UNIT-I

Syllabus: A Brief Introduction to Internet, The World Wide Web, Web Browsers, Web Servers, Uniform Resource Locators, MIME, HTTP. **HTML5:** Evolution of HTML and XHTML, Basic Syntax, Document Structure, Links, Images, Multimedia, Lists, Tables, Creating Forms, Cascading Style sheets.

Introduction to Internet

The Internet is a worldwide system of interconnected computer networks. The computers and computer networks exchange information using TCP/IP (Transmission Control Protocol/Internet Protocol) to communicate with each other. The computers are connected via the telecommunications networks, and the Internet can be used for e-mailing, transferring files and accessing information on the World Wide Web.

The World Wide Web is a system of Internet servers that use HTTP (Hypertext Transfer Protocol) to transfer documents formatted in HTML (Hypertext Mark-up Language). These are viewed by using software for web browsers such as Netscape, Safari, Google Chrome and Internet Explorer. Hypertext enables a document to be connected to other documents on the web through hyperlinks. It is possible to move from one document to another by using hyperlinked text found within web pages.

History of Internet

The history of the Internet begins with the development of electronic computers in the 1950s. Initial concepts of wide area networking originated in several computer science laboratories in the United States, United Kingdom, and France. The US Department of Defense awarded contracts as early as the 1960s, including for the development of the ARPANET project, directed by Robert Taylor and managed by Lawrence Roberts.

The first message was sent over the ARPANET in 1969 from computer science Professor Leonard Kleinrock's laboratory at the University of California, Los Angeles (UCLA) to the second network node at Stanford Research Institute (SRI).

Packet switching networks such as the NPL network, ARPANET, Tymnet, Merit Network, CYCLADES, and Telnet, were developed in the late 1960s and early 1970s using a variety of communications protocols. Donald Davies first demonstrated packet switching in 1967 at the National Physics Laboratory (NPL) in the UK, which became a test-bed for UK research for almost two decades.

The ARPANET project led to the development of protocols for internet-working, in which multiple separate networks could be joined into a network of networks. The Internet protocol suite (TCP/IP) was developed by Robert E. Kahn and Vint Cerf in the 1970s and became the standard networking protocol on the ARPANET. The ARPANET was decommissioned in 1990. In the 1980s, research at CERN in Switzerland by British computer scientist Tim Berners-Lee resulted in the World Wide Web, linking hypertext documents into an information system, accessible from any node on the network.

Since the mid-1990s, the Internet has had a revolutionary impact on culture, commerce, and technology, including the rise of near-instant communication by electronic mail, instant messaging, voice over Internet Protocol (VoIP) telephone calls, two-way interactive video calls, and the World Wide Web with its discussion forums, blogs, social networking, and online shopping sites.

The World Wide Web:

The World Wide Web (abbreviated WWW or the Web) is an information space where documents and other web resources are identified by Uniform Resource Locators (URLs), interlinked by hypertext links, and can be accessed via the Internet.The World Wide Web has been central to the development of the Information Age and is the primary tool billions of people use to interact on the Internet.[4][5][6]Web pages are primarily text documents formatted and annotated with Hypertext Markup Language (HTML). In addition to formatted text, web pages may contain images, video, audio, and software components that are rendered in the user's web browser as coherent pages of multimedia content.

Difference between the Internet and the World Wide Web :

The terms Internet and World Wide Web, although often used synonymously, are different. The term Internet is a nominalised abbreviation of Internetworking, and came into use in 1982. The Internet identifies not a single network, but a vast network of networks. These networks communicate with each other via the existing telecommunications networks. The Internet offers several different services including email and File Transfer Protocol (FTP). The World Wide Web, commonly known as "the Web," is the largest and fastest growing area of the Internet (Worsley, 2000). The Web uses the network of the Internet to access and link Web sites. The Internet essentially provides the infrastructure over which the Web is able to operate (Figure 1). The Web is a way of organizing information so that any workstation or computer around the world can access it through the Internet via any means of connectivity.

WEB BROWSER

Web Browser is an application software that allows us to view and explore information on the web. User can request for any web page by just entering a URL into address bar. Web browser can show text, audio, video, animation and more. It is the responsibility of a web browser to interpret text and commands contained in the web page.

Earlier the web browsers were text-based while now a days graphical-based or voice-based web browsers are also available. Following are the most common web browser available today:

Browser	Vendor
Internet Explorer/Microsoft Edge	Microsoft
Google Chrome	Google
Mozilla Firefox	Mozilla
Netscape Navigator	Netscape Communications Corp.
Opera	Opera Software
Safari	Apple

On a network, a web browser can retrieve a web page from a remote web server. The web server may restrict access to a private network such as a corporate intranet. The web browser uses the Hypertext Transfer Protocol (HTTP) to make such requests.

The browser does not display the HTML tags but uses them to determine how to display the document. Web browsers coordinate various web resource elements for the written web page, such as style sheets, scripts, and images, to present the web page.

Architecture

There are a lot of web browser available in the market. All of them interpret and display information on the screen however their capabilities and structure vary depending upon implementation. But the most basic component that all web browser must exhibit are listed below:

- Controller/Dispatcher
- Interpreter
- Client Programs

Controller works as a control unit in CPU. It takes input from the keyboard or mouse, interpret it and make other services to work based on input it receives.

Interpreter receives the information from the controller and execute the instruction line by line. Some interpreter are mandatory while some are optional For example, HTML interpreter program is mandatory and java interpreter is optional.

Client Program describes the specific protocol that will be used to access a service. Following are the client programs that are commonly used: HTTP, SMTP, FTP, NNTP, POP

WEB SERVERS

Web servers are computers that deliver (serves up) Web pages. Every Web server has an IP address and possibly a domain name. For example, if you enter the URL http://www.webopedia.com/index.html in your browser, this sends a request to the Web server whose domain name is webopedia.com. The server then fetches the page named index.html and sends it to your browser.

Any computer can be turned into a Web server by installing server software and connecting the machine to the Internet. There are many Web server software applications, including public domain software and commercial packages.

Web Server Working

Web server respond to the client request in either of the following two ways:

- Sending the file to the client associated with the requested URL.
- Generating response by invoking a script and communicating with database

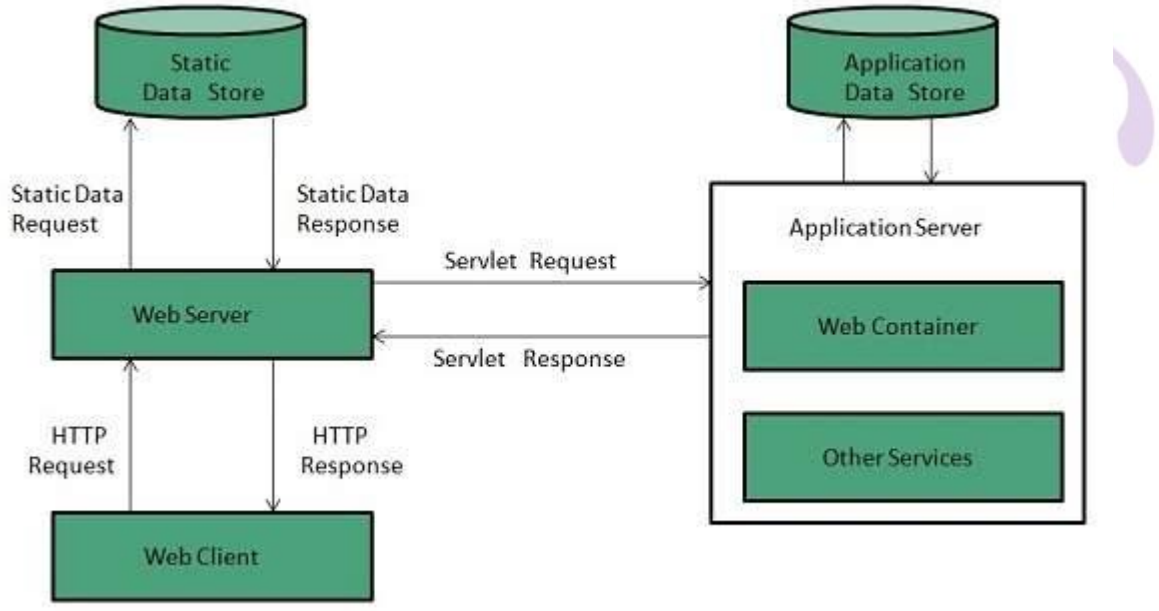


Fig: Architecture

Architecture

Web Server Architecture follows the following two approaches:

- Concurrent Approach
- Single-Process-Event-Driven Approach.

Concurrent approach allows the web server to handle multiple client requests at the same time. It can be achieved by following methods:

- Multi-process
- Multi-threaded
- Hybrid method.
- Multi-processing

In this a single process (parent process) initiates several single-threaded child processes and distribute incoming requests to these child processes. Each of the child processes are responsible for handling single request.

It is the responsibility of parent process to monitor the load and decide if processes should be killed or forked.

Multi-threaded: Unlike Multi-process, it creates multiple single-threaded process.

Hybrid: It is combination of above two approaches. In this approach multiple process are created and each process initiates multiple threads. Each of the threads handles one connection. Using multiple threads in single process results in less load on system resources.

Examples

Following table describes the most leading web servers available today:

1. Apache HTTP Server

This is the most popular web server in the world developed by the Apache Software Foundation. Apache web server is an open source software and can be installed on almost all operating systems including Linux, UNIX, Windows, FreeBSD, Mac OS X and more. About 60% of the web server machines run the Apache Web Server.

2. Internet Information Services (IIS)

The Internet Information Server (IIS) is a high-performance Web Server from Microsoft. This web server runs on Windows NT/2000 and 2003 platforms (and may be on upcoming new Windows version also). IIS comes bundled with Windows NT/2000 and 2003; Because IIS is tightly integrated with the operating system, so it is relatively easy to administer it.

3.Lighttpd

The lighttpd, pronounced lighty is also a free web server that is distributed with the FreeBSD operating system. This open source web server is fast, secure and consumes much less CPU power. Lighttpd can also run on Windows, Mac OS X, Linux and Solaris operating systems.

4.Sun Java System Web Server

This web server from Sun Microsystems is suited for medium and large web sites. Though the server is free it is not open source. It, however, runs on Windows, Linux and UNIX platforms. The Sun Java System web server supports various languages, scripts and technologies required for Web 2.0 such as JSP, Java Servlets, PHP, Perl, Python, and Ruby on Rails, ASP and ColdFusion etc.

5.Jigsaw Server

Jigsaw (W3C's Server) comes from the World Wide Web Consortium. It is open source and free and can run on various platforms like Linux, UNIX, Windows, and Mac OS X Free BSD etc. Jigsaw has been written in Java and can run CGI scripts and PHP programs.

UNIFORM RESOURCE LOCATORS:

URL is the abbreviation of Uniform Resource Locator and is defined as the global address of documents and other resources on the World Wide Web. The URL is an address that sends users to a specific resource online, such as a webpage, video or other document or resource. When you search Google, for example, the search results will display the URL of the resources that match your search query. The title in search results is simply a hyperlink to the URL of the resource.

A URL is one type of Uniform Resource Identifier (URI); the generic term for all types of names and addresses that refer to objects on the World Wide Web.

Parts of a URL:

A URL has two main components:

- Protocol identifier: For the URL http://example.com, the protocol identifier is http.
- Resource name: For the URL http://example.com, the resource name is example.com.

The protocol identifier indicates the name of the protocol to be used to fetch the resource. The example uses the Hypertext Transfer Protocol (HTTP), which is typically used to serve up hypertext documents. HTTP is just one of many different protocols used to access different types of resources on the net. Other protocols include File Transfer Protocol (FTP), Gopher, File, and News.

The resource name is the complete address to the resource. The format of the resource name depends entirely on the protocol used, but for many protocols, including HTTP, the resource name contains one or more of the following components:

Host Name The name of the machine on which the resource lives.

Filename The pathname to the file on the machine.

Port Number The port number to which to connect (typically optional).

Reference A reference to a named anchor within a resource that usually identifies a specific location within a file (typically optional).

For many protocols, the host name and the filename are required, while the port number and reference are optional. For example, the resource name for an HTTP URL must specify a server on the network (Host Name) and the path to the document on that machine (Filename); it also can specify a port number and a reference.

Every HTTP URL conforms to the syntax of a generic URI. The URI generic syntax consists of a hierarchical sequence of five components:

URI = scheme:[//authority]path[?query][#fragment]

where the authority component divides into three subcomponents:

authority = [userinfo@]host[:port]

MIME :

Multipurpose Internet Mail Extensions (MIME) is an Internet standard that extends the format of email to support:

- Text in character sets other than ASCII
- Non-text attachments: audio, video, images, application programs etc.
- Message bodies with multiple parts
- Header information in non-ASCII character sets

MIME type names follow a given format:

media-type/subtype-identifier

o image/jpeg is an example of a MIME type where image is the media type, and jpeg is the subtype identifier.

HTTP:

HTTP means Hyper Text Transfer Protocol. HTTP is the underlying protocol used by the World Wide Web and this protocol defines how messages are formatted and transmitted, and what actions Web servers and browsers should take in response to various commands. For example, when you enter a URL in your browser, this sends an HTTP command to the Web server directing it to fetch and transmit the requested Web page. The other main standard that controls how the World Wide Web works is HTML, which covers how Web pages are formatted and displayed.

HTTP is based on the client-server architecture model and a stateless request/response protocol that operates by exchanging messages across a reliable TCP/IP connection. Hypertext is structured text that uses logical links (hyperlinks) between nodes containing text. HTTP is the protocol to exchange or transfer hypertext.

An HTTP "client" is a program (Web browser or any other client) that establishes a connection to a server for the purpose of sending one or more HTTP request messages. An HTTP "server" is a program (generally a web server like Apache Web Server or Internet Information Services IIS, etc.) that accepts connections in order to serve HTTP requests by sending HTTP response messages.

HTTP makes use of the Uniform Resource Identifier (URI) to identify a given resource and to establish a connection. Once the connection is established, HTTP messages are passed in a format similar to that used by the Internet mail and the Multipurpose Internet Mail Extensions (MIME).

HTML

HTML is the World Wide Web's core markup language. Originally, HTML was primarily designed as a language for semantically describing scientific documents.

For its first five years (1990-1995), HTML went through several revisions and experienced a number of extensions, primarily hosted first at CERN, and then at the IETF. With the creation of the W3C, HTML's development changed venue again. A first abortive attempt at extending HTML in 1995 known as HTML 3.0 then made way to a more pragmatic approach known as HTML 3.2, which was completed in 1997. HTML4 quickly followed later that same year.

In 2006, the W3C indicated an interest to participate in the development of HTML5 after all, and in 2007 formed a working group chartered to work with the WHATWG on the development of the HTML5 specification.

New Features

HTML5 introduces a number of new elements and attributes that can help you in building modern websites. Here is a set of some of the most prominent features introduced in HTML5.

New Semantic Elements − These are like <header>, <footer>, and <section>.

Forms 2.0 − Improvements to HTML web forms where new attributes have been introduced for <input> tag.

Persistent Local Storage − To achieve without resorting to third-party plugins.

WebSocket − A next-generation bidirectional communication technology for web applications.

Server-Sent Events − HTML5 introduces events which flow from web server to the web browsers and they are called Server-Sent Events (SSE).

Canvas − This supports a two-dimensional drawing surface that you can program with JavaScript.

Audio & Video − You can embed audio or video on your webpages without resorting to third-party plugins.

Geolocation − Now visitors can choose to share their physical location with your web application.

Microdata − This lets you create your own vocabularies beyond HTML5 and extend your web pages with custom semantics.
Drag and drop − Drag and drop the items from one location to another location on the same webpage.
The HTML 5 language has a "custom" HTML syntax that is compatible with HTML 4 and XHTML1 documents published on the Web, but is not compatible with the more esoteric SGML features of HTML 4.HTML 5 does not have the same syntax rules as XHTML where we needed lower case tag names, quoting our attributes, an attribute had to have a value and to close all empty elements.
HTML5 comes with a lot of flexibility and it supports the following features −

- Uppercase tag names.
- Quotes are optional for attributes.
- Attribute values are optional.
- Closing empty elements are optional.

Introduction to HTML and HTML5

What is an HTML File?

- HTML stands for Hyper Text Markup Language
- An HTML file is a text file containing small markup tags
- The markup tags tell the Web browser how to display the page
- An HTML file must have an htm or html file extension
- An HTML file can be created using a simple text editor

To write and execute a simple static webpage, use the following procedure:

Step1: Open notepad application in your computer.
Step2: Type the following simple code and save the file with .html extension and double click on the created file. The file will be opened by the default browser of your computer.

Code:

```
<html>
<head>
<title>Title of page</title>
</head>
<body>
This is my first homepage. <b>This text is bold</b>
</body>
</html>
```

BASIC SYNTAX :
HTML elements are represented by tags
HTML tags label pieces of content such as "heading", "paragraph", "table", and so on
Browsers do not display the HTML tags, but use them to render the content of the page
Example

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

```
<head>
<title>Page Title</title>
</head>
<body>
<h1>My First Heading</h1>
<p>My first paragraph.</p>
</body>
</html>
```

DOCUMENT STRUCTURE

An HTML Document is mainly divided into two parts:

HEAD: This contains the information about the HTML document. For Example, Title of the page, version of HTML, Meta Data etc.
BODY: This contains everything you want to display on the Web Page.

Example Explained

The first tag in your HTML document is <html>. This tag tells your browser that this is the start of an HTML document. The last tag in the above code is </html>. This tag tells your browser that this is the end of the HTML document.

The text between the <head> tag and the </head> tag is header information. Header information is not displayed in the browser window. This section is used to provide meta data about the page. The text between the <title> tags is the title of your document. The title is displayed in your browser's caption.

The text between the <body> tags is the text that will be displayed in your browser. The text between the and tags will be displayed in a bold font. When you save an HTML file,use .html extension.

You can easily edit HTML files using a WYSIWYG (what you see is what you get) editor like notepad plus, sublime text, FrontPage editors. But if you want to be a skillful Web developer, we strongly recommend that you use a plain text editor like notepad to learn HTML in the earlier days of learning.

HTML Elements

HTML documents are text files made up of HTML elements.
HTML elements are defined using HTML tags.
HTML Tags

- HTML tags are used to mark-up HTML elements
- HTML tags are surrounded by the two characters < and >
- The surrounding characters are called angle brackets
- HTML tags normally come in pairs like and
- The first tag in a pair is the start tag, the second tag is the end tag
- The text between the start and end tags is the element content
- HTML tags are not case sensitive, means the same as <B>

Remember the HTML example from the previous page:

```
<html>
<head><title>Title of page</title></head>
<body> This is my first homepage. <b>This text is bold</b></body>
</html>
```

This is an HTML element: This text is bold

The HTML element starts with a start tag:
The content of the HTML element is: This text is bold
The HTML element ends with an end tag:
The purpose of the tag is to define an HTML element that should be displayed as bold.

This is also an HTML element:

```
<body>
This is my first homepage. <b>This text is bold</b>
</body>
```

This HTML element starts with the start tag <body>, and ends with the end tag </body>.
The purpose of the <body> tag is to define the HTML element that contains the body of the HTML document.

Tag Attributes

Tags can have attributes. Attributes can provide additional information about the HTML elements on your page.
This tag defines the body element of your HTML page: <body>. With an added bgcolor attribute, you can tell the browser that the background color of your page should be red, like this:

```
<body bgcolor="red">.
```

This tag defines an HTML table: <table>. With an added border attribute, you can tell the browser that the table should have no borders:

```
<table border="0">
```

Attributes always come in name/value pairs like this: name="value". Attributes are always added to the start tag of an HTML element. Quote Styles, "red" or 'red'?

Attribute values should always be enclosed in quotes. Double style quotes are the most common, but single style quotes are also allowed. In some rare situations, like when the attribute value itself contains quotes, it is necessary to use single quotes:
name='John "ShotGun" Nelson'

Basic HTML Tags

The most important tags in HTML are tags that define headings, paragraphs and line breaks.

Headings

Headings are defined with the <h1> to <h6> tags. <h1> defines the largest

heading. <h6> defines the smallest heading.

```
<h1>This is a heading</h1>
<h2>This is a heading</h2>
<h3>This is a heading</h3>
<h4>This is a heading</h4>
<h5>This is a heading</h5>
<h6>This is a heading</h6>
```

HTML automatically adds an extra blank line before and after a heading.

Paragraphs

Paragraphs are defined with the <p> tag.

```
<p>This is a paragraph</p>
<p>This is another paragraph</p>
```

HTML automatically adds an extra blank line before and after a paragraph.

Line Breaks

The
 tag is used when you want to end a line, but don't want to start a new paragraph. The
 tag forces a line break wherever you place it.

```
<p>This <br> is a para<br>graph with line breaks</p>
```

The
 tag is an empty tag. It has no closing tag.

Comments in HTML

The comment tag is used to insert a comment in the HTML source code. A comment will be ignored by the browser. You can use comments to explain your code, which can help you when you edit the source code at a later date.

```
<!-- This is a comment -->
```

Note that you need an exclamation point after the opening bracket, but not before the closing bracket.

Basic HTML Tags

<html>	Defines an HTML document
<body>	Defines the document's body
<h1> to <h6>	Defines header 1 to header 6
<p>	Defines a paragraph
 	Inserts a single line break
<hr>	Defines a horizontal rule/line
<!-->	Defines a comment

HTML Text Formatting

HTML defines a lot of elements for formatting output, like bold or italic text.

How to View HTML Source

Have you ever seen a Web page and wondered "How do they do that?". To find out, simply click on the VIEW option in your browsers toolbar and select SOURCE or PAGE SOURCE. This will open a window that shows you the actual HTML of the page.

Text Formatting Tags

<b>	Defines bold text
<big>	Defines big text
	Defines emphasized text
<i>	Defines italic text
<small>	Defines small text
	Defines strong text
<sub>	Defines subscripted text
<sup>	Defines superscripted text
<ins>	Defines inserted text
<del>	Defines deleted text
<strike>	Defines Strike through text
<u>	Defines Underlined text

HTML Character Entities

Some characters like the < character, have a special meaning in HTML, and therefore cannot be used in the text. To display a less than sign (<) in HTML, we have to use a character entity.

Character Entities

Some characters have a special meaning in HTML, like the less than sign (<) that defines the start of an HTML tag. If we want the browser to actually display these characters we must insert character entities in the HTML source.

A character entity has three parts: an ampersand (&), an entity name or a # and an entity number, and finally a semicolon (;).

To display a less than sign in an HTML document we must write: < or <

The advantage of using a name instead of a number is that a name is easier to remember. The disadvantage is that not all browsers support the newest entity names, while the support for entity numbers is very good in almost all browsers. Note that the entities are case sensitive.

Non-breaking Space

The most common character entity in HTML is the non-breaking space.

Normally HTML will truncate spaces in your text. If you write 10 spaces in your text HTML will remove 9 of them. To add spaces to your text, use the   character entity.

The Most Common Character Entities:

Result	Description	Entity Name	Entity Number
	non-breaking space		
<	Lessthan	&lt:	<
>	greater than	>	>
&	ampersand	&	&
“	Double Quote	&quot	"
‘	Single Quote	'	'

LINKS:

HTML Anchor Tag

You might know that **hyperlink** which is a powerful means of browsing from web pages and websites. It is developed for sending the readers or those who will perform surfing from one web page to another without opening a new tab or window. This is in general term said to as link and is

given a reference to jump to another page, document, or from one part of the same page to another using a hypertext.

Hypertext can be defined as a text shown on your PC screen, which holds the hyperlink data (here data means from which document to which document, it will move) and hence take the readers to different web pages by clicking it. In this chapter, you will be learning about how to create your very own hyperlink and use them in creating your website and web pages.

What is Anchor Tag?

The Anchor tag in HTML can be defined as a means to create a hyperlink that can link your current page on which the text is being converted to hypertext via <a> (anchor tag) to another page. This anchoring from one page to another is made possible by the attribute "href", which can be abbreviated as (hypertext reference).

HREF attribute

The attribute 'HREF' of the Anchor tag is implemented for defining the address or path to which this hypertext will get linked. In other words, it can be said that it directs you out of your page to that destination page, whose link you have mentioned within the double quotes of the href attribute as value.

The syntax for an anchor tag with href attribute looks something like this:

Text Here

Appearance of HTML <a> tag

Since Anchor tags are used for giving hyperlinks, you might have noticed that they change color based on certain situations, such as unvisited, clicked, visited, etc. Let's see their color code concerning their activity:

An unvisited link gets displayed with properties like underlined and the color is blue

A link which is visited gets displayed as underlined with color as purple

A link which is active link gets displayed as underlined with red color

Still, many web developers will deposit their link-colors in their web pages to match the color scheme of their site. Here's the format:

<body bgcolor="red" text="yellow" link="blue" alink = "#FFFF00" vlink ="#FF00FF">

You can also provide hex code for colors as well (as you can see from the above example), or else you can use specific words for each color, which is acceptable by the browser.

The Target Attribute

With the target attribute, you can define where the linked document will be opened.

The line below will open the document in a new browser window:

Visit W3Schools!

The Anchor Tag and the Name Attribute

The name attribute is used to create a named anchor. When using named anchors we can create links that can jump directly into a specific section on a page, instead of letting the user scroll around to find what he/she is looking for.

Below is the syntax of a named anchor:

Text to be displayed

The name attribute is used to create a named anchor. The name of the anchor can be any text you care to use.

The line below defines a named anchor:

Useful Tips Section

You should notice that a named anchor is not displayed in a special way.

To link directly to the "tips" section, add a # sign and the name of the anchor to the end of a URL, like this:

Jump to the Useful Tips Section

A hyperlink to the Useful Tips Section from WITHIN the file "html_links.asp" will look like this:

Jump to the Useful Tips Section

IMAGES

HTML Images

With HTML you can display images in a document.

The Image Tag and the Src Attribute

In HTML, images are defined with the <img> tag. The <img> tag is empty, which means that it contains attributes only and it has no closing tag.

To display an image on a page, you need to use the src attribute. Src stands for "source". The value of the src attribute is the URL of the image you want to display on your page.

The syntax of defining an image:

<img src="url">

The URL points to the location where the image is stored. An image named "boat.gif" located in the directory "images" on "www.w3schools.com" has the URL:

Example: http://www.w3schools.com/images/boat.gif.

The browser puts the image where the image tag occurs in the document. If you put an image tag between two paragraphs, the browser shows the first paragraph, then the image, and then the second paragraph.

The Alt Attribute

The alt attribute is used to define an "alternate text" for an image. The value of the alt attribute is an author-defined text:

<img src="boat.gif" alt="Big Boat">

The "alt" attribute tells the reader what he or she is missing on a page if the browser can't load images. The browser will then display the alternate text instead of the image. It is a good practice to include the "alt" attribute for each image on a page, to improve the display and usefulness of your document for people who have text-only browsers.

Image Tags

<img> Defines an image

<map> Defines an image map

<area> Defines an area inside an image map

MULTIMEDIA

What is Multimedia?

Multimedia is available in many different formats. It can be defined as almost anything which one can hear or see.

Examples: Pictures, sound, music, videos, records, films, animations, and more.

Web pages consists of multimedia elements belonging to different types and formats.

In this chapter you will be learning about the different multimedia formats.

BROWSER SUPPORT

The first web browsers was having support for text only, which was also limited to a single font and a single color.

Later browsers were developed to support for colors and fonts. They also supported for pictures!

The support for various multimedia like sounds, animations, and videos are handled in a different manner by different browsers. Different types and formats are supported, and some formats require extra helper programs (plug-ins) to work.

Hopefully, all the defects will become history. HTML5 multimedia is promising a bright future for multimedia.

MULTIMEDIA FORMATS

Multimedia elements (like sounds or videos) are getting stored as media files.

The most easy and common way to find or discover the type of a file, is to have a look at the extension of the file. When a browser locate the file extension .htm or .html, it will handle the file as an HTML file. The .xml extension is used for indicating an XML file, and the .css extension is used to indicate a style sheet file. Images are recognized using extensions like .gif, .png and .jpg.

Multimedia files also tend to have their own extensions, formats and like: .swf, .wav, .mp3, .mp4, .mpg, .wmv, and .avi.

HTML LISTS

HTML supports ordered, unordered and definition lists.

Unordered Lists

An unordered list is a list of items. The list items are marked with bullets (typically small black circles).

An unordered list starts with the <ul> tag. Each list item starts with the <li> tag.

```
<ul>
<li>Coffee</li>
<li>Milk</li>
</ul>
```

Here is how it looks in a browser:

- Coffee
- Milk

Inside a list item you can put paragraphs, line breaks, images, links, other lists, etc.

Ordered Lists

An ordered list is also a list of items. The list items are marked with numbers.

An ordered list starts with the <ol> tag. Each list item starts with the <li> tag.

```
<ol>
<li>Coffee</li>
```

```
<li>Milk</li>
</ol>
```

Here is how it looks in a browser:

1. Coffee
2. Milk

Inside a list item you can put paragraphs, line breaks, images, links, other lists, etc.

Definition Lists

A definition list is not a list of items. This is a list of terms and explanation of the terms.

A definition list starts with the <dl> tag. Each definition-list term starts with the <dt> tag. Each definition-list definition starts with the <dd> tag.

```
<dl>
<dt>Coffee</dt>
<dd>Black hot drink</dd>
<dt>Milk</dt>
<dd>White cold drink</dd>
</dl>
```

Here is how it looks in a browser:

Coffee
    Black hot drink
Milk
    White cold drink

Inside a definition-list definition (the <dd> tag) you can put paragraphs, line breaks, images, links, other lists, etc.

List Tags

	Defines an ordered list
	Defines an unordered list
	Defines a list item
<dl>	Defines a definition list
<dt>	Defines a definition term
<dd>	Defines a definition description
<dir>	Deprecated. Use instead
<menu>	Deprecated. Use instead

HTML TABLES

With HTML you can create tables.

Tables

Tables are defined with the <table> tag. A table is divided into rows (with the <tr> tag), and each row is divided into data cells (with the <td> tag). The letters td stands for "table data," which is the content of a data cell. A data cell can contain text, images, lists, paragraphs, forms, horizontal rules, tables, etc.

```
<table border="1">
```

```
        <tr>
            <td>row 1, cell 1</td>
            <td>row 1, cell 2</td>
        </tr>
        <tr>
            <td>row 2, cell 1</td>
            <td>row 2, cell 2</td>
        </tr>
    </table>
```

How it looks in a browser:

row 1, cell 1	row 1, cell 2
row 2, cell 1	row 2, cell 2

Tables and the Border Attribute

If you do not specify a border attribute the table will be displayed without any borders. Sometimes this can be useful, but most of the time, you want the borders to show.
To display a table with borders, you will have to use the border attribute:

```
<table border="1">
    <tr>
        <td>Row 1, cell 1</td>
        <td>Row 1, cell 2</td>
    </tr>
</table>
```

Headings in a Table

Headings in a table are defined with the <th> tag.

```
<table border="1">
    <tr>
        <th>Heading</th>
        <th>Another Heading</th>
    </tr>
    <tr>
        <td>row 1, cell 1</td>
        <td>row 1, cell 2</td>
    </tr>
    <tr>
        <td>row 2, cell 1</td>
        <td>row 2, cell 2</td>
    </tr>
</table>
```

How it looks in a browser:

Heading	**Another Heading**
row 1, cell 1	row 1, cell 2

row 2, cell 1	row 2, cell 2

Empty Cells in a Table

Table cells with no content are not displayed very well in most browsers.

```
<table border="1">
    <tr>
        <td>row 1, cell 1</td>
        <td>row 1, cell 2</td>
    </tr>
    <tr>
        <td>row 2, cell 1</td>
        <td></td>
    </tr>
</table>
```

How it looks in a browser:

row 1, cell 1	row 1, cell 2
row 2, cell 1	

Note that the borders around the empty table cell are missing.

To avoid this, add a non-breaking space ( ) to empty data cells, to make the borders visible:

```
<table border="1">
    <tr>
        <td>row 1, cell 1</td>
        <td>row 1, cell 2</td>
    </tr>
    <tr>
        <td>row 2, cell 1</td>
        <td> </td>
    </tr>
</table>
```

How it looks in a browser:

row 1, cell 1	row 1, cell 2
row 2, cell 1	

Table Tags

<table>	Defines a table
<th>	Defines a table header
<tr>	Defines a table row
<td>	Defines a table cell
<caption>	Defines a table caption
<colgroup>	Defines groups of table columns
<colspan>	Defines the number of columns to be merged in a row
<rowspan>	Defines the number of rows to be merged in a column
<col>	Defines the attribute values for one or more columns in a table

<thead>	Defines a table head
<tbody>	Defines a table body
<tfoot>	Defines a table footer

CREATING HTML FORMS

HTML Forms are used to select different kinds of user input.

Forms

A form is an area that can contain form elements. Form elements are elements that allow the user to enter information (like text fields, textarea fields, drop-down menus, radio buttons, checkboxes, etc.) in a form.

A form is defined with the <form> tag.

```
<form>
<input>
</form>
```

Input

The most used form tag is the <input> tag. The type of input is specified with the type attribute. The most commonly used input types are explained below.

Text Fields

Text fields are used when you want the user to type letters, numbers, etc. in a form.

```
<form>
First name: <input type="text" name="firstname">
<br>
Last name: <input type="text" name="lastname">
</form>
```

How it looks in a browser:

First name:

Last name:

Note that the form itself is not visible. Also note that in most browsers, the width of the text field is 20 characters by default.

Radio Buttons

Radio Buttons are used when you want the user to select one of a limited number of choices.

```
<form>
<input type="radio" name="sex" value="male"> Male
<br>
<input type="radio" name="sex" value="female"> Female
</form>
```

How it looks in a browser:

○ Male

○ Female

Note that radio button has to allow only one option to be chosen. To do so we have to keep the name attribute value should be same for all options.

Checkbox

Checkboxes are used when you want the user to select one or more options of a limited number of choices.

```
<form>
<input type="checkbox" name="bike">I have a bike
<br>
<input type="checkbox" name="car">I have a car
</form>
```

How it looks in a browser:

☐ I have a bike
☐ I have a car

The Form's Action Attribute and the Submit Button

When the user clicks on the "Submit" button, the content of the form is sent to another file. The form's action attribute defines the name of the file to send the content to. The file defined in the action attribute usually does something with the received input.

```
<form name="input" action="html_form_action.asp" method="get">
Username: <input type="text" name="user">
<input type="submit" value="Submit">
</form>
```

How it looks in a browser:

Username: [ ] Submit

If you type some characters in the text field above, and click the "Submit" button, you will send your input to a page called "html_form_action.asp". That page will show you the received input.

The <button> Element

The <button> element defines a clickable button:

```
<button type="button" onclick="alert('Hello World!')">Click Me!</button>
```

How it looks in a browser:

The button Element

Click Me!

Form Tags

<form>	Defines a form for user input
<input>	Defines an input field
<textarea>	Defines a text-area (a multi-line text input control)
<label>	Defines a label to a control
<fieldset>	Defines a fieldset
<legend>	Defines a caption for a fieldset
<select>	Defines a selectable list (a drop-down box)
<optgroup>	Defines an option group
<option>	Defines an option in the drop-down box
<button>	Defines a push button

<isindex> Deprecated. Use <input> instead

HTML Fonts

The <font> tag in HTML is deprecated. It is supposed to be removed in a future version of HTML. Even if a lot of people are using it, you should try to avoid it, and use styles instead.

The HTML <font> Tag

With HTML code like this, you can specify both the size and the type of the browser output :

```
<p><font size="2" face="Verdana">This is a paragraph.</font></p>
<p><font size="3" face="Times">This is another paragraph.</font></p>
```

The <font> Tag Should NOT be Used

The <font> tag is deprecated in the latest versions of HTML (HTML 4 and XHTML). It can be sone by using sylesheets.

CASCADING STYLE SHEET:

What is CSS? (Cascasding Style Sheet)

- CSS was developed in 1997 for Web designers, as a way for Web designers to define the attractive look and feel of their Web pages.
- CSS is used for design web pages.
- CSS works with HTML only.
- HTML is only Markup language, we can use some attributes and values in tags, but it's not capable of creating attractive web page design.
- The World Wide Web Consortium (W3C) has created CSS to solve this problem.
- In HTML 4.0, all formatting could be removed from the HTML document and can be stored in a separate CSS file.
- Today all web browsers are supporting CSS.

CSS Example

```
<!DOCTYPE html>
<html>
    <head><title>Sample CSS</title></head>
    <body>
        <style>
            h2 { font-size: 30px; line-height: 36px }
            p { padding: 10px 10px 10px 40px }
            p:hover { text-decoration: underline }
        </style>
        <h2>This is page heading.</h2>
```

```
        <p>This is my first <strong>paragraph text</strong>.</p>
    </body>
</html>
```

Introduction to CSS

Styling your web page to make it look appealing is also an essential element of web designing and development. CSS helps in giving styles for describing the presentation of the markup based documents.

CSS is abbreviated as Cascading Style Sheets and is used for describing how HTML elements need to be displayed when they are represented in a web page format or other media. It also helps in saving a lot of work because controlling the layout of multiple web pages can be done all at a time. It helps in representing how markup based documents can be presented in conjunction with HTML. The latest version of CSS released in CSS3.
CSS is said to as the cornerstone design tool of the World Wide Web along with HTML and JavaScript. CSS is intended for enabling the separation of appearance with content, which includes layout, coloring, and font styles. Such a smart approach can progress the accessibility of content, offer more flexibility and organize in the order of presentation where it allows multiple web pages to contribute to formatting by giving relevant CSS instructions in another file which has the file extension .css. This separation of design implementation helps in reducing complexity as well as repetition in the structural content.

Benifits of CSS

Compatible with Old Versions: This is the most beautiful feature of CSS3 as it gets compatible with its older versions. This is advantageous as designers won't have to relinquish their previous work because of the arrival of CSS3.

Simple yet Independent: Like that of CSS2, CSS3 is created using tiny modules that compose the application more straightforward and more comfortable to exercise. Some of the most vital components and features offered by CSS3 are selectors, box model, color, borders, backgrounds, 2D and 3D transformations, Text Effects, and user interface.

View and Change-Friendly: As the various CSS concepts have been broken down into tiny chunks or modules, it has become easier to alter the elements individually without particularly disturbing the remaining components. Also, CSS has become compatible with all kinds of platform.

Rapid development: Since it is not dependent on JavaScript, CSS3 has become a lot swifter than its older versions and is relatively more compatible with every browser available in the market.

Eye-catching backgrounds: CSS helps in making attractive backgrounds, especially using CSS3. It permits web designers to prefer from a wide range of backgrounds that can be useful and catch the

eye with ease. Also, these backgrounds have the feature of gradient colors and resizing so that they can fit for the project.

Images and Animations: CSS helps in ornamenting the appearance and feel of a website or web pages immeasurably by making use of some simple styles. It permits straightforward incorporation of a variety of images-including 3D images in your web development project. It also makes easy insertion of videos or animations in your web pages along with flexibility for customization.

Testing its features: Because of the smaller modules (specifically in CSS3), testing of every entity becomes smooth, which ultimately saves a lot of time and effort.

CSS Syntax

It is a standard set of rules which has three parts, a selector, a property, and a value. But, you do not have to recall this every time to code your web designing thing using CSS. Selectors will be discussed separately in the upcoming chapter. At the fundamental level of CSS, it has the two building blocks that define how CSS code will perform:

Selector: selectors in CSS help in selecting the styles in elements you want to put for designing your web pages.

Properties: These are understandable identifiers which point to different stylistic characteristics (such as font, background color, width, etc.) you wish for change.

Values: Every particular property is set with a value that tells in what way you wish to alter those stylistic characteristics within the web page (such as what font size you want, the width of any particular CSS entity, or the background color).

The property and value together as a single pair are called as CSS declaration. These declarations are placed inside CSS declaration Blocks.

Types of Style Sheets

There are three different ways you can use to insert CSS definitions in your web page. These are:

1. Inline Style
2. Embedded Style Sheet
3. External Style Sheet

Inline Style

Styles sheets that are of type inline refer to information related to the style being functional to the existing HTML element. The use of inline approach, rather than defining your style once, here, you have to write the style in every HTML element that you use for designing your web page. It can be more precisely called an inline style rather than the inline style sheet. It uses the style attribute within that HTML element.

Syntax:

<HTML ELEMENT style="properties: value"> .... </HTML ELEMENT>

Example:

The text gets the effect of inline style.

Drawbacks of Inline Styles

- Using inline styles is not as powerful as other types of style sheets available in CSS. If you define your styles for many HTML tags, which are in use to design a web page, then you will not be able to use most of the power associated with CSS.
- For example, you have to define the style over and over again whenever it is used, instead of defining it only once.
- Furthermore, if you want to change a specific style, you have to change it in many places in your documents instead of changing it to one place.

Embedded Style Sheets

Embedded Style Sheets is a style sheet where designers can embed information of the style sheet in an HTML document by making use of the <style> element. This embedding of style sheet info within <style> .... </style> tags are done within head section of HTML.
The syntax for embedded style sheets has no such exception. Simply you have to place the style sheet code between the <head>.....</head> tags where <style> .... </style> is nested within head element:
Example:

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Embedded Style Sheets Example</title>
<style>
    h1 {
        border-bottom: 1px solid #DDDDDD;
        color: #069;
        font-family: Helvetica, Arial;
        font-size: 25px;
```

```
        font-weight: normal;
        line-height: 34px;
        margin-bottom: 10px;
        outline: 0 none;
        padding-bottom: 3px;
        padding-top: 0;
        text-decoration: none;
}
</style></head>
```
```<body>
<h1>Welcome to w3schools.in</h1>
</body>
</html>
```

External Style sheets

This type of style sheet gets a separate file in which designers can state every CSS styles which seems relevant for your web site. Then this has to be linked with the external style sheet from your HTML page. You have to follow some specific steps to make this conceptual style sheet implementable.

Steps to create External Style Sheets:

1. Build the Style Sheet by typing the CSS code in a plain text file (using text editor, usually), and then save the with as a .css extension.
2. Now, you have to link the Style Sheet with the HTML document by using an HTML link element.

Example:

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

----------END OF UNIT – I----------

UNIT-II

JavaScript: Overview, Object Orientation and JavaScript, Syntactic Characteristics, Primitives, Operators,Expressions, Input and Output, Control Statements, Objects Creation and modification, Arrays, Functions,Constructors, Pattern Matching, Manipulating DOM, HTML DOM Events, Basics of AJAX with example.

OVERVIEW

JavaScript is a dynamic computer programming language. It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages. It is an interpreted programming language with object-oriented capabilities.

JavaScript was first known as **LiveScript,** but Netscape changed its name to JavaScript, possibly because of the excitement being generated by Java. JavaScript made its first appearance in Netscape 2.0 in 1995 with the name **LiveScript**. The general-purpose core of the language has been embedded in Netscape, Internet Explorer, and other web browsers.

Java Script:

- JavaScript was designed to add interactivity to HTML pages
- JavaScript is a scripting language
- A scripting language is a lightweight programming language
- JavaScript is usually embedded directly into HTML pages
- JavaScript is an interpreted language (means that scripts execute without preliminary compilation)
- Everyone can use JavaScript without purchasing a license

Note: Java and JavaScript are two completely different languages in both concept and design!

Advantages:

- JavaScript can interact with events - A JavaScript can be set to execute when something happens,
- like when a page has finished loading or when a user clicks on an HTML element
- JavaScript can read and write HTML elements - A JavaScript can read and change the content of an HTML element
- JavaScript can be used to validate data - A JavaScript can be used to validate form data before it is submitted to a server. This saves the server from extra processing
- JavaScript can be used to detect the visitor's browser - A JavaScript can be used to detect the visitor's browser, and - depending on the browser - load another page specifically designed for that browser
- JavaScript can be used to create cookies - A JavaScript can be used to store and retrieve information on the visitor's computer

Disadvantages

- Security Issues: Javascript snippets, once appended onto web pages execute on client servers immediately and therefore can also be used to exploit the user's system. While a certain

restriction is set by modern web standards on browsers, malicious code can still be executed complying with the restrictions set.

- Javascript rendering varies

The HTML <script> tag is used to insert a JavaScript into an HTML page.

```
<html>
<body>
<script type="text/javascript">
document.write("Hello World!");
</script>
</body>
 </html>
```

To insert a JavaScript into an HTML page, we use the <script> tag. Inside the <script> tag we use the type attribute to define the scripting language.

So, the <script type="text/javascript"> and </script> tells where the JavaScript starts and The document.write command is a standard JavaScript command for writing output to a page. By entering the document.write command between the <script> and </script> tags, the browser will recognize it as a JavaScript command and execute the code line. In this case the browser will write Hello World!.

JavaScript's in a page will be executed immediately while the page loads into the browser. This is not always what we want. Sometimes we want to execute a script when a page loads, other times when a user triggers an event.

Object Orientation and JavaScript

There are certain features or mechanisms which makes a Language Object Oriented like:

- **Object**
- **Classes**
- **Encapsulation**
- **Inheritance**

Object– An Object is a **unique** entity which contains **property** and **methods**. For example "car" is a real life Object, which have some characteristics like color, type, model, horsepower and performs certain action like drive. The characteristics of an Object are called as Property, in Object Oriented Programming and the actions are called methods. An Object is an **instance** of a class. Objects are everywhere in JavaScript almost every element is an Object whether it is a function,arrays and string.

Classes– Classes are **blueprint** of an Object. A class can have many Object, because class is a **template** while Object are **instances** of the class or the concrete implementation.

Before we move further into implementation, we should know unlike other Object Oriented Language there is **no classes in JavaScript** we have only Object. To be more precise, JavaScript is a prototype based object oriented language, which means it doesn't have classes rather it define behaviors using constructor function and then reuse it using the prototype.

Encapsulation – The process of **wrapping property and function** within a **single unit** is known as encapsulation.

Syntactic Characteristics

1.JavaScript are embedded either directly or indirectly in XHTML documents.

2.Scripts can appear directly as the content of a <script> tag.

3.The type attribute of <script> must be set to “text/JavaScript”.

4.The JavaScript can be indirectly embedded in an XHTML document using the srcattribute of a <script> tag, whose value is name of a file that contains the script.

Eg. <script type=”text/JavaScript” src=”tst_number.js”>

</script>

Closing tag is required even if script element has src attribute included.The indirect method of embedding JavaScript in XHTML has advantages of

1)Hiding the script from the browser user.

2)It also avoids the problem of hiding scripts from older browsers.

3)It is good to separate the computation provided by JavaScript from the layoutand presentation provided by XHTML and CSS respectively.

But it is sometimes not convenient and cumbersome to place all JavaScriptcode in separate file JavaScript identifiers or names are similar to programming languages.

1. Must begin with (-), or a letter. Subsequent characters may be letters, underscores or digits.
2. No length limitations for identifiers.
3. Case sensitive
4. No uppercase letters.

Reserved words of JavaScript are as follows:

abstract	else	instanceof	super
boolean	enum	int	switch
break	export	interface	synchronized
byte	extends	let	this
case	false	long	throw
catch	final	native	throws
char	finally	new	transient
class	float	null	true
const	for	package	try
continue	function	private	typeof
debugger	goto	protected	var
default	if	public	void
delete	implements	return	volatile
do	import	short	while
double	in	static	with

JavaScript primitives

Primitives are the simplest elements of a programming language. JavaScript currently has six **primitive types**: string, number, boolean, null, undefined, symbol , everything else is an object — yes, including arrays and dates.

All the types are used to define immutable values — values which can not be changed. We call values of these types **primitive values**.

A primitive type has a fixed size in memory. For example, a number occupies eight bytes of memory, and a boolean value can be represented with only one bit. The number type is the largest of the primitive types.

Variables hold the actual values of primitive types, but they hold only references to the values of reference types

Example

```
var a_num = 1.234 // a_num holds the actual value
var an_obj = { type: 'cat' } // an_obj holds a reference to the obj
```

Boolean type

- Allowed values: `true`,`false`.

Null type

- Allowed values: `null`.

It is worth noting that the typeof operator returns `object` for null [because the spec says so](#).

Undefined type

- Allowed values: `undefined`.

The Undefined type has exactly one value, called **undefined**. Any variable that has not been assigned a value has the value **undefined**.

Number type

- Size: 8 bytes
- Allowed values: 18437736874454810624 finite numbers (half positive half negative, including positive and negative zero) and three symbolic values NaN, Positive Infinity & Negative Infinity

String type

- Size: variable (string.length * 2 bytes)
- Max size: 18014398509481982 bytes (~16gB)
- Allowed values: 16-bit unsigned integer values (0 to 65535 inclusive)

JavaScript strings are immutable (unchangeable), once a string is created it is impossible to modify it. However, it is still possible to create another string based on the original by using string functions such as substr and concat.

Symbol type

Symbols are new to ES6 they are **unique** and **immutable** primitive values and may be used as the key of an Object property.

Example:

```
var key = Symbol()
var an_obj = {  [key]: 'cat'}
Symbol() === Symbol()     /// false
```

Auto-boxing of primitive types

When 'auto-boxing' JavaScript will automatically convert the primitive value to an object, it happens in two situations:

When accessing a property on a primitive, such as in the following example:

```
var part = 'cat'.substr(0,2)
```

2. When a primitive is passed as the this argument to .call or .apply

Example:

```
var part = String.prototype.slice.call('cat', 0, 2)
var partArr = Array.prototype.slice.apply('cat', [0, 2])
```

OPERATORS in JavaScript

JavaScript has the following types of operators. This section describes the operators and contains information about operator precedence.

- Assignment operators
- Comparison operators
- Arithmetic operators
- Bitwise operators
- Logical operators
- String operators
- Conditional (ternary) operator
- Comma operator
- Unary operators
- Relational operators

Expressions

Any unit of code that can be evaluated to a value is an expression. Since expressions produce values, they can appear anywhere in a program where JavaScript expects a value such as the arguments of a function invocation. As per the MDN documentation, JavaScript has the following expression categories.

Arithmetic Expressions:

Arithmetic expressions evaluate to a numeric value. Examples include the following

```
10;     // Here 10 is an expression that is evaluated to the numeric value 10 by the JS interpreter
10+13; // This is another expression that is evaluated to produce the numeric value 23
```

String Expressions:

String expressions are expressions that evaluate to a string. Examples include the following

```
'hello';
'hello' + 'world'; // evaluates to the string 'hello world'
```

Logical Expressions:

Expressions that evaluate to the boolean value true or false are considered to be logical expressions. This set of expressions often involve the usage of logical operators && (AND), ||(OR) and !(NOT). Examples include

```
10 > 9;   // evaluates to boolean value true
10 < 20;  // evaluates to boolean value false
true;     //evaluates to boolean value true
a===20 && b===30; // evaluates to true or false based on the values of a and b
```

Primary Expressions:

Primary expressions refer to stand alone expressions such as literal values, certain keywords and variable values. Examples include the following

```
'hello world'; // A string literal
23;            // A numeric literal
true;          // Boolean value true
sum;           // Value of variable sum
this;          // A keyword that evaluates to the current object
```

Left-hand-side Expressions:

Also known as lvalues, left-hand-side expressions are those that can appear on the left side of an assignment expression. Examples of left-hand-side expressions include the following

```
// variables such as i and total
i = 10;
total = 0;
```

User Input and Output in JavaScript

Working with any dynamic language requires the ability to read, process and output user data. JavaScript is especially useful when you want to take user information and process it without sending the data back to the server. JavaScript is much faster than sending everything to the server to process, but you must be able to read user input and use the right syntax to work with that input. This article will focus on retrieving user input and displaying it on the screen through HTML elements or prompts.

Displaying Prompts and Retrieving User Responses

JavaScript has a few window object methods that you can use to interact with your users. The prompt() method lets you open a client-side window and take input from a user. For instance, maybe you want the user to enter a first and last name. Normally, you can use an HTML form for user input, but you might need to get the information before sending the form back to your server for processing. You can use the window.prompt() method for small amounts of information. It's not meant for large blocks of text, but it's useful when you need information before the user continues to another page.

The following code is an example of the window.prompt method.

```
var customerName = prompt("Please enter your name", "<name goes here>");

if (customerName!= null) {

    document.getElementById("welcome").innerHTML =

    "Hello " + customerName + "! How are you today?";

}
```

The first line of code uses the prompt method. Notice that we don't need the "window" object since JavaScript inherits the main DOM methods and understands that this is an internal method. The first parameter in the prompt is what you want to show the user. In this example, we want the user to enter a name, so the prompt displays "Please enter your name." The second parameter is the default text. This default text helps the user understand where to type the name, but if he clicks "OK" without entering a name, the "<name goes here>" text will be used for the username. You can create checks in your JavaScript code that detects when the user doesn't enter a name and just clicks OK, but this code assumes any text entered is the user's name.

Notice the "if" statement that checks if the "customerName" variable is null. This logic checks to make sure that the user entered something. It only checks that some character was entered, so the user can type anything to bypass the "if" statement. The code within the "if" statement then displays the message to the user in the "welcome" div. We've covered reading and writing text to a web page, and this is another example of writing text to the inner HTML section of a div. Remember that the innerHTML property writes any tags or text within the opening and closing div tag.

Use this method to get a short string response from your users before they access a page or before they move on to another page in your site structure.

The JavaScript Confirmation Message Box

The window.prompt method is one way to read user input, but JavaScript also provides a way to get confirmation from the user. For instance, you might want to confirm that the user has entered the right information and wants to continue with payment. The confirmation window displays the amount the user will be charged, and the user has the option to confirm or cancel. You could write a complex JavaScript function to create a new window for confirmation or you can use the internal window.confirm method. This method returns either a boolean true result if the user clicks "OK" or the boolean false result if the user clicks "Cancel." This prompt is a quick way to get confirmation without using any complex coding logic.

Look at the following JavaScript code.

```
var r = confirm("Are you sure you want to send a payment?");

if (r == true) {

   x = "Payment sent!";

} else {

   x = "Payment cancelled!";

}

alert (x);
```

The main utility in the above code is the "confirm" function. This is an internal JavaScript function from the window object. In other words, using "window.confirm()" and "confirm" results in the same function. JavaScript handles the inheritance for you, so you don't need to remember to use the window object. The confirmation window is also pre-built and shown to the user. In this example, a prompt displays with the text "Are you sure you want to send a payment?" The only options for the user are to click the Cancel button or the OK button. OK is the confirmation that returns "true." If the user clicks "OK," the variable x contains the text "Payment sent!" Conversely, if the user clicks the "Cancel" button, x contains the text "Payment cancelled!" The text is then displayed to the user using the "alert" function. This is the first time we've seen the alert function, which is also a part of the window object. Typing "alert" and "window.alert" results in the same method call. The alert prompt is common during debugging and development of a web application, because it's a quick way to check your logic and see the control flow of your code. The alert function in this example checks that the confirm window is responding with the right result and the x variable contains the right text.

Displaying Text from User Input

There are three main HTML tags used to display text after you prompt users. You've seen two internal JavaScript functions that get user input, but how do you display it back to the user? The three tags used are the div, span and text tags. The third one, the text tag, is a form field used to take string input from the user. You usually use this tag to get input such as a name or an address, but you can also display input from the user.

The span and div tags are a way to just display the information you've read in JavaScript prompt windows. You can also display data you've retrieved from a database.

Let's use the previous example where we prompted the user to enter a name. We then use the input to display output in a div element. We've seen this before, but this is the first time we've used input directly from the user.

```
<!DOCTYPE html>
<html>

<head>

<script>

var customerName = prompt("Please enter your name", "");
if (customerName!= null) {
   document.getElementById("welcome").innerHTML =
   "Hello " + customerName + "! How are you today?";
}



</script>

</head>

<body>

<div id="welcome">My First JavaScript code.</div>

</body>
</html>
```

We've included the HTML this time to show you how JavaScript prompts work with HTML elements. The element we're using is the div HTML tag with the id of "welcome." The JavaScript prompt asks the user for his name and then displays the result in the "welcome" div. You can also use this same code to display the result in the span element using the same code except change the div tag to a span opening and closing tag.

You've seen the innerHTML property several times, which controls the text within the opening and closing div or span tags, but what if you want to use the input to pre-populate a form field? The input text field lets users input string values, but you can use the methods we've seen to pre-populate string values automatically. Using the same input example as we previously used, the following code uses the input prompt to enter the user's name in the "username" text field.

```
<!DOCTYPE html>
<html>

<head>

<script>

var customerName = prompt("Please enter your name", "");
if (customerName!= null) {
   document.getElementById("username").value = customerName;
}



</script>

</head>
```

Interested in learning more? Why not take an online class in JavaScript?

```
<body>

<input type="text" id="username" />

</body>
</html>
```


The above code varies slightly from the previous div example. This code uses the same prompt method to get a user's name, but then it displays the information in an "input" element with the type set as "text." Again, you'll notice that the id tag property is used, which allows us to grab the DOM element by id and change its properties. The form input element uses the "value" property and not the innerHTML property. You'll get to know the DOM element properties by heart as you use them more frequently. Also notice that the input HTML tag does not have a closing tag. The terminator "/>" can be used when you don't need to contain text within an opening and closing tag. Using the terminator is cleaner and follows HTML5 standards.

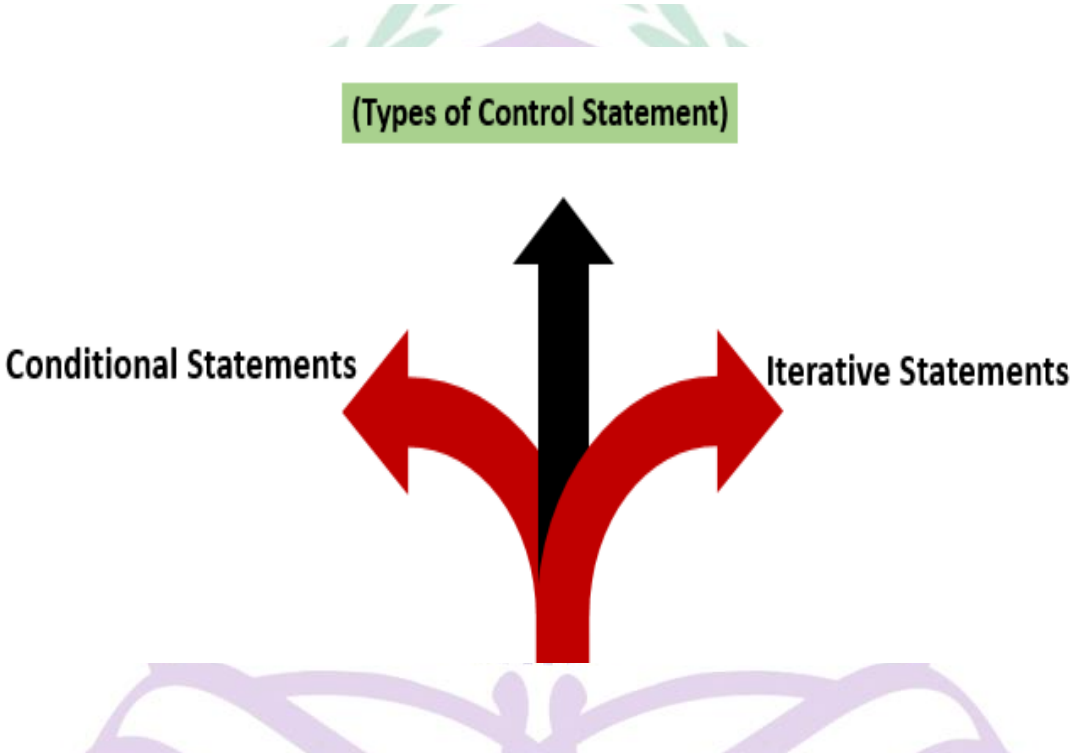
Control statements in JavaScript

While writing a program, there may be a situation when you need to adopt one out of a given set of paths. In such cases, you need to use conditional statements that allow your program to make correct decisions and perform right actions.

JavaScript supports control statements which are used to perform different actions based on different conditions

Type of Control Statement in JavaScript

Every programming language, basically, has two types of control statements as follows:



- **Conditional Statements:** based on an expression passed, a conditional statement makes a decision, which results in either YES or NO.
- **Iterative Statements (Loop):** Until and unless the expression or the condition given is satisfied, these statements repeat themselves.

Conditional Statements

- **IF**

when you want to check for a specific condition. With the IF condition, the inner code block is executed if the condition provided is satisfied.

Syntax:

```
if (condition) {
//code block to be executed if condition is satisfied
}
```

IF-ELSE

an extended version of IF. When you want to check a specific condition and two

Syntax:

```
if (condition)
{
// code to be executed of condition is true
}
else {
// code to be executed of condition is false
}
```

SWITCH

A switch statement is similar to IF and is of use when you need to execute one code out of the multiple code block execution possibilities, based on the result of the expression passed. Switch statements carry an expression, which is compared with values of the following cases and once a match is found, code associated with that case executes.

Syntax:

```
switch (expression) {

case a:

//code block to be executed

Break;

case b:

//code block to be executed

Break;

case n:

//code block to be executed

Break;

default:

//default code to be executed if none of the above case is executed

}
```

Iterative Statement

WHILE

one of the control flow statement, which executes a code block when the condition is satisfied. But unlike IF, while keeps repeating itself until the condition is satisfied. Difference between IF and while can be, IF executes code ‘if’ the condition is satisfied while the while keeps repeating itself until the condition is satisfied.

Syntax:

```
while (condition)

{

//code block to be executed when condition is satisfied

}
```

DO-WHILE

Similar to a while loop, with a twist that keeps a condition at the end of the loop. Also known as Exit Control Loop, DO-WHILE executes the code and then checks for the condition.

Syntax:

while

{

//code block to be executed when condition is satisfied

} (condition)

If the condition at the end is satisfied, the loop will repeat.

FOR

a for loop will execute a code block for a number of times. Compared to other loops, FOR is shorter and easy to debug as it contains initialization, condition and increment or decrement in a single line.

Syntax:

for (initialize; condition; increment/decrement)

{

//code block to be executed

}

Objects Creation and modification

A javaScript object is an entity having state and behavior (properties and method). For example: car, pen, bike, chair, glass, keyboard, monitor etc.

JavaScript is an object-based language. Everything is an object in JavaScript.

JavaScript is template based not class based. Here, we don't create class to get the object. But, we direct create objects.

next →← prev

JavaScript Objects

A javaScript object is an entity having state and behavior (properties and method). For example: car, pen, bike, chair, glass, keyboard, monitor etc.

JavaScript is an object-based language. Everything is an object in JavaScript.

JavaScript is template based not class based. Here, we don't create class to get the object. But, we direct create objects.

Creating Objects in JavaScript

There are 3 ways to create objects.

1. By object literal
2. By creating instance of Object directly (using new keyword)
3. By using an object constructor (using new keyword)

1) JavaScript Object by object literal

The syntax of creating object using object literal is given below:

Syntax:

```
object={property1:value1,property2:value2.....propertyN:valueN}
```

example:

```
<script>
emp={id:102,name:"Shyam Kumar",salary:40000}
document.write(emp.id+" "+emp.name+" "+emp.salary);
</script>
```

2) By creating instance of Object

The syntax of creating object directly is given below:

Syntax:

```
var objectname=new Object();
```

example:

```
<script>
var emp=new Object();
emp.id=101;
emp.name="Ravi Malik";
emp.salary=50000;
document.write(emp.id+" "+emp.name+" "+emp.salary);
</script>
```

3) By using an Object constructor

Here, you need to create function with arguments. Each argument value can be assigned in the current object by using this keyword.

Example:

```
<script>
function emp(id,name,salary){
this.id=id;
this.name=name;
this.salary=salary;
}
e=new emp(103,"Vimal Jaiswal",30000);

document.write(e.id+" "+e.name+" "+e.salary);
</script>
```

The various methods of Object are as follows:

S.No	Methods	Description
1	Object.assign()	This method is used to copy enumerable and own properties from a source object to a target object
2	Object.create()	This method is used to create a new object with the specified prototype object and properties.
3	Object.defineProperty()	This method is used to describe some behavioral attributes of the property.
4	Object.defineProperties()	This method is used to create or configure multiple object properties.
5	Object.entries()	This method returns an array with arrays of the key, value pairs.
6	Object.freeze()	This method prevents existing properties from being removed.
7	Object.getOwnPropertyDescriptor()	This method returns a property descriptor for the specified property of

		the specified object.
8	Object.getOwnPropertyDescriptors()	This method returns all own property descriptors of a given object.
9	Object.getOwnPropertyNames()	This method returns an array of all properties (enumerable or not) found.
10	Object.getOwnPropertySymbols()	This method returns an array of all own symbol key properties.
11	Object.getPrototypeOf()	This method returns the prototype of the specified object.
12	Object.is()	This method determines whether two values are the same value.
13	Object.isExtensible()	This method determines if an object is extensible
14	Object.isFrozen()	This method determines if an object was frozen.
15	Object.isSealed()	This method determines if an object is sealed.
16	Object.keys()	This method returns an array of a given object's own property names.
17	Object.preventExtensions()	This method is used to prevent any extensions of an object.
18	Object.seal()	This method prevents new properties from being added and marks all existing properties as non-configurable.
19	Object.setPrototypeOf()	This method sets the prototype of a specified object to another object.
20	Object.values()	This method returns an array of values.

JavaScript Array's

JavaScript array is an object that represents a collection of similar type of elements.

There are 3 ways to construct array in JavaScript

1. By array literal
2. By creating instance of Array directly (using new keyword)
3. By using an Array constructor (using new keyword)

1) JavaScript array literal

The syntax of creating array using array literal is given below:

Syntax:

```
var arrayname=[value1,value2.....valueN];
```

example:

```
<script>
var emp=["Sonoo","Vimal","Ratan"];
for (i=0;i<emp.length;i++){
document.write(emp[i] + "<br/>");
}
</script>
```

2) JavaScript Array directly (new keyword)

The syntax of creating array directly is given below:

Syntax:

```
var arrayname=new Array();
```

example:

```
<script>
var i;
var emp = new Array();
emp[0] = "Arun";
emp[1] = "Varun";
emp[2] = "John";

for (i=0;i<emp.length;i++){
document.write(emp[i] + "<br>");
}
</script>
```

3) JavaScript array constructor (new keyword)

Here, you need to create instance of array by passing arguments in constructor so that we don't have to provide value explicitly

```
<script>
var emp=new Array("Jai","Vijay","Smith");
for (i=0;i<emp.length;i++){
document.write(emp[i] + "<br>");
}
</script>
```

List of JavaScript array methods with their description.

Methods	Description
concat()	It returns a new array object that contains two or more merged arrays.
copywithin()	It copies the part of the given array with its own elements and returns the modified array.
entries()	It creates an iterator object and a loop that iterates over each key/value pair.
every()	It determines whether all the elements of an array are satisfying the provided function conditions.
flat()	It creates a new array carrying sub-array elements concatenated recursively till the specified depth.
flatMap()	It maps all array elements via mapping function, then flattens the result into a new array.
fill()	It fills elements into an array with static values.
from()	It creates a new array carrying the exact copy of another array element.
filter()	It returns the new array containing the elements that pass the provided function conditions.
find()	It returns the value of the first element in the given array that satisfies the specified condition.

JavaScript Functions

JavaScript functions are used to perform operations. We can call JavaScript function many times to reuse the code.

Advantage of JavaScript function

There are mainly two advantages of JavaScript functions.

1. **Code reusability**: We can call a function several times so it save coding.
2. **Less coding**: It makes our program compact. We don't need to write many lines of code each time to perform a common task.

Syntax:

```
function functionName([arg1, arg2, ...argN]){
 //code to be executed
}
```

Example:

```
<html>
<body>
<script>
function msg(){
alert("hello! this is message");
}
</script>
<input type="button" onclick="msg()" value="call function"/>
</body>
</html>
```

JavaScript Constructor Method

A JavaScript constructor method is a special type of method which is used to initialize and create an object. It is called when memory is allocated for an object.

Points to remember

- The constructor keyword is used to declare a constructor method.
- The class can contain one constructor method only.
- JavaScript allows us to use parent class constructor through super keyword.

Example

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

```
<script>
class Employee {
  constructor() {
    this.id=101;
    this.name = "Martin Roy";
  }
}
var emp = new Employee();
document.writeln(emp.id+" "+emp.name);
</script>

</body>
</html>
```

Pattern matching in javascript

The **JavaScript string** is an object that represents a sequence of characters.

There are 2 ways to create string in JavaScript

1. By string literal
2. By string object (using new keyword)

Example:

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

<p>Click the button to perfom a global, case-insensitive search for the letters "ain" in a string, and display the matches.</p>

<button onclick="myFunction()">Try it</button>

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

<script>
function myFunction() {
  var str = "The rain in SPAIN stays mainly in the plain";
  var res = str.match(/ain/gi);
  document.getElementById("demo").innerHTML = res;
}
</script>

</body>
</html>
```

List of JavaScript string methods with examples.

Methods	Description
charAt()	It provides the char value present at the specified index.
charCodeAt()	It provides the Unicode value of a character present at the specified index.
concat()	It provides a combination of two or more strings.
indexOf()	It provides the position of a char value present in the given string.
lastIndexOf()	It provides the position of a char value present in the given string by searching a character from the last position.
search()	It searches a specified regular expression in a given string and returns its position if a match occurs.
match()	It searches a specified regular expression in a given string and returns that regular expression if a match occurs.
replace()	It replaces a given string with the specified replacement.

Manipulating DOM using JavaScript

The Document Object Model (DOM) is a programming interface for HTML and XML documents. It represents the page so that programs can change the document structure, style, and content. The DOM represents the document as nodes and objects. That way, programming languages can connect to the page.

The DOM is not a programming language, but without it, the JavaScript language wouldn't have any model or notion of web pages, HTML documents, XML documents, and their component parts (e.g. elements). Every element in a document—the document as a whole, the head, tables within the document, table headers, text within the table cells—is part of the document object model for that document, so they can all be accessed and manipulated using the DOM and a scripting language like JavaScript.

Example:

```
<html>
  <head>
    <script>
      // run this function when the document is loaded
```

```
    window.onload = function() {

      // create a couple of elements in an otherwise empty HTML page
      const heading = document.createElement("h1");
      const heading_text = document.createTextNode("Big Head!");
      heading.appendChild(heading_text);
      document.body.appendChild(heading);
    }
  </script>
 </head>
 <body>
 </body>
</html>
```

HTML DOM Events

HTML DOM events allow JavaScript to register different event handlers on elements in an HTML document.

Events are normally used in combination with functions, and the function will not be executed before the event occurs, such as when a user clicks a button.

Event	Description	Belongs To
abort	The event occurs when the loading of a media is aborted	UiEvent, Event
afterprint	The event occurs when a page has started printing, or if the print dialogue box has been closed	Event
animationend	The event occurs when a CSS animation has completed	AnimationEvent
animationiteration	The event occurs when a CSS animation is repeated	AnimationEvent
animationstart	The event occurs when a CSS	AnimationEvent

	animation has started	
beforeprint	The event occurs when a page is about to be printed	Event
beforeunload	The event occurs before the document is about to be unloaded	UiEvent, Event
blur	The event occurs when an element loses focus	FocusEvent
canplay	The event occurs when the browser can start playing the media (when it has buffered enough to begin)	Event
canplaythrough	The event occurs when the browser can play through the media without stopping for buffering	Event
change	The event occurs when the content of a form element, the selection, or the checked state have changed (for <input>, <select>, and <textarea>)	Event
click	The event occurs when the user clicks on an element	MouseEvent
contextmenu	The event occurs when the user right-clicks on an element to open a context menu	MouseEvent
copy	The event occurs when the user	ClipboardEvent

	copies the content of an element	
cut	The event occurs when the user cuts the content of an element	ClipboardEvent
dblclick	The event occurs when the user double-clicks on an element	MouseEvent

Introduction to AJAX (Asynchronous JavaScript and XML)

With the growth of Web applications, websites have been trying to make user interaction similar to that found in

desktop applications. The largest push is towards making Web interfaces more dynamic: if data changes, say a new email arrives in your inbox in GMAIL, the Web page should update and display the new email without reloading the whole page. The page should always be available to the user and always responsive to their input. While the DOM allows for such dynamic updates to occur, AJAX is an important group of technologies that allows a Web page to request more information from the server (such as an updated list of email in an inbox) without having to reload the whole page. Together, AJAX and the DOM can be used to create dynamic Web pages.

AJAX stands for Asynchronous JavaScript and XML. It allows a Web page to make a request to a Web server for information using standard HTTP, but without reloading the page, and without automatically displaying the information returned from the server. These requests are all made programmatically, using JavaScript, and data communication is often done using XML, as JavaScript can easily parse this data. After receiving the data from the server, the JavaScript script can use the returned data however it wishes. The requests can also be made in a such a way that the JavaScript code does not have to wait for a reply from the server. Instead, the JavaScript code is notified when the page has finished receiving the information. In this way, the script can continue to perform useful actions while the data downloads from the server – this makes the communication asynchronous to any action that the Web page is performing.

It is important to realise that AJAX itself refers to a group of related technologies, not all of which are standardised, and not all of which need to be used at once. For instance, it can often be more convenient to communicate with the server using plain text rather than XML, and these communications need not occur asynchronously. Various other technologies are often employed in addition to those making up the AJAX name itself. The original article that defined the term AJAX lists the following technologies:

• XHTML and CSS, which defines what is being displayed and how it is displayed.

• The Document Object Model, which allows us to programmatically alter the content and how it is displayed.

• XML and XSLT, which is used to transfer data between the server and Web browser (using XML), and to manipulate that data (using XSLT).

• XMLHttpRequest, which is the object used to communicate with the Web server over HTTP. This object provides the asynchronous communication abilities.

• JavaScript, which is the programming language implemented in most Web browsers, and is used to bring together all the other technologies listed above.

Initialization

The first step to making an HTTP request using AJAX is to create an instance of one of the XMLHttpFunction function objects. There are two different objects that can be used, depending on the browser: Mozilla based browsers – such as Firefox – and Safari (used on OS X) both use XMLHttpRequest() objects. Internet Explorer, on the other hand, uses ActiveXObject(). An object can easily be created using the following code:

```
var httpRequest;
if (window.XMLHttpRequest) { // Mozilla, Safari, …
 httpRequest = new XMLHttpRequest();
} else if (window.ActiveXObject) { // IE
httpRequest = new ActiveXObject("Microsoft.XMLHTTP");
}
```

The methods and properties associated with both of these objects are identical: the major difference between them is how they are initialised. During these notes we will speak of XMLHttpRequest to mean both the Mozilla / Safari XMLHttpRequest objects

---------- END OF UNIT-II ----------.

UNIT-III

XML: Introduction to XML, Syntax, XML document structure, Document Type Definition, Name spaces, XML Schemas, Display in raw XML documents, Displaying XML documents with CSS, XPath Basics, XSLT, XML Processors.
J2EE: Exploring Enterprise architecture styles, Features of EE platform, Web servers and application servers.**Database programming with JDBC:** JDBC Drivers, Exploring JDBC Processes with the java.sql Package

Introduction to XML

XML is a software- and hardware-independent tool for storing and transporting data.

What is XML?

XML stands for extensible Markup Language
XML is a markup language much like HTML
XML was designed to store and transport data
XML was designed to be self-descriptive
XML is a W3C Recommendation
XML Does Not DO Anything

Maybe it is a little hard to understand, but XML does not DO anything.

Example:

This note is a note to Tove from Jani, stored as XML:

```
<note>
    <to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body>Don't forget me this weekend!</body>
</note>
```

The XML above is quite self-descriptive:

- It has sender information.
- It has receiver information.
- It has a heading.
- It has a message body.

But still, the XML above does not DO anything. XML is just information wrapped in tags.

The Difference Between XML and HTML

XML and HTML were designed with different goals:

- XML was designed to carry data - with focus on what data is
- HTML was designed to display data - with focus on how data looks
- XML tags are not predefined like HTML tags are
- XML Does Not Use Predefined Tags.

The XML language has no predefined tags.

The tags in the example above (like <to> and <from>) are not defined in any XMLstandard. These tags are "invented" by the author of the XML document.

HTML works with predefined tags like <p>, <h1>, <table>, etc.

With XML, the author must define both the tags and the document structure.

XML is Extensible.
Most XML applications will work as expected even if new data is added (or removed).

Imagine an application designed to display the original version of note.xml (<to> <from><heading><body>).
Then imagine a newer version of note.xml with added <date> and <hour> elements,and a removed <heading>.
The way XML is constructed, older version of the application can still work:

```
<note>
<date>2015-09-01</date>
<hour>08:30</hour>
<to>Tove</to>
<from>Jani</from>
<body>Don't forget me this weekend!</body>
</note>
```

XML Simplifies Things

- It simplifies data sharing
- It simplifies data transport
- It simplifies platform changes
- It simplifies data availability

Many computer systems contain data in incompatible formats. Exchanging data betweenincompatible systems (or upgraded systems) is a time-consuming task for webdevelopers. Large amounts of data must be converted, and incompatible data is oftenlost.
XML stores data in plain text format. This provides a software- and hardware-independent way of storing, transporting, and sharing data.
XML also makes it easier to expand or upgrade to new operating systems, newapplications, or new browsers, without losing data.
With XML, data can be available to all kinds of "reading machines" like people,computers, voice machines, news feeds, etc.

How Can XML be Used?

XML is used in many aspects of web development.
XML is often used to separate data from presentation.
XML Separates Data from Presentation
XML does not carry any information about how to be displayed.

The same XML data can be used in many different presentation scenarios.Because of this, with XML, there is a full separation between data and presentation.

XML is Often a Complement to HTML: In many HTML applications, XML is used to store or transport data, while HTML is usedto format and display the same data.
XML Separates Data from HTML: When displaying data in HTML, you should not have to edit the HTML file when the datachanges.With XML, the data can be stored in separate XML files.With a few lines of JavaScript code, you can read an XML file and update the datacontent of any HTML page.

XML Syntax
Self-Describing Syntax

XML uses a much self-describing syntax.
A prolog defines the XML version and the character encoding:
<?xml version="1.0" encoding="UTF-8"?>
The next line is the root element of the document:

```
<bookstore>
```

The next line starts a <book> element:

```
<book category="cooking">
```

The <book> elements have 4 child elements: <title>, <author>, <year>, <price>.

```
<title lang="en">Everyday Italian</title>
<author>Giada De Laurentiis</author>
<year>2005</year>
<price>30.00</price>
```

The next line ends the book element:

```
</book>
```

Syntax Rules for XML Declaration

- The XML declaration is case sensitive and must begin with "**<?xml>**" where "**xml**" is written in lower-case.
- If document contains XML declaration, then it strictly needs to be the first statement of the XML document.
- The XML declaration strictly needs be the first statement in the XML document.
- An HTTP protocol can override the value of *encoding* that you put in the XML declaration.
- XML Documents Must Have a Root Element
- XML documents must contain one root element that is the parent of all other elements:

In this example <note> is the root element:

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

The XML Prolog(Processing Instruction)
This line is called the XML prolog:

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

The XML prolog is optional. If it exists, it must come first in the document.XML documents can contain international characters, like Norwegian øæå or French êèé.To avoid errors, you should specify the encoding used, or save your XML files as UTF-8.UTF-8 is the default character encoding for XML documents.Character encoding can be studied in our Character Set Tutorial.
UTF-8 is also the default encoding for HTML5, CSS, JavaScript, PHP, and SQL.

All XML Elements Must Have a Closing Tag
In XML, it is illegal to omit the closing tag. All elements must have a closing tag:

Note: The XML prolog does not have a closing tag! This is not an error. The prolog is not a part of the XML document.

XML Tags are Case Sensitive: XML tags are case sensitive. The tag <Letter> is different from the tag <letter>.
Opening and closing tags must be written with the same case:<message>This is correct</message>
"Opening and closing tags" are often referred to as "Start and end tags". Use whateveryou prefer. It is exactly the same thing.
XML Elements Must be Properly Nested
XML Attribute Values Must Always be Quoted
XML elements can have attributes in name/value pairs just like in HTML.

In XML, the attribute values must always be quoted:
<note date="12/11/2007">
<to>Tove</to>
<from>Jani</from>
</note>

Entity References
Some characters have a special meaning in XML.
If you place a character like "<" inside an XML element, it will generate an error becausethe parser interprets it as the start of a new element.
This will generate an XML error:<message>salary < 1000</message>
To avoid this error, replace the "<" character with an entity reference:
<message>salary < 1000</message>

There are 5 pre-defined entity references in XML:
< < less than
> > greater than
& & ampersand
' ' apostrophe
" " quotation mark
Only < and & are strictly illegal in XML, but it is a good habit to replace > with > aswell. User can also create user-defined entities in an XML document.

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

```
<!-- This is a comment -->
```

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

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

White-space is Preserved in XML

XML does not truncate multiple white-spaces (HTML truncates multiple white-spaces toone single white-space):
XML: Hello Tove
HTML: Hello Tove
XML Stores New Line as LF
Windows applications store a new line as: carriage return and line feed (CR+LF).
Unix and Mac OSX use LF.
Old Mac systems use CR.
XML stores a new line as LF.

Well Formed XML
XML documents that conform to the syntax rules above are said to be "Well Formed"XML documents. DTD and Schema are used to validate an XML document.

XML Elements
An XML document contains XML Elements.
What is an XML Element?
An XML element is everything from (including) the element's start tag to (including) theelement's end tag.
<price>29.99</price>
An element can contain:

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

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

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

Empty XML Elements

An element with no content is said to be empty.In XML, you can indicate an empty element like this:<element></element>

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

The two forms produce identical results in XML software (Readers, Parsers, Browsers).

Empty elements can have attributes.

Display Books.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
    <book category="cooking">
        <title lang="en">Everyday Italian</title>
        <author>Giada De Laurentiis</author>
        <year>2005</year>
        <price>30.00</price>
    </book>
    <book category="web">
        <title lang="en">XQuery Kick Start</title>
        <author>James McGovern</author>
        <year>2003</year>
        <price>49.99</price>
    </book>
</bookstore>
```

Thousands of XML formats exists, in many different industries, to describe day-to-daydata transactions:

- Stocks and Shares
- Financial transactions
- Medical data
- Mathematical data
- Scientific measurements
- News information
- Weather services

XML Document structure

XML Tree

XML documents form a tree structure that starts at "the root" and branches to "theleaves".

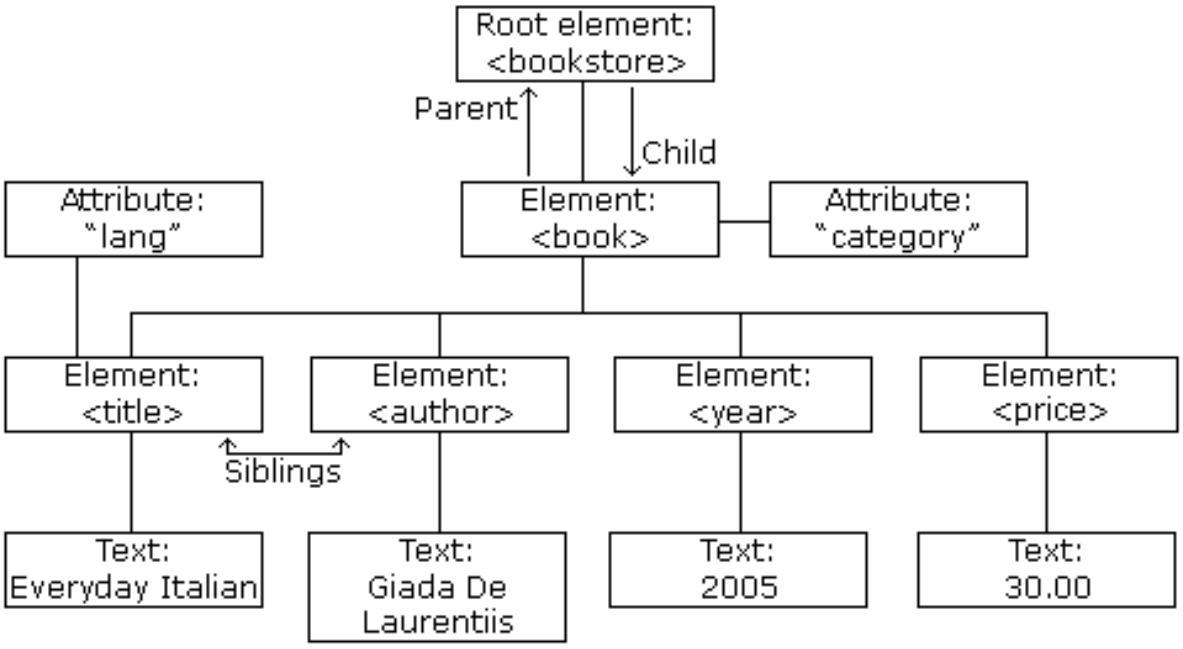
XML Tree Structure

An Example XML Document

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

```
<bookstore>
        <book category="cooking">
            <title lang="en">Everyday Italian</title>
            <author>Giada De Laurentiis</author>
            <year>2005</year>
            <price>30.00</price>
        </book>
        <book category="children">
            <title lang="en">Harry Potter</title>
            <author>J K. Rowling</author>
            <year>2005</year>
            <price>29.99</price>
        </book>
        <book category="web">
            <title lang="en">Learning XML</title>
            <author>Erik T. Ray</author>
            <year>2003</year>
            <price>39.95</price>
        </book>
</bookstore>
```

XML Tree Structure



XML documents are formed as element trees.
An XML tree starts at a root element and branches from the root to child elements.All elements can have sub elements (child elements):

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

```
    </child>
</root>
```

The terms parent, child, and sibling are used to describe the relationships betweenelements.Parents have children. Children have parents. Siblings are children on the same level(brothers and sisters).
All elements can have text content (Harry Potter) and attributes (category="cooking").

DTD(Document Type Definition)

An XML document with correct syntax is called "Well Formed".An XML document validated against a DTD is both "Well Formed" and "Valid".

What is a DTD?

DTD stands for Document Type Definition.A DTD defines the structure and the legal elements and attributes of an XML document.

Valid XML Documents: 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.

XML DTD

The purpose of a DTD is to define the structure and the legal elements and attributes ofan 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.

Using DTD for Entity Declaration

A DOCTYPE declaration can also be used to define special characters or strings, used inthe document:

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE note [
<!ENTITY nbsp " ">
<!ENTITY writer "Writer: Donald Duck.">
<!ENTITY copyright "Copyright: W3Schools.">
]>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
<footer>&writer; &copyright;</footer>
</note>
```

Tip: An entity has three parts: it starts with an ampersand (&), then comes the entity name, and it ends with a semicolon (;).

When to Use a DTD?

With a DTD, independent groups of people can agree to use a standard DTD for interchanging data.With a DTD, you can verify that the data you receive from the outside world is valid.You can also use a DTD to verify your own data.

Namespaces

XML Namespaces provide a method to avoid element name conflicts.

Name Conflicts: In XML, element names are defined by the developer. This often results in a conflictwhen trying to mix XML documents from different XML applications.

This XML carries HTML table information:

```
<table>
<tr>
<td>Apples</td>
<td>Bananas</td>
</tr>
</table>
```

This XML carries information about a table (a piece of furniture):

```
<table>
<name>African Coffee Table</name>
<width>80</width>
<length>120</length>
</table>
```

If these XML fragments were added together, there would be a name conflict. Bothcontain a <table> element, but the elements have different content and meaning.

A user or an XML application will not know how to handle these differences.Solving the Name Conflict Using a Prefix. Name conflicts in XML can easily be avoided using a name prefix.

This XML carries information about an HTML table, and a piece of furniture:

```
<h:table>
<h:tr>
<h:td>Apples</h:td>
<h:td>Bananas</h:td>
</h:tr>
</h:table>
<f:table>
<f:name>African Coffee Table</f:name>
<f:width>80</f:width>
<f:length>120</f:length>
</f:table>
```

In the example above, there will be no conflict because the two <table> elements havedifferent names.

XML Namespaces - The xmlns Attribute

When using prefixes in XML, a namespace for the prefix must be defined.The namespace can be defined by an xmlns attribute in the start tag of an element.The namespace declaration has the following syntax. xmlns:prefix="URI".

```
<root>
<h:table xmlns:h="http://www.w3.org/TR/html4/">
<h:tr>
<h:td>Apples</h:td>
<h:td>Bananas</h:td>
</h:tr>
</h:table>
<f:table xmlns:f="https://www.w3schools.com/furniture">
<f:name>African Coffee Table</f:name>
<f:width>80</f:width>
<f:length>120</f:length>
</f:table>
</root>
```

In the example above:
The xmlns attribute in the first <table> element gives the h: prefix a qualifiednamespace.
The xmlns attribute in the second <table> element gives the f: prefix a qualifiednamespace.
When a namespace is defined for an element, all child elements with the same prefix areassociated with the same namespace.Namespaces can also be declared in the XML root element:
<root xmlns:h="http://www.w3.org/TR/html4/"xmlns:f="https://www.w3schools.com/furniture">

Note: The namespace URI is not used by the parser to look up information.
The purpose of using an URI is to give the namespace a unique name.However, companies often use the namespace as a pointer to a web page containingnamespace information.

XML Naming Rules

XML elements must follow these naming rules:

- Element names are case-sensitive
- Element names must start with a letter or underscore
- Element names cannot start with the letters xml (or XML, or Xml, etc)
- Element names can contain letters, digits, hyphens, underscores, and periods
- Element names cannot contain spaces

Any name can be used, no words are reserved (except xml).

Best Naming Practices

Create descriptive names, like this: <person>, <firstname>, <lastname>.

Create short and simple names, like this: <book_title> not like this: <the_title_of_the_book>.

Avoid "-". If you name something "first-name", some software may think you want tosubtract "name" from "first".

Avoid ".". If you name something "first.name", some software may think that "name" isa property of the object "first".

Avoid ":". Colons are reserved for namespaces (more later).

Non-English letters like éòá are perfectly legal in XML, but watch out for problems if yoursoftware doesn't support them.

XML Attribute

XML elements can have attributes, just like HTML.Attributes are designed to contain data related to a specific element.

XML Attributes Must be Quoted. Attribute values must always be quoted. Either single or double quotes can be used.

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

<person gender="female">

or like this:

<person gender='female'>

If the attribute value itself contains double quotes you can use single quotes, like in this example:

<gangster name='George "Shotgun" Ziegler'>

or you can use character entities:

<gangster name="George "Shotgun" Ziegler">

Note: Some things to consider when using attributes are:

- attributes cannot contain multiple values (elements can)
- attributes cannot contain tree structures (elements can)
- attributes are not easily expandable (for future changes)

Uniform Resource Identifier (URI)

A Uniform Resource Identifier (URI) is a string of characters which identifies anInternet Resource.

The most common URI is the Uniform Resource Locator (URL) which identifies anInternet domain address. Another, not so common type of URI is the UniformResource Name (URN).

Default Namespaces

Defining a default namespace for an element saves us from using prefixes in all the childelements. It has the following syntax:

xmlns="namespaceURI"

This XML carries HTML table information:

```
<table xmlns="http://www.w3.org/TR/html4/">
<tr>
<td>Apples</td>
<td>Bananas</td>
</tr>
</table>
```

This XML carries information about a piece of furniture:

```
<table xmlns="https://www.w3schools.com/furniture">
<name>African Coffee Table</name>
<width>80</width>
<length>120</length>
</table>
```

Namespaces in Real Use

XSLT is a language that can be used to transform XML documents into other formats.

The XML document below, is a document used to transform XML into HTML.The namespace "http://www.w3.org/1999/XSL/Transform" identifies XSLT elementsinside an HTML document:

```
<?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>
<th style="text-align:left">Title</th>
<th style="text-align:left">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>
```

XML Schema

An XML Schema describes the structure of an XML document, just like a DTD.An XML document with correct syntax is called "Well Formed".An XML document validated against an XML Schema is both "Well Formed" and"Valid".

XML Schema

XML Schema is an XML-based alternative to DTD:

```
<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>
```

The Schema above is interpreted like this:

- <xs:element name="note"> defines the element called "note"
- <xs:complexType> the "note" element is a complex type
- <xs:sequence> the complex type is a sequence of elements
- <xs:element name="to" type="xs:string"> the element "to" is of type string (text)
- <xs:element name="from" type="xs:string"> the element "from" is of type string
- <xs:element name="heading" type="xs:string"> the element "heading" is of type string
- <xs:element name="body" type="xs:string"> the element "body" is of type string

- XML Schemas are More Powerful than DTD
- XML Schemas are written in XML
- XML Schemas are extensible to additions
- XML Schemas support data types
- XML Schemas support namespaces

Why Use an XML Schema?

With XML Schema, your XML files can carry a description of its own format.With XML Schema, independent groups of people can agree on a standard forinterchanging data.With XML Schema, you can verify data.

XML Schemas Support Data Types

One of the greatest strengths of XML Schemas is the support for data types:

- It is easier to describe document content
- It is easier to define restrictions on data
- It is easier to validate the correctness of data
- It is easier to convert data between different data types

XML Schemas use XML Syntax

Another great strength about XML Schemas is that they are written in XML:

- You don't have to learn a new language
- You can use your XML editor to edit your Schema files
- You can use your XML parser to parse your Schema files
- You can manipulate your Schemas with the XML DOM
- You can transform your Schemas with XSLT

Displaying XML Documents

Raw XML files can be viewed in all major browsers.Don't expect XML files to be displayed as HTML pages.

Viewing XML Files: note.xml

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

Look at the XML file above in your browser: note.xml

Most browsers will display an XML document with color-coded elements.

Often a plus (+) or minus sign (-) to the left of the elements can be clicked to expand orcollapse the element structure.To view raw XML source, try to select "View Page Source" or "View Source" from thebrowser menu.

Note: In Safari 5 (and earlier), only the element text will be displayed. To view the rawXML, you must right click the page and select "View Source".

Viewing an Invalid XML File. If an erroneous XML file is opened, some browsers will report the error, and some willdisplay it, or display it incorrectly.

XML Documents with CSS

boostore.xml

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/css" href="bookstore.css"?>
<bookstores xmlns:xs="bookstore.xsd">
    <books category="CSE">
        <title>Web Programming</title>
        <author>Chris Bates</author>
        <publisher>Willey</publisher>
        <price>300</price>
    </books>
    <books category="CSE">
        <title>Computer Networks</title>
        <author>Tanenbaum</author>
        <publisher>Pearson</publisher>
        <price>400</price>
```

```
        </books>
</bookstores>
```

bookstore.css

```
bookstore{
    color: blue;
    width: 100;
        }
author{
    display:block;
        }
publisher{
    display:block;
        }
price{
    display:block;
        }
 title {
    font-size:25px;
    font-weight:bold;
        display:block; }
```

XPath

XPath is a major element in the XSLT standard.

XPath can be used to navigate through elements and attributes in an XML document.

- XPath stands for XML Path Language
- XPath uses "path like" syntax to identify and navigate nodes in an XML document
- XPath contains over 200 built-in functions
- XPath is a major element in the XSLT standard
- XPath is a W3C recommendation

XPath Path Expressions

XPath uses path expressions to select nodes or node-sets in an XML document.

These path expressions look very much like the path expressions you use with traditional computer file systems:

XSLT

With XSLT you can transform an XML document into HTML.

Displaying XML with XSLT

XSLT (eXtensible Stylesheet Language Transformations) is the recommended style sheetlanguage for XML.XSLT is far more sophisticated than CSS. With XSLT you can add/remove elements andattributes to or from the output file. You can also rearrange and sort elements, performtests and make decisions about which elements to hide and display, and a lot more.XSLT uses XPath to find information in an XML document.

XSLT Example(menu.xml)

We will use the following XML document:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="menu.xsl"?>
<breakfast_menu>
<food>
<name>Belgian Waffles</name>
<price>$5.95</price>
<description>Two of our famous Belgian Waffles with plenty of real maple
syrup</description>
<calories>650</calories>
</food>
<food>
<name>Strawberry Belgian Waffles</name>
<price>$7.95</price>
<description>Light Belgian waffles covered with strawberries and whipped
cream</description>
<calories>900</calories>
</food>
<food>
<name>Berry-Berry Belgian Waffles</name>
<price>$8.95</price>
<description>Light Belgian waffles covered with an assortment of fresh berries
and whipped cream</description>
<calories>900</calories>
</food>
<food>
<name>French Toast</name>
<price>$4.50</price>
<description>Thick slices made from our homemade sourdough bread</description>
<calories>600</calories>
</food>
<food>
<name>Homestyle Breakfast</name>
<price>$6.95</price>
<description>Two eggs, bacon or sausage, toast, and our ever-popular hash
browns</description>
```

```
<calories>950</calories>
</food>
</breakfast_menu>
```

Use XSLT to transform XML into HTML, before it is displayed in a browser:

Example XSLT Stylesheet:(menu.xsl)

```
<?xml version="1.0" encoding="UTF-8"?>
<html xsl:version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<body style="font-family:Arial;font-size:12pt;background-color:#EEEEEE">
<xsl:for-each select="breakfast_menu/food">
<div style="background-color:teal;color:white;padding:4px">
<span style="font-weight:bold"><xsl:value-of select="name"/> - </span>
<xsl:value-of select="price"/>
</div>
<div style="margin-left:20px;margin-bottom:1em;font-size:10pt">
<p>
<xsl:value-of select="description"/>
<span style="font-style:italic"> (<xsl:value-of select="calories"/> calories
per serving)</span>
</p>
</div>
</xsl:for-each>
</body>
</html>
```

XML Processors

XMLHttpRequest Object

All modern browsers have a built-in XMLHttpRequest object to request data from aserver.

The XMLHttpRequest Object

The XMLHttpRequest object can be used to request data from a web server.The XMLHttpRequest object is a 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 Example

When you type a character in the input field below, an XMLHttpRequest is sent to theserver, and some name suggestions are returned (from the server):

A common JavaScript syntax for using the XMLHttpRequest object looks much like this:

Example

```
var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
if (this.readyState == 4 && this.status == 200) {
```

```
// Typical action to be performed when the document is ready:
document.getElementById("demo").innerHTML = xhttp.responseText;
}
};
xhttp.open("GET", "filename", true);
xhttp.send();
```

Example Explained

The first line in the example above creates an XMLHttpRequest object:

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

The onreadystatechange property specifies a function to be executed every time the status of the XMLHttpRequest object changes:

```
xhttp.onreadystatechange = function()
```

When readyState property is 4 and the status property is 200, the response is ready:

```
if (this.readyState == 4 && this.status == 200)
```

The responseText property returns the server response as a text string.

The text string can be used to update a web page:

```
document.getElementById("demo").innerHTML = xhttp.responseText;
```

XML Parser

All major browsers have a built-in XML parser to access and manipulate XML.

XML Parser

There are two types of parser available: 1) DOM Parser, 2) SAX Parser

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 DOMobject.All modern browsers have a built-in XML parser that can convert text into an XML DOMobject.

Parsing a Text String

This example parses a text string into an XML DOM object, and extracts the info from it with JavaScript:

The XML DOM is a standard for how to get, change, add, and delete XML elements.

This example loads 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>
```

Example Explained

A text string is defined:

```
text = "<bookstore><book>" +
"<title>Everyday Italian</title>" +
"<author>Giada De Laurentiis</author>" +
"<year>2005</year>" +
"</book></bookstore>";
```

An XML DOM parser is created:

```
parser = new DOMParser();
```

The parser creates a new XML DOM object using the text string:

```
xmlDoc = parser.parseFromString(text,"text/xml");
```

The XMLHttpRequest Object

The XMLHttpRequest Object has a built in XML Parser.

The responseText property returns the response as a string.

The responseXML property returns the response as an XML DOM object.

If you want to use the response as an XML DOM object, you can use the responseXMLproperty.

Example

Request the file cd_catalog.xml and use the response as an XML DOM object:

```
xmlDoc = xmlhttp.responseXML;
txt = "";
x = xmlDoc.getElementsByTagName("ARTIST");
for (i = 0; i < x.length; i++) {
txt += x[i].childNodes[0].nodeValue + "<br>";
}
document.getElementById("demo").innerHTML = txt;
```

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>
```

Enterprise Architecture Styles

J2EE Architecture

- The client/server application architecture.
- Which was a two-tier architecture.
- Evolved over time to a multitier architecture.
- This natural progression occurred as additional tiers were introduced between the end-user clients and back-end systems.
- Although a multitier architecture brings greater flexibility of design.
- It also increases the
  - complexity of building,
  - testing, deploying,
  - administering, and
  - maintaining application components.

Two-Tier Architecture

- The two-tier architecture is also known as the client/server architecture.
- It consists mainly of two tiers:

o data and
o client (GUI).

- The application logic can be located in either the client tier.
- Which results in a fat client or located in the data tier.
- which results in a fat server (see Figure).

[Figure : Two-tier architecture.]

- This type of architecture suffers from a lack of scalability.
- Because both the client and the server have limited resources.
- In addition to the negative effect of network traffic to transfer data to the fat client.
- Another issue is maintainability.
- We have to roll out the new system version to all system users.

Three-Tier Architecture

- To address the issues of the two-tier architecture, the application logic will be placed in its own tier.
- Thus applications can be portioned into three tiers.

- The first tier is referred to as the presentation layer, and consists of the application GUI.
- The middle tier, or the business layer, consists of the business logic to retrieve data for the user requests.
- The back-end tier, or data layer, consists of the data needed by the application.
- Figure illustrates the three-tier architecture.

[Figure : Three-tier architecture.]

- The decoupling of application logic from either presentation or data increases the flexibility of the application design.
- Multiple views or a GUI can be added without changing the existing application logic.
- Similarly, multiple applications can be created using the same data model.
- Changing the components of one tier should not impact the other two tiers.
- For example, any change to the data or GUI will not affect the application logic.

Note :- 1
- The three-tier architecture is the basis for J2EE applications.
- In which EJBs provide a mechanism to build application logic.
- While JSPs and servlets abstract the presentation layer and allow interaction with the business layer.
- One important feature of the three-tier architecture is sharing system resources by all clients.
- Which results in :
o highly efficient,
o scalable,
o secure, and
o reliable applications.

Multitier J2EE Architecture
- Multitier (or n-tier) architecture differs from the three-tier architecture in viewing each tier logically.
- Rather than physically.
- The application logic, for example, can be split into more than one layer; the business logic tier and the presentation logic tier.
- Similarly, the user interface is partitioned into the client tier and the presentation tier.
- A multitier architecture determines where the software components that make up a computing system are executed in relation to each other and to the hardware, network, and users.
- J2EE is a multitier architecture, which partitions the application into client,
o presentation logic,
o business logic, and
o enterprise information tiers.

Note : - 2
- The J2EE platform is designed not only to support a multitier architecture to partition applications.
- But also to provide infrastructure common services to reduce the complexity of developing and deploying these applications.
- Other than multitier, the J2EE architecture provides the enterprise with common infrastructure services.
- Which help in developing and deploying :
o portable,
o secure and

o transactional applications.

- The J2EE architecture partitions enterprise applications into three fundamental parts:
o components,
o containers, and
o connectors.

- Components are the key focus of application developers.
- Whereas system vendors implement containers and
- Connectors to hide complexity and enhance portability.
- Enterprise Java applications can run on any J2EE-compliant application server.

Note :- 3
- Multitier distributed applications follow the Model-View-Controller (MVC) paradigm,.
- This design pattern provides clean separation between tiers.
- Using this paradigm, the model (data tier) is separated from the view (client and presentation tiers).
- Similarly, the controller (the application logic tier) is separated from both the view and the model.
- Containers transparently provide common services :
o including transaction,
o security,
o persistence, and
o resource pooling, to both clients and components.

- A container allows the configuration of applications.
- And components at deployment.
- Rather than hard-coding them in program code.
- Connectors extend the J2EE platform.
- By defining a portable client service API to plug into existing enterprise vendor products.
- Connectors promote flexibility by enabling a variety of implementations of specific services.

Features of EE platform

The Java EE stands for Java Enterprise Edition, which was earlier known as J2EE and is currently known as Jakarta EE. It is a set of specifications wrapping around Java SE (Standard Edition). The Java EE provides a platform for developers with enterprise features such as distributed computing and web services. Java EE applications are usually run on reference run times such as microservers or application servers. Examples of some contexts where Java EE is used are e-commerce, accounting, banking information systems.

Specifications of Java EE
Java EE has several specifications which are useful in making web pages, reading and writing from database in a transactional way, managing distributed queues. The Java EE contains several APIs which have the functionalities of base Java SE APIs such as Enterprise JavaBeans, connectors, Servlets, Java Server Pages and several web service technologies.

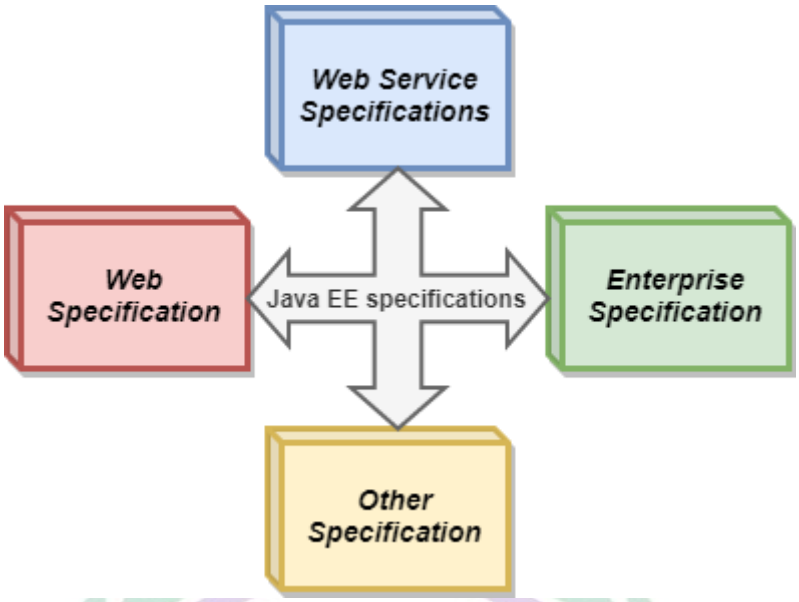


Fig: specifications of EE platform.

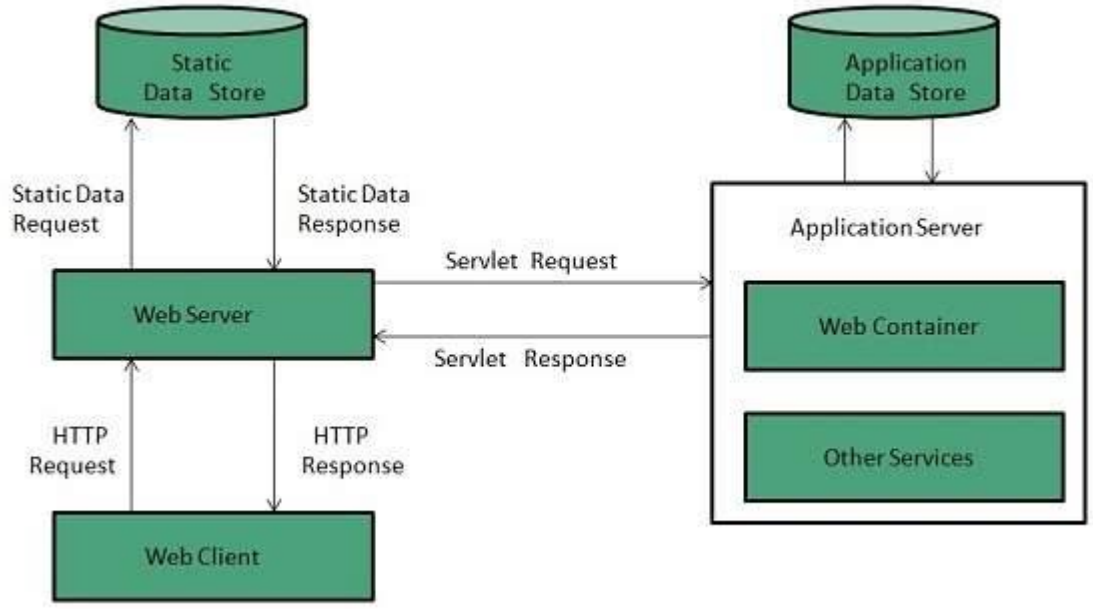
Web servers and application servers

Web server is a computer where the web content is stored. Basically web server is used to host the web sites but there exists other web servers also such as gaming, storage, FTP, email etc.

Web Server Working

Web server respond to the client request in either of the following two ways:

- Sending the file to the client associated with the requested URL.
- Generating response by invoking a script and communicating with database



Key Points

- When client sends request for a web page, the web server search for the requested page if requested page is found then it will send it to client with an HTTP response.
- If the requested web page is not found, web server will the send an **HTTP response:Error 404 Not found.**
- If client has requested for some other resources then the web server will contact to the application server and data store to construct the HTTP response.

Architecture

Web Server Architecture follows the following two approaches:

1. Concurrent Approach
2. Single-Process-Event-Driven Approach.

Concurrent Approach

Concurrent approach allows the web server to handle multiple client requests at the same time. It can be achieved by following methods:

- Multi-process
- Multi-threaded
- Hybrid method.

Application Servers

An application server is a server specifically designed to run applications. The "server" includes both the hardware and software that provide an environment for programs to run.

Application servers are used for many purposes. Several examples are listed below:

- running web applications
- hosting a hypervisor that manages virtual machines
- distributing and monitoring software updates
- processing data sent from another server

An application server provides the processing power and memory to run these applications in real-time. It also provides the environment to run specific applications. For example, a cloud service may need to process data on a Windows machine. A Linux-based server may provide the web interface for the cloud service, but it cannot run Windows applications. Therefore, it may send input data to a Windows-based application server. The application server can process the data, then return the result to the web server, which can output the result in a web browser.

JDBC DRIVERS

JDBC Driver is a software component that enables java application to interact with the database. There are 4 types of JDBC drivers:

1. JDBC-ODBC bridge driver

2. Native-API driver (partially java driver)
3. Network Protocol driver (fully java driver)
4. Thin driver (fully java driver)

1) JDBC-ODBC bridge driver

The JDBC-ODBC bridge driver uses ODBC driver to connect to the database. The JDBC-ODBC bridge driver converts JDBC method calls into the ODBC function calls. This is now discouraged because of thin driver.

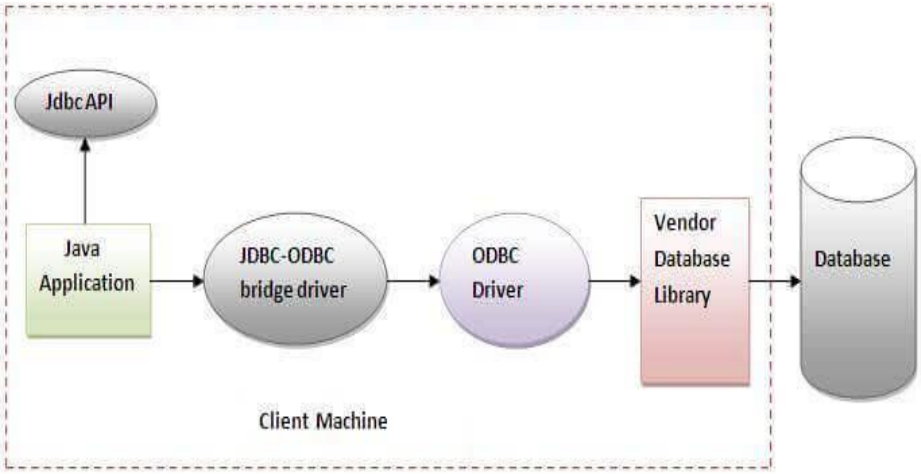


Figure- JDBC-ODBC Bridge Driver

Oracle does not support the JDBC-ODBC Bridge from Java 8. Oracle recommends that you use JDBC drivers provided by the vendor of your database instead of the JDBC-ODBC Bridge.

Advantages:

- easy to use.
- can be easily connected to any database.

Disadvantages:

- Performance degraded because JDBC method call is converted into the ODBC function calls.
- The ODBC driver needs to be installed on the client machine.

2) Native-API driver

The Native API driver uses the client-side libraries of the database. The driver converts JDBC method calls into native calls of the database API. It is not written entirely in java.

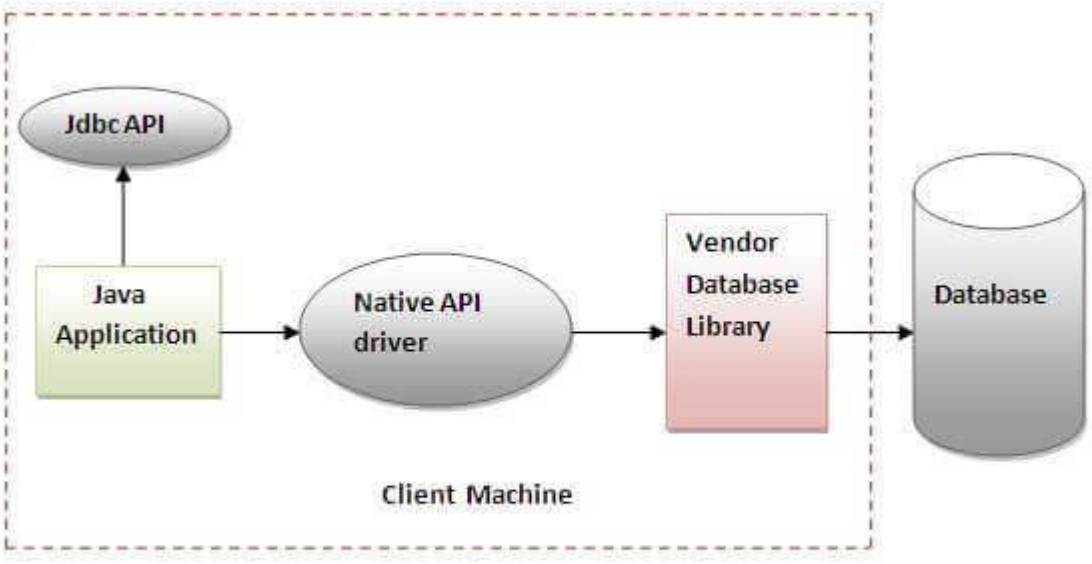


Figure- Native API Driver

Advantage:

- performance upgraded than JDBC-ODBC bridge driver.

Disadvantage:

- The Native driver needs to be installed on the each client machine.
- The Vendor client library needs to be installed on client machine.

3) Network Protocol driver

The Network Protocol driver uses middleware (application server) that converts JDBC calls directly or indirectly into the vendor-specific database protocol. It is fully written in java.

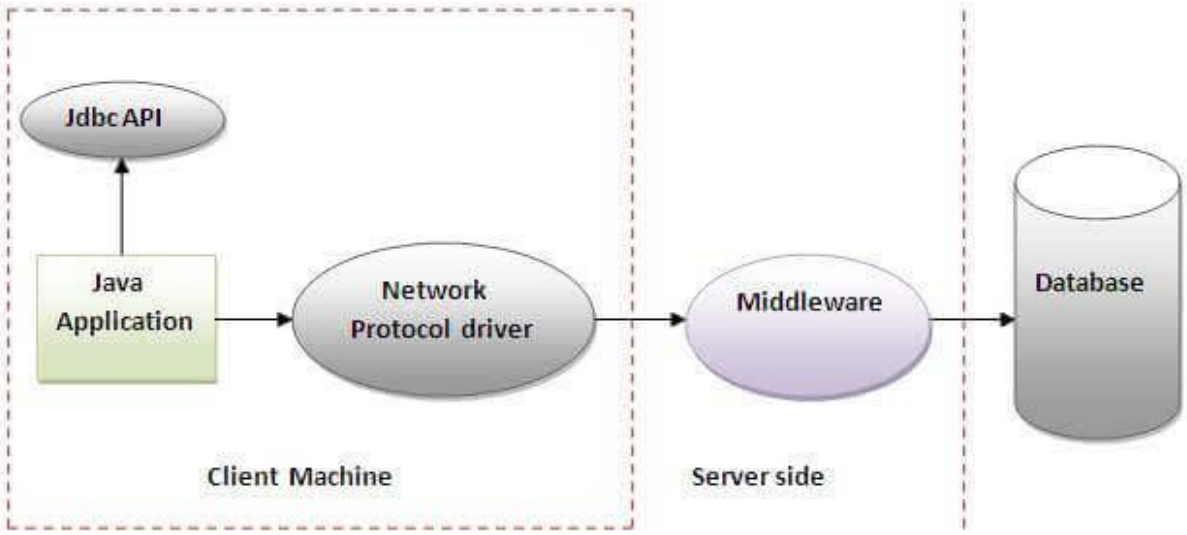


Figure- Network Protocol Driver

4) Thin driver

The thin driver converts JDBC calls directly into the vendor-specific database protocol. That is why it is known as thin driver. It is fully written in Java language.

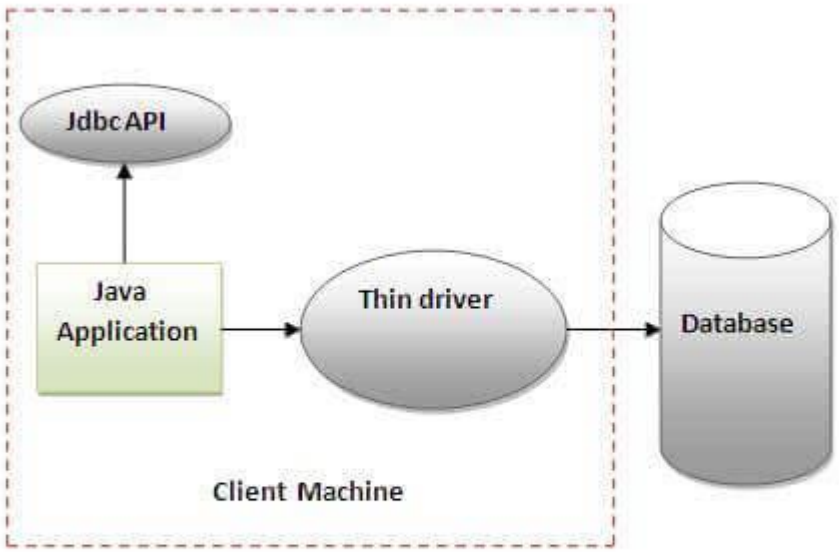


Figure- Thin Driver

Advantage:

- Better performance than all other drivers.
- No software is required at client side or server side.

Disadvantage:

- Drivers depend on the Database.

Exploring JDBC Processes with the java.sql Package.

The java.sql package provides the API for accessing and processing data stored in a data source (usually a relational database) using the Java programming language. This API includes a framework whereby different drivers can be installed dynamically to access different data sources. Although the JDBC API is mainly geared to passing SQL statements to a database, it provides for reading and writing data from any data source with a tabular format. The reader/writer facility, available through the *javax.sql.RowSet* group of interfaces can be customized to use and update data from a spreadsheet, flat file, or any other tabular data source.

What the java.sql Package Contains

The java.sql package contains API for the following:

1. Making a connection with a database via the DriverManager facility

- DriverManager class -- makes a connection with a driver
- SQLPermission class -- provides permission when code running within a Security Manager, such as an applet, attempts to set up a logging stream through the DriverManager
- Driver interface -- provides the API for registering and connecting drivers based on JDBC technology ("JDBC drivers"); generally used only by the DriverManager class
- DriverPropertyInfo class -- provides properties for a JDBC driver; not used by the general user

2. Sending SQL statements to a database

- Statement -- used to send basic SQL statements
- PreparedStatement -- used to send prepared statements or basic SQL statements (derived from Statement)
- CallableStatement -- used to call database stored procedures (derived from PreparedStatement)
- Connection interface -- provides methods for creating statements and managing connections and their properties
- Savepoint -- provides savepoints in a transaction

3. Retrieving and updating the results of a query

- ResultSet interface

4. Standard mappings for SQL types to classes and interfaces in the Java programming language

- Array interface -- mapping for SQL ARRAY
- Blob interface -- mapping for SQL BLOB
- Clob interface -- mapping for SQL CLOB
- Date class -- mapping for SQL DATE
- NClob interface -- mapping for SQL NCLOB
- Ref interface -- mapping for SQL REF
- RowId interface -- mapping for SQL ROWID
- Struct interface -- mapping for SQL STRUCT
- SQLXML interface -- mapping for SQL XML
- Time class -- mapping for SQL TIME
- Timestamp class -- mapping for SQL TIMESTAMP
- Types class -- provides constants for SQL types

5. Custom mapping an SQL user-defined type (UDT) to a class in the Java programming language

1. SQLData interface -- specifies the mapping of a UDT to an instance of this class
2. SQLInput interface -- provides methods for reading UDT attributes from a stream
3. SQLOutput interface -- provides methods for writing UDT attributes back to a stream

6. Metadata

- DatabaseMetaData interface -- provides information about the database
- ResultSetMetaData interface -- provides information about the columns of a ResultSet object
- ParameterMetaData interface -- provides information about the parameters to PreparedStatement commands

7. Exceptions

- SQLException -- thrown by most methods when there is a problem accessing data and by some methods for other reasons
- SQLWarning -- thrown to indicate a warning
- DataTruncation -- thrown to indicate that data may have been truncated
- BatchUpdateException -- thrown to indicate that not all commands in a batch update executed successfully

----------END OF UNIT -III ----------

UNIT-IV

Servlets Technology: Exploring the Features of Java Servlet, Exploring the Servlet API, Explaining the Servlet Life Cycle, Creating a Sample Servlet, Working with ServletConfig and ServletContext Objects, Implementing Servlet Collaboration, Exploring the Session Tracking Mechanisms.

Servlets

Servlet technology is used to create a web application (resides at server side and generates a dynamic web page).

Servlet technology is robust and scalable because of java language. Before Servlet, CGI (Common Gateway Interface) scripting language was common as a server-side programming language. However, there were many disadvantages to this technology.

There are many interfaces and classes in the Servlet API such as Servlet, GenericServlet, HttpServlet, ServletRequest, ServletResponse, etc.

What is a Servlet?

Servlet can be described in many ways, depending on the context.

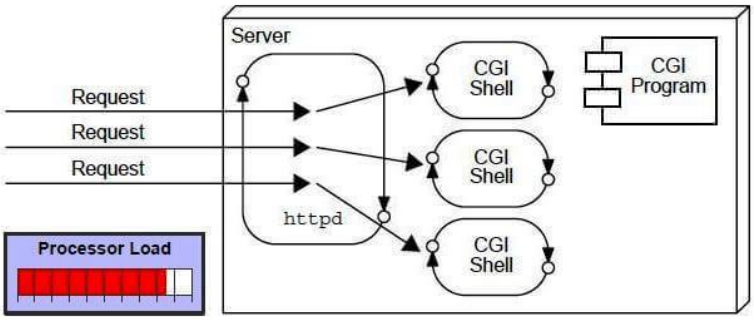
1. Servlet is a technology which is used to create a web application.
2. Servlet is an API that provides many interfaces and classes including documentation.
3. Servlet is an interface that must be implemented for creating any Servlet.
4. Servlet is a class that extends the capabilities of the servers and responds to the incoming requests. It can respond to any requests.
5. Servlet is a web component that is deployed on the server to create a dynamic web page.

What is a web application?

A web application is an application accessible from the web. A web application is composed of web components like Servlet, JSP, Filter, etc. and other elements such as HTML, CSS, and JavaScript. The web components typically execute in Web Server and respond to the HTTP request.

CGI (Common Gateway Interface)

CGI technology enables the web server to call an external program and pass HTTP request information to the external program to process the request. For each request, it starts a new process.

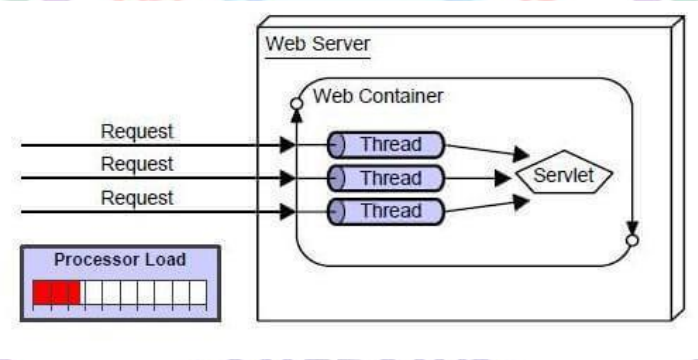


Disadvantages of CGI

There are many problems in CGI technology:

1. If the number of clients increases, it takes more time for sending the response.
2. For each request, it starts a process, and the web server is limited to start processes.
3. It uses platform dependent language e.g. C, C++, perl.

Advantages of Servlet:

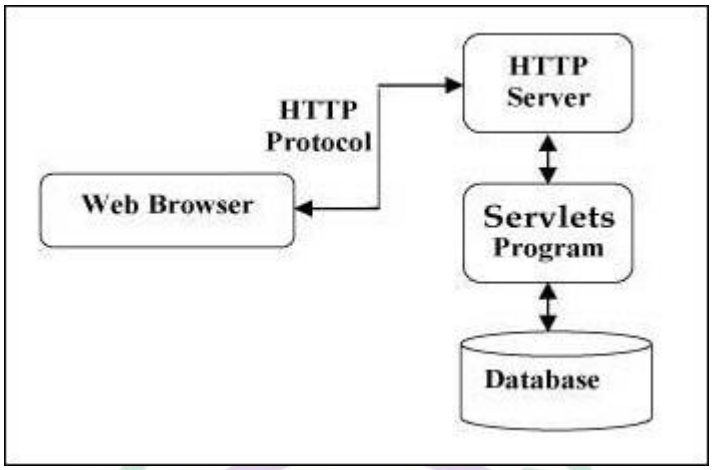


There are many advantages of Servlet over CGI. The web container creates threads for handling the multiple requests to the Servlet. Threads have many benefits over the Processes such as they share a common memory area, lightweight, cost of communication between the threads are low. The advantages of Servlet are as follows:

1. **Better performance:** because it creates a thread for each request, not process.
2. **Portability:** because it uses Java language.
3. **Robust:** JVM manages Servlets, so we don't need to worry about the memory leak, garbage collection, etc.
4. **Secure:** because it uses java language.

Servlets Architecture

The following diagram shows the position of Servlets in a Web Application.



Servlets Tasks

Servlets perform the following major tasks −

- 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.

Servlets Packages

The **javax.servlet** and **javax.servlet.http** packages represent interfaces and classes for servlet api.

The **javax.servlet** package contains many interfaces and classes that are used by the servlet or web container. These are not specific to any protocol.

The **javax.servlet.http** package contains interfaces and classes that are responsible for http requests only.

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.

1. The servlet is initialized by calling the init() method.
2. The servlet calls service() method to process a client's request.
3. The servlet is terminated by calling the destroy() method.
4. Finally, servlet is garbage collected by the garbage collector of the JVM.

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.

Here is the signature of this method −

```
public void service(ServletRequest request, ServletResponse response)
  throws ServletException, IOException {
}
```

The service () method is called by the container and service method invokes doGet, doPost, doPut, doDelete, etc. methods as appropriate. So you have nothing to do with service() method but you override either doGet() or doPost() depending on what type of request you receive from the client.

The doGet() and doPost() are most frequently used methods with in each service request. Here is the signature of these two methods.

The doGet() Method

A GET request results from a normal request for a URL or from an HTML form that has no METHOD specified and it should be handled by doGet() method.

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
// Servlet code
}
```

The doPost() Method

A POST request results from an HTML form that specifically lists POST as the METHOD and it should be handled by doPost() method.

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
// Servlet code
}
```

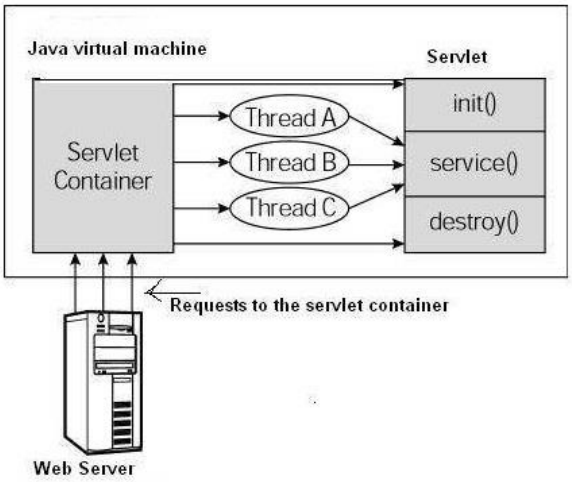
The destroy() Method

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...
}
```

The following figure depicts a typical servlet life-cycle scenario.

- First the HTTP requests coming to the server are delegated to the servlet container.
- The servlet container loads the servlet before invoking the service() method.
- Then the servlet container handles multiple requests by spawning multiple threads, each thread executing the service() method of a single instance of the servlet.



Reading Form Data using Servlet

Servlets handles form data parsing automatically using the following methods depending on the situation −

- **getParameter()** − You call request.getParameter() method to get the value of a form parameter.
- **getParameterValues()** − Call this method if the parameter appears more than once and returns multiple values, for example checkbox.
- **getParameterNames()** − Call this method if you want a complete list of all parameters in the current request.

GenericServlet class implements Servlet, ServletConfig and Serializable interfaces. It provides the implementation of all the methods of these interfaces except the service method. GenericServlet class can handle any type of request so it is protocol-independent. You may create a generic servlet by inheriting the GenericServlet class and providing the implementation of the service method.

Methods of GenericServlet class

There are many methods in GenericServlet class. They are as follows:

1) **public void init(ServletConfig config)** is used to initialize the servlet.
2) **public abstract void service(ServletRequest request, ServletResponse response)** provides service for the incoming request. It is invoked at each time when user requests for a servlet.
3) **public void destroy()** is invoked only once throughout the life cycle and indicates that servlet is being destroyed.
4) **public ServletConfig getServletConfig()** returns the object of ServletConfig.

5) **public String getServletInfo()** returns information about servlet such as writer, copyright, version etc.
6) **public void init()** it is a convenient method for the servlet programmers, now there is no need to call super.init(config)
7) **public ServletContext getServletContext()** returns the object of ServletContext.
8) **public String getInitParameter(String name)** returns the parameter value for the given parameter name.
9) **public Enumeration getInitParameterNames()** returns all the parameters defined in the web.xml file.
10) **public String getServletName()** returns the name of the servlet object.
11) **public void log(String msg)** writes the given message in the servlet log file.
12) **public void log(String msg,Throwable t)** writes the explanatory message in the servlet log file and a stack trace.

The **HttpServlet** class extends the GenericServlet class and implements Serializable interface. It provides http specific methods such as doGet, doPost, doHead, doTrace etc. There are many methods in HttpServlet class. They are as follows:

1) **public void service(ServletRequest req,ServletResponse res)** dispatches the request to the protected service method by converting the request and response object into http type.
2) **protected void service(HttpServletRequest req, HttpServletResponse res**) receives the request from the service method, and dispatches the request to the doXXX() method depending on the incoming http request type.
3) **protected void doGet(HttpServletRequest req, HttpServletResponse res)** handles the GET request. It is invoked by the web container.
4) **protected void doPost(HttpServletRequest req, HttpServletResponse res)** handles the POST request. It is invoked by the web container.
5) **protected void doHead(HttpServletRequest req, HttpServletResponse res**) handles the HEAD request. It is invoked by the web container.
6) **protected void doOptions(HttpServletRequest req, HttpServletResponse res)** handles the OPTIONS request. It is invoked by the web container.
7) **protected void doPut(HttpServletRequest req, HttpServletResponse res)** handles the PUT request. It is invoked by the web container.
8) **protected void doTrace(HttpServletRequest req, HttpServletResponse res)** handles the TRACE request. It is invoked by the web container.

9) **protected void doDelete(HttpServletRequest req, HttpServletResponse res)** handles the DELETE request. It is invoked by the web container.

10) **protected long getLastModified(HttpServletRequest req)** returns the time when HttpServletRequest was last modified since midnight January 1, 1970 GMT.

Steps to create a servlet example

There are given 6 steps to create a servlet example. These steps are required for all the servers.

The servlet example can be created by three ways:

1. By implementing Servlet interface,
2. By inheriting GenericServlet class, (or)
3. By inheriting HttpServlet class

The mostly used approach is by extending HttpServlet because it provides http request specific method such as doGet(), doPost(), doHead() etc.

Here, we are going to use apache tomcat server in this example. The steps are as follows:

1. Create a directory structure
2. Create a Servlet
3. Compile the Servlet
4. Create a deployment descriptor
5. Start the server and deploy the project
6. Access the servlet

How web container handles the servlet request?

The web container is responsible to handle the request. Let's see how it handles the request.

- maps the request with the servlet in the web.xml file.
- creates request and response objects for this request
- calls the service method on the thread
- The public service method internally calls the protected service method
- The protected service method calls the doGet method depending on the type of request.
- The doGet method generates the response and it is passed to the client.
- After sending the response, the web container deletes the request and response objects. The thread is contained in the thread pool or deleted depends on the server implementation.

ServletRequest Interface

An object of ServletRequest is used to provide the client request information to a servlet such as content type, content length, parameter names and values, header informations, attributes etc.

Methods of ServletRequest interface

There are many methods defined in the ServletRequest interface. Some of them are as follows:

Method	Description
public String getParameter(String name)	is used to obtain the value of a parameter by name.
public String[] getParameterValues(String name)	returns an array of String containing all values of given parameter name. It is mainly used to obtain values of a Multi select list box.
java.util.Enumeration getParameterNames()	returns an enumeration of all of the request parameter names.
public int getContentLength()	Returns the size of the request entity data, or -1 if not known.
public String getCharacterEncoding()	Returns the character set encoding for the input of this request.
public String getContentType()	Returns the Internet Media Type of the request entity data, or null if not known.
PublicServletInputStream getInputStream() throws IOException	Returns an input stream for reading binary data in the request body.
public abstract String getServerName()	Returns the host name of the server that received the request.
public int getServerPort()	Returns the port number on which this request was received.

RequestDispatcher in Servlet

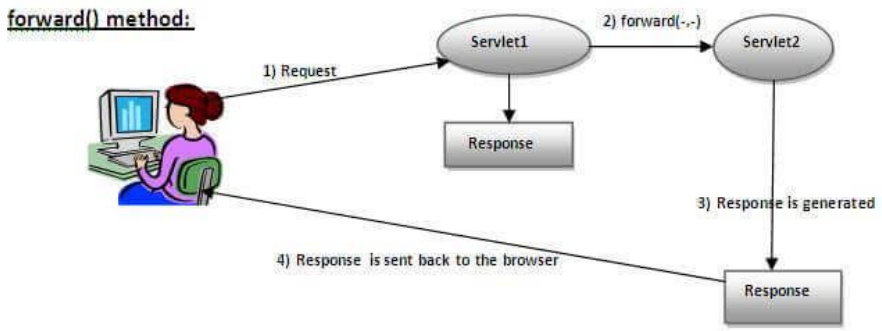
The RequestDispatcher interface provides the facility of dispatching the request to another resource it may be html, servlet or jsp. This interface can also be used to include the content of another resource also. It is one of the way of servlet collaboration. There are two methods defined in the RequestDispatcher interface.

Methods of RequestDispatcher interface

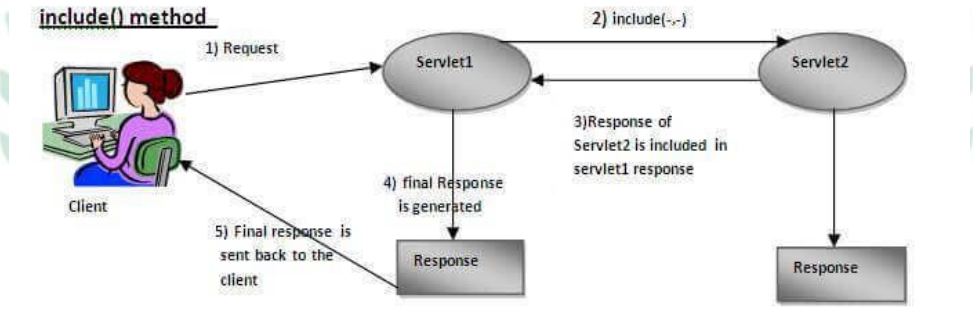
The RequestDispatcher interface provides two methods. They are:

1. **public void forward(ServletRequest request,ServletResponse response)**throws ServletException,java.io.IOException:Forwards a request from a servlet to another resource (servlet, JSP file, or HTML file) on the server.

2. **public void include(ServletRequest request,ServletResponse response**)throws ServletException,java.io.IOException:Includes the content of a resource (servlet, JSP page, or HTML file) in the response.



As you see in the above figure, response of second servlet is sent to the client. Response of the first servlet is not displayed to the user.



As you can see in the above figure, response of second servlet is included in the response of the first servlet that is being sent to the client.

SendRedirect in servlet

The sendRedirect() method of HttpServletResponse interface can be used to redirect response to another resource, it may be servlet, jsp or html file. It accepts relative as well as absolute URL. It works at client side because it uses the url bar of the browser to make another request. So, it can work inside and outside the server.

Difference between forward() and sendRedirect() method

There are many differences between the forward() method of RequestDispatcher and sendRedirect() method of HttpServletResponse interface. They are given below:

forward() method	sendRedirect() method

The forward() method works at server side.	The sendRedirect() method works at client side.
It sends the same request and response objects to another servlet.	It always sends a new request.
It can work within the server only.	It can be used within and outside the server.
Example:request.getRequestDispacher("servlet2").forward(request,response);	Example: response.sendRedirect("servlet2");

Session Management (Session Tracking)

HTTP is a stateless protocol.It means, each request made to server is independent of the previous one. However, in some applications, it is necessary to save state information. State information is collected from several interactions between a browser and a server. Sessions provide such a mechanism.

Session: A sequence of requests made by some client to a web-server.

In client-server environment, while working in www, client and server communicate with each other using HTTP protocol. It is a stateless protocol, ie, once the request was processed, neither the client nor the server were able to identity the other.Because of this whenever a client sends a second request, server is unable to identify the client, which forces the client to login to server once again.

To overcome the above mentioned problem and making the serverside program to keep track of the client we use session management.

We might have seen sometime in client server communication that, “session timeout”.

Different Session Tracking Techniques

There are four techniques used in Session tracking:

1. **Hidden Form Field**
2. **URL Rewriting**
3. **Cookies**
4. **HttpSession**

Hidden form fields:

In html form fields, we can have an element whose type="hidden". It is an invisible component. Whenever the client is login on to the server, we will be validating the user identity and generates some response. While writing the response to client, we will be adding the hidden field in the response that contains some data regarding the client.

Whenever the client send the second request to server, with the help of javascript we will retrieve the data from the hidden field and submit to server as a part of request to server.

Implementation of this concept is very simple, but this leads to security problems.

URL Rewriting:

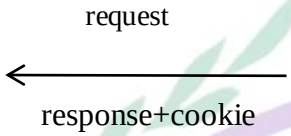
display text

In client-server communication, we will be writing lots of URLs to the client machine as part of the response. In URL-Rewriting, we are going to change the way we write URL. While writing some URL to client machine, we will append a query string(name=value) that contains client data.

Whenever the client submits this re-written URL to the server, server retrieves the client data form the URL, and generates the response back to the intended client.

Cookies:

A Cookie is a small piece of data about the client, which will be stored on the client machine by the server machine.
Cookies are used by server. To store a cookie, 1Kb of memory is required.



Client stores the cookies in the form a text file.

In cookies, just prior to writing some response to the client, we will create a cookie object and place it in http headers, which will be delivered to the client. Once these http headers were received by the client browser, it will retrieve the cookie from the header and stores them in a separate text file.

Whenever the client tries to send another request to the same server, browser will retrieve the cookie associated with the server and places them as a part of http header, and send the request. Withe the help of cookie, server remember the client.

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.
3. No limitation on number of cookies.

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.
3. They must be written to client prior to any response to client.

Cookie class
javax.servlet.http.Cookie class provides the functionality of using cookies. It provides a lot of useful methods for cookies.

Constructor of Cookie class

Constructor	Description

Cookie()	constructs a cookie.
Cookie(String name, String value)	constructs a cookie with a specified name and value.

Useful Methods of Cookie class

There are given some commonly used methods of the Cookie class.

Method	Description
public void setMaxAge(int expiry)	Sets the maximum age of the cookie in seconds. -ive value: retain cookie for ever 0: immediately removes cookie +ive value: retains it upto specified time
public String getName()	Returns the name of the cookie. The name cannot be changed after creation.
public String getValue()	Returns the value of the cookie.
public void setName(String name)	changes the name of the cookie.
public void setValue(String value)	changes the value of the cookie.

public void addCookie(Cookie ck):method of HttpServletResponse interface is used to add cookie in response object.

public Cookie[] getCookies():method of HttpServletRequest interface is used to return all the cookies from the browser in an array.

Cookie is created with the following syntax

```
Cookie ck=new Cookie("user","praveen");          //creating cookie object
response.addCookie(ck);                          //adding cookie in the response
```

HttpSession:

Purely handled by the server only.

A session can be created via the getSession( ) method of HttpServletRequest. An HttpSession object is returned. This object can store a set of bindings that associate names with objects. The

setAttribute( ), getAttribute( ), getAttributeNames( ), and removeAttribute( ) methods of HttpSession manage these bindings. It is important to note that session state is shared among all the servlets that are associated with a particular client.

Advantages: Session tracking will available always

We can store java objects also

It can handle more than one value with single session object

The following servlet illustrates how to use session state. The **getSession( )** method gets the current session. A new session is created if one does not already exist. The **getAttribute( )** method is called to obtain the object that is bound to the name "date". That object is a **Date** object that encapsulates the date and time when this page was last accessed. (Of course, there is no such binding when the page is first accessed.) A **Date** object encapsulating the current date and time is then created. The **setAttribute( )** method is called to bind the name "date" to this object.

```
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class DateServlet extends HttpServlet {
public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
{   // Get the HttpSession object.
HttpSession hs = request.getSession(true);
// Get writer.
response.setContentType("text/html");
PrintWriter pw = response.getWriter();
pw.print("<B>");
// Display date/time of last access.
Date date = (Date)hs.getAttribute("date");
if(date != null) {
pw.print("Last access: " + date + "<br>");
}
// Display current date/time.
date = new Date();
hs.setAttribute("date", date);
pw.println("Current date: " + date);
}
}
```

When you first request this servlet, the browser displays one line with the current date and time information. On subsequent invocations, two lines are displayed. The first line shows the date and time when the servlet was last accessed. The second line shows the current date and time.

----------END OF UNIT – IV ----------

UNIT-V

Syllabus: **JSP Technology**: Advantages of JSP over Java Servlet, Architecture of a JSP Page, Life Cycle of a JSP Page, Working with JSP Basic Tags and Implicit Objects, Working with Action Tags in JSP, Exploring EL, Exploring the Elements of Tag Extensions, Tag Extension API, Working with Simple Tag Handlers Accessing Database from Servlet and JSP.

Introduction to JSP

Java Server Pages: It is a technology which is used to develop severside programs for generating dynamic content. It is an extension to servlet technology.

Drawbacks in servlets:

1) To develop a serverside program using servlets, it requires two categories of people. One of them is Web designers. Second are web developers.

2) For different display devices, different servlets are to be developed. i.e, because of writing presentation logic inside business logic.

i.e Servlet => HTML+Business Logic

In servlets, html tags are embedded in between business logic. This increases the complexity of development. And servlet programs are very lengthy.

So, to overcome these difficulty, JSP was introduced.

In JSP, data presentation was separated from business logic. i.e, in JSP we can embed business logic in between html tags. With respect to efficiency, JSP file will be served with the same efficiency as that of a servlet, because, a JSP file will be internally converted into a servlet and then the request will be processed.

JSP's Execution process(can be called as JSP life cycle)

It is 2 step process: 1) Translation and Compilation 2) Execution

This entire process requires two engines. 1) JSPEngine 2) ServletEngine

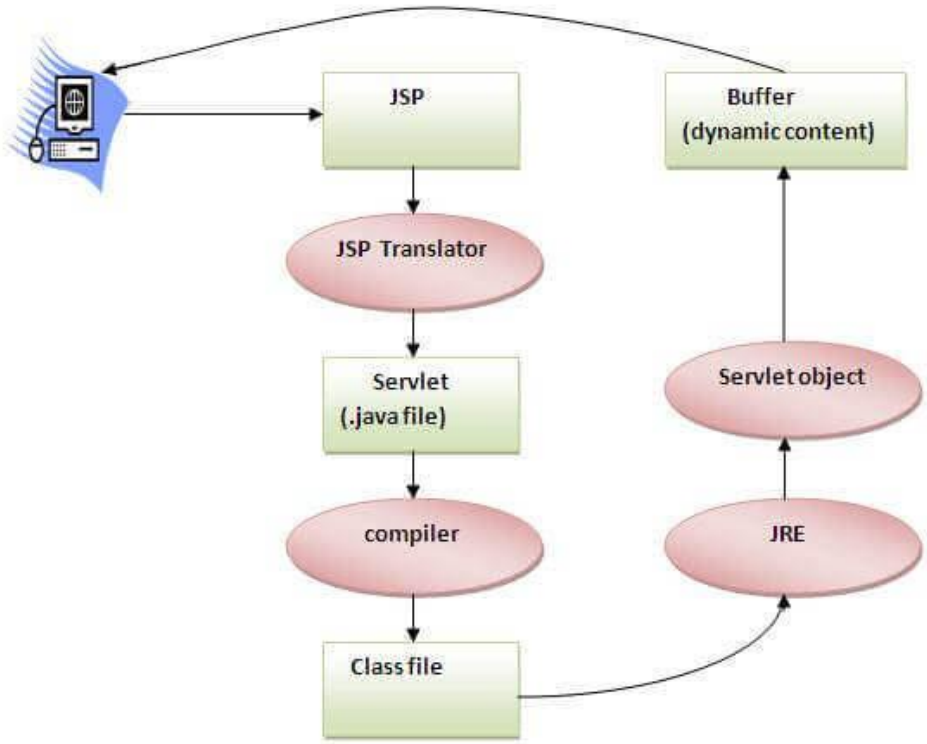


Fig: JSP Execution Process

Whenever a webserver receives a request for .jsp file, it will deligate the request to a virtual component called JSPEngine. This engine will process the .jsp file in a 2 step process.

1) Translation and Compilation

In this step JSPEngine translates a jsp file into a servlet based on the content of the jsp file. i.e, .java file (servlet) will be created for .jsp file. Then the created .java file is compiled into .class file by JSP engine.

2) Once the .class file is created, JSPEngine will forward it to ServletEngine to execute it. This .class file is served to client by ServletEngine according to the life cycle of servlet.

JSP tags

Four categories of JSP tags are available

1) Directives: page, include, taglib

2) Scripting Elements: declaration, expression, scriplet

3) Action tags: JSP:forward, JSP:include, JSP:usebean, JSP:plugin

4) Custom tags: user defined tags (open foe developers)

JSP Directive tags: These tags are used to define how to translate a JSP page into the corresponding servlet.

Syntax: <%@ directive attribute=”value” %>

There are three types of directives:

- page directive
- include directive
- taglib directive

page directive: This tag is used to define how exactly a servlet is created for the JSP

Syntax: <%@ page attribute="value", attribute="value", attribute="value", …… %>

This must be written/placed as the first line of any JSP program.

Attributes associated with page directive are

1) language="java" : with which business logic is defined
2) extends="name of the superclass"
3) import="packages" : external packages that are required.
   By default the following packages are imported into any JSP program
   javax.servlet.*
   javax.servlet.http.*
   javax.servlet.jsp.*
4) session="true/false" : enables implicit object for session tracking
5) buffer="n.o of kilo bytes used for file/None"
6) autoflush="true/false" : to flush data from buffer to client, default is true
7) isThreadSafe="false/true": to avoid a deadlock situation, default is false
8) errorPage=".jsp file" : this page is displayed when an error occurs, unchecked exception
9) isErrorPage="false/true" : it enables errorPage, default is false
10) contentType="text/html" :defines the MIMEtype of the HTTP response

Example:

```
<%@ page import="java.util.Date" language="java" isThreadSafe="true" %>
<html>
<body>
<p>A simple example on JSP page directive tag which displays current time and date</p>
Today is: <%= new Date() %>
</body>
</html>
```

Save the above code as example.jsp, and place this file in "C:\Program Files\Apache Software Foundation\Tomcat 9.0\webapps\New Folder"

……. New Folder

WEB-INF

example.jsp

Then, start the tomact server, open browser, type "localhost:8080" and press enter

Then click on Manager APP, enter username and password, then click signin, then select New Folder, then type localhost:8080/New Folder/example.jsp

include directive:

Syntax: <%@ include file="url" %>

It is used to include any external text based files(like .html, .txt). include directive will be executed at the time of translation.

taglib directive:

It is used to include user defined tags. All the tags created by using custom tags are user defined tags

Syntax: <%@ taglib uri="identifier" prefix="tag prefix" %>

JSP Scripting Elements: these are used to place some valid java code or business logic in the JSP program

3 types of scripting elements are available

1) declaration 2) expression 3) scriplet

1) declaration: It is used to declare a global variable in JSP program

It is used to declare user defined methods, and also used to override the methods

Syntax: <%! declaration %>

Ex: <%! int x, y; %>

<%! public void validate()

{

} %>

2) expression:The code placed within **JSP expression tag** is *written to the output stream of the response*.

Syntax: <%= expression %>

ex: <%= i %>

<%= (i+j)/x %>

<%= sum() %>

All the above expressions are evaluated at runtime and the output is written to the client

3) scriplet:A scriptlet tag is used to place java code in JSP

Syntax:

<%

```
-----
%>
```

Example:

```
<html><body>
<% for(int i=0; i<=10;i++)  { %>
  <p> i = <%= i %></p>
  <% } %>
</body></html>
```

JSP Action Tags:

JSP action tag is used to perform some specific tasks. They are used to control the flow between pages and to use Java Bean.

The following tags are available: 1) jsp:include 2) jsp:forward 3) jsp:usebean 4) jsp:setproperty

jsp:getproperty 6) jsp:plugin 7) jsp:param

1) Include action tag: It is used to include static dynamic content in the form of another JSP, html, servlet file. It is evaluated at execution time.

The difference between include directive and include action tags is, first one ie evaluated at execution time, so it is used to include dynamic content into JSP file. Second one is evaluated at translation time, so it is used to include only static content.

Syntax: <jsp:include page=".jsp/.html">……..</jsp:include>

2) Forward action tag: It is used to forward the request from one jsp to another jsp file.

Syntax: <jsp:forward page=".jsp">…….</jsp:forward>

It works like sendRedirect() of HttpServlet, the only difference is, sendRedirect() requires 2 pairs of request and response objects, whereas jsp:forward requires only one pair of request and response objects.

So network traffic is less when we use jsp:forward when compared to sendRedirect().

3) UseBean action tag:

it is used to separate the business logic from jsp file.

So jsp file allows to include java's class file developed by java beans. It is used to achieve code reusability.

A java bean is a simple java class which offers some properties with the help of getter and setter methods.

Syntax:

<jsp:useBean id="bean identifier" class="bean class name" scope="page/request/session/ application">

…….

</jsp:useBean>

Attributed and Usage of jsp:useBean action tag:

1. **id:** is used to identify the bean in the specified scope.
2. **class:** instantiates the specified bean class (i.e. creates an object of the bean class). It should not contain constructor and must not be abstract.
3. **scope:** represents the scope of the bean. It may be page, request, session or application. The default scope is page.
    - **page:** specifies that you can use this bean within the JSP page. The default scope is page.
    - **request:** specifies that you can use this bean from any JSP page that processes the same request. It has wider scope than page.
    - **session:** specifies that you can use this bean from any JSP page in the same session whether processes the same request or not. It has wider scope than request.
    - **application:** specifies that you can use this bean from any JSP page in the same application. It has wider scope than session.

We use jsp:setProterty action tag for calling the setter methods of java bean

We use jsp:getProperty action tag to call the getter methods of java bean

4) Plugin action tag:

It is used to embed applet in the jsp file. The jsp:plugin action tag downloads plugin at client side to execute an applet or bean.

Syntax: **<jsp:plugin** type= "applet | bean" code= "name of the class file"
codebase= "directory name that class file contains"
</jsp:plugin>

Attributes of jsp:plugin tag are
name, code, codebase, type, width, height, align, vspace, hspace, pluginurl, nspluginurl.
type, code and codebase are mandatory attributes, remaining are optional attributes.

JSP Custom tags

These are used to separate business logic completely from jsp file.

In a normal jsp file, we can place any business logic, because of this jsp file looks like a java file. To eliminate this we can use jsp:useBean action tag. Upto some extent it is capable of separating business logic from jsp file. In case if you want to remove the business logic from jsp file, we have to implement custom tags.

To work with custom tags, we have to create the following:

1) Tag Library Descriptor
2) Tag Handler class
3) Web.xml and jsp file

1) Tag Library Descriptor (TLD): It is an XML file in which we describe custom tags. i.e, in this file we are going to specify the name of the custom tag, content and attributes. This TLD can be used to describe any number of custom tags.

To create TLD, the following syntax is used:

```
<taglib>
<tlib-version>1.0</tlib-version>
<jsp-version>1.2</jsp-version>
<short-name>sample</short-name>
<uri>http://.....</uri>
<tag>
<name>userdefined name</name>
<tag-class>class name</tag-class>
</tag>
</taglib>
```

Once we create a TLD file, it must be described in the web.xml file by using the element <taglib>. while describing the TLD, we have to specify a unique URI and the location of corresponding TLD file.

2) Tag handler class: It is used to implement the business logic that is associated with the custom tag.

To create tag handler class, we have to follow the following procedure:

To create the Tag Handler, we are inheriting the **TagSupport class** and overriding itsmethod**doStartTag()**.To write data for the jsp, we need to use the **JspWriter class**.

The **PageContext** class provides **getOut()** method that returns the instance of JspWriter class. TagSupport class provides instance of pageContext bydefault.

JSP Implicit Objects

There are **9 jsp implicit objects**. These objects are *created by the web container* that are available to all the jsp pages. These objects are references to interfaces and classes that are available in servlets.

Object	Reference	Description
request	HttpServletRequest	This object is created for each jsp request by the web container. It can be used to get request information such as parameter, header information, remote address, server name, server port, content type, character encoding etc.
response	HttpServletResponse	It is created by the web container for each jsp request, used to add or manipulate response such as redirect response to another resource, send error etc.
out	javax.servlet.jsp.JspWriter	It is used to write any data to the buffer
session	HttpSession	this object is used to set,get or remove attribute or to get session information.
application	ServletContext	This object is used to get initialization parameter from configuration file (web.xml). It can also be used to get, set or remove attribute from the

		application scope.
page	this	This object is assigned to the reference of auto generated servlet class
exception	java.lang.Exception	This object is used to print the exception. But it can only be used in error pages
config	ServletConfig	This object can be used to get initialization parameter for a particular JSP page. The config object is created by the web container for each jsp page.Generally, it is used to get initialization parameter from the web.xml file.
pageContext	javax.servlet.jsp.PageContext	The pageContext object can be used to set,get or remove attribute from one of the following scopes: 1) page 2) request 3) session 4) application In JSP, page scope is the default scope.

Session Management:

In Java Sever Pages, the state information of a client is maintained by session object. In JSP, session is an implicit object of type HttpSession.The Java developer can use this object to set,get or remove attribute or to get session information.

Syntax to create session object: session.setAttribute("user",name); // user is the name of the session object

Syntax to retrieve session object: String name=(String)session.getAttribute("user");

Cookies

Cookies are text files stored on the client computer and they are kept for various information tracking purposes. JSP transparently supports HTTP cookies using underlying servlet technology.

There are three steps involved in identifying and returning users −

- Server sends a set of cookies to the browser. It contains name, age, or identification number, etc.
- Browser stores this information on the local machine for future use.
- When the next time the browser sends any request to the web server then it sends those cookies information to the server and server uses that information to identify the user or may be for some other purpose as well.

Cookies are usually set in an HTTP header. A JSP script will then have access to the cookies through the request method ***request.getCookies()*** which returns an array of *Cookie* objects.

Setting Cookies with JSP

It involves three steps −

Step 1: Creating a Cookie object

You call the Cookie constructor with a cookie name and a cookie value, both of which are strings.

Syntax:Cookie cookie = new Cookie("key","value");

Keep in mind, neither the name nor the value should contain white space or any of the following characters −[ ] ( ) = , " / ? @ : ;

Step 2: Setting the maximum age

You use setMaxAge to specify how long (in seconds) the cookie should be valid. The following code will set up a cookie for 24 hours.

Syntax: cookie.setMaxAge(60*60*24);

Step 3: Sending the Cookie into the HTTP response headers

You use response.addCookie to add cookies in the HTTP response header as follows

Syntax:response.addCookie(cookie);

Reading Cookies with JSP

To read cookies, you need to create an array of *javax.servlet.http.Cookie* objects by calling the **getCookies( )** method of *HttpServletRequest*. Then cycle through the array, and use **getName()** and **getValue()** methods to access each cookie and associated value.

Delete Cookies with JSP

To delete cookies is very simple. If you want to delete a cookie, then you simply need to follow these three steps −

- Read an already existing cookie and store it in Cookie object.
- Set cookie age as zero using the **setMaxAge()** method to delete an existing cookie.
- Add this cookie back into the response header.

----------END OF UNIT – V ----------

QUESTION BANK

<table>
<tr><th colspan="4">UNIT-1 (HTML5)</th></tr>
<tr><th colspan="4">PART-A: SHORT ANSWER QUESTIONS</th></tr>
<tr><th>S.NO</th><th>QUESTION</th><th>Blooms Taxonomy Level</th><th>Course Outcome</th></tr>
<tr><td>1</td><td>Write HTML code to create a link for other section within the same page.</td><td>L2</td><td></td></tr>
<tr><td>2</td><td>List out the differences between html and xml.</td><td>L2</td><td></td></tr>
<tr><td>3</td><td>What is URI? Write its format.</td><td>L1</td><td></td></tr>
<tr><td>4</td><td>What is the format of URL?Illustrate.</td><td>L1</td><td></td></tr>
<tr><td>5</td><td>Illustrate a) WWW b)URL c)HTTP d) Web Browser</td><td>L1</td><td></td></tr>
<tr><td>6</td><td>What is CSS? How many ways are there to include styles into a WebPage.</td><td>L2</td><td></td></tr>
<tr><td>7</td><td>Define CSS. Illustrate with an example the concept of external styling in HTML5.</td><td>L2</td><td></td></tr>
<tr><td>8</td><td>How many types of lists are supported by HTML5. Explain</td><td>L1</td><td></td></tr>
<tr><td>9</td><td>How many ways are there to embedded scripts into HTML5 code. Explain.</td><td>L2</td><td></td></tr>
<tr><th colspan="4">PART-B: LONG ANSWER QUESTIONS</th></tr>
<tr><th>S.NO</th><th>QUESTION</th><th>Blooms Taxonomy Level</th><th>Course Outcome</th></tr>
<tr><td>1</td><td>Write short note on the following tags.< img >, < table >, <a>, < pre ></td><td>L1</td><td></td></tr>
<tr><td>2</td><td>Demonstrate with a sample code to include different multimedia content and external files into a WebPage using HTML5.</td><td>L3</td><td></td></tr>
<tr><td>3</td><td>Create a XHTML registration form to accept the details of a student : Name,Address (Street Name, Country and Pincode), Sex (Male/Femal), Branch(chosen from a list box) and Courses (Check box). Provide submit and resetbuttons on it.</td><td>L3</td><td></td></tr>
<tr><td>4</td><td>Create a form to accept the details of a student : Name, Address, Sex(Male/Female), Electives (Check box), and Branch (chosen from a list box).Provide submit and Reset buttons on it.</td><td>L3</td><td></td></tr>
<tr><td>5</td><td>Write short notes on WWW?</td><td>L1</td><td></td></tr>
<tr><td>6</td><td>Write down the HTML5 code for creating the below table specification and insert two students information.
<table>
<tr><th colspan="6">Students Information</th></tr>
<tr><td>Roll No.</td><td>Name of the Student</td><td>Branch</td><td>Class</td><td>Course</td><td>Email ID</td></tr>
</table></td><td>L3</td><td></td></tr>
<tr><th colspan="4">UNIT-2(JAVASCRIPT)</th></tr>
<tr><th colspan="4">PART-A: SHORT ANSWER QUESTIONS</th></tr>
<tr><th>S.NO</th><th>QUESTION</th><th>Blooms Taxonomy Level</th><th>Course Outcome</th></tr>
<tr><td>1</td><td>Write a JavaScript code for ONFOCUS event.</td><td>L2</td><td></td></tr>
<tr><td>2</td><td>List out various Data types supported in python.</td><td>L2</td><td></td></tr>
</table>

3	Write JavaScript function to validate email-id.	**L3**	
4	Write JavaScript function to validate phone number.	**L3**	
5	Define AJAX and its importance.	**L1**	
6	What are the data formats supported by AJAX. Explain.	**L1**	
PART-B: LONG ANSWER QUESTIONS			
S.NO	**QUESTION**	**Blooms Taxonomy Level**	**Course Outcome**
1	Write a JavaScript code to find sum of n even numbers? Read the value of n from user?	**L3**	
2	Write JavaScript code to validate the form fields username (alphabets onlymaximum 10 characters), password and retype password (must match withpassword).	**L3**	
3	Write JavaScript code to validate the form fields username, password andemail id.	**L3**	
4	Write a JavaScript program to validate the following form fields *Username(Must start with a Capital Letter, alphabets only)* *Password(Must contain one Special character and number)* *Email(universally accepted email pattern)* *Mobile(must contain exactly 10 digits)*	**L3**	
5	What is AJAX? List out various examples with AJAX?	**L2**	
6	Difference between JavaScript and Ajax?	**L2**	
7	What are the key features of AJAX?	**L2**	
UNIT-3(XML)			
PART-A: SHORT ANSWER QUESTIONS			
S.NO	**QUESTION**	**Blooms Taxonomy Level**	**Course Outcome**
1	Explain the purposes of XML Processors.	**L2**	
2	What are the advantages of XML schema over DTD's?	**L2**	
3	Differentiate DOM and SAX parses.	**L2**	
4	Explain DTD. What is the use of DTD. How to link an external DTD into and XML Document.	**L2**	
5	What are the different elements we see in an XML document	**L2**	
6	Differentiate DTD and Schema.	**L2**	
PART-B: LONG ANSWER QUESTIONS			
S.NO	**QUESTION**	**Blooms Taxonomy Level**	**Course Outcome**
1	What is the use of XML namespaces? Explain in detail with an example?	**L2**	
2	Why XSLT is important for XML?	**L2**	
3	Write short notes on XML parsers	**L1**	
4	*University(department+,instructor+)* *department(dept_name,building,budget,course+)* *course(course_id,title,credits)* *instructor(IID,name,dept_name,salary,course_id)* A) Create an XML document for the above specification. **B)** Define the XML Schema for the above XML document	**L3**	

5	Illustrate the concept of XML Schema? How XML is different from XML DTD. Explain?	**L2**	
6	Write a short note on DOM and SAX.	**L2**	
7	How to validate any XML document. Explain.	**L3**	
UNIT-4(SERVLET)			
PART-A: SHORT ANSWER QUESTIONS			
S.NO	**QUESTION**	**Blooms Taxonomy Level**	**Course Outcome**
1	What is the difference between Servlet Context and Servlet Config objects?	**L2**	
2	What is persistent cookie?	**L2**	
3	How many servlet config objects are created for a web application?	**L1**	
4	Distinguish between Servlet and Filter.	**L2**	
5	What is Servlet chaining?	**L2**	
PART-B: LONG ANSWER QUESTIONS			
S.NO	**QUESTION**	**Blooms Taxonomy Level**	**Course Outcome**
1	Draw the life cycle of Servlet. Explain in detail steps involved in deploying a web application	**L2**	
2	Define a Session. Explain Different Session handling Mechanisms in web applications? What is session tracking? Write a program to track session using Http sessionobject.	**L3**	
3	Explain the functionality of javax.servlet.http package by discussing about classes, interfaces and methods of this package	**L2**	
4	What is a Container? Explain in detail steps involved in deploying a web application.	**L2**	
5	Explain in detail the mechanisms to secure a web application.	**L3**	
UNIT-5(JSP)			
PART-A: SHORT ANSWER QUESTIONS			
S.NO	**QUESTION**	**Blooms Taxonomy Level**	**Course Outcome**
1	List out JSP implicit Objects.	**L1**	
2	Differentiate JSP include and JSP forward action tags	**L2**	
3	Explain the anatomy of JSP page?	**L2**	
4	What is deployment description?	**L1**	
5	Explain the concept of Cookie.	**L2**	
6	Interpret in your own words about session tracking	**L3**	
7	What is JSPEngine. What is the use of it.	**L2**	
PART-B: LONG ANSWER QUESTIONS			
S.NO	**QUESTION**	**Blooms Taxonomy Level**	**Course Outcome**

1	Explain in detail different JSP elements.	**L2**	
2	Explain Different Types of JSP Directives in detail.	**L2**	
3	Explain about implicit objects in JSP?	**L2**	
4	Write a jsp program to accept two values from the user using forms and display the sum to the user.	**L3**	
5	Write a program to create iterative custom tag using Tag Extension.	**L3**	
7	Write a short note on the following JSP tags a)Directives b) Scripting Elements c) Action Tags d) Custom Tags	**L2**	
8	Explain in detail the life cycle methods of a Filter.	**L2**	

PREVIOUS UNIVERSITY QUESTION PAPERS

Code No.11591/CBCS

FACULTY OF ENGINEERING

B. E. VI – Semester (CBCS) (CSE) (Main) Examination, May / June 2019

Subject: Web Programming

Time: 3 Hours Max. Marks: 70

Note: Answer All Questions from Part-A, & Any Five Questions from Part – B

Part– A (2 x 10 = 20 Marks)

1) Define Web Server List out various Web Server operations.
2) Write HTML code to create a link for other section within the same page.
3) List outthe differences between HTML and XML.
4) Explain thepurposes of XML Processors.
5) Write a JavaScript code for ONFOCUS event.
6) List outvarious Data types supported in python.
7) What isthe difference between Servlet Context and Servlet Config objects?
8) List outJSPimplicit Objects.
9) Is multiple Inheritance is supported in PHP ? Justify your Answer.
10)What is persistent cookie?

Part – B (5 x 10 = 50 Marks)

11. a) What is HTTP and Explain Http Request and Response Formats? 5M
    b)Write short noteonthe following tags. < img >, < table >, < frameset >, < pre > 5M

12.a) What is the use ofXML namespaces? Explain in detail with an example? 5M
    b)Why XSLT is importantfor XML? 5M

13.a) Write ajava script code to find sum of n even numbers? Read the value of n from user? 6M
    b)What is AJAX? List out various security issues with AJAX? 4M

14.a) Definea Session.Explain Different Session handling Mechanisms in web applications. 6M
    b)Explain Different Types of JSP Directives in detail. 4M

15.a) What is Servlet chaining? 4M
    b)Write a jsp program to accept two values from the user using forms and display the sum to user? 6M

16.a) Develop PHPpage for validating username and password? 6M
    b)Explain about pattern matching in PHP. 4M

17.a) What are the key features of python? 3M
    b)Difference between JavaScript and Ajax? 3M
    c) Write a note on PHP files? 4M

****

Code No. 3477 / N

FACULTY OF ENGINEERING

B.E. 3/4 (CSE) II – Semester (New) (Main) Examination, May / June 2017

Subject: Web Programming and Services

Time: 3 Hours Max.Marks: 75

Note: Answer all questions from Part A and any five questions from Part B.

PART – A (25 Marks)

1 Mention the MIME format for PDF documents. 2M
2 Write Javascript code to validate range from 10-100. 3M
3 Differentiate between Servlet and CGI? 3M
4 What is Session? 2M
5 What is the purpose of deployment descriptor? 2M
6 List the implicit objects in JSP. 2M
7 Write the syntax of UseBean tag. 3M
8 Why type 4 driver referred to as thin driver? 3M
9 Distinguish between Statement and PreparedStatement object. 3M
10Name the server on which ASP pages are deployed. 2M

PART – B (5x10 = 50 Marks)

11. a)Write the XHTML program to create employee registration form with fieldsemployee id, name, gender (radio button), designation, DOB, email-id, skillset(check boxes), and provision to upload photo of the employee. WriteJavaScript code to validate employee name and email-id. 6M
    b)Write JavaScript code to greet the user based on time. 4M

12. a)Write a Servlet program to read the details from the form and store them indatabaseusing JDBC. 6M
    b)What is session tracking? How do we track user sessions with Cookies?Giveanexample. 4M

13. a)What is webserver? Explain different webservers and describe the sequenceof steps for deploying a web application. 6M
    b)Write a JSP page demonstrating the usage of JSP include and forward tag. 4M

14. a)Describe in detail the types of JDBC drivers. 5M
    b)Write a Java Mail program to read the details from the form and send a mail tothe user. 5M
15. a)Describe the trends in ASP. Write an ASP.Net program to validate fields anddisplay date. 6M
    b)Write a program demonstrating .Net Remoting. 4M

16. a)Explain different types of hyperlinks. Design a webpage to display chapters andchapter contents, apply hyperlink within a page to navigate to chapter contents. 6M
    b)Describe enterprise architectural styles. 4M
17. Write short notes on:
    a) Filter. 5M
    b) Connection Pooling. 5M

****