UNIT 1 – INTRODUCTION TO INTERNET, BROWSER ARCHITECTURE, HTML5 AND CSS

1.1 Introduction to the Internet

The Internet is a worldwide system of interconnected computer networks. These networks communicate using a common set of protocols, mainly the Internet Protocol (IP) suite. It connects personal computers, mobile devices, servers and many other network-enabled systems. Through the Internet, users can exchange messages, transfer files, attend online classes, access databases and use web applications.

The World Wide Web (WWW) is one of the services available through the Internet. It is a collection of interlinked documents and resources that are generally accessed using a web browser. The Internet is the underlying network, whereas the Web is an information service that operates on it.

Term	Meaning
Internet	Global network of interconnected networks.
World Wide Web	Collection of linked web pages and resources accessed over the Internet.
Website	A collection of related web pages and resources under a common domain.
Web page	A document displayed by a browser, usually written in HTML.
URL	Uniform Resource Locator; the address used to identify a resource.
Protocol	A set of rules that controls communication between devices.

1.2 Basic working of the Web

When a user enters a website address, the browser identifies the server that hosts the requested resource. Domain Name System (DNS) translates a human-readable domain name into an IP address. The browser then establishes a connection and sends an HTTP or HTTPS request. The server processes the request and returns a response. The browser interprets the received HTML, CSS, JavaScript and media files and displays the page.

1. The user enters a URL or selects a link.
2. The browser uses DNS to identify the server address.
3. The browser sends an HTTP/HTTPS request to the server.
4. The server locates or generates the requested resource.
5. The server sends an HTTP response with a status code and content.
6. The browser processes the response and renders the web page.

1.3 Browser architecture

A web browser is application software that retrieves web resources and presents them in a readable and interactive form. A browser does more than display text: it interprets markup, applies styles, executes scripts, manages user interaction and protects the user through security controls.

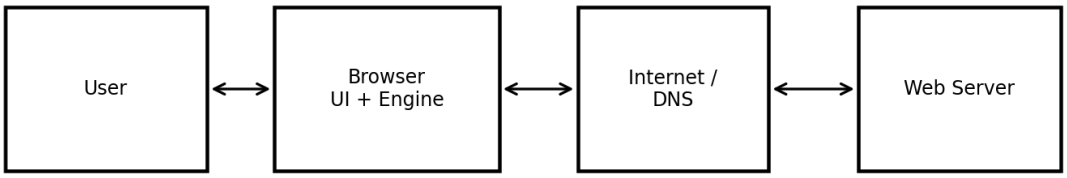


Figure 1.1: Simplified communication between a user, browser, network and web server.

Major components of a browser

Component	Function
User Interface	Provides the address bar, tabs, menus, bookmarks and navigation controls.
Browser Engine	Coordinates communication between the user interface and the rendering engine.
Rendering Engine	Parses HTML and CSS and displays the visual page. It constructs layout information and paints content.
JavaScript Engine	Executes JavaScript code and manages its runtime operations.
Networking	Handles network requests, responses, caching and communication protocols.
UI Backend	Draws basic interface elements such as windows, controls and text.
Data Storage	Stores cookies, cache, local storage and other site-related information.

The rendering engine converts HTML into a Document Object Model (DOM) tree and CSS into style information. It combines these structures to calculate layout and paint the page. JavaScript can modify the DOM, causing the browser to update the displayed content.

1.4 Internet Explorer and Google Chrome

Internet Explorer (IE) was Microsoft's web browser and was widely used on Windows computers for many years. It supported browsing, bookmarks, security zones and integration with Windows. It is now retired and is mainly relevant when studying the history of web browsers or maintaining older systems.

Google Chrome is a modern web browser developed by Google and based on the Chromium project. It supports current web standards and provides tabbed browsing, extensions, synchronisation, developer tools, private browsing and security features. Its developer tools help programmers inspect HTML, CSS, JavaScript, network requests and page performance.

Feature	Internet Explorer	Google Chrome
Status	Retired legacy browser	Actively maintained modern browser
Platform association	Historically integrated with Windows	Available across major desktop and mobile platforms
Developer support	Older development tools and legacy compatibility	Modern developer tools and web platform support

Current relevance	Legacy systems and browser history	Current web development and general browsing

1.5 Search engines

A search engine is a web-based information retrieval system that helps users locate information from websites. Users enter keywords or questions, and the search engine displays a list of relevant results. Search engines use automated systems to discover web pages, organise information and determine which results are most useful for a query.

Working stages of a search engine

- Crawling: automated programs, commonly called crawlers or bots, visit pages and follow links to discover content.
- Indexing: the search engine analyses page content and stores selected information in a large searchable index.
- Ranking: when a user searches, the system orders results using relevance, quality, freshness, location and other signals.
- Result presentation: the search engine displays titles, summaries, links, images or other useful result formats.

Search engines are useful for research, education, news, product discovery and locating services. Examples include Google, Microsoft Bing and DuckDuckGo. Search results may be organic or sponsored; users should evaluate the reliability, date and authority of information before using it.

1.6 Introduction to HTML5

HTML stands for HyperText Markup Language. It is the standard markup language used to describe the structure and meaning of web content. HTML uses elements represented by tags. A browser reads these elements and constructs a document structure that can be displayed to the user.

HTML5 is the modern generation of HTML that introduced semantic elements, native audio and video, improved form controls and several browser APIs. HTML is not a programming language: it describes the structure of a document rather than performing calculations or controlling application logic.

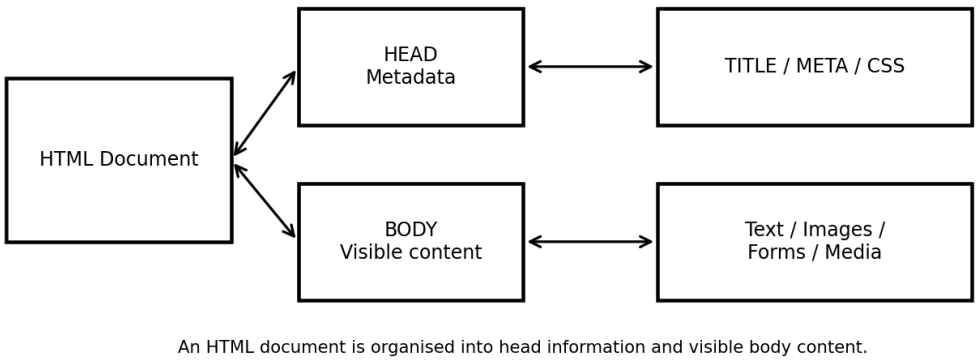


Figure 1.2: Basic organisation of an HTML document.

Structure of an HTML5 document

A standard HTML5 document begins with a doctype declaration. The html element is the root. The head section contains metadata, the page title and references to stylesheets or scripts. The body section contains the visible page content.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Department Website</title>
</head>
<body>
  <h1>Welcome to MCA</h1>
  <p>Introduction to Web Technologies.</p>
</body>
</html>
```

1.7 Important HTML5 tags and their uses

An HTML tag identifies the type of content or structure represented by an element. Most elements have an opening tag and a closing tag, with content placed between them. Some elements, such as img and input, are void elements and do not have closing tags.

Tag / Element	Purpose
h1 to h6	Represent headings from highest to lower levels.
p	Defines a paragraph.
a	Creates a hyperlink to another resource or location.
img	Embeds an image; alternative text should be provided using alt.
ul, ol, li	Create unordered lists, ordered lists and list items.
table, tr, th, td	Represent tabular information.
form, input, label	Create a form and its user-input controls.
div and span	Generic containers used for grouping and styling.
header and footer	Represent introductory and closing areas of a page or section.
nav	Identifies a group of navigation links.
main	Represents the main content of the page.
section and article	Group thematic content or self-contained content.
audio and video	Embed sound and video media.

Semantic HTML elements

Semantic elements communicate the meaning of content to browsers, developers and assistive technologies. They make a page easier to understand and maintain. For example, nav identifies navigation links, main identifies the central content, article represents an independent item and footer identifies closing information. Semantic markup can improve accessibility and support search engines in understanding page structure.

```
<header><h1>College Information</h1></header>
<nav>
  <a href="home.html">Home</a>
  <a href="courses.html">Courses</a>
</nav>
<main>
  <section>
    <h2>About the Department</h2>
```

```
    <p>The department offers MCA education.</p>
  </section>
</main>
<footer>Department Information</footer>
```

1.8 HTML5 audio and video tags

Earlier websites often depended on browser plug-ins to play multimedia. HTML5 introduced the audio and video elements, allowing browsers to play supported media formats directly. The controls attribute displays built-in controls such as play, pause, volume and progress.

The source element can be placed inside audio or video to provide one or more media files. The browser selects a format it can play. The poster attribute of video can display an image before playback. The loop attribute repeats playback, while muted starts media without sound.

Attribute	Purpose
src	Specifies the media file location.
controls	Displays the browser's playback controls.
autoplay	Requests automatic playback; browser policies may restrict it.
loop	Repeats the media after it finishes.
muted	Starts playback without sound.
poster	Specifies a preview image for a video.
width / height	Sets the displayed dimensions of a video.

```
<audio controls>
  <source src="lecture.mp3" type="audio/mpeg">
  Audio is not supported by this browser.
</audio>

<video width="480" controls poster="preview.jpg">
  <source src="lesson.mp4" type="video/mp4">
  Video is not supported by this browser.
</video>
```

1.9 HTML5 forms and controls

An HTML form is used to collect information from a user and submit it for processing. Forms are important in registration pages, login screens, feedback systems, online applications and search facilities. The form element groups the controls and defines how the information is submitted.

The action attribute specifies the server-side resource that receives the submitted data. The method attribute specifies the HTTP method, usually GET or POST. Each form control should have a meaningful name because the name is used to identify its submitted value.

Common form controls

Control	Purpose / Example
text	Accepts a short text value such as a student's name.
password	Accepts a password; the browser masks the entered characters.
email	Accepts an email address and provides basic format validation.
number	Accepts a numeric value, with optional min and max.

date	Provides a date input control.
radio	Allows one selection from a related group.
checkbox	Allows independent yes/no or multiple selections.
file	Allows a user to select a file for upload.
select	Provides a drop-down list of options.
textarea	Accepts multi-line text.
submit / reset	Submits the form or resets controls to their initial values.

HTML5 validation attributes include required, minlength, maxlength, min, max, step and pattern. These attributes improve the user experience, but server-side validation is still necessary because browser validation can be bypassed.

GET and POST methods

GET	POST
Data is generally appended to the URL as a query string.	Data is sent in the request body.
Useful for searches and retrieving information.	Useful for submitting data or requesting a change.
Can be bookmarked and may be stored in browser history.	Not normally represented as a bookmarkable query URL.
Should not be used for sensitive information.	Does not itself encrypt information; HTTPS is required.

1.10 CSS styling

CSS stands for Cascading Style Sheets. It is used to control the appearance and layout of HTML documents. CSS can define colours, fonts, borders, spacing, alignment, backgrounds, responsive layouts and visual effects. By separating content from presentation, CSS makes a website easier to maintain and gives multiple pages a consistent appearance.

CSS rule and syntax

A CSS rule consists of a selector and a declaration block. The selector identifies the HTML elements to be styled. Each declaration contains a property and a value.

```
h1 {
  color: navy;
  font-size: 28px;
  text-align: center;
}
```

Ways of applying CSS

- Inline CSS: written in the style attribute of an individual HTML element. It is useful for a small one-off style but can make maintenance difficult.
- Internal CSS: written inside a style element in the document head. It is useful when styles apply to a single page.
- External CSS: written in a separate .css file and connected using a link element. It is generally preferred for multi-page websites because it supports reuse and consistency.

CSS selectors

Selector	Example	Meaning
Element	p { color: blue; }	Selects all paragraph elements.
Class	.notice { padding: 10px; }	Selects elements having class notice.
ID	#header { height: 80px; }	Selects the element with the specified ID.

Universal	* { box-sizing: border-box; }	Selects all elements.
Grouping	h1, h2 { color: navy; }	Applies the same rule to several selectors.
Descendant	.menu a { text-decoration: none; }	Selects links contained within an element having class menu.

CSS properties and attributes commonly used

Property group	Examples	Purpose
Text and font	color, font-family, font-size, font-weight, line-height	Controls text appearance.
Background	background-color, background-image	Sets the background of an element.
Spacing	margin, padding	Controls outside and inside spacing.
Border	border, border-radius	Defines an element boundary and rounded corners.
Dimensions	width, height, max-width	Controls element size.
Layout	display, position, flex, grid, gap	Controls placement and page structure.
Visual effects	box-shadow, opacity, transform	Adds visual effects and transformations.

CSS box model

The CSS box model explains how the size of an element is calculated. Every element is treated as a rectangular box consisting of content, padding, border and margin. Content is the text, image or other material inside the element. Padding creates space between content and border. Border surrounds the padding. Margin creates space outside the border.

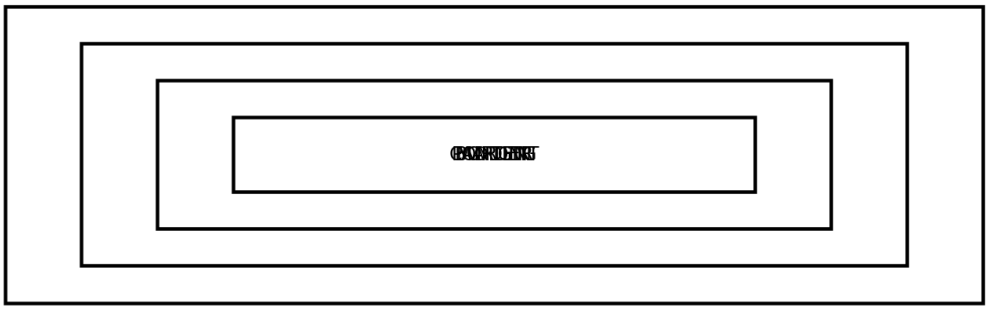


Figure 1.3: CSS box model showing the nested spacing areas.

The box-sizing property affects how width and height are calculated. With content-box, the declared width applies to the content only. With border-box, the declared width includes content, padding and border, which often makes layout calculations easier.

Responsive design and media queries

Responsive web design allows a website to adapt to different screen sizes such as desktops, tablets and mobile phones. Flexible widths, responsive images, CSS Flexbox, CSS Grid and media queries are common techniques. A media query applies selected styles only when a condition, such as a maximum screen width, is met.

```
/* Example of a small responsive layout */
.card { width: 50%; padding: 16px; }
@media (max-width: 600px) {
  .card { width: 100%; }
```

```
}
```

2.1 Introduction to JavaScript

JavaScript is a high-level, interpreted scripting language that is widely used in web browsers. It works with HTML and CSS to create dynamic and interactive web pages. HTML defines the structure, CSS defines the presentation and JavaScript controls behaviour.

JavaScript can be written inside a script element or stored in an external file with a .js extension. External files are useful because the same script can be reused on multiple pages. Modern JavaScript also runs outside browsers in server-side environments.

Features and applications of JavaScript

- It is event-driven and can respond to clicks, keyboard actions and form submissions.
- It can access and modify the Document Object Model (DOM).
- It supports functions, objects, arrays and modern programming structures.
- It can validate user input before sending data to a server.
- It can exchange information with a server using AJAX techniques.
- It is used in interactive forms, menus, games, dashboards and web applications.

2.2 JavaScript programming scripts and basic syntax

A JavaScript statement is an instruction executed by the JavaScript engine. Statements may declare variables, call functions, evaluate expressions or control program flow. JavaScript is case-sensitive. Comments are used to explain code and are ignored during execution.

Feature	Description
Variable	A named storage location for a value.
Expression	A combination of values, variables and operators that produces a result.
Statement	An instruction executed by the program.
Comment	A note for developers; single-line comments use // and multi-line comments use /* */.
Block	A group of statements enclosed in curly braces.

Variables and data types

JavaScript provides let, const and var for declaring variables. let is suitable for values that may change. const is used when a variable should not be reassigned. var is found in older programs and has function-level scope. Common data types include string, number, boolean, undefined, null, object, symbol and bigint.

JavaScript is dynamically typed, meaning that a variable can hold values of different types at different times. The typeof operator can be used to examine the type of a value, although null is a historical special case for which typeof returns object.

Operators

Operator category	Examples	Purpose
Arithmetic	+ - * / % **	Performs mathematical calculations.
Assignment	= += -= *= /=	Assigns or updates values.
Comparison	=== !== > < >= <=	Compares values; strict equality checks

		both value and type.
Logical	&& || !	Combines or reverses Boolean conditions.
String	+, template literals	Combines or formats text.
Conditional	condition ? a : b	Selects one of two expressions.

2.3 Control structures

Control structures determine the order in which statements are executed. They allow a program to make decisions and repeat operations. The three broad categories are sequence, selection and iteration.

Selection statements

The if statement executes a block when a condition is true. The if–else statement selects between two alternatives. An else-if ladder checks multiple conditions in order. The switch statement selects one branch from several possible cases.

Iteration statements

Loops repeat a set of statements. A for loop is commonly used when the number of repetitions is known. A while loop repeats while a condition remains true. A do–while loop executes the body at least once before checking the condition. The break statement exits a loop, while continue skips the remaining statements in the current iteration.

```
// Example: result classification
let marks = 76;
if (marks >= 75) {
  console.log("Distinction");
} else if (marks >= 40) {
  console.log("Pass");
} else {
  console.log("Fail");
}

// Example: repetition
for (let i = 1; i <= 5; i++) {
  console.log(i);
}
```

2.4 Functions

A function is a reusable block of statements designed to perform a particular task. Functions help divide a large program into smaller logical parts. A function may accept input values called parameters and may return a result using the return statement.

JavaScript supports function declarations, function expressions and arrow functions. A function can be called from another function or in response to an event. Local variables declared inside a function are generally available only within that function's scope.

```
function calculateTotal(a, b) {
  return a + b;
}
let total = calculateTotal(25, 35);
console.log("Total = " + total);
```

2.5 Document object

The Document object represents the HTML document loaded in the browser. It is part of the Document Object Model (DOM). The DOM represents the page as a tree of objects, where elements, attributes and text can be accessed and modified by JavaScript.

Common document methods include getElementById(), getElementsByClassName(), getElementsByTagName(), querySelector(), querySelectorAll(), createElement() and write(). Modern applications generally use DOM methods such as textContent, innerHTML, setAttribute() and appendChild() carefully to update the page.

- getElementById() selects an element using its ID.
- querySelector() returns the first element matching a CSS selector.
- createElement() creates a new HTML element.
- appendChild() adds a node to another node.
- textContent reads or changes text without interpreting it as HTML.

2.6 Browser objects

The Browser Object Model (BOM) provides objects that allow JavaScript to interact with the browser environment. The window object is the top-level object in browser-side JavaScript. Its properties and related objects provide information about the browser window, current page, browsing history and device.

Object	Purpose
window	Represents the browser window and provides methods such as alert(), confirm() and setTimeout().
location	Provides information about the current URL and can navigate to another address.
history	Provides limited access to the browser's session history.
navigator	Provides information about the browser and its environment.
screen	Provides information about the user's display screen.

2.7 Date object

The Date object is used to work with dates and times. A Date instance can represent the current date and time or a specified date. Common methods retrieve the day, month, year, hours and minutes. JavaScript months are numbered from 0 to 11, so January is 0 and December is 11.

```
let today = new Date();
console.log(today.getDate());
console.log(today.getMonth() + 1);
console.log(today.getFullYear());
```

2.8 Math object

The Math object provides mathematical constants and functions. It does not need to be created with new. Common methods include Math.sqrt() for square roots, Math.pow() for powers, Math.round() for rounding, Math.floor() for rounding down, Math.ceil() for rounding up and Math.random() for generating a pseudo-random number between 0 inclusive and 1 exclusive.

2.9 String object

A string represents a sequence of characters. JavaScript strings are immutable, which means string methods return new strings rather than changing the original string. Useful properties and methods include length, charAt(), indexOf(), includes(), substring(), slice(), replace(), toUpperCase(), toLowerCase() and trim().

2.10 JavaScript events

An event is an action or occurrence that the browser can detect. Events may be triggered by the user, by the browser or by changes in the document. JavaScript event handling allows a program to respond to these actions and provide interactive behaviour.

Event	Description
click	Occurs when an element is activated by a click.
dblclick	Occurs when an element is double-clicked.
mouseover / mouseout	Occurs when the pointer enters or leaves an element.
keydown / keyup	Occurs when a keyboard key is pressed or released.
input	Occurs when the value of an input control changes.
change	Occurs when a control's value is committed or changed.
submit	Occurs when a form is submitted.
load	Occurs when a page or resource has finished loading.

Event listeners are commonly attached using addEventListener(). This approach separates the event-handling code from HTML markup and allows multiple listeners to be attached to an element.

2.11 Introduction to jQuery

jQuery is a JavaScript library designed to make common browser scripting tasks shorter and more consistent. It provides methods for selecting elements, changing content, manipulating CSS, handling events, performing animations and sending AJAX requests. The jQuery function is commonly represented by the dollar sign ($).

A jQuery library file must be loaded before the code that uses it. The document-ready function ensures that the HTML document has been parsed before the script tries to access its elements. In modern JavaScript, the equivalent DOMContentLoaded event is also widely used.

jQuery objects and selectors

A jQuery selector identifies one or more elements in the page. The result is a jQuery object, which provides methods that can operate on the selected elements. This is different from a plain DOM element, although jQuery provides ways to access the underlying DOM element.

Selector	Meaning
$('p')	Selects all paragraph elements.
$('#notice')	Selects the element with ID notice.
$('.item')	Selects all elements with class item.
$('input[type=text]')	Selects text input controls.
$('ul li')	Selects list items contained inside unordered lists.

jQuery functions and common methods

Method	Use
text()	Gets or sets text content.
html()	Gets or sets HTML content.
val()	Gets or sets the value of a form control.
css()	Gets or sets CSS properties.

addClass() / removeClass()	Adds or removes a CSS class.
append()	Adds content at the end of selected elements.
remove()	Removes selected elements.
attr()	Gets or sets an attribute.

2.12 jQuery events

jQuery provides event methods such as click(), change(), submit(), keyup() and on(). The on() method is flexible and is commonly preferred for attaching event handlers. Event handlers execute when the selected event occurs.

2.13 jQuery animations

jQuery includes simple effects that can change the visibility or appearance of elements. hide() and show() change visibility, toggle() switches between them, fadeIn() and fadeOut() change opacity, and slideUp() and slideDown() create sliding effects. animate() can change selected numeric CSS properties over a period of time.

```
$(document).ready(function () {
  $("#hideButton").on("click", function () {
    $("#message").slideUp(400);
  });
  $("#showButton").on("click", function () {
    $("#message").slideDown(400);
  });
});
```

UNIT

UNIT 3 – AJAX CONCEPTS, WEB SERVERS AND WEBSITE HOSTING

3.1 Introduction to AJAX

AJAX stands for Asynchronous JavaScript and XML. It is a web development technique that allows a browser to communicate with a server in the background and update part of a web page without reloading the entire page. AJAX is not a separate programming language. It combines JavaScript, browser request objects and server-side processing.

Although the name contains XML, modern AJAX applications commonly exchange data in JSON format because JSON is compact and maps naturally to JavaScript objects. HTML, plain text and XML can also be used as response formats.

Need and advantages of AJAX

- Improves user experience by avoiding a full page reload for every small operation.
- Allows the browser to request only the information it needs.
- Can make interfaces more responsive, for example in live search and form validation.
- Reduces unnecessary transfer of page content when only a small data update is required.
- Supports interactive dashboards, notifications and dynamic lists.

3.2 Working of AJAX

When an event occurs, JavaScript creates an asynchronous request and sends it to the server. The server processes the request and sends back a response. JavaScript receives the response, interprets the data and updates the relevant part of the DOM. The user can continue interacting with the page while the request is being processed.

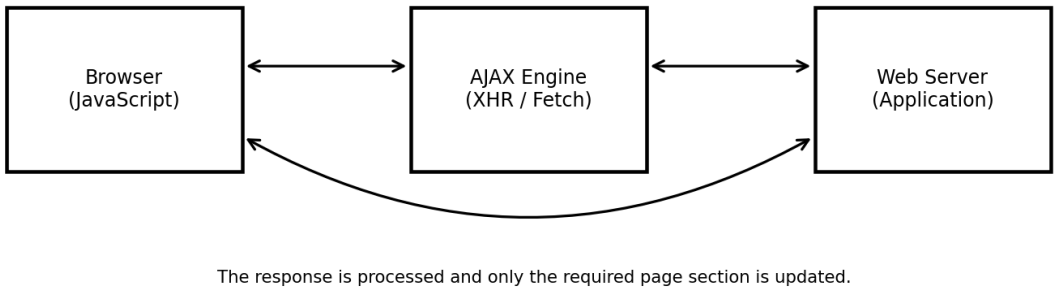


Figure 3.1: Simplified AJAX communication between the browser and web server.

7. The user performs an action, such as clicking a button.
8. JavaScript creates a request using XMLHttpRequest or Fetch.
9. The request is sent to a server-side resource.
10. The server processes the request and prepares a response.
11. The browser receives the response asynchronously.

12. JavaScript updates the required section of the page.

3.3 XMLHttpRequest object

The XMLHttpRequest (XHR) object is a browser API used to send HTTP requests and receive responses. It can work asynchronously, meaning that the browser does not need to stop all other activity while waiting for the server. Important properties include readyState, status, responseText and responseXML. The onload and onerror event handlers can be used to process success and failure conditions.

Property / Method	Description
open(method, url, async)	Initialises a request with an HTTP method, URL and asynchronous setting.
send()	Sends the request to the server.
readyState	Indicates the current state of the request.
status	Contains the HTTP status code, such as 200 or 404.
responseText	Contains the response as text.
onload	Runs when the request completes successfully at the network level.
onerror	Runs when a network-level error occurs.

Simple AJAX example using XHR

```
const xhr = new XMLHttpRequest();
xhr.open("GET", "message.txt", true);
xhr.onload = function () {
  if (xhr.status === 200) {
    document.getElementById("result").textContent =
      xhr.responseText;
  }
};
xhr.send();
```

The example assumes that message.txt is available from the same web server and that the page is being served through HTTP or HTTPS. In a complete application, error handling and loading indicators should also be included.

3.4 AJAX libraries and modern request methods

Libraries can reduce the amount of code required for AJAX operations. jQuery provides the $.ajax() method with options for the URL, request type, expected data type and success or error callbacks. Modern JavaScript provides the Fetch API, which uses Promises and is commonly used in new applications.

Method	Characteristics
XMLHttpRequest	Traditional browser API; uses properties and event handlers to monitor request progress.
jQuery $.ajax()	Library method that simplifies request configuration and callback handling.
Fetch API	Modern browser API that returns a Promise and supports readable response-processing steps.

Fetch example

```
fetch("student.json")
  .then(response => {
    if (!response.ok) throw new Error("Request failed");
```

```
  return response.json();
})
.then(data => {
  document.getElementById("result").textContent = data.name;
})
.catch(error => console.log(error));
```

A Promise represents the eventual completion or failure of an asynchronous operation. The then() method processes a successful result, while catch() handles errors. The response.json() method reads the response body and converts valid JSON into a JavaScript value.

3.5 AJAX applications

- Live search: suggestions are displayed while the user types.
- Form validation: a server checks whether a username or other value is already registered.
- Online shopping: cart quantities and totals can be updated without reloading the page.
- Dashboards: selected figures can be refreshed in the background.
- Chat and notification interfaces: new information can be retrieved periodically or through real-time communication technologies.

3.6 Web servers

A web server is software that receives requests from clients, usually through HTTP or HTTPS, and returns responses. A client is commonly a browser or mobile application. A web server may deliver static resources such as HTML, CSS, images and JavaScript files. It may also forward dynamic requests to application software that generates content.

A server can perform several tasks, including request handling, authentication, access control, logging, connection management, caching and secure communication. A web server is not the same as a physical server computer: the term web server commonly refers to the software, although it can also describe the computer running that software.

3.7 Internet Information Services (IIS)

Internet Information Services (IIS) is Microsoft's web server platform for Windows. It is used to host websites, web applications and services. IIS supports HTTP and HTTPS, site configuration, application pools, authentication, logging, request filtering and integration with Microsoft technologies such as ASP.NET.

An application pool provides an isolated environment for one or more web applications. Isolation can improve reliability because problems in one application pool need not affect every other site hosted by the server. IIS Manager provides a graphical interface for configuring websites, bindings, application pools and security settings.

3.8 Apache Tomcat

Apache Tomcat is an open-source implementation of Java Servlet and JavaServer Pages technologies. It acts as a servlet container and provides a runtime environment for Java web applications. Tomcat receives HTTP requests, maps them to the appropriate web application and manages servlet execution.

Tomcat is commonly used in Java web development and education because it is relatively lightweight and supports deployment of web applications packaged as WAR files. It is not a general replacement for every feature of a full enterprise application server, but it is well suited to Servlet and JSP applications.

Basis	IIS	Apache Tomcat
Provider	Microsoft	Apache Software Foundation
Typical environment	Windows Server	Java runtime on supported operating systems
Main role	General web server and application hosting platform	Servlet container for Java web applications
Common technologies	Static sites, ASP.NET and other configured applications	Java Servlets and JSP
Administration	IIS Manager and configuration files	Configuration files, management tools and deployment folders

3.9 Hosting a website on a web server

Website hosting is the process of placing website files or an application on a server so that users can access it through a network. Hosting may be local for development, within an organisation's network, or publicly available through the Internet. The hosting environment must support the technologies used by the website.

General steps in website hosting

13. Prepare the website files and test them in a development environment.
14. Select a server and hosting plan that supports the required technologies.
15. Configure the web server, site directory or application deployment location.
16. Upload static files or deploy the application package.
17. Configure the domain name and DNS records when using a public domain.
18. Configure HTTPS, access permissions and application settings.
19. Test pages, forms, database operations and links from a browser.
20. Monitor logs, update software and maintain backups.

Important hosting terms

Term	Explanation
Domain name	Human-readable name used to access a website.
DNS	System that maps domain names to network addresses.
Document root	Directory from which the web server serves website files.
Virtual host / site binding	Configuration that allows a server to host sites using names, addresses or ports.
HTTPS	Secure HTTP communication protected using TLS.
Deployment	The process of placing an application into its target runtime environment.
Server log	Record of requests, errors and other server events.

Technology requirement: A static HTML/CSS/JavaScript site needs a web server for normal public hosting. PHP applications need a PHP-enabled environment. Java Servlet and JSP applications need a compatible servlet container such as Tomcat.

UNIT 4 – INTRODUCTION TO PHP, DATABASE CONNECTIVITY AND ZEND FRAMEWORK

4.1 Introduction to PHP

PHP stands for PHP: Hypertext Preprocessor. It is a widely used server-side scripting language designed for web development. PHP code is executed on the web server. The server processes the code and sends the generated output, commonly HTML or JSON, to the browser. The browser does not normally receive the PHP source code.

PHP can be embedded within HTML and is commonly used for login systems, student portals, content management systems, e-commerce applications and database-driven websites. PHP files generally use the .php extension and must be processed by a PHP-enabled server.

Features of PHP

- Open-source and available for major operating systems.
- Can be embedded in HTML and works with common web servers.
- Supports procedural and object-oriented programming.
- Provides built-in facilities for handling forms, sessions, cookies and files.
- Supports database access through extensions such as MySQLi and PDO.
- Can generate HTML pages, JSON responses and other server-side output.

4.2 Basic PHP syntax and variables

PHP code is normally placed between <?php and ?> tags. Statements commonly end with a semicolon. Variables begin with the dollar sign ($), and variable names are case-sensitive. PHP is dynamically typed, so the type of a variable is associated with the value assigned to it.

Data type	Description
Integer	Whole-number values.
Float	Numbers containing a fractional part.
String	Sequence of characters.
Boolean	A true or false value.
Array	Collection of values stored under keys.
Object	Instance of a class.
NULL	Represents a variable with no value.

4.3 PHP operators

Operators are symbols used to perform calculations, comparisons and logical operations. PHP supports arithmetic operators such as +, -, *, / and %. Assignment operators include =, += and -=. Comparison operators include ==, ===, !=, !==, > and <. Logical operators include &&, || and !. The dot (.) operator joins strings.

4.4 Control structures in PHP

Control structures determine which statements are executed and how many times they are repeated. PHP provides if, if–else, elseif and switch for decision-making. It provides for, while, do–while and foreach for iteration. break exits a loop or switch, while continue moves to the next loop iteration.

Selection statements

The if statement executes a block when a condition evaluates to true. The if–else statement selects between two alternatives. The elseif form checks further conditions. The switch statement is useful when a variable is compared against several possible values.

Looping statements

A for loop is suitable when the number of iterations is known or controlled by a counter. A while loop checks its condition before each iteration. A do–while loop checks the condition after executing the body, so the body runs at least once. A foreach loop is designed to process each element of an array or iterable.

```
<?php
$marks = 68;
if ($marks >= 40) {
   echo "Pass";
} else {
   echo "Fail";
}

for ($i = 1; $i <= 5; $i++) {
   echo $i . " ";
}
?>
```

4.5 Arrays in PHP

An array is a data structure that stores multiple values in a single variable. PHP arrays are ordered maps, meaning that values are associated with keys. The key may be an integer or a string. Arrays are useful for storing lists of students, marks, products and other related information.

Array type	Explanation
Indexed array	Uses numeric indexes, commonly starting at zero.
Associative array	Uses named keys to identify values.
Multidimensional array	Contains one or more arrays as its elements.

```
<?php
$subjects = ["PHP", "Java", "AJAX"];
echo $subjects[0];

$student = ["name" => "Anu", "marks" => 85];
echo $student["name"];

foreach ($subjects as $subject) {
   echo $subject . "<br>";
}
?>
```

Common array functions include count() to find the number of elements, sort() to sort values, array_push() to add an item, array_pop() to remove the last item and array_merge() to combine arrays.

4.6 Functions in PHP

A function is a named block of code that performs a particular task. It can be called from different parts of a program. Functions improve reuse, readability, testing and maintenance. A function may accept zero or more parameters and may return a value.

PHP supports built-in functions and user-defined functions. Built-in functions perform common tasks such as string processing, array operations and mathematical calculations. User-defined functions are written by the programmer for application-specific tasks.

```
<?php
function calculateTotal($a, $b) {
   return $a + $b;
}
$total = calculateTotal(20, 30);
echo "Total = " . $total;
?>
```

4.7 PHP form handling

PHP can receive form data submitted by an HTML page. The $_GET and $_POST superglobal arrays provide access to values submitted using the corresponding HTTP methods. The $_REQUEST array may contain values from more than one request source depending on configuration, so using the expected specific source is clearer.

User input must be validated on the server. Data displayed in HTML should be escaped to prevent cross-site scripting. Sensitive information should be transmitted over HTTPS, and passwords should be stored using secure password-hashing functions rather than plain text.

4.8 Database connectivity in PHP

A database stores information in an organised form so that it can be retrieved, inserted, updated and deleted. A web application may store student details, course information, login records or transaction data in a relational database such as MySQL. PHP communicates with a database through database extensions.

Two common ways to connect PHP to MySQL are MySQLi and PDO. MySQLi is designed for MySQL databases. PDO provides a consistent interface for several database systems. Both support prepared statements, which separate SQL structure from user-supplied values.

Basic stages of database connectivity

21. Create a connection using the database host, username, password and database name.
22. Check whether the connection was successful.
23. Prepare and execute an SQL statement.
24. Process the returned records when the query is a SELECT operation.
25. Handle errors safely and close resources when finished.

Illustrative PHP–MySQL example

```
<?php
$conn = new mysqli("localhost", "appuser", "password", "college");


if ($conn->connect_error) {
   die("Database connection failed");
```

```
}

$stmt = $conn->prepare(
   "SELECT name FROM students WHERE id = ?"
);
$id = 5;
$stmt->bind_param("i", $id);
$stmt->execute();

$result = $stmt->get_result();
while ($row = $result->fetch_assoc()) {
   echo htmlspecialchars($row["name"]);
}

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

This example assumes that the database and table already exist and that the MySQLi extension is enabled. The database account and password must be replaced with suitable local values. The prepared statement protects the query parameter from being treated as SQL code. Output escaping is used when displaying data in HTML.

4.9 Introduction to Zend Framework

Zend Framework is a PHP framework that provides reusable components and an organised structure for developing web applications. It encourages separation of responsibilities, code reuse and maintainability. The Zend Framework project has evolved into the Laminas Project; the name Zend Framework continues to appear in older applications and academic syllabi.

A framework differs from a simple library because it provides an overall structure and conventions for building an application. Developers use its components and application patterns rather than writing every feature from the beginning.

MVC architecture

Model–View–Controller (MVC) is an architectural pattern used to separate an application into three main responsibilities. The Model represents application data and business rules. The View presents information to the user. The Controller receives requests, coordinates application work and selects the response.

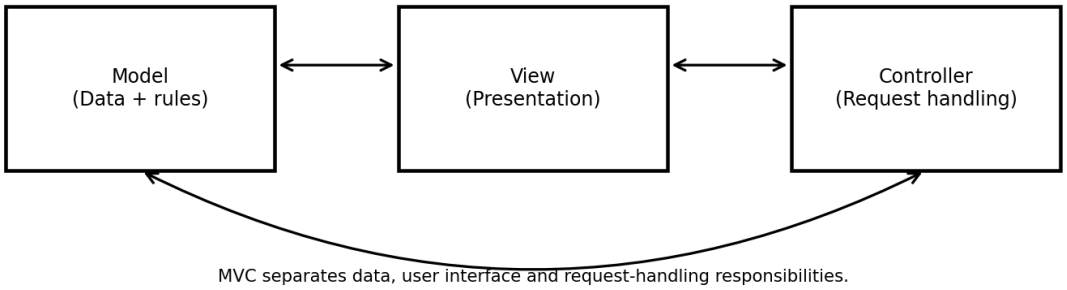


Figure 4.1: Conceptual Model–View–Controller architecture.

Part	Responsibility	Example in a college application
Model	Manages data and business rules.	Student record and course data operations.
View	Displays information to the user.	Student details page or registration form.
Controller	Handles requests and coordinates model and view.	Processes a request to display student information.

Typical MVC request flow

26. The browser sends a request to the application.
27. The routing mechanism identifies the appropriate controller action.
28. The controller validates the request and calls the model when data is required.
29. The model retrieves or updates the data source.
30. The controller provides the result to the view.
31. The view generates the presentation returned to the browser.

Zend Framework / Laminas components and applications

Framework components provide reusable facilities for common application tasks. Depending on the framework version and selected components, these may include routing, HTTP request and response handling, form validation, database access, authentication, logging and dependency management. The framework's modular design allows developers to include the components needed by an application.

- College management systems for student, course and staff records.
- Content management systems for publishing and updating information.
- Online shopping applications with product, cart and order modules.
- Business portals with login, authentication and role-based access.
- Web services that exchange data with other applications.

Advantages of using a framework

- Encourages a consistent project structure.
- Promotes code reuse through components.
- Separates presentation, request handling and business logic.
- Can improve maintainability and testing.
- Provides established approaches for common web development tasks.

5.1 Introduction to Java Servlets

A Servlet is a Java class that runs inside a servlet container and handles requests from clients. It commonly receives HTTP requests from a browser, performs processing and sends an HTTP response. Servlets are used for form processing, session management, authentication, database interaction and controlling the flow of a web application.

A servlet container, such as Apache Tomcat, provides the runtime environment for servlets. It manages servlet loading, object creation, life-cycle methods, request and response objects, URL mapping and communication with clients.

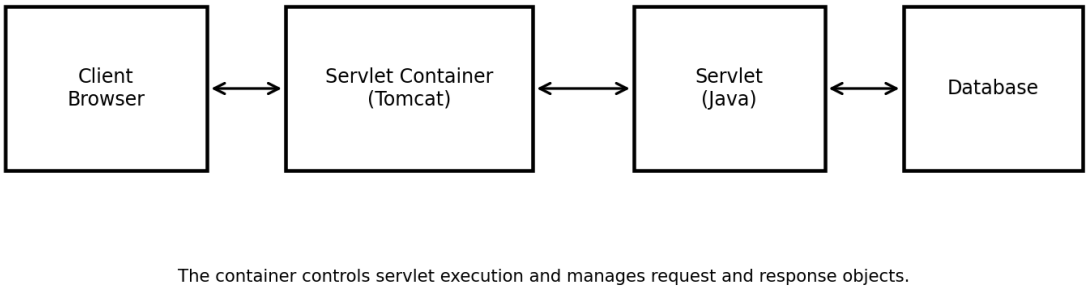


Figure 5.1: Simplified servlet application architecture.

Features and uses of Servlets

- Servlets are written in Java and run inside a compatible servlet container.
- They are platform-independent when supported by a compatible Java runtime and container.
- They can process HTTP GET and POST requests.
- They can access request parameters, headers and session information.
- They can communicate with databases through JDBC.
- They are commonly used as controllers in MVC-based web applications.

5.2 Servlet classes and interfaces

Servlet technology defines interfaces and classes that specify how a servlet is configured and executed. The Servlet interface defines the basic life-cycle contract. GenericServlet is a protocol-independent implementation. HttpServlet extends GenericServlet and provides methods designed for HTTP requests.

Class / Interface	Purpose
Servlet	Defines the servlet life-cycle methods init(), service() and destroy().
GenericServlet	Abstract class that provides a general implementation of the Servlet interface.
HttpServlet	Base class for servlets that handle HTTP requests.

ServletRequest	Provides client request information and request parameters.
ServletResponse	Provides facilities for creating a response.
HttpServletRequest	HTTP-specific request information such as headers, parameters and session access.
HttpServletResponse	HTTP-specific response operations such as content type and status.
ServletConfig	Provides configuration information for an individual servlet.
ServletContext	Provides application-wide information and shared application access.

5.3 Servlet life cycle

The servlet life cycle is managed by the servlet container. The container loads the servlet class, creates an instance, initialises it, invokes it to handle requests and eventually removes it from service.

32. Loading and instantiation: the container loads the servlet class and creates a servlet object.
33. Initialisation: the container calls init() once after creating the servlet. Initial setup tasks can be performed here.
34. Request processing: the container calls service() for incoming requests. In HttpServlet, service() dispatches the request to methods such as doGet() and doPost().
35. Destruction: the container calls destroy() before removing the servlet, allowing it to release resources.

Important: A servlet instance may process multiple requests. Avoid storing request-specific information in ordinary shared instance fields unless thread safety is handled correctly.

5.4 HTTP methods in Servlets

The doGet() method handles HTTP GET requests and is generally used to retrieve information. The doPost() method handles HTTP POST requests and is generally used to submit information or request an operation that changes application state. Other methods include doPut(), doDelete() and doHead(). The appropriate method depends on the purpose of the request.

Simple HttpServlet example

```
import java.io.*;
import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.WebServlet;

@WebServlet("/hello")
public class HelloServlet extends HttpServlet {
   protected void doGet(HttpServletRequest request,
                        HttpServletResponse response)
                        throws IOException {
      response.setContentType("text/html");
      PrintWriter out = response.getWriter();
      out.println("<h2>Hello from Servlet</h2>");
   }
}
```

This example uses the Jakarta Servlet API. Older Java EE applications may use the javax.servlet package instead. The application must be deployed to a servlet container compatible with the API version used to compile it.

5.5 Java Database Connectivity (JDBC)

Java Database Connectivity (JDBC) is a standard Java API used to connect Java applications to relational databases. It provides a common way to establish connections, send SQL statements and process query results. A JDBC driver translates JDBC calls into the communication required by a particular database.

JDBC architecture and components

Component	Function
JDBC API	Provides standard interfaces and classes for database operations.
JDBC Driver	Enables communication between a Java program and a specific database.
DriverManager / DataSource	Provides or manages database connections.
Connection	Represents an active connection to a database.
Statement	Executes a simple SQL statement.
PreparedStatement	Executes a parameterised SQL statement and supports safer value binding.
CallableStatement	Supports calling stored procedures.
ResultSet	Represents rows returned by a query.
SQLException	Provides information about database errors.

Steps involved in JDBC connectivity

36. Import the required JDBC packages.
37. Make the correct JDBC driver available to the application.
38. Obtain a database connection using a URL and credentials.
39. Create a Statement or PreparedStatement.
40. Execute the SQL query or update.
41. Process the ResultSet if the operation returns rows.
42. Close the ResultSet, statement and connection after use.

JDBC program to retrieve student records

```
import java.sql.*;

public class JdbcExample {
    public static void main(String[] args) {
        String url = "jdbc:mysql://localhost:3306/college";
        String user = "appuser";
        String password = "password";
        String sql = "SELECT id, name FROM students";

        try (Connection con =
                DriverManager.getConnection(url, user, password);
             PreparedStatement ps = con.prepareStatement(sql);
             ResultSet rs = ps.executeQuery()) {

            while (rs.next()) {
                System.out.println(rs.getInt("id") + " "
                                   + rs.getString("name"));
            }
        } catch (SQLException e) {
            System.out.println("Database error: " + e.getMessage());
        }
    }
```

```
}
```

The example assumes that MySQL is running, the college database and students table exist, and the MySQL JDBC driver is included in the project. In a real application, credentials should be stored securely and connection pooling may be used.

Advantages of JDBC

- Provides a standard API for database access from Java.
- Supports SQL queries, updates and stored procedures.
- Works with different database systems when an appropriate driver is available.
- Supports prepared statements and transaction management.
- Can be used by standalone Java programs, Servlets and other Java applications.

5.6 Introduction to JSP

JSP stands for JavaServer Pages. It is a server-side technology used to create dynamic web pages. A JSP page contains HTML and JSP elements that can display information generated by a Java application. The web container translates a JSP page into a servlet and compiles it before execution.

JSP is particularly useful for the presentation layer of a Java web application. A servlet can process a request, obtain data and forward it to a JSP page for display. This separation helps keep request-handling logic and page presentation organised.

Features and advantages of JSP

- Supports dynamic content within HTML pages.
- Can use Java-based application data and server-side objects.
- Is translated into a servlet and managed by the web container.
- Supports reusable components and tag libraries.
- Can be used as the View in an MVC application.
- Separates presentation from much of the application's request-processing logic.

5.7 JSP scripting elements

Traditional JSP provides three scripting elements: declarations, scriptlets and expressions. These elements allow Java code to be included directly in a JSP page. They are important for understanding older JSP applications and syllabus concepts. In modern development, Expression Language (EL), JSTL and controller-based MVC design are generally preferred because they keep Java code out of presentation pages.

Element	Syntax	Purpose
Declaration	<%! ... %>	Declares fields or methods at the generated servlet class level.
Scriptlet	<% ... %>	Contains Java statements executed during request processing.
Expression	<%= ... %>	Evaluates an expression and writes its value to the response.

Example of JSP scripting elements

```
<%@ page contentType="text/html;charset=UTF-8" %>
<html>
```

```
<body>
  <h2>Welcome to JSP</h2>
  <%
    String studentName = "Anu";
  %>
  <p>Student Name: <%= studentName %></p>
</body>
</html>
```

5.8 JSP implicit objects

JSP provides predefined objects called implicit objects. They are made available by the JSP container, so the programmer does not need to create them explicitly. These objects provide access to request data, response handling, output, session information and application context.

Implicit object	Purpose
request	Represents the client's request and provides access to parameters, headers and attributes.
response	Represents the response that will be sent to the client.
out	Writes content to the response output stream.
session	Stores information associated with a particular user's session.
application	Represents the ServletContext and provides application-wide information.
config	Provides configuration information for the JSP's servlet.
pageContext	Provides access to page-related context and scoped attributes.
page	Refers to the current generated servlet instance.
exception	Provides exception information on a JSP error page.

5.9 JSP directives and actions

JSP directives provide instructions to the container during translation. The page directive defines page-level settings such as content type and imports. The include directive includes another file during translation. The taglib directive declares a tag library. JSP actions are XML-style elements that perform standard tasks at request time, such as including another resource or forwarding a request.

Directive / Action	Purpose
page directive	Sets page properties such as content type, imports and error-page settings.
include directive	Includes the content of another file during translation.
taglib directive	Declares a tag library for use in the JSP page.
jsp:include	Includes the output of another resource at request time.
jsp:forward	Forwards the current request to another resource.
jsp:useBean	Locates or creates a JavaBean for use by the page.

5.10 JSP web applications

A JSP web application is a collection of resources deployed to a servlet container. It may include JSP pages, servlets, Java classes, libraries, images, CSS files and configuration files. A common structure separates presentation pages from controller classes and data-access components.

For example, a student registration application may use an HTML form to collect details, a servlet to validate and process the request, a Java class or JDBC component to store the data, and a JSP page to display a

confirmation message. This approach is more maintainable than placing all application logic inside a single JSP page.

Servlet and JSP comparison

Basis	Servlet	JSP
Main purpose	Request processing and application control.	Dynamic presentation of information.
Source form	Java class.	Page containing HTML and JSP elements.
Execution	Managed by the servlet container.	Translated into a servlet and then executed.
Typical MVC role	Controller.	View.
Best use	Processing input, coordinating services and navigation.	Displaying data and page output.

EXAMINATION PREPARATION

Important long-answer questions

- Explain the Internet and the working of the World Wide Web. Describe browser architecture with a diagram.
- Explain the functions of browser components. Compare Internet Explorer and Google Chrome.
- Describe the working stages of a search engine.
- Explain the structure of an HTML5 document and discuss important semantic tags.
- Discuss HTML5 audio, video and form controls with suitable examples.
- Explain CSS types, selectors, properties and the CSS box model with a diagram.
- Explain JavaScript data types, operators and control structures.
- Discuss JavaScript functions and the Document, Browser, Date, Math and String objects.
- Explain JavaScript events and event handling with examples.
- Describe jQuery objects, selectors, functions, events and animation methods.
- Explain AJAX architecture and the working of XMLHttpRequest.
- Discuss AJAX libraries and explain how asynchronous requests update a web page.
- Compare IIS and Tomcat. Explain the procedure for hosting a website.
- Explain PHP variables, operators, control structures, arrays and functions.
- Describe PHP database connectivity and the use of prepared statements.
- Explain Zend Framework and MVC architecture with a diagram.
- Explain Servlet classes, interfaces and the servlet life cycle.
- Describe JDBC architecture, components and steps involved in database connectivity.
- Explain JSP scripting elements, directives and implicit objects.
- Compare Servlets and JSP and explain the structure of a JSP web application.

Short-answer revision questions

- Define Internet, WWW, URL, HTTP and HTTPS.
- What is DNS? Why is it required?
- What is the role of a rendering engine?
- Differentiate HTML and CSS.
- What is a semantic HTML element?
- Differentiate GET and POST.
- What is the DOM?
- What is an event listener?
- What is a jQuery selector?
- Expand AJAX and JSON.
- What is the purpose of XMLHttpRequest?
- What is a web server?
- What is an application pool in IIS?
- What is a servlet container?
- What is a PHP associative array?

- What is a prepared statement?
- Expand MVC and explain its three parts.
- What is the purpose of the Servlet interface?
- What is JDBC?
- What are JSP implicit objects?

Important terms and glossary

Term	Meaning
AJAX	Asynchronous JavaScript and XML; a technique for background data exchange.
API	Application Programming Interface; a defined way for software components to interact.
BOM	Browser Object Model; browser-related objects exposed to JavaScript.
CSS	Cascading Style Sheets; controls web page presentation.
DOM	Document Object Model; a structured representation of an HTML document.
DNS	Domain Name System; translates domain names into IP addresses.
HTML	HyperText Markup Language; structures web content.
HTTP / HTTPS	Protocols used for web communication; HTTPS uses TLS protection.
JDBC	Java Database Connectivity; API for Java database access.
JSON	JavaScript Object Notation; a lightweight data interchange format.
PHP	PHP: Hypertext Preprocessor; a server-side scripting language.
Servlet	A Java server-side component managed by a servlet container.
JSP	JavaServer Pages; technology for dynamic server-side web pages.
URL	Uniform Resource Locator; address of a web resource.
Web server	Software that receives web requests and returns responses.

Practical topics for laboratory practice

- Create a structured HTML5 page using headings, paragraphs, lists, links, images and semantic elements.
- Create a student registration form using HTML5 controls and validation attributes.
- Apply external CSS using selectors, box model properties and a responsive media query.
- Write JavaScript examples for conditions, loops, functions and DOM events.
- Use jQuery to select elements, change content, handle events and apply a simple animation.
- Create an AJAX request to load a text or JSON response from a local server.
- Write a PHP page that receives form data and displays validated output.
- Connect PHP to a MySQL database and retrieve records using a prepared statement.
- Create a basic Servlet and deploy it on a compatible Tomcat server.
- Write a JDBC program to retrieve database records and display them.
- Create a JSP page that displays values passed from a form or servlet.