VG

MCA-106

WEB TECHNOLOGIES

PREFACE

Dear Reader,

It is a privilege to present this comprehensive collection of **RTU MCA Semester Examination Notes**, designed to cover the complete syllabus with clarity and academic accuracy. Every concept, definition, explanation, and example has been organized to support thorough understanding and effective exam preparation.

The purpose of these notes is to simplify complex topics and provide a reliable study resource that can assist in both detailed learning and quick revision. Continuous effort has been made to ensure correctness and relevance; however, learning grows when readers engage, question, and explore further.

May these notes serve as a strong academic foundation and contribute meaningfully to your preparation and future growth.

Warm regards,
Virendra Goura
Author
www.virendragoura.com

DISCLAIMER

This e-book has been created with utmost care, sincere effort, and extensive proofreading. However, despite all attempts to avoid mistakes, there may still be some errors, omissions, or inaccuracies that remain unnoticed.
This e-book is issued with the understanding that neither the author nor the publisher shall be held responsible for any loss, damage, or misunderstanding arising from the use of the information contained within.
All content provided is for educational and informational purposes only.

MCA-106 Web Technologies

Unit-1

The internet: history of the World Wide Web, hardware and software trend, object, technology - java script object, scripting for the web-browser portability.

Introduction of HTML: introduction, markup language, editing HTML : common tags, headers, text styles, linking, images, formatting text, horizontal rules and more line breaks, unordered lists, nested and ordered lists, basic HTML tables : intermediate HTML tables and formatting : basic HTML forms, more complex HTML forms, HTML5: Input Types & Attributes, internal linking, creating and using image maps.

Unit-2

Introduction to scripting: introduction- memory concepts- arithmetic- decision making. Java script control structures, Java script functions: introduction - program Units in java script - function definitions, duration of identifiers, scope rules, recursion, java script global functions.

Java script arrays: introduction, array-declaring and allocating arrays, references and reference parameters - passing arrays to functions, multiple subscripted arrays. Java script objects: introduction, math, string, date, Boolean and number objects.

Unit-3

CSS: introduction - inline styles, creating style sheets with the style element, conflicting styles, linking external style sheets, positioning elements, backgrounds, element dimensions, text flow and the CSS box model, user style sheets, Filter and Transitions, HTML DOM, Browser BOM.

Event model: introduction, event ON CLICK, event ON LOAD error handling with ON ERROR, tracking the mouse with event, more DHTML events.

Unit-4

Overview of PHP Capabilities, PHP HTML embedding tags & syntax, Simple script examples, PHP & HTTP Environment variables. PHP Language Core- Variables. Constants, Data Types, PHP: Operators, Flow Control & Loops. Arrays, String, Functions Include & require statements, Simple File & Directory Access operations.

Unit-5

Error handling, Processing HTML form using GET, POST, REQUEST, SESSION, COOKIE variables, Sending E-mail, Database Operations with PHP, Connecting to My-SQL (or any other database), Selecting a db, building & Sending Query, retrieving, updating & inserting data, CMS: Wordpress. Note: XAMMP is used for PHP

UNIT – 1: Introduction to HTML

The internet: history of the World Wide Web, hardware and software trend, object technology - java script object, scripting for the web-browser portability.

Introduction of HTML: introduction, markup language, editing HTML: common tags, headers, text styles, linking, images, formatting text, horizontal rules and more line breaks, unordered lists, nested and ordered lists, basic HTML tables :intermediate HTML tables and formatting: basic HTML forms, more complex HTML forms, HTML5: Input Types & Attributes, internal linking, creating and using image maps.

1. The Internet

1.1 Introduction

The **Internet** is a global network of interconnected computers that communicate using standardized protocols to share information, resources, and services.
It connects millions of private, public, academic, business, and government networks.

The Internet is a worldwide system of computer networks that use the TCP/IP protocol to link devices globally.

1.2 History of the World Wide Web (WWW)

Year	Development
1969	ARPANET (Advanced Research Projects Agency Network) — first internet prototype by the U.S. Department of Defense.
1974	TCP/IP model proposed by Vinton Cerf & Robert Kahn.
1983	TCP/IP adopted as standard for ARPANET.
1989	Tim Berners-Lee at CERN developed the World Wide Web (WWW).
1991	First website went live (info.cern.ch).
1993	Mosaic — first graphical browser introduced.
1995	Internet became commercial; HTML 2.0 released.
2004–Present	Web 2.0 introduced (interactive & social web); HTML5 and modern JavaScript frameworks dominate.

1.3 Hardware and Software Trends in Internet

Hardware Trends

- High-speed fiber-optic networks
- Cloud computing servers (AWS, Azure)
- Mobile devices & IoT (Internet of Things)

- High-performance routers and wireless access points

Software Trends

- Web browsers (Chrome, Firefox, Edge)
- Dynamic web applications using JavaScript, React, Angular
- Cloud-based OS and apps
- AI integration into websites (chatbots, recommendations)

2. Object Technology

Object Technology uses the concept of **objects** to design software and web applications. Each object combines **data** (properties) and **functions** (methods).

JavaScript Objects

JavaScript is an **object-based scripting language** used for adding interactivity and logic to web pages.

Example:

```
<script>
let student = {
   name: "Veer",
   course: "MCA",
   greet: function() {
      return "Hello, " + this.name;
   }
};
document.write(student.greet());
</script>
```

Output:
Hello, Veer

Objects in JavaScript can represent:

- HTML elements
- Browser windows (window object)
- Forms, events, and user interactions

3. Scripting for the Web

Scripting allows web pages to become dynamic and interactive.

Type of Script	Executed By	Example
Client-side script	Web browser	JavaScript, VBScript
Server-side script	Web server	PHP, Python, Node.js

Example (Client-Side JavaScript):

```
<script>
alert("Welcome to Saklume Website!");
</script>
```

Browser Portability

Browser Portability ensures that a webpage or script behaves consistently across different browsers (Chrome, Firefox, Safari, Edge).

Techniques:

- Follow **HTML5** and **CSS3** standards
- Avoid browser-specific extensions
- Test web pages across multiple browsers using tools like BrowserStack

INTRODUCTION TO HTML

1. What is HTML?

HTML stands for **HyperText Markup Language** — the standard language for creating and structuring web pages.

Definition:

HTML is a markup language that uses *tags* to define the structure, layout, and elements of a webpage.

HTML Document Structure

```
<!DOCTYPE html>
<html>
<head>
  <title>My First Web Page</title>
</head>
<body>
  <h1>Welcome to HTML</h1>
  <p>This is my first HTML page.</p>
</body>
</html>
```

Tag	Meaning
<!DOCTYPE html>	Declares HTML5 document type
<html>	Root element
<head>	Contains metadata, title, CSS links
<body>	Contains visible page content

2. Markup Language and Tags

- HTML is not a programming language but a **markup language**.
- It uses **tags** enclosed in angle brackets (< >) to describe content.

Example:
This is a paragraph.

Tag Types

Type	Example	Description
Paired Tags	...	Opening & closing tags
Empty Tags	 , <hr>, <img>	Do not require closing tags

3. Editing HTML

You can create or edit HTML files using any text editor:

- Notepad / VS Code / Sublime Text / Brackets

Save file as **.html** (e.g., index.html).

4. Common HTML Tags

Tag	Purpose
<html>	Defines start of HTML document
<head>	Header information
<title>	Page title (appears in browser tab)
<body>	Visible content
<h1>–<h6>	Headings (largest to smallest)
	Paragraph
 	Line break
<hr>	Horizontal line
<b> / <i>	Bold / Italic
<u>	Underline

5. Headers and Text Styles

Example:

```
<h1>Main Heading</h1>
<h2>Sub Heading</h2>
<p><b>Bold</b>, <i>Italic</i>, and <u>Underline</u></p>
```

Output:
Main Heading
Sub Heading
Bold, Italic, and Underline

6. Linking Web Pages

Links connect one page to another using the <a> tag.

Example:

About Us

Attributes:

- href → URL of the link destination
- target="_blank" → Opens link in new tab

7. Inserting Images

Images are inserted using the <img> tag.

<img src="saklume.jpg" alt="Saklume Jewelry" width="400" height="250">

Attribute	Purpose
src	Image file path
alt	Alternate text if image fails to load
width, height	Size of the image

8. Formatting Text

Tag	Function
<b>	Bold
<i>	Italic
<u>	Underline
<sup>	Superscript (e.g., X2)
<sub>	Subscript (e.g., H2O)
<small>	Smaller text
<big>	Larger text

9. Horizontal Rules and Line Breaks

Welcome to Saklume!
<hr>
New collection coming soon.
Stay tuned!
Output: Adds a line and a new line break.

10. Lists

HTML supports three types of lists:

a) Unordered List

```
<ul>
  <li>Gold</li>
  <li>Silver</li>
  <li>Diamond</li>
</ul>
```

Output: Bulleted list.

b) Ordered List

```
<ol type="A">
  <li>Home</li>
  <li>Shop</li>
  <li>Contact</li>
</ol>
```

Output: A, B, C...

c) Nested List

```
<ul>
  <li>Fruits
    <ul>
      <li>Apple</li>
      <li>Mango</li>
    </ul>
  </li>
</ul>
```

11. Basic HTML Tables

Tables are created using <table> tag.

```
<table border="1">
<tr><th>Name</th><th>Price</th></tr>
<tr><td>Ring</td><td>₹2500</td></tr>
<tr><td>Necklace</td><td>₹6000</td></tr>
</table>
```

Output Table:

Name	Price
Ring	₹2500
Necklace	₹6000

12. Intermediate Tables and Formatting

You can merge cells or add colors:

```
<table border="1" cellpadding="5" cellspacing="2" bgcolor="#f9f9f9">
<tr><th colspan="2">Saklume Jewelry</th></tr>
<tr><td>Gold Ring</td><td>₹2500</td></tr>
<tr><td colspan="2" align="center">Limited Offer</td></tr>
</table>
```

Attributes:

- colspan → Merges columns
- cellpadding → Space inside cell
- cellspacing → Space between cells

13. Basic HTML Forms

Forms collect user input.

```
<form action="/submit">
  Name: <input type="text" name="username"><br>
  Password: <input type="password" name="pass"><br>
  <input type="submit" value="Login">
</form>
```

Elements:

- <input> → Text boxes, radio buttons, etc.
- <textarea> → Multi-line text
- <select> → Dropdown menu
- <button> → Clickable button

14. More Complex Forms

```
<form>
  Gender:
  <input type="radio" name="g" value="M">Male
  <input type="radio" name="g" value="F">Female<br>

  Choose Items:
  <input type="checkbox" name="jewelry" value="Ring">Ring
  <input type="checkbox" name="jewelry" value="Necklace">Necklace<br>

  <select name="city">
    <option>Delhi</option>
    <option>Mumbai</option>
  </select><br>

  <input type="reset" value="Clear">
  <input type="submit" value="Submit">
</form>
```

15. HTML5 Input Types & Attributes

HTML5 introduced new <input> types and attributes for better validation.

Input Type	Purpose
email	Validates email addresses
number	Accepts numeric values
date	Selects date
url	Validates website URLs
range	Slider input
color	Color picker

Attributes:

- placeholder → Hint text
- required → Mandatory input
- pattern → Regular expression validation

Example:

<input type="email" placeholder="Enter your email" required>

16. Internal Linking

Internal linking connects sections within the same page.

Go to Contact

<h2 id="contact">Contact Us</h2>
Email: support@saklume.com

17. Image Maps

An **image map** allows clicking on different parts of an image to navigate to different links.

<img src="store-map.jpg" usemap="#shopmap">
<map name="shopmap">
  <area shape="rect" coords="0,0,100,100" href="rings.html">
  <area shape="circle" coords="150,150,50" href="necklaces.html">
</map>

Attributes:

- usemap → Links image to map name
- coords → Defines clickable region coordinates
- shape → Rectangle, circle, or polygon

UNIT – 2: Java Script

Introduction to scripting: introduction- memory concepts- arithmetic- decision making. Java script control structures, Java script functions: introduction - program Units in java script - function definitions, duration of identifiers, scope rules, recursion, java script global functions.

Java script arrays: introduction, array-declaring and allocating arrays, references and reference parameters - passing arrays to functions, multiple subscripted arrays. Java script objects: introduction, math, string, date, Boolean and number objects.

1. Introduction to Scripting

What is Scripting?

Scripting is a technique used to write small programs (scripts) that control or automate tasks inside another software environment, such as a web browser.

Scripts are **interpreted**, not compiled, and executed line by line.

Why Scripting is Used?

- To make web pages **interactive**
- To validate user input
- To perform calculations
- To dynamically update content
- To control browser behavior

Examples of Scripting Languages

- JavaScript
- Python
- PHP
- Ruby
- VBScript

2. Introduction to JavaScript

JavaScript is a **high-level, interpreted, object-based scripting language** used mainly to create **dynamic and interactive web pages**.

Features of JavaScript

- Lightweight
- Interpreted language
- Event-driven

- Platform independent
- Supports OOP concepts
- Runs inside the browser

Uses of JavaScript

- Form validation
- Dynamic HTML content
- Animations
- Games
- Web applications

3. Memory Concepts in JavaScript

Memory in JavaScript refers to the storage area where **variables, functions, and objects** are stored during program execution.

Types of Memory

1. **Stack Memory**
2. **Heap Memory**

3.1 Stack Memory

- Stores **primitive data types**
- Uses **Last In First Out (LIFO)**
- Faster access

Stored in Stack:

- Number
- String
- Boolean
- Undefined
- Null

Example

```
let a = 10;
let b = 20;
```
Here a and b are stored in **stack memory**.

3.2 Heap Memory

- Stores **non-primitive data types**
- Used for dynamic memory allocation

Stored in Heap:

- Objects
- Arrays
- Functions

Example

let obj = { name: "John", age: 25 };
The object is stored in **heap memory**, while reference is stored in stack.

4. Arithmetic in JavaScript

Arithmetic Operators

Used to perform mathematical operations.

Operator	Meaning
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus
++	Increment
--	Decrement

Example

```
let a = 10;
let b = 3;

console.log(a + b);
console.log(a - b);
console.log(a * b);
console.log(a / b);
console.log(a % b);
```

Output

```
13
7
30
3.333333333
1
```

5. Decision Making in JavaScript

Decision making allows the program to **choose different paths** based on conditions.

5.1 if Statement

```
let age = 20;

if(age >= 18){
  console.log("Eligible to vote");
}
```

Output

```
Eligible to vote
```

5.2 if–else Statement

```
let num = 5;

if(num % 2 == 0){
  console.log("Even");
}else{
  console.log("Odd");
}
```

Output

```
Odd
```

5.3 else–if Ladder

```
let marks = 75;

if(marks >= 90){
  console.log("A Grade");
}else if(marks >= 60){
  console.log("B Grade");
}else{
  console.log("Fail");
}
```

Output

```
B Grade
```

5.4 switch Statement

```
let day = 2;

switch(day){
  case 1: console.log("Monday"); break;
  case 2: console.log("Tuesday"); break;
  default: console.log("Invalid Day");
}
```

Output

```
Tuesday
```

6. JavaScript Control Structures

Control structures control the **flow of execution**.

Types

1. Sequential
2. Selection
3. Iteration (Loops)

6.1 Looping Statements

for Loop

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

Output

1 2 3 4 5

while Loop

```
let i = 1;
while(i <= 3){
  console.log(i);
  i++;
}
```

Output

1 2 3

do-while Loop

```
let i = 1;
do{
  console.log(i);
  i++;
}while(i <= 2);
```

Output

1 2

7. JavaScript Functions

A function is a **reusable block of code** that performs a specific task.

7.1 Program Units in JavaScript

Program units include:

- Functions

- Variables
- Objects
- Events

Functions are the **main program units**.

7.2 Function Definition

```
function add(a, b){
  return a + b;
}
```

7.3 Function Call

```
let result = add(5, 3);
console.log(result);
```

Output

8

8. Duration of Identifiers

Duration refers to **how long an identifier exists in memory**.

Types

1. Local
2. Global
3. Static (function-level)

9. Scope Rules in JavaScript

Scope defines **where a variable can be accessed**.

Types of Scope

9.1 Global Scope

```
let x = 10;

function show(){
  console.log(x);
}
show();
```

Output

10

9.2 Local Scope

```
function test(){
```

```
  let y = 5;
  console.log(y);
}
test();
```

Output

5

9.3 Block Scope (let / const)

```
if(true){
  let a = 20;
}
```

a cannot be accessed outside the block.

10. Recursion in JavaScript

Recursion is a technique where a function **calls itself** to solve a problem.

Example: Factorial

```
function factorial(n){
  if(n == 1){
    return 1;
  }
  return n * factorial(n - 1);
}

console.log(factorial(5));
```

Output

120

11. JavaScript Global Functions

Global functions can be used **without creating any object**.

Common Global Functions

Function	Purpose
alert()	Displays message
prompt()	Takes input
confirm()	OK / Cancel
parseInt()	Converts string to integer
parseFloat()	Converts string to float
isNaN()	Checks NaN

Example

```
let num = "10";
console.log(parseInt(num) + 5);
```

Output

15

JAVASCRIPT ARRAYS

1. Introduction to JavaScript Arrays

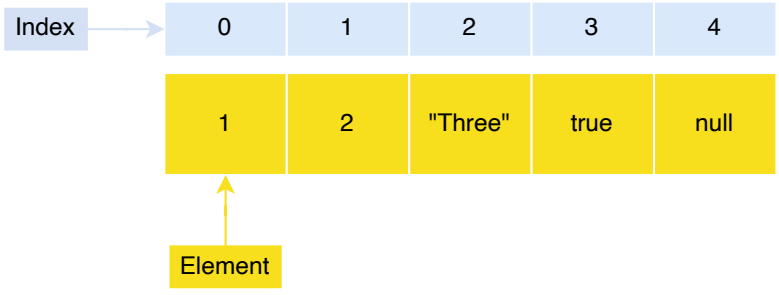
An **array** in JavaScript is a special variable that can store **multiple values in a single variable**, instead of declaring separate variables for each value.

Why Arrays are Needed

- Store large amount of data efficiently
- Easy data access using index
- Useful for loops and data processing
- Improves program readability

Key Characteristics

- JavaScript arrays are **dynamic**
- Index starts from **0**
- Can store **different data types** together
- Array size can grow or shrink automatically



2. Declaring and Allocating Arrays

2.1 Array Declaration Using Array Literal

This is the **most common method**.

```
let numbers = [10, 20, 30, 40];
```

Accessing Elements

```
console.log(numbers[0]);
console.log(numbers[2]);
```

Output

```
10
30
```

2.2 Array Declaration Using new Array()

```
let fruits = new Array("Apple", "Banana", "Mango");
```

2.3 Declaring Empty Array and Allocating Later

```
let arr = [];
arr[0] = 5;
arr[1] = 10;
```

2.4 Array with Mixed Data Types

```
let data = [10, "Hello", true, 3.5];
```

3. References and Reference Parameters

In JavaScript, arrays are treated as **objects**, so they are passed to functions **by reference**, not by value.

This means:

- Changes inside the function **affect the original array**

Example: Reference Behavior

```
let arr = [1, 2, 3];

function modifyArray(a){
  a[0] = 100;
}

modifyArray(arr);
console.log(arr);
```

Output

```
[100, 2, 3]
```

4. Passing Arrays to Functions

An array can be passed as an argument to a function and accessed using parameters.

Example: Sum of Array Elements

```
function sumArray(a){
  let sum = 0;
  for(let i = 0; i < a.length; i++){
    sum += a[i];
  }
  return sum;
}

let nums = [10, 20, 30];
console.log(sumArray(nums));
```

Output

```
60
```

Passing Array and Returning Array

```
function doubleArray(a){
  for(let i = 0; i < a.length; i++){
    a[i] = a[i] * 2;
  }
  return a;
}

console.log(doubleArray([1,2,3]));
```

Output

```
[2, 4, 6]
```

5. Multiple Subscripted Arrays (Multidimensional Arrays)

A **multiple subscripted array** is an array that contains **other arrays as elements**.
It is also called a **multidimensional array**.

2D Array Example

```
let matrix = [
  [1, 2, 3],
  [4, 5, 6],
  [7, 8, 9]
];
```

Accessing Elements

```
console.log(matrix[0][1]);
console.log(matrix[2][2]);
```

Output

```
2
9
```

Traversing 2D Array

```
for(let i = 0; i < matrix.length; i++){
  for(let j = 0; j < matrix[i].length; j++){
    console.log(matrix[i][j]);
  }
}
```

JAVASCRIPT OBJECTS

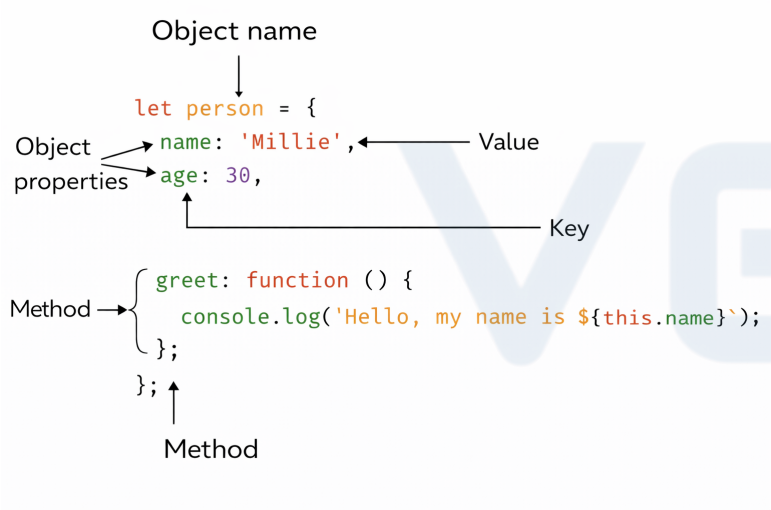
6. Introduction to JavaScript Objects

An **object** is a collection of **properties and methods** that represent a real-world entity.

Why Objects are Used

- Organize related data
- Represent real-world entities
- Improve modularity
- Support object-oriented programming

Object Creation



7. Math Object

The **Math object** provides mathematical constants and functions.

Important Properties and Methods

Method	Description
Math.PI	Value of π
Math.sqrt()	Square root

Math.pow()	Power
Math.max()	Maximum
Math.min()	Minimum
Math.round()	Round value
Math.floor()	Lower integer
Math.ceil()	Upper integer

Example

```
console.log(Math.sqrt(25));
console.log(Math.pow(2,3));
console.log(Math.round(4.6));
```

Output

```
5
8
5
```

8. String Object

The **String object** is used to manipulate text data.

Common String Methods

Method	Purpose
length	String length
toUpperCase()	Uppercase
toLowerCase()	Lowercase
charAt()	Character
indexOf()	Position
substring()	Extract part
replace()	Replace text

Example

```
let str = "JavaScript";

console.log(str.length);
console.log(str.toUpperCase());
console.log(str.substring(0,4));
```

Output

```
10
JAVASCRIPT
Java
```

9. Date Object

The **Date object** is used to work with **date and time**.

Creating Date Object

```
let today = new Date();
console.log(today);
```

Common Date Methods

Method	Description
getDate()	Day
getMonth()	Month
getFullYear()	Year
getHours()	Hours
getMinutes()	Minutes

Example

```
let d = new Date();
console.log(d.getFullYear());
```

Output

2025

10. Boolean Object

The **Boolean object** represents **true or false** values.

Example

```
let x = new Boolean(false);

if(x){
  console.log("True value");
}else{
  console.log("False value");
}
```

Output

True value
(Note: Boolean objects are always true when used in conditions.)

11. Number Object

The **Number object** is used to work with numeric values.

Common Number Methods

Method	Description
toFixed()	Fixed decimals
toString()	Convert to string
parseInt()	Convert to integer
parseFloat()	Convert to float

Example

```
let num = 12.3456;

console.log(num.toFixed(2));
console.log(num.toString());
```

Output

```
12.35
12.3456
```

UNIT –3: Dynamic HTML

CSS: introduction - inline styles, creating style sheets with the style element, conflicting styles, linking external style sheets, positioning elements, backgrounds, element dimensions, text flow and the CSS box model, user style sheets, Filter and Transitions, HTML DOM, Browser BOM.

Event model: introduction, event ON CLICK, event ON LOAD - error handling with ON ERROR, tracking the mouse with event, more DHTML events.

1. Introduction to DHTML

Dynamic HTML (DHTML) is an advanced form of HTML that combines **HTML**, **CSS**, **JavaScript**, and **Document Object Model (DOM)** to create interactive, animated, and responsive web pages without reloading.

DHTML = HTML + CSS + JavaScript + DOM

Key Concept

Unlike static HTML pages that show fixed content, DHTML allows pages to **change dynamically** based on **user interaction, time, or input**.

Example of dynamic change:

- Changing text color when a user hovers over it.
- Expanding menus when clicked.
- Real-time validation of form inputs.
- Animating images and elements on scroll.

1.1 Characteristics of DHTML

1. **Dynamic Content Update:**
   Page content changes instantly using scripts (no refresh needed).
2. **Client-Side Execution:**
   Operations are handled by the user's browser, reducing server load.
3. **Event-Driven Programming:**
   Reacts to user actions like clicks, keystrokes, and mouse movement.
4. **Animation and Effects:**
   Elements can move, fade, or resize using CSS transitions or JavaScript.
5. **Object Manipulation:**
   DOM allows elements to be modified or created at runtime.

1.2 Difference Between HTML and DHTML

Feature	HTML	DHTML
Nature	Static	Dynamic and interactive

Interactivity	No	Yes, through JavaScript
Technology Used	Only HTML	HTML + CSS + JS + DOM
Page Reload	Required for changes	Not required
Animation	Not possible	Possible using CSS/JS
User Control	Limited	Extensive (real-time interaction)

1.3 Advantages of DHTML

- Enhances **user experience** through interactivity.
- Reduces **server requests** (client-side processing).
- Supports **animations**, **menus**, and **responsive design**.
- Enables **form validation** and **data verification** on the client side.
- Provides **real-time feedback** to users.

2. Cascading Style Sheets (CSS)

2.1 Introduction

CSS stands for **Cascading Style Sheets** — a language used to define the **visual appearance** of web elements.

It separates *content (HTML)* from *presentation (design)*.
DHTML uses CSS for **styling and animation**, enabling developers to control **layout, color, font, and effects** dynamically.

2.2 The Cascade Concept

The term *"Cascading"* means that styles are applied in a hierarchical order.
If multiple rules apply to an element, the one with **highest priority** overrides others.

2.3 Syntax of CSS Rule

```
selector {
  property: value;
}
```

Example:

```
p {
  color: blue;
  font-size: 18px;
}
```

2.4 Ways to Apply CSS

1. Inline CSS

Applied directly to an element using the style attribute.
Used for small, specific changes.

```
<h2 style="color:purple;">Welcome to Saklume</h2>
```

2. Internal CSS

Defined inside <style> tag in the <head> section.
Applies styles to elements within the same document.

```
<head>
<style>
body { background-color: lavender; }
h1 { text-align: center; }
</style>
</head>
```

3. External CSS

Saved in a separate .css file and linked to multiple HTML pages.

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

```
p { color: darkblue; }
```

2.5 Conflicting Styles

If multiple styles are applied to the same element, the browser decides which one to use based on **priority**.

Priority Order:

1. Inline CSS (highest)
2. Internal CSS
3. External CSS
4. Browser default (lowest)

Example:
If color is defined differently in each method, the **inline color** will appear.

2.6 Positioning Elements in DHTML

Positioning allows you to **control the layout** and **movement** of elements.

Value	Description
static	Default position (normal flow)
relative	Moves element relative to its normal position

absolute	Positions element relative to nearest ancestor
fixed	Stays fixed on screen (useful for headers/menus)
sticky	Acts as relative until a scroll threshold is met

Example:

```
#menu {
  position: fixed;
  top: 0;
  width: 100%;
}
```

2.7 CSS Box Model

The **CSS Box Model** defines how space is distributed around elements.

Margin → Border → Padding → Content
Example:

```
div {
  margin: 20px;
  padding: 10px;
  border: 2px solid black;
  width: 300px;
}
```

2.8 Background and Colors

Backgrounds can have **solid colors, gradients, or images**.

```
body {
  background-image: url('jewelry-bg.jpg');
  background-size: cover;
  background-repeat: no-repeat;
}
```

2.9 Transitions and Filters

Transitions add smooth animation between CSS property changes.
Filters apply visual effects like blur or brightness.

```
img {
  filter: brightness(1.2) contrast(1.5);
  transition: transform 1s;
}
img:hover {
  transform: scale(1.1);
}
```

2.10 User Style Sheets

Users can define personal style sheets in browsers for accessibility — changing colors, fonts, or sizes as per preference.
Used especially for **visually impaired users**.

3. Document Object Model (DOM)

The **DOM** represents an HTML document as a **tree of objects**.
Each element, attribute, or text node in the HTML becomes a **DOM node**, which can be accessed and modified using JavaScript.

DOM Hierarchy Example

```
<html>
  <body>
    <h1>Hello</h1>
    <p>Welcome!</p>
  </body>
</html>
```

DOM Tree Structure:

```
Document
 └── html
     └── body
         ├── h1
         └── p
```

3.2 DOM Interfaces

Object	Description
document	Root of all HTML elements
element	Represents HTML tags like <p>,
attribute	Represents HTML attributes
text	Represents text content inside tags

3.3 Accessing and Modifying DOM Elements

Access Methods

Method	Use
getElementById()	Select element by ID
getElementsByClassName()	Select elements by class
getElementsByTagName()	Select elements by tag
querySelector()	Selects first element matching CSS selector

Example:

```
<p id="msg">Welcome</p>
<script>
document.getElementById("msg").innerHTML = "Dynamic Text Updated!";
</script>
```

Creating Elements Dynamically

```
let newDiv = document.createElement("div");
newDiv.innerHTML = "New section added!";
```

document.body.appendChild(newDiv);

Removing Elements

document.body.removeChild(newDiv);

4. Browser Object Model (BOM)

The **BOM** provides an interface for interacting with the **browser window** and its properties. It allows control over features like windows, history, location, and screen.

Main BOM Objects

Object	Purpose	Example
window	Represents browser window	window.open("page.html")
document	Represents current HTML document	document.title
navigator	Provides browser info	navigator.userAgent
screen	Screen dimensions	screen.width, screen.height
history	Browser navigation history	history.back()
location	Current page URL	location.href

Example:

```
document.write("Browser: " + navigator.appName);
document.write("<br>Screen Width: " + screen.width);
```

5. EVENT MODEL IN DHTML

5.1 Introduction

An **event** is any user action or system-generated signal that can trigger JavaScript code. DHTML uses an **event-driven programming model** to create interactivity.

Common Events

Event	Triggered When
onclick	Element is clicked
onload	Page finishes loading
onmouseover	Mouse moves over element
onmouseout	Mouse leaves element
onfocus	Element gains focus
onblur	Element loses focus
onchange	Value of an element changes

onerror	Error occurs while loading content

5.2 ONCLICK Event

Triggered when user clicks an element.

```
<button onclick="showMessage()">Click Me!</button>
<script>
function showMessage() {
   alert("Button clicked!");
}
</script>
```

5.3 ONLOAD Event

Executed when a webpage or image has completely loaded.

```
<body onload="alert('Welcome, Veer! Page loaded successfully.')">
```

5.4 ONERROR Event

Handles errors such as missing images or script failures.

```
<img src="missing.jpg" onerror="alert('Image not found!')">
```

5.5 Tracking Mouse Movement

You can capture mouse coordinates in real time.

```
<body onmousemove="showCoords(event)">
<p id="coord"></p>
<script>
function showCoords(event) {
  document.getElementById("coord").innerHTML =
  "X: " + event.clientX + ", Y: " + event.clientY;
}
</script>
</body>
```

5.6 More DHTML Events

Event	Description	Use Case
ondblclick	Triggered on double-click	Open or expand menus
onkeydown	Key pressed down	Typing input validation
onkeyup	Key released	Search-as-you-type
onsubmit	Form submission	Form validation
onresize	Browser window resized	Responsive layouts
onscroll	Page scrolling	Trigger animations

Example – Form Validation

```
<form onsubmit="return validate()">
<input type="text" id="name" placeholder="Enter Name">
```

```
<input type="submit">
</form>

<script>
function validate(){
  let name = document.getElementById("name").value;
  if(name == ""){
     alert("Name cannot be empty!");
     return false;
  }
  return true;
}
</script>
```

6. Real-Life Applications of DHTML

1. **Interactive Menus** – Dropdowns and navigation bars
2. **Form Validation** – Checking input fields dynamically
3. **Animations** – Fading images, transitions
4. **Dynamic Content Loading** – Chat boxes, live feeds
5. **Games and Simulations** – Real-time browser interactions

UNIT –4: Introduction to PHP & Web Server Architecture

Overview of PHP Capabilities, PHP HTML embedding tags & syntax, Simple script examples, PHP & HTTP Environment variables. PHP Language Core- Variables. Constants, Data Types, PHP: Operators, Flow Control & Loops. Arrays, String, Functions Include & require statements, Simple File & Directory Access operations.

1. Introduction to PHP

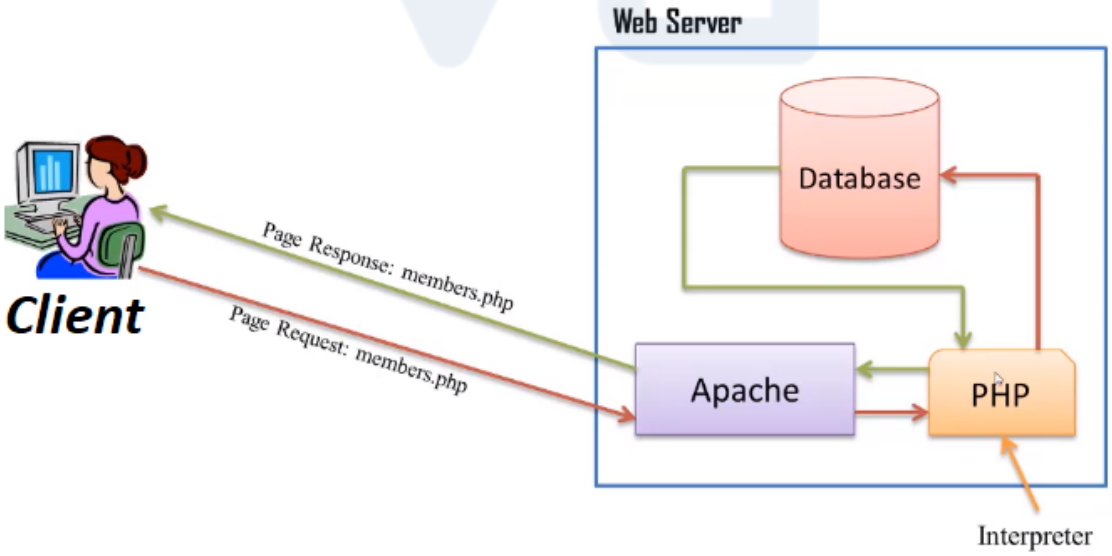
PHP (Hypertext Preprocessor) is a **server-side scripting language** designed mainly for **web development**.
It executes on the server and sends processed HTML output to the client browser.

Key Points

- PHP code runs on the **server**
- Output is sent to the **browser as HTML**
- Open-source and platform independent
- Works well with databases like MySQL

2. Web Server Architecture

Web server architecture explains how a **client (browser)** communicates with a **server** to request and receive web pages.



Architecture Flow

1. User enters URL in browser
2. Browser sends HTTP request
3. Web server receives request
4. PHP interpreter executes PHP code

5. Server sends HTML response
6. Browser displays output

Components

- Client (Browser)
- Web Server (Apache / Nginx)
- PHP Engine
- Database (optional)

3. Overview of PHP Capabilities

PHP is capable of:

- Generating dynamic page content
- Handling forms
- Connecting with databases
- Managing sessions & cookies
- File handling
- Sending emails
- Creating REST APIs
- Authentication & authorization

4. PHP HTML Embedding Tags & Syntax

PHP allows embedding server-side code within HTML documents using special PHP tags, enabling dynamic content generation inside static web page structures.

PHP Tags

PHP code is written inside special tags.

```
<?php
  // PHP code here
?>
```

Embedding PHP in HTML

```
<html>
<body>
  <h1>
    <?php echo "Welcome to PHP"; ?>
  </h1>
</body>
</html>
```

Output

Welcome to PHP

5. Simple PHP Script Examples

A simple PHP script demonstrates basic syntax, variable usage, and output generation, helping beginners understand PHP execution flow and server-side processing.

Example 1: Hello World

```
<?php
echo "Hello World";
?>
```

Output

Hello World

Example 2: Arithmetic Operation

```
<?php
$a = 10;
$b = 5;
echo $a + $b;
?>
```

Output

15

6. PHP & HTTP Environment Variables

Environment variables store information about:

- Server
- Request
- Client
- Script execution

They are accessed using **$_SERVER** superglobal.

Common PHP Environment Variables

Variable	Description
$_SERVER['PHP_SELF']	Current script name
$_SERVER['SERVER_NAME']	Server name
$_SERVER['HTTP_HOST']	Host name
$_SERVER['REQUEST_METHOD']	GET / POST
$_SERVER['REMOTE_ADDR']	Client IP

Example

```
<?php
echo $_SERVER['SERVER_NAME'];
?>
```

7. PHP Language Core

7.1 Variables

A variable is a **named memory location** used to store data.

Rules

- Starts with $
- Case-sensitive
- No declaration needed

Example

```
<?php
$name = "John";
$age = 25;
echo $name;
?>
```

7.2 Constants

A constant is a value that **cannot be changed** during script execution.

Syntax

```
define("SITE", "MyWebsite");
```

Example

```
<?php
define("PI", 3.14);
echo PI;
?>
```

7.3 Data Types

PHP Data Types

Type	Example
Integer	10
Float	10.5
String	"PHP"
Boolean	true / false
Array	[]

Object	new Class
NULL	null

Example

```
<?php
$x = 10;
$y = "PHP";
$z = true;
?>
```

8. PHP Operators

Operators in PHP are symbols used to perform operations like arithmetic calculation, comparison, logical evaluation, assignment, and increment or decrement of values.

Types of Operators

Operator Type	Examples
Arithmetic	+ − * / %
Assignment	= += -=
Comparison	== != > <
Logical	&&
Increment	++ --

Example

```
<?php
$a = 10;
$b = 5;
echo $a > $b;
?>
```

Output

```
1
```

9. Flow Control & Loops

Loops are used to repeatedly execute a block of code until a specified condition becomes false, reducing code redundancy and improving efficiency.

9.1 if Statement

The if statement is used to execute a block of code only when a given condition evaluates to true.

Program

```
<?php
```

```
$age = 18;

if ($age >= 18) {
  echo "Adult";
}
?>
```

Output

Adult

9.2 if–else Statement

The if–else statement allows execution of one block when the condition is true and another block when the condition is false.

Program

```
<?php
$num = 5;

if ($num % 2 == 0) {
  echo "Even";
} else {
  echo "Odd";
}
?>
```

Output

Odd

9.3 switch Statement

The switch statement is used to select one execution path from multiple options based on the value of a variable.

Program

```
<?php
$day = 1;

switch ($day) {
  case 1:
    echo "Monday";
    break;
  default:
    echo "Invalid";
}
?>
```

Output

Monday

9.4 Looping Statements

(a) for Loop

The for loop is used when the number of iterations is known in advance and executes code repeatedly based on a loop counter.

Program

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

Output

```
1 2 3 4 5
```

(b) while Loop

The while loop executes a block of code repeatedly as long as the specified condition remains true.

Program

```
<?php
$i = 1;

while ($i <= 3) {
  echo $i . " ";
  $i++;
}
?>
```

Output

```
1 2 3
```

10. PHP Arrays

Arrays store multiple values in a single variable, allowing organized data storage and easy access using index or key-based referencing.

10.1 Indexed Array

An indexed array stores elements with numeric index values starting from zero. It is mainly used to store simple lists of data.

Program

```
<?php
$colors = array("Red", "Green", "Blue");
echo $colors[0];
?>
```

Output

Red

10.2 Associative Array

An associative array stores elements in key–value pairs, allowing data to be accessed using meaningful string keys instead of numeric indexes.

Program

```
<?php
$student = array("name" => "John", "age" => 20);
echo $student["name"];
?>
```

Output

John

10.3 Multidimensional Array

A multidimensional array contains one or more arrays as its elements. It is used to store data in tabular or matrix form.

Program

```
<?php
$marks = array(
  array(10, 20),
  array(30, 40)
);
echo $marks[0][1];
?>
```

Output

20

11. PHP String Functions

String functions in PHP are used to manipulate text data, such as finding length, converting case, extracting substrings, and replacing characters.

Common String Functions

Function	Purpose
strlen()	Length
strtoupper()	Uppercase
strtolower()	Lowercase
substr()	Substring
str_replace()	Replace

Example

```
<?php
$str = "PHP Language";
```

```
echo strlen($str);
?>
```

12. PHP Functions

Functions are reusable blocks of code that perform a specific task, improving modularity, readability, and maintainability of PHP programs.

User Defined Function

```
<?php
function add($a,$b){
  return $a + $b;
}
echo add(5,3);
?>
```

13. Include & Require Statements

Used to include one PHP file into another.

include

The include statement inserts external PHP files into a script. If the file is missing, a warning is generated but script execution continues.

```
include "header.php";
```

require

The require statement includes external PHP files into a script. If the file is missing, a fatal error occurs and execution stops immediately.

```
require "config.php";
```

14. Simple File & Directory Access Operations

14.1 File Handling

PHP provides built-in functions to create, read, write, and delete files, allowing efficient management of server-side data storage.

Reading File

```
<?php
echo file_get_contents("data.txt");
?>
```

Writing File

```
<?php
file_put_contents("data.txt","Hello PHP");
?>
```

14.2 Directory Handling

Directory functions in PHP allow creation, deletion, and management of folders on the server, supporting structured file organization.

Create Directory

```
<?php
mkdir("myfolder");
?>
```

Delete Directory

```
<?php
rmdir("myfolder");
?>
```

UNIT – 5: PHP Advanced Features & CMS

Error handling, Processing HTML form using GET, POST, REQUEST, SESSION, COOKIE variables, Sending E-mail, Database Operations with PHP, Connecting to My-SQL (or any other database), Selecting a db, building & Sending Query, retrieving, updating & inserting data, CMS: Wordpress. Note: XAMMP is used for PHP

1. Error Handling in PHP

Error handling in PHP helps identify and manage runtime errors during script execution. It improves debugging, prevents unexpected script termination, and ensures smooth program execution.

Types of Errors

1. **Notice** – Minor issues, script continues
2. **Warning** – Serious problem, script continues
3. **Fatal Error** – Script stops execution

Example: Notice Error

```
<?php
echo $x;
?>
```

Output

Notice: Undefined variable: x

Example: Warning Error

```
<?php
include("file.php");
?>
```

Output

Warning: include(file.php): failed to open stream

Custom Error Handler

```
<?php
function myErrorHandler($errno, $errstr) {
  echo "Custom Error: $errstr";
}

set_error_handler("myErrorHandler");
echo $a;
?>
```

Output

Custom Error: Undefined variable: a

2. Processing HTML Forms Using PHP

PHP processes HTML form data using superglobal variables. These variables collect user input and allow server-side validation, processing, and storage of submitted data.

2.1 GET Method

- Data sent via URL
- Less secure
- Suitable for search forms

HTML Form

```
<form method="get" action="get.php">
  Name: <input type="text" name="name">
  <input type="submit">
</form>
```

PHP Code (get.php)

```
<?php
echo "Name is: " . $_GET["name"];
?>
```

Output (URL)

Name is: John

2.2 POST Method

- Data hidden from URL
- More secure
- Used for login & registration

HTML Form

```
<form method="post" action="post.php">
  Age: <input type="text" name="age">
  <input type="submit">
</form>
```

PHP Code (post.php)

```
<?php
echo "Age is: " . $_POST["age"];
?>
```

Output

Age is: 20

2.3 REQUEST Variable

$_REQUEST is a superglobal variable that collects form data from GET, POST, and COOKIE methods, allowing flexible access to user input.

Example

```
<?php
echo $_REQUEST["name"];
?>
```

3. SESSION Variables

Sessions store user information on the server and maintain data across multiple pages, commonly used for login systems and user authentication.

Example

```
<?php
session_start();
$_SESSION["username"] = "Admin";
echo $_SESSION["username"];
?>
```

Output

Admin

Destroy Session

```
<?php
session_start();
session_destroy();
?>
```

4. COOKIE Variables

Cookies store small data on the client's browser for a specific time period and are used to remember user preferences and login information.

Create Cookie

```
<?php
setcookie("user", "John", time() + 3600);
?>
```

Read Cookie

```
<?php
echo $_COOKIE["user"];
?>
```

Output

John

5. Sending E-mail Using PHP

PHP provides the mail() function to send emails from the server, commonly used in contact forms, notifications, and system alerts.

Example

```
<?php
$to = "example@test.com";
$subject = "PHP Test Mail";
$message = "This is a test email";
$headers = "From: admin@test.com";

mail($to, $subject, $message, $headers);
?>
```

Output

Mail sent successfully
(Server configuration required)

6. Database Operations with PHP

PHP interacts with databases to store, retrieve, update, and delete data dynamically using SQL queries, enabling data-driven web applications.

6.1 Connecting to MySQL (Using XAMPP)

Connection Code

```
<?php
$conn = mysqli_connect("localhost", "root", "", "college");

if (!$conn) {
  die("Connection failed");
}
echo "Database Connected";
?>
```

Output

Database Connected

6.2 Selecting Database

```
<?php
mysqli_select_db($conn, "college");
?>
```

6.3 Inserting Data

```
<?php
$sql = "INSERT INTO students(name, age) VALUES('John', 20)";
mysqli_query($conn, $sql);
echo "Data Inserted";
?>
```

Output

Data Inserted

6.4 Retrieving Data

```
<?php
$result = mysqli_query($conn, "SELECT * FROM students");

while($row = mysqli_fetch_assoc($result)) {
  echo $row["name"] . " " . $row["age"] . "<br>";
}
?>
```

Output

John 20

6.5 Updating Data

```
<?php
$sql = "UPDATE students SET age=21 WHERE name='John'";
mysqli_query($conn, $sql);
echo "Data Updated";
?>
```

Output

Data Updated

7. CMS – WordPress

WordPress is an open-source Content Management System developed using PHP and MySQL. It enables users to create, manage, and publish dynamic websites easily without requiring advanced programming skills.

Key Features of WordPress

1. **Themes & Plugins**
   WordPress provides thousands of free and paid themes and plugins that allow users to customize website design and add advanced functionality without coding.

2. **User-Friendly Dashboard**
   The WordPress dashboard offers an intuitive interface for managing content, media, users, and settings, making website management simple for non-technical users.

3. **SEO Support**
   WordPress is search-engine friendly and supports SEO plugins that help improve website visibility and ranking on search engines like Google.

4. **Database-Driven (MySQL)**
   WordPress stores all website content such as posts, pages, and user data in a MySQL database, allowing efficient data management and dynamic content delivery.

Uses of WordPress

- Used to create **blogs** and personal websites

- Used for **business and corporate websites**
- Used for **portfolios and educational websites**
- Used for **e-commerce websites** using plugins like WooCommerce

8. XAMPP

XAMPP is a free and open-source local web server package that provides Apache, MySQL, PHP, and Perl, enabling developers to develop and test PHP applications locally.

XAMPP Components

1. **X – Cross-Platform**
   XAMPP works on multiple operating systems such as Windows, Linux, and macOS, making it suitable for cross-platform web development.

2. **A – Apache**
   Apache is the web server that processes HTTP requests and delivers web pages to the client browser.

3. **M – MySQL**
   MySQL is the database management system used to store and manage application data efficiently.

4. **P – PHP**
   PHP is the server-side scripting language used to create dynamic and interactive web applications.

5. **P – Perl**
   Perl is a scripting language included in XAMPP, mainly used for text processing and legacy web applications.