simple	➢ The HTTP or Hypertext Transfer Protocol (HTTP) is an application protocol which is used for distributed, collaborative and hypermedia information systems. ➢ HTTP is widely regarded as the foundation of data communication for the World Wide Web (WWW). ➢ Hypertext is a structured text that uses logical links or hyperlinks between those nodes that containing text. HTTP is the protocol

	Mail Transfer Protocol (SMTP). ➢ It is used for message negotiation and transmission mainly. ➢ SOAP forms the foundation layer of a web services protocol stack.	for exchanging or transferring hypertext. ➢ The standards development of HTTP when it was innovated was coordinated by the Internet Engineering Task Force and the World Wide Web Consortium also called as W3C.
13.	**Summarize** the need for enhancing security in web services. **Definition - What does *Web Services Security (WS Security)* mean?** Web Services Security (WS Security) is a specification that defines how security measures are implemented in web services to protect them from external attacks. It is a set of protocols that ensure security for SOAP-based messages by implementing the principles of confidentiality, integrity and authentication. Because Web services are independent of any hardware and software implementations, WS-Security protocols need to be flexible enough to accommodate new security mechanisms and provide alternative mechanisms if an approach is not suitable. Because SOAP-based messages traverse multiple intermediaries, security protocols need to be able to identify fake nodes and prevent data interpretation at any nodes. WS-Security combines the best approaches to tackle different security problems by allowing the developer to customize a particular security solution for a part of the problem. For example, the developer can select digital signatures for non-repudiation and Kerberos for authentication.	
14.	**Name** the types of indicators along with the definition. **Web Services Security (WSS)** Web Services Security (WSS or WS-Security) describes enhancements to SOAP messaging in order to provide quality of protection through message integrity, and single message authentication. These mechanisms can be used to accommodate a wide variety of security models and encryption technologies. The scope of the Web Services Security Technical Committee is the support of security mechanisms in the following areas:	

	Using XML Signature to provide SOAP message integrity for Web Services Using XML Encryption to provide SOAP message confidentiality for Web Services Attaching and/or referencing security tokens in headers of SOAP messages. Options include: Username token SAML XrML Kerberos X.509 Carrying security information for potentially multiple, designated actors Associating signatures with security tokens Each of the security mechanisms will use implementation and language neutral XML formats defined in XML Schema.
15.	**Classify** the basic concepts behind JAX-RPC technology. JAX-RPC Java APIs for XML-based Remote Procedure Call (JAX-RPC) help with Web service interoperability and accessibility by defining Java APIs that Java applications use to develop and access Web services. JAX-RPC fully embraces the heterogeneous nature of Web services -- it allows a JAX-RPC client to talk to another Web service deployed on a different platform and coded in a different language. Similarly, it also allows clients on other platforms and coded in different languages to talk to a JAX-RPC service. JAX-RPC also defines the mapping between WSDL service descriptions and Java interfaces. Th the JAX-RPC technology and describes its client and server programming models. JAX-RPC hides the complexity of underlying protocols and message-level processing from application developers crafting Web services using the Java 2 platform. The API combines XML with Remote Procedure Call (RPC), which is a mechanism enabling clients to execute procedures on distributed or remote systems, so that developers can build Web services and clients. The JAX-RPC remote procedure calls are represented by an XML infoset and they are carried over a network transport. While the JAX-RPC APIs rely on a XML-based protocol and a network transport, the APIs themselves are independent of a specific protocol or transport. The current JAX-RPC implementation relies on the SOAP 1.1 protocol and HTTP 1.1 network transport.
16.	**What are the benefits of UDDI?** Problems the UDDI specification can help to solve: • Making it possible to discover the right business from the millions currently online • Defining how to enable commerce once the preferred business is discovered • Reaching new customers and increasing access to current customers • Expanding offerings and extending market reach

	• Solving customer-driven need to remove barriers to allow for rapid participation in the global Internet economy • Describing services and business processes programmatically in a single, open, and secure environment
17.	**What are the core elements of UDDI?** UDDI defines four core data elements within the data model: • businessEntity (modeling business information) • businessService (describing a service) • tModel (describing specifications, classifications, or identifications) • binding Template (mapping between a businessService and the set of tModels that describe its technical fingerprint)
18.	**Rewrite** the definition for UDDI. UDDI is an XML-based standard for describing, publishing, and finding web services. • UDDI stands for **Universal Description, Discovery, and Integration.** • UDDI is a specification for a distributed registry of web services. • UDDI is a platform-independent, open framework. • UDDI can communicate via SOAP, CORBA, Java RMI Protocol. • UDDI uses Web Service Definition Language(WSDL) to describe interfaces to web services. • UDDI is seen with SOAP and WSDL as one of the three foundation standards of web services. • UDDI is an open industry initiative, enabling businesses to discover each other and define how they interact over the Internet. UDDI has two sections − • A registry of all web service's metadata, including a pointer to the WSDL description of a service. • A set of WSDL port type definitions for manipulating and searching that registry.
19.	**Give** the usage of UDDI in web service. UDDI is an XML-based standard for describing, publishing, and finding web services. • UDDI stands for **Universal Description, Discovery, and Integration.** • UDDI is a specification for a distributed registry of web services. • UDDI is a platform-independent, open framework.

	• UDDI can communicate via SOAP, CORBA, Java RMI Protocol. • UDDI uses Web Service Definition Language(WSDL) to describe interfaces to web services. • UDDI is seen with SOAP and WSDL as one of the three foundation standards of web services. • UDDI is an open industry initiative, enabling businesses to discover each other and define how they interact over the Internet. UDDI has two sections − • A registry of all web service's metadata, including a pointer to the WSDL description of a service. • A set of WSDL port type definitions for manipulating and searching that registry.
20.	**Define** WSDL. WSDL stands for Web Services Description Language. It is the standard format for describing a web service. WSDL was developed jointly by Microsoft and IBM. ## Features of WSDL • WSDL is an XML-based protocol for information exchange in decentralized and distributed environments. • WSDL definitions describe how to access a web service and what operations it will perform. • WSDL is a language for describing how to interface with XML-based services. • WSDL is an integral part of Universal Description, Discovery, and Integration (UDDI), an XML-based worldwide business registry. • WSDL is the language that UDDI uses. • WSDL is pronounced as 'wiz-dull' and spelled out as 'W-S-D-L'. ## WSDL Usage WSDL is often used in combination with SOAP and XML Schema to provide web services over the Internet. A client program connecting to a web service can read the WSDL to determine what functions are available on the server. Any special datatypes used are embedded in the WSDL file in the form of XML Schema. The client can then use SOAP to actually call one of the functions listed in the WSDL.
	PART-B
1.	(i) **Describe** in detail about the AJAX architecture.

	(ii) **List out** the call back methods.
2.	(i) **Analyze** various concepts of RPC. ## Remote Procedure Call (RPC) A remote procedure call is an interprocess communication technique that is used for client-server based applications. It is also known as a subroutine call or a function call. A client has a request message that the RPC translates and sends to the server. This request may be a procedure or a function call to a remote server. When the server receives the request, it sends the required response back to the client. The client is blocked while the server is processing the call and only resumed execution after the server is finished. The sequence of events in a remote procedure call are given as follows: • The client stub is called by the client. • The client stub makes a system call to send the message to the server and puts the parameters in the message. • The message is sent from the client to the server by the client's operating system. • The message is passed to the server stub by the server operating system. • The parameters are removed from the message by the server stub. • Then, the server procedure is called by the server stub. A diagram that demonstrates this is as follows: 

Advantages of Remote Procedure Call

Some of the advantages of RPC are as follows:

- Remote procedure calls support process oriented and thread oriented models.
- The internal message passing mechanism of RPC is hidden from the user.
- The effort to re-write and re-develop the code is minimum in remote procedure calls.
- Remote procedure calls can be used in distributed environment as well as the local environment.
- Many of the protocol layers are omitted by RPC to improve performance.

Disadvantages of Remote Procedure Call

Some of the disadvantages of RPC are as follows:

- The remote procedure call is a concept that can be implemented in different ways. It is not a standard.
- There is no flexibility in RPC for hardware architecture. It is only interaction based.
- There is an increase in costs because of remote procedure call.

(ii) **Explain** the basic concepts behind JAX-RPC.

JAX-RPC stands for Java API for XML-based RPC. It's an API for building Web services and clients that used remote procedure calls (RPC) and XML. Often used in a distributed client/server model, an RPC mechanism enables clients to execute procedures on other systems.

In JAX-RPC, a remote procedure call is represented by an XML-based protocol such as SOAP. The SOAP specification defines envelope structure, encoding rules, and a convention for representing remote procedure calls and responses. These calls and responses are transmitted as SOAP messages over HTTP. In this release, JAX-RPC relies on SOAP 1.1 and HTTP 1.1.

Although JAX-RPC relies on complex protocols, the API hides this complexity from the application developer. On the server side, the developer specifies the remote procedures by defining methods in an interface written in the Java programming language. The developer also codes one or more classes that implement those methods. Client programs are also easy to code. A client creates a proxy, a local object representing the service, and then simply invokes methods on the proxy.

With JAX-RPC, clients and Web services have a big advantage--the platform independence of the Java programming language. In addition, JAX-RPC is not restrictive: a JAX-RPC client can access a Web service that is not running on the Java platform and vice versa. This flexibility is possible because JAX-RPC uses technologies defined by the World Wide

	Web Consortium (W3C): HTTP, SOAP, and the Web Service Description Language (WSDL). WSDL specifies an XML format for describing a service as a set of endpoints operating on messages.
3.	**Explain** in detail with an example of Java Web Services. **With a simple example illustrate the steps to create a java web service. (NOV/DEC 2012)** **Writing a java web service** **Currency conversion Service** ◆ Writing a server for a service using JWSDP 1.3 tools ◆ Application: currency converter ■ Three operations: ● fromDollars ● fromEuros ● fromYen ■ Input: value in specified currency ■ Output: object containing input value and equivalent values in other two currencies **Writing server software** 1. Write service endpoint interface • May need to write additional classes representing data structures 2. Write class implementing the interface 3. Compile classes 4. Create configuration files and run JWSDP tools to create web service 5. Deploy web service to Tomcat **service endpoint interface** ◆ The Web service endpoint interface is used to define the 'Web services methods'. ◆ A Web service endpoint interface must conform to the rules of a JAX-RPC service definition interface. ◆ a service endpoint interface (SEI) that defines the interface of the web service. ◆ Configuration files are XML files that can be changed as needed. Developers can use configuration files to change settings without recompiling applications. Administrators can use configuration files to set policies that affect how applications run on their computers. ◆ config.xml : Defines the URL for WSDL file location. Each Web services has a corresponding WSDL (Web service Definition Language) document. **JWSDP: Server** **Rules for Service endpoint interface** ■ Must extend java.rmi.Remote ● declares a set of methods that may be invoked from a remote Java Virtual Machine(JVM) ■ Every method must throw java.rmi.RemoteException

- Parameter/return value data types are restricted
- No public static final declarations (global constants) It must not have constant declarations

- **Allowable parameter/return value data types**
- Java primitives (int, boolean, *etc*.)
- Primitive wrapper classes (Integer, *etc*.)
- String, Date, Calendar, BigDecimal, BigInteger
- java.xml.namespace.QName, java.net.URI
- Struct: class consisting entirely of public instance variables
- Array of any of the above

- **Struct for currency converter app (data type for return values)**

```
package myCurCon;

public class ExchangeValues {
    public double dollars;
    public double euros;
    public double yen;
}
```

- **Service endpoint interface**

```
package myCurCon;

public interface CurCon extends java.rmi.Remote {
    public ExchangeValues fromDollars(double dollars)
        throws java.rmi.RemoteException;
    public ExchangeValues fromEuros(double euros)
        throws java.rmi.RemoteException;
    public ExchangeValues fromYen(double yen)
        throws java.rmi.RemoteException;
}
```

- Three files ExchangeValues.java, CurCon.java and CurConImpl.java written for the web se
- Class CurConImpl contains methods implements sevice endpoint interface, for *example:*

```
public ExchangeValues fromDollars(double dollars)
    throws java.rmi.RemoteException
{
    ExchangeValues ev = new ExchangeValues();
    ev.dollars = dollars;
    ev.euros = dollars * dollar2euro;
    ev.yen = dollars * dollar2yen;
    return ev;
}
```

Packaging server software

- Configuration file input to wscompile to create server

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration
  xmlns="http://java.sun.com/xml/ns/jax-rpc/ri/config">
  <service
    name="HistoricCurrencyConverter"
    targetNamespace="http://tempuri.org/wsdl"
    typeNamespace="http://tempuri.org/types"
    packageName="myCurCon">
    <interface name="myCurCon.CurCon" />
  </service>
</configuration>
```

- Configuration file for web service

```
<?xml version="1.0" encoding="UTF-8"?>
<webServices
  xmlns="http://java.sun.com/xml/ns/jax-rpc/ri/dd"
  version="1.0"
  targetNamespaceBase="http://tempuri.org/wsdl"
  typeNamespaceBase="http://tempuri.org/types"
```

- Configuration file for web service

```
  urlPatternBase="/converter">

  <endpoint
    name="CurrConverter"
    displayName="Currency Converter"
    description=
      "Converts between dollars, euros, and yen."
    interface="myCurCon.CurCon"
    model="/WEB-INF/model.xml.gz"
    implementation="myCurCon.CurConImpl"/>

  <endpointMapping
    endpointName="CurrConverter"
    urlPattern="/currency" />

</webServices>
```

Context path

Like servlet in web.xml

Like servlet-mapping in web.xml

- Also need a minimal web.xml

```
<web-app
  xmlns="http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"
  version="2.4">

  <display-name>Historic Currency Converter</display-name>
  <description>
    This web service converts between three currencies using their
    exchange rates as of a fixed date.

  </description>
</web-app>
```

Run jar and wsdeploy to create a Web Archive (WAR) file converter.war

- Name must match urlPatternBase value

- jaxrpc-ri.xml: Defines the various end points for referencing a Web service.
- *wscompile*: The wscompile tool generates stubs, and WSDL files used in JAX-RPC clients and

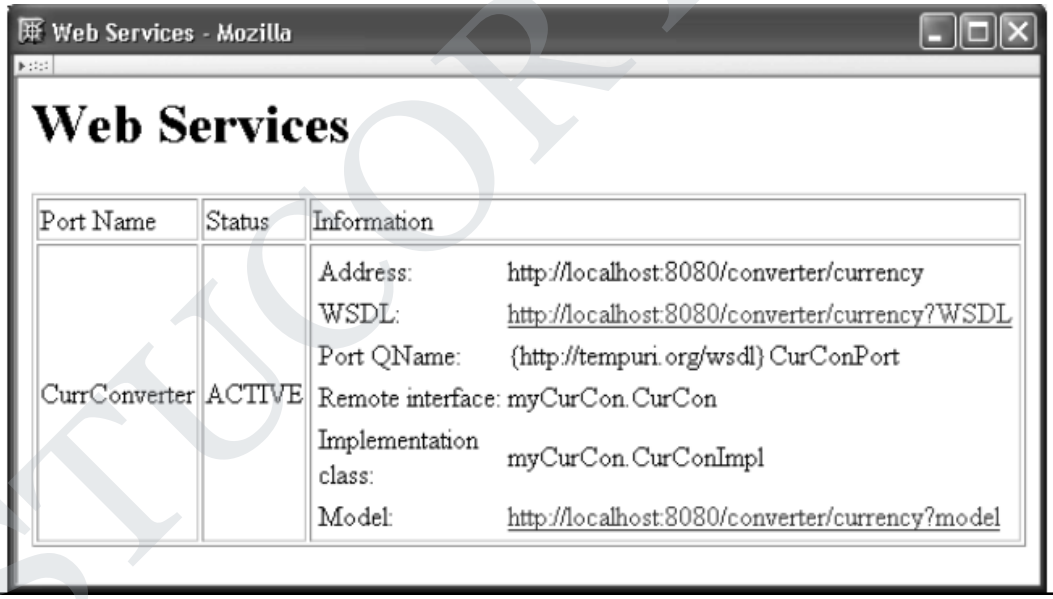
services. The tool reads as input a configuration file and either a WSDL file or an RMI interface

that defines the service.

- *wsdeploy*: Reads a WAR file (something like Jar file) and the jaxrpc-ri.xml file and then

generates another WAR file that is ready for deployment

- Write service endpoint interface
- May need to write additional classes representing data structures
- Write class implementing the interface
- Compile classes
- Create configuration files and run JWSDP tools to create web service
- Deploy web service to Tomcat
- Just copy converter.war to Tomcat webapps directory
- May need to use Manager app to deploy
- Enter converter.war in "WAR or Directory URL" text box
- Testing success:
- Visit http://localhost:8080/converter/currency



4. **Discuss** in detail the architecture of web services.

Architecture of Web Services

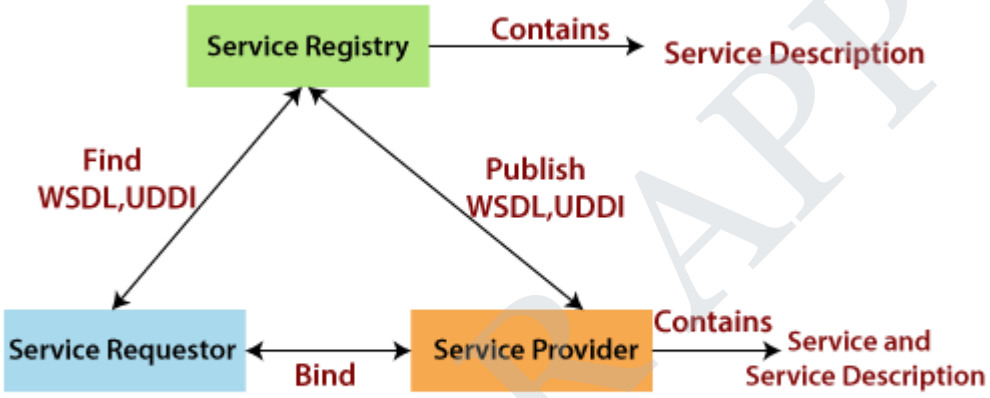
The Web Services architecture describes how to instantiate the elements and implement the operations in an interoperable manner.

The architecture of web service interacts among three roles: **service provider, service requester,** and **service registry**. The interaction involves the three operations: **publish, find,** and **bind**. These operations and roles act upon

the **web services artifacts**. The web service artifacts are the web service software module and its description.

The service provider hosts a network-associable module (web service). It defines a service description for the web service and publishes it to a service requestor or service registry. These service requestor uses a find operation to retrieve the service description locally or from the service registry. It uses the service description to bind with the service provider and invoke with the web service implementation.

The following figure illustrates the operations, roles, and their interaction.



Web Service Roles, Operations and Artifacts

Roles in a Web Service Architecture

There are three roles in web service architecture:

- Service Provider
- Service Requestor
- Service Registry

Service Provider

From an architectural perspective, it is the platform that hosts the services.

Service Requestor

Service requestor is the application that is looking for and invoking or initiating an interaction with a service. The browser plays the requester role, driven by a consumer or a program without a user interface.

Service Registry

	Service requestors find service and obtain binding information for services during development. ## Operations in a Web Service Architecture Three behaviors that take place in the microservices: ○ Publication of service descriptions **(Publish)** ○ Finding of services descriptions **(Find)** ○ Invoking of service based on service descriptions **(Bind)** **Publish:** In the publish operation, a service description must be published so that a service requester can find the service. **Find:** In the find operation, the service requestor retrieves the service description directly. It can be involved in two different lifecycle phases for the service requestor: ○ At design, time to retrieve the service's interface description for program development. ○ And, at the runtime to retrieve the service's binding and location description for invocation. **Bind:** In the bind operation, the service requestor invokes or initiates an interaction with the service at runtime using the binding details in the service description to locate, contact, and invoke the service. ## Artifacts of the web service There are two artifacts of web services: ○ Service ○ Service Registry **Service:** A service is an **interface** described by a service description. The service description is the implementation of the service. A service is a software module deployed on network-accessible platforms provided by the service provider. It interacts with a service requestor. Sometimes it also functions as a requestor, using other Web Services in its implementation. **Service Description:** The service description comprises the details of the **interface** and **implementation** of the service. It includes its **data types, operations, binding information,** and **network location**. It can also categorize other metadata to enable discovery and utilize by service requestors. It can be published to a service requestor or a service registry.
5.	(i) **Deduce** any two elements of WSDL.

WSDL (Web services description language)

A web service cannot be used if it cannot be found. The client invoking the web service should know where the web service actually resides.

Secondly, the client application needs to know what the web service actually does, so that it can invoke the right web service. This is done with the help of the WSDL, known as the Web services description language. The WSDL file is again an XML-based file which basically tells the client application what the web service does. By using the WSDL document, the client application would be able to understand where the web service is located and how it can be utilized.

Web Service Example

An example of a WSDL file is given below.

```
<definitions>
   <message name="TutorialRequest">
      <part name="TutorialID" type="xsd:string"/>
   </message>

   <message name="TutorialResponse">
      <part name="TutorialName" type="xsd:string"/>
   </message>

   <portType name="Tutorial_PortType">
      <operation name="Tutorial">
         <input message="tns:TutorialRequest"/>
         <output message="tns:TutorialResponse"/>
      </operation>
   </portType>

   <binding name="Tutorial_Binding" type="tns:Tutorial_PortType">
      <soap:binding style="rpc"
         transport="http://schemas.xmlsoap.org/soap/http"/>
      <operation name="Tutorial">
         <soap:operation soapAction="Tutorial"/>
         <input>
            <soap:body
               encodingStyle="http://schemas.xmlsoap.org/soap/encodi
ng/"
               namespace="urn:examples:Tutorialservice"
```

```
                use="encoded"/>
            </input>

              <output>
               <soap:body
                  encodingStyle="http://schemas.xmlsoap.org/soap/encodi
ng/"
                  namespace="urn:examples:Tutorialservice"
                  use="encoded"/>
            </output>
         </operation>
      </binding>
</definitions>
```

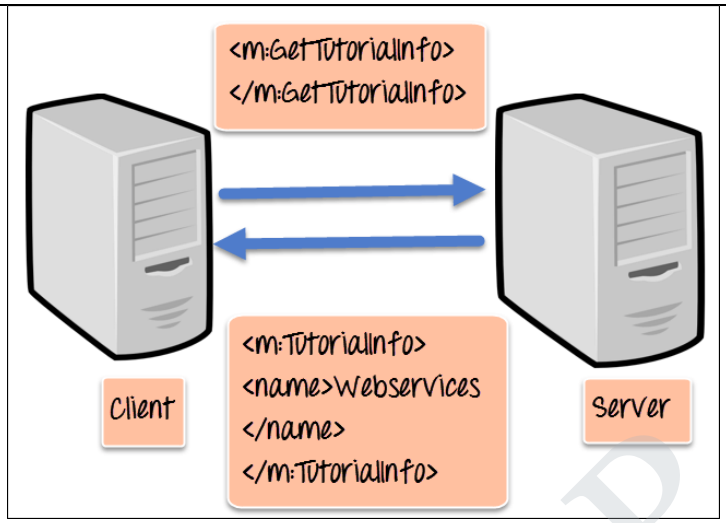
(ii) **Explain** the steps for writing web service.

6. **Describe** briefly about the elements of WSDL.

he important aspects to note about the above WSDL declaration are as follows;

1. **<message>** - The message parameter in the WSDL definition is used to define the different data elements for each operation performed by the web service. So in the example above, we have 2 messages which can be exchanged between the web service and the client application, one is the "TutorialRequest", and the other is the "TutorialResponse" operation. The TutorialRequest contains an element called "TutorialID" which is of the type string. Similarly, the TutorialResponse operation contains an element called "TutorialName" which is also a type string.
2. **<portType>** - This actually describes the operation which can be performed by the web service, which in our case is called Tutorial. This operation can take 2 messages; one is an input message, and the other is the output message.
3. **<binding>** - This element contains the protocol which is used. So in our case, we are defining it to use http (**http://schemas.xmlsoap.org/soap/http**). We also specify other details for the body of the operation, like the

	namespace and whether the message should be encoded.
7.	(i) **Summarize** on the structure of SOAP. **SOAP (Simple Object Access Protocol)** SOAP is known as a transport-independent messaging protocol. SOAP is based on transferring XML data as SOAP Messages. Each message has something which is known as an XML document. Only the structure of the XML document follows a specific pattern, but not the content. The best part of Web services and SOAP is that its all sent via HTTP, which is the standard web protocol. Here is what a SOAP message consists of ○ Each SOAP document needs to have a root element known as the <Envelope> element. The root element is the first element in an XML document. ○ The "envelope" is in turn divided into 2 parts. The first is the header, and the next is the body. ○ The header contains the routing data which is basically the information which tells the XML document to which client it needs to be sent to. ○ The body will contain the actual message. The diagram below shows a simple example of the communication via SOAP.



(ii) **Describe** briefly about SOAP & HTTP.

8. (i) **Demonstrate** the building blocks of SOAP.

XML Soap

- SOAP stands for **S**imple **O**bject **A**ccess **P**rotocol
- SOAP is an application communication protocol
- SOAP is a format for sending and receiving messages
- SOAP is platform independent
- SOAP is based on XML
- SOAP is a W3C recommendation

Why SOAP?

It is important for web applications to be able to communicate over the Internet.

The best way to communicate between applications is over HTTP, because HTTP is supported by all Internet browsers and servers. SOAP was created to accomplish this.

SOAP provides a way to communicate between applications running on different operating systems, with different technologies and programming languages.

SOAP Building Blocks

A SOAP message is an ordinary XML document containing the following elements:

- An Envelope element that identifies the XML document as a SOAP message
- A Header element that contains header information
- A Body element that contains call and response information
- A Fault element containing errors and status information

Syntax Rules

Here are some important syntax rules:

- A SOAP message MUST be encoded using XML
- A SOAP message MUST use the SOAP Envelope namespace
- A SOAP message must NOT contain a DTD reference
- A SOAP message must NOT contain XML Processing Instructions

(ii) **Classify** the encoding of struct data and array.

SOAP - Encoding

SOAP includes a built-in set of rules for encoding data types. It enables the SOAP message to indicate specific data types, such as integers, floats, doubles, or arrays.

- SOAP data types are divided into two broad categories − scalar types and compound types.
- Scalar types contain exactly one value such as a last name, price, or product description.
- Compound types contain multiple values such as a purchase order or a list of stock quotes.
- Compound types are further subdivided into arrays and structs.

Compound Types

SOAP arrays have a very specific set of rules, which require that you specify both the element type and array size. SOAP also supports multidimensional arrays, but not all SOAP implementations support multidimensional functionality.

To create an array, you must specify it as an *xsi:type* of array. The array must also include an *arrayType* attribute. This attribute is required to specify the data type for the contained elements and the dimension(s) of the array.

For example, the following attribute specifies an array of 10 double values −

```
arrayType = "xsd:double[10]"
```

In contrast, the following attribute specifies a two-dimensional array of strings −

```
arrayType = "xsd:string[5,5]"
```

Here is a sample SOAP response with an array of double values −

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV = "http://www.w3.org/2001/12/soap-
envelope"
   xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd = "http://www.w3.org/2001/XMLSchema">

   <SOAP-ENV:Body>
      <ns1:getPriceListResponse
         xmlns:ns1 = "urn:examples:pricelistservice"
         SOAP-ENV:encodingStyle =
"http://www.w3.org/2001/12/soap-encoding">

         <return xmlns:ns2 =
"http://www.w3.org/2001/09/soap-encoding"
            xsi:type = "ns2:Array" ns2:arrayType =
"xsd:double[2]">
            <item xsi:type = "xsd:double">54.99</item>
            <item xsi:type = "xsd:double">19.99</item>
         </return>
      </ns1:getPriceListResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Structs contain multiple values, but each element is specified with a unique accessor element. For example, consider an item within a product catalog. In this case, the struct might contain a product SKU, product name, description, and price. Here is how such a struct would be represented in a SOAP message −

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV = "http://www.w3.org/2001/12/soap-
envelope"
   xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd = "http://www.w3.org/2001/XMLSchema">

   <SOAP-ENV:Body>
      <ns1:getProductResponse
         xmlns:ns1 = "urn:examples:productservice"
```

```
          SOAP-ENV:encodingStyle =
"http://www.w3.org/2001/12/soap-encoding">

          <return xmlns:ns2 = "urn:examples" xsi:type =
"ns2:product">
             <name xsi:type = "xsd:string">Red Hat
Linux</name>
             <price xsi:type = "xsd:double">54.99</price>
             <description xsi:type = "xsd:string">
                Red Hat Linux Operating System
             </description>
             <SKU xsi:type = "xsd:string">A358185</SKU>
          </return>
       </ns1:getProductResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

9. **Analyze** the various steps in database driven web service with some example.

Overview of Database Web Services

Web services enable application-to-application interaction over the Web, regardless of platform, language, or data formats. The key ingredients, including Extensible Markup Language (XML), Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL), and Universal Description, Discovery, and Integration (UDDI), have been adopted across the entire software industry. Web services usually refer to services implemented and deployed in middle-tier application servers. However, in heterogeneous and disconnected environments, there is an increasing need to access stored procedures, as well as data and metadata, through Web services interfaces.

The Database Web services technology is a database approach to Web services. It works in the following two directions:

- Accessing database resources as a Web service
- Consuming external Web services from the database

Oracle Database can access Web services through PL/SQL packages and Java classes deployed within the database. Turning Oracle Database into a Web service provider leverages investment in Java stored procedures, PL/SQL packages, predefined SQL queries, and data manipulation language (DML). Conversely, consuming external Web services from the database, together with integration with the SQL engine, enables Enterprise Information Integration.

Using Oracle Database as Web Services Provider

Web Services use industry-standard mechanisms to provide easy access to remote content and applications, regardless of the platform and location of the provider and implementation and data format. Client applications can query and retrieve data from Oracle Database and call stored procedures using standard Web service protocols. There is no dependency on Oracle-specific

database connectivity protocols. This approach is highly beneficial in heterogeneous, distributed, and disconnected environments.

You can call into the database from a Web service, using the database as a service provider. This enables you to leverage existing or new SQL, PL/SQL, Java stored procedures, or Java classes within Oracle Database. You can access and manipulate database tables from a Web service client.

10.

Illustrate on web services for writing web service client along with the description of WSDL.

Creating a Web Service Client

Creating a web service client application always starts with an existing WSDL file.

Typically, you retrieve the WSDL directly from a web service provider using the `wsimport` tool. The `wsimport` tool then generates the corresponding Java source code for the interface described by the WSDL. The Java compiler, `javac`, is then called to compile the source into class files. The programming code uses the generated classes to access the web service.

Creating a Client from WSDL

To create a client from WSDL, you must create the following files:

- Client Java File (fromwsdl)
- Client Configuration File (fromwsdl)
- `build.xml`
- `build.properties`

Client Java File (fromwsdl)

The client Java file defines the functionality of the web service client. The following code shows the `AddNumbersClient.java` file that is provided in the sample.

```
package fromjava.client;

import com.sun.xml.ws.Closeable;
import java.rmi.RemoteException;

public class AddNumbersClient {
  public static void main (String[] args) {
    AddNumbersImpl port = null;
    try {
      port = new
AddNumbersImplService().getAddNumbersImplPort();
      int number1 = 10;
      int number2 = 20;
      System.out.printf ("Invoking addNumbers(%d, %d)\n",
          number1, number2);
      int result = port.addNumbers (number1, number2);
      System.out.printf (
          "The result of adding %d and %d is %d.\n\n",
          number1, number2, result);

      number1 = -10;
```

```
        System.out.printf ("Invoking addNumbers(%d, %d)\n",
            number1, number2);
        result = port.addNumbers (number1, number2);
        System.out.printf (
            "The result of adding %d and %d is %d.\n",
            number1, number2, result);
      } catch (AddNumbersException_Exception ex) {
        System.out.printf (
            "Caught AddNumbersException_Exception: %s\n",
            ex.getFaultInfo ().getDetail ());
      } finally {
        ((Closeable)port).close();
      }
    }
}
```

This file specifies two positive integers that are to be added by the web service, passes the integers to the web service and gets the results from the web service via the `port.addNumbers` method, and prints the results to the screen. It then specifies a negative number to be added, gets the results (which should be an exception), and prints the results (the exception) to the screen.

Client Configuration File (fromwsdl)

This is a sample `custom-client.xml` file. The `wsdlLocation`, `package name`, and `jaxb:package name` xml tags are unique to each client and are highlighted in bold text

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<bindings
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:jaxb="http://java.sun.com/xml/ns/jaxb"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  wsdlLocation="http://localhost:8080/wsit-enabled-fromwsdl/
      addnumbers?wsdl"
  xmlns="http://java.sun.com/xml/ns/jaxws">
  <bindings node="ns1:definitions"
      xmlns:ns1="http://schemas.xmlsoap.org/wsdl/">
    <package name="fromwsdl.client"/>
  </bindings>
  <bindings node="ns1:definitions/ns1:types/xsd:schema
      [@targetNamespace='http://duke.org']"
      xmlns:xs="http://www.w3.org/2001/XMLSchema"
      xmlns:ns1="http://schemas.xmlsoap.org/wsdl/">
    <jaxb:schemaBindings>
      <jaxb:package name="fromwsdl.client"/>
    </jaxb:schemaBindings>
  </bindings>
</bindings>
```

11. (i) **List out** the installation steps of JWSDP.

JWSDP

1. Install JDK 6.0 (i.e., JDK 1.6.0)

   Set up the following environment variables:
   JAVA_HOME C:\Program Files\Java\jdk1.6.0_07

   Add the following path:
   C:\Program Files\Java\jdk1.6.0_07\bin

2. Install JWSDP 2.0 & Tomcat 5.0 for JWSDP (based upon Tomcat 5.0.19 that implements
the Java Server Pages 2.0 and Java Servlet 2.4 specifications)

Set up the following environment variables:
JWSDP_HOME C:\Sun\jwsdp-2.0
ANT_HOME C:\Sun\jwsdp-2.0\apache-ant

Add the following path:
C:\Sun\jwsdp-2.0\jwsdp-shared\bin;C:\Sun\jwsdp-2.0\apache-ant\bin

3. Copy examples.zip into C:\ and extract here

4. Copy lib.zip into C:\Sun\jwsdp-2.0\server directory and extract here
Delete the file "lib.zip"

5. Replace saaj-impl.jar file at the following directories by saaj-impl-1.3.jar.
Rename it to saaj-impl.jar.

C:\Sun\jwsdp-2.0\saaj\lib
C:\Sun\tomcat50-jwsdp\saaj\lib

6. Modify C:\examples\common\build.properties for the first four lines as follows:

tutorial.home=C:
tutorial.install=${tutorial.home}
username=hxu
password=12345

where "hxu" and "12345" are the username and password for the Tomcat server.

7. Build server:
cd C:\examples\jaxrpc\helloservice
ant build

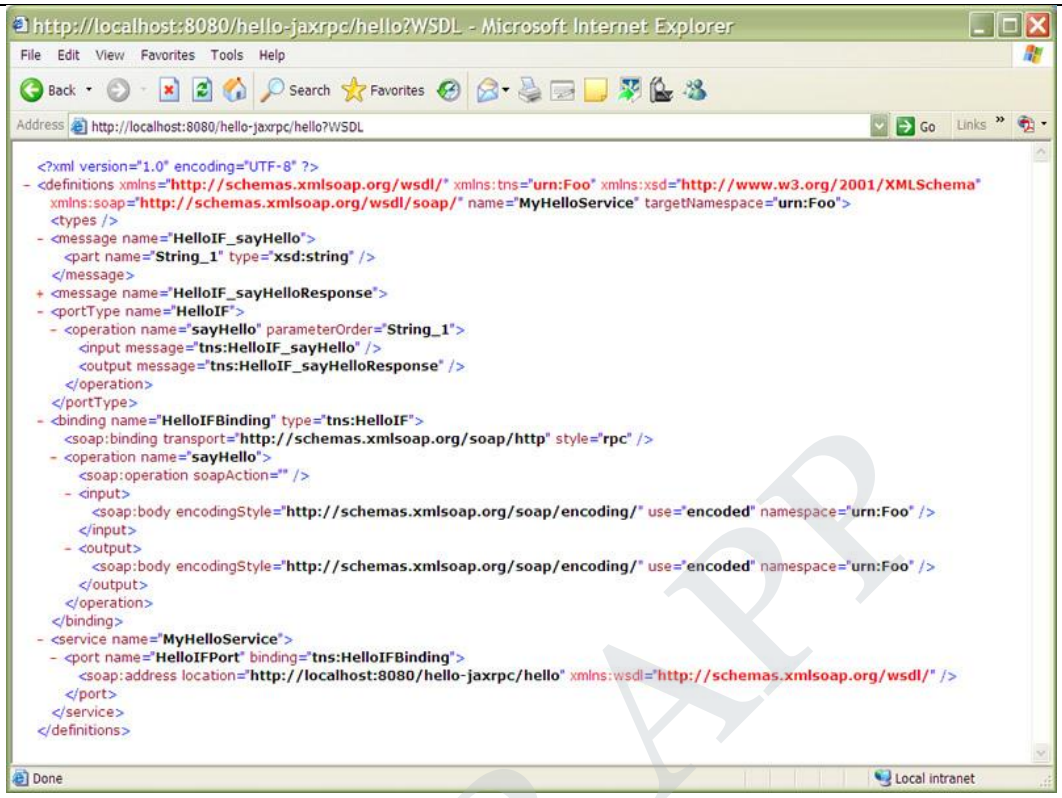
Start Tomcat from JWSDP 2.0

Deploy server:
ant deploy

Note: If application already exists at path /hello-jaxrpc, you should use the
command "ant undeploy" to undeploy the web service first.

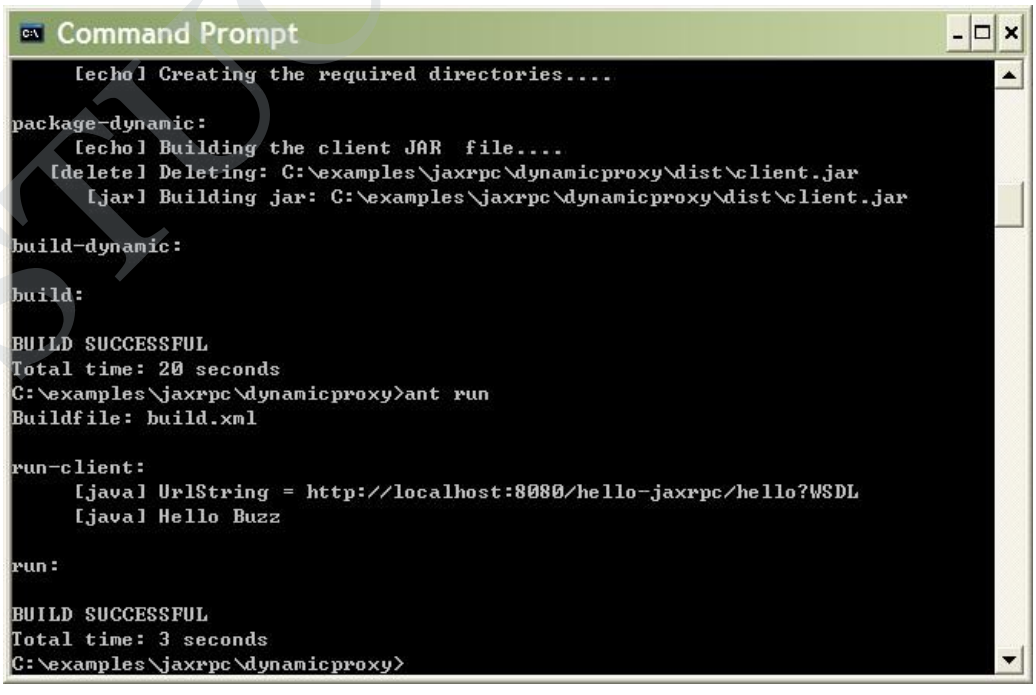
Verify the deployment:
To verify that the service has been successfully deployed, open a browser window
and specify the service endpoint's URL as follows:

http://localhost:8080/hello-jaxrpc/hello?WSDL
You should get the following display.



8. Build client:
   cd C:\examples\jaxrpc\dynamicproxy
   ant build

   Run client:
   ant run



(ii) **Describ**e on Simple Object Access Protocol.

XML Soap

- SOAP stands for **S**imple **O**bject **A**ccess **P**rotocol
- SOAP is an application communication protocol
- SOAP is a format for sending and receiving messages
- SOAP is platform independent
- SOAP is based on XML
- SOAP is a W3C recommendation

Why SOAP?

It is important for web applications to be able to communicate over the Internet.

The best way to communicate between applications is over HTTP, because HTTP is supported by all Internet browsers and servers. SOAP was created to accomplish this.

SOAP provides a way to communicate between applications running on different operating systems, with different technologies and programming languages.

SOAP Building Blocks

A SOAP message is an ordinary XML document containing the following elements:

- An Envelope element that identifies the XML document as a SOAP message
- A Header element that contains header information
- A Body element that contains call and response information
- A Fault element containing errors and status information

Syntax Rules

Here are some important syntax rules:

- A SOAP message MUST be encoded using XML
- A SOAP message MUST use the SOAP Envelope namespace
- A SOAP message must NOT contain a DTD reference
- A SOAP message must NOT contain XML Processing Instructions

12.	(i) **Discuss** the XMLHttpRequest Object with example.

AJAX - The XMLHttpRequest Object

The keystone of AJAX is the XMLHttpRequest object.

The XMLHttpRequest Object

All modern browsers support the XMLHttpRequest object.

The XMLHttpRequest object can be used to exchange data with a server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.

Create an XMLHttpRequest Object

All modern browsers (Chrome, Firefox, IE7+, Edge, Safari Opera) have a built-in XMLHttpRequest object.

Syntax for creating an XMLHttpRequest object:

```
variable = new XMLHttpRequest();
```

Example

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

Access Across Domains

For security reasons, modern browsers do not allow access across domains.

This means that both the web page and the XML file it tries to load, must be located on the same server.

The examples on W3Schools all open XML files located on the W3Schools domain.

If you want to use the example above on one of your own web pages, the XML files you load must be located on your own server.

Example

```
if (window.XMLHttpRequest) {
    // code for modern browsers
    xmlhttp = new XMLHttpRequest();
 } else {
    // code for old IE browsers
```

```
    xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
}
```

(ii) **Describe** about Java web service basics.

JAX-WS Example RPC Style

Creating JAX-WS example is a easy task because it requires no extra configuration settings.

JAX-WS API is inbuilt in JDK, so you don't need to load any extra jar file for it. Let's see a simple example of JAX-WS example in RPC style.

There are created 4 files for hello world JAX-WS example:

1. HelloWorld.java
2. HelloWorldImpl.java
3. Publisher.java
4. HelloWorldClient.java

The first 3 files are created for server side and 1 application for client side.

JAX-WS Server Code

File: HelloWorld.java

```
package com.javatpoint;
import javax.jws.WebMethod;
import javax.jws.WebService;
import javax.jws.soap.SOAPBinding;
import javax.jws.soap.SOAPBinding.Style;
//Service Endpoint Interface
@WebService
@SOAPBinding(style = Style.RPC)
public interface HelloWorld{
    @WebMethod String getHelloWorldAsString(String name);
}
```

File: HelloWorldImpl.java

```
package com.javatpoint;
import javax.jws.WebService;
//Service Implementation
@WebService(endpointInterface = "com.javatpoint.HelloWorld")
```

```
public class HelloWorldImpl implements HelloWorld{
    @Override
    public String getHelloWorldAsString(String name) {
        return "Hello World JAX-WS " + name;
    }
}
```

File: Publisher.java

```
package com.javatpoint;
import javax.xml.ws.Endpoint;
//Endpoint publisher
public class HelloWorldPublisher{
    public static void main(String[] args) {
        Endpoint.publish("http://localhost:7779/ws/hello", new HelloWorldImpl());
    }
}
```

How to view generated WSDL

After running the publisher code, you can see the generated WSDL file by visiting the URL:

http://localhost:7779/ws/hello?wsdl

JAX-WS Client Code

File: HelloWorldClient.java

```
package com.javatpoint;
import java.net.URL;
import javax.xml.namespace.QName;
import javax.xml.ws.Service;
public class HelloWorldClient{
    public static void main(String[] args) throws Exception {
    URL url = new URL("http://localhost:7779/ws/hello?wsdl");

        //1st argument service URI, refer to wsdl document above
    //2nd argument is service name, refer to wsdl document above
        QName qname = new QName("http://javatpoint.com/", "HelloWorldImplService");
        Service service = Service.create(url, qname);
        HelloWorld hello = service.getPort(HelloWorld.class);
```

```
        System.out.println(hello.getHelloWorldAsString("javatpoint rpc"));
    }
}
```

Output:

```
Hello World JAX-WS javatpoint rpc
```

13.

(i) **Explain** in detail about SOAP encoding.

- For transfer between client and server in a SOAP message, we encode them in XML.

SOAP Encoding is an extension of the SOAP framework specification that defines how a data value should be encoded in an XML format. SOAP Data Model is defined as an adjunct in SOAP 1.2 specification.

SOAP encoding offers the following rules to convert any data value defined in SOAP data model into XML format. Converting a data value into XML format is called serialization or encoding.

Rule 1. A simple value node with a labeled inbound edge will be serialized into a single XML element with the edge's label as the element's name and node value as the element's text content.

Rule 2. When serializing a node into an XML element, an "xsi:type" attribute can be added to specify the value type of this note. For more information on "xsi:type", see the other sections in this book.

Rule 3. A compound value node with labeled outbound edges, a data structure, will be serialized into a single XML element with child elements. One outbound edge will be serialized into one child element with element's name equal to the edge's label. The order of child elements is not significant.

Rule 4. A compound value node with non-labeled outbound edges, a data array, will be serialized into a single XML element with child elements. One outbound edge will be serialized into one child element with element's name equal to any label as long as it's the same for all child elements. The order of child elements signifies the position values of outbound edges.

Rule 5. When serializing an array, an "enc:itemType" attribute can be added to specify the value type of its sub nodes, and an "enc:arraySize" attribute can be added to specify the number of values in the array.

(ii) **Point out** the RPC representation model.

What Is JAX-RPC?

JAX-RPC stands for Java API for XML-based RPC. It's an API for building Web services and clients that used remote procedure calls (RPC) and XML. Often used in a distributed client/server model, an RPC mechanism enables clients to execute procedures on other systems.

	In JAX-RPC, a remote procedure call is represented by an XML-based protocol such as SOAP. The SOAP specification defines envelope structure, encoding rules, and a convention for representing remote procedure calls and responses. These calls and responses are transmitted as SOAP messages over HTTP. In this release, JAX-RPC relies on SOAP 1.1 and HTTP 1.1. Although JAX-RPC relies on complex protocols, the API hides this complexity from the application developer. On the server side, the developer specifies the remote procedures by defining methods in an interface written in the Java programming language. The developer also codes one or more classes that implement those methods. Client programs are also easy to code. A client creates a proxy, a local object representing the service, and then simply invokes methods on the proxy. With JAX-RPC, clients and Web services have a big advantage--the platform independence of the Java programming language. In addition, JAX-RPC is not restrictive: a JAX-RPC client can access a Web service that is not running on the Java platform and vice versa. This flexibility is possible because JAX-RPC uses technologies defined by the World Wide Web Consortium (W3C): HTTP, SOAP, and the Web Service Description Language (WSDL). WSDL specifies an XML format for describing a service as a set of endpoints operating on messages.
14.	**Explain** the structure of a WSDL document, its elements and their purposes with appropriate examples. A WSDL document defines **services** as collections of network endpoints, or **ports**. In WSDL, the abstract definition of endpoints and messages is separated from their concrete network deployment or data format bindings. This allows the reuse of abstract definitions: **messages**, which are abstract descriptions of the data being exchanged, and **port types** which are abstract collections of **operations**. The concrete protocol and data format specifications for a particular port type constitutes a reusable **binding**. A port is defined by associating a network address with a reusable binding, and a collection of ports define a service. Hence, a WSDL document uses the following elements in the definition of network services: • **Types**– a container for data type definitions using some type system (such as XSD). • **Message**– an abstract, typed definition of the data being communicated.

- **Operation**– an abstract description of an action supported by the service.
- **Port Type**–an abstract set of operations supported by one or more endpoints.
- **Binding**– a concrete protocol and data format specification for a particular port type.
- **Port**– a single endpoint defined as a combination of a binding and a network address.
- **Service**– a collection of related endpoints.

In addition, WSDL defines a common **binding** mechanism. This is used to attach a specific protocol or data format or structure to an abstract message, operation, or endpoint. It allows the reuse of abstract definitions.

WSDL Document Example

The following example shows the WSDL definition of a simple service providing stock quotes. The service supports a single operation called GetLastTradePrice, which is deployed using the SOAP 1.1 protocol over HTTP. The request takes a ticker symbol of type string, and returns the price as a float. A detailed description of the elements used in this definition can be found in Section 2 (core language) and Section 3 (SOAP binding).

This example uses a fixed XML format instead of the SOAP encoding (for an example using the SOAP encoding, see Example 4).

Example 1 SOAP 1.1 Request/Response via HTTP

```
<?xml version="1.0"?>
<definitions name="StockQuote"

targetNamespace="http://example.com/stockquote.wsdl"
        xmlns:tns="http://example.com/stockquote.wsdl"
        xmlns:xsd1="http://example.com/stockquote.xsd"
        xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
        xmlns="http://schemas.xmlsoap.org/wsdl/">

    <types>
      <schema targetNamespace="http://example.com/stockquote.xsd"
            xmlns="http://www.w3.org/2000/10/XMLSchema">
        <element name="TradePriceRequest">
          <complexType>
            <all>
              <element name="tickerSymbol" type="string"/>
            </all>
          </complexType>
        </element>
        <element name="TradePrice">
          <complexType>
            <all>
              <element name="price" type="float"/>
```

```
            </all>
          </complexType>
        </element>
      </schema>
    </types>

    <message name="GetLastTradePriceInput">
       <part name="body" element="xsd1:TradePriceRequest"/>
    </message>

    <message name="GetLastTradePriceOutput">
       <part name="body" element="xsd1:TradePrice"/>
    </message>

    <portType name="StockQuotePortType">
       <operation name="GetLastTradePrice">
          <input message="tns:GetLastTradePriceInput"/>
          <output message="tns:GetLastTradePriceOutput"/>
       </operation>
    </portType>

    <binding name="StockQuoteSoapBinding" type="tns:StockQuotePortType">
       <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
       <operation name="GetLastTradePrice">
          <soap:operation soapAction="http://example.com/GetLastTradePrice"/>
          <input>
             <soap:body use="literal"/>
          </input>
          <output>
             <soap:body use="literal"/>
          </output>
       </operation>
    </binding>

    <service name="StockQuoteService">
       <documentation>My first service</documentation>
       <port name="StockQuotePort" binding="tns:StockQuoteBinding">
          <soap:address location="http://example.com/stockquote"/>
       </port>
    </service>

 </definitions>
```

PART – C	
Q.No	**Questions**
1.	**Create** an XML HttpRequest to retrieve data from an XML file and display the data in an HTML table. The data to be retrieved is a collection of stationary items stored in an XML file. The XML Document Used INPUT: XML file called "cd_catalog.xml".

Display XML Data in an HTML Table

This example loops through each <CD> element, and displays the values of the <ARTIST> and the <TITLE> elements in an HTML table:

Example

```
<html>
<head>
<style>
table, th, td {
  border: 1px solid black;
  border-collapse:collapse;
}
th, td {
  padding: 5px;
}
</style>
</head>
<body>

<table id="demo"></table>

<script>
function loadXMLDoc() {
  var xmlhttp = new XMLHttpRequest();
  xmlhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      myFunction(this);
    }
  };
  xmlhttp.open("GET", "cd_catalog.xml", true);
  xmlhttp.send();
}
function myFunction(xml) {
  var i;
  var xmlDoc = xml.responseXML;
  var table="<tr><th>Artist</th><th>Title</th></tr>";
  var x = xmlDoc.getElementsByTagName("CD");
  for (i = 0; i <x.length; i++) {
    table += "<tr><td>" +
    x[i].getElementsByTagName("ARTIST")[0].childNodes[0].nodeValu
e +
    "</td><td>" +
    x[i].getElementsByTagName("TITLE")[0].childNodes[0].nodeValue
 +
    "</td></tr>";
```

```
  }
  document.getElementById("demo").innerHTML = table;
}
</script>

</body>
</html>
```

OUTPUT

Artist	Title
Bob Dylan	Empire Burlesque
Bonnie Tyler	Hide your heart
Dolly Parton	Greatest Hits
Gary Moore	Still got the blues
Eros Ramazzotti	Eros
Bee Gees	One night only
Dr.Hook	Sylvias Mother

2.

Summarize Ajax Client server architecture in detail.

What is AJAX?

AJAX = **A**synchronous **J**avaScript **A**nd **X**ML.

AJAX is not a programming language.

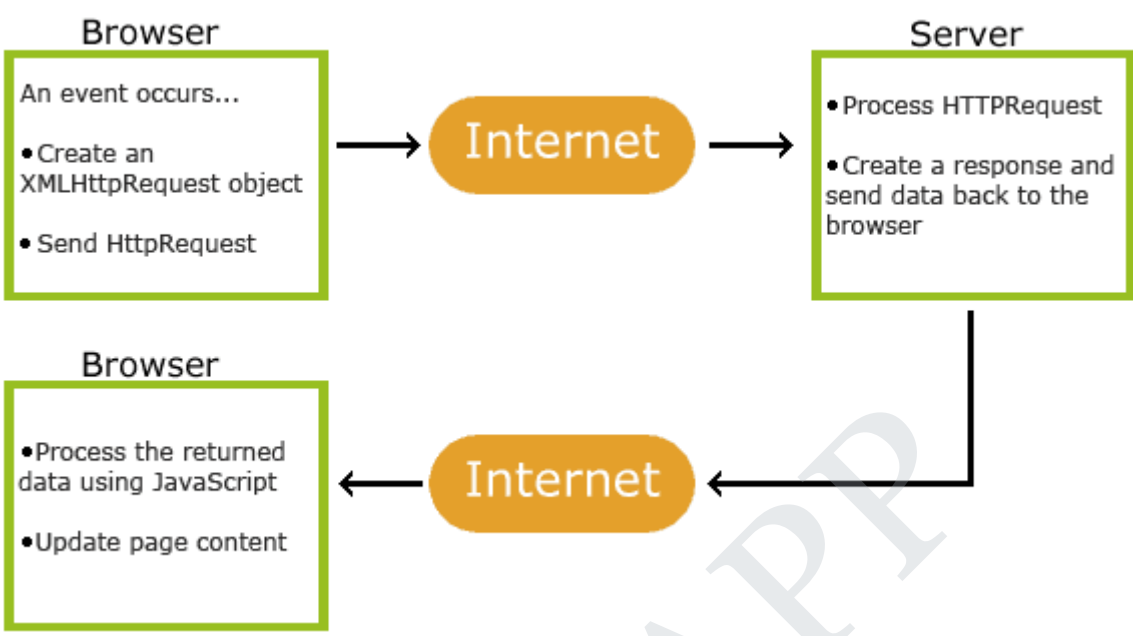
AJAX just uses a combination of:

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

AJAX is a misleading name. AJAX applications might use XML to transport data, but it is equally common to transport data as plain text or JSON text.

AJAX allows web pages to be updated asynchronously by exchanging data with a web server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.

How AJAX Works

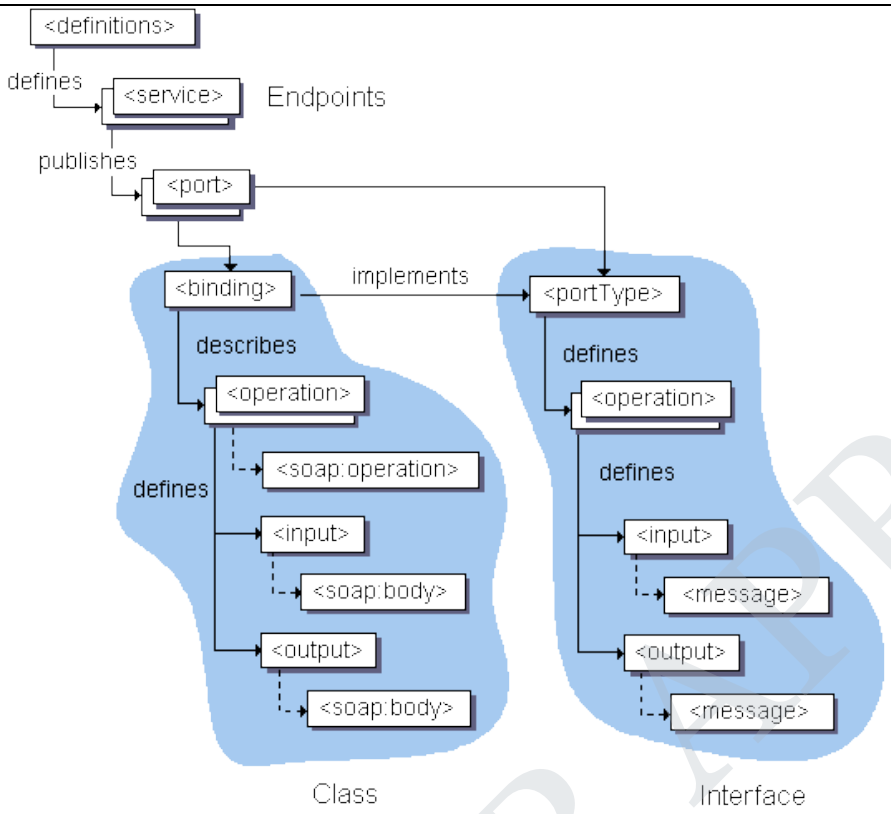


1. An event occurs in a web page (the page is loaded, a button is clicked)
2. An XMLHttpRequest object is created by JavaScript
3. The XMLHttpRequest object sends a request to a web server
4. The server processes the request
5. The server sends a response back to the web page
6. The response is read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

3. **Give** the basic structure of a WSDL and show how they are used to create, publish, test and describe web services.

Structure of a WSDL Document

Web Services Description Language (WSDL) is an XML grammar for describing network services as collections of communication endpoints capable of exchanging messages. The diagram below illustrates the elements that are present in a WSDL document, and indicates their relationships. To see an example of how this is implemented in a WSDL document, see Example of a WSDL Document .



WSDL Document Elements

A WSDL document has a definitions element that contains the other five elements, types, message, portType, binding and service. The following sections describe the features of the generated client code.

WSDL supports the XML Schemas specification (XSD) as its type system.

definitions

Contains the definition of one or more services. JDeveloper generates the following attribute declarations for this section:

- `name` is optional.
- `targetNamespace` is the logical namespace for information about this service. WSDL documents can import other WSDL documents, and setting targetNamespace to a unique value ensures that the namespaces do not clash.
- `xmlns` is the default namespace of the WSDL document, and it is set to `http://schemas.xmlsoap.org/wsdl/`.
- All the WSDL elements, such as `<definitions>`, `<types>` and `<message>` reside in this namespace.

	• `xmlns:xsd` and `xmlns:soap` are standard namespace definitions that are used for specifying SOAP-specific information as well as data types. • `xmlns:tns` stands for this namespace. • `xmlns:ns1` is set to the value of the `schema targetNamespace`, in the `<types>` section.
4.	**Compare** and **contrast** the additional web application architecture and AJAX Based web application architecture. **Traditional Web Applications vs. Ajax Applications** The following highlights the key differences between traditional web applications and Ajax-based 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 processed on the server, the user cannot interact with the client web page.

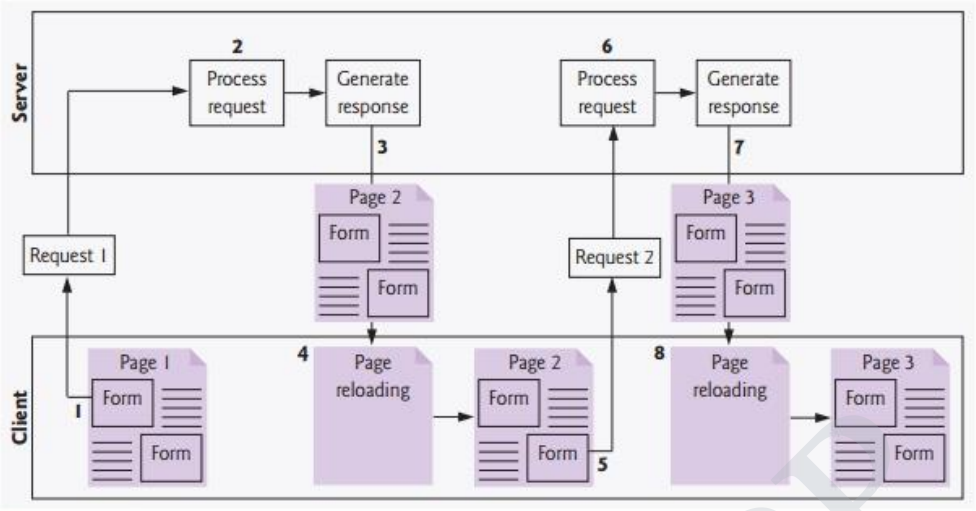


Fig. 15.1 | Classic web application reloading the page for every user interaction.

- Frequent long periods 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.

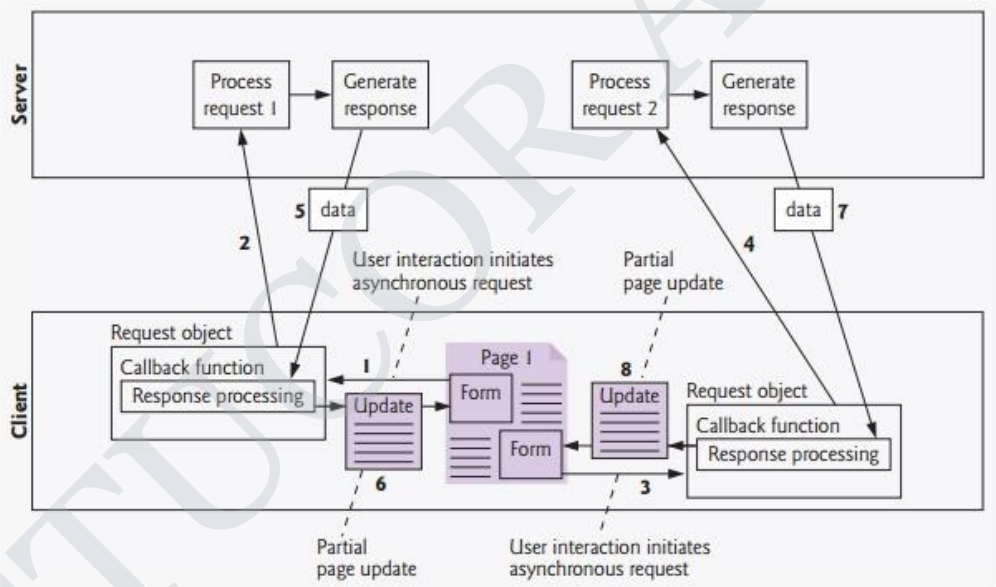


Fig. 15.2 | Ajax-enabled web application interacting with the server asynchronously.