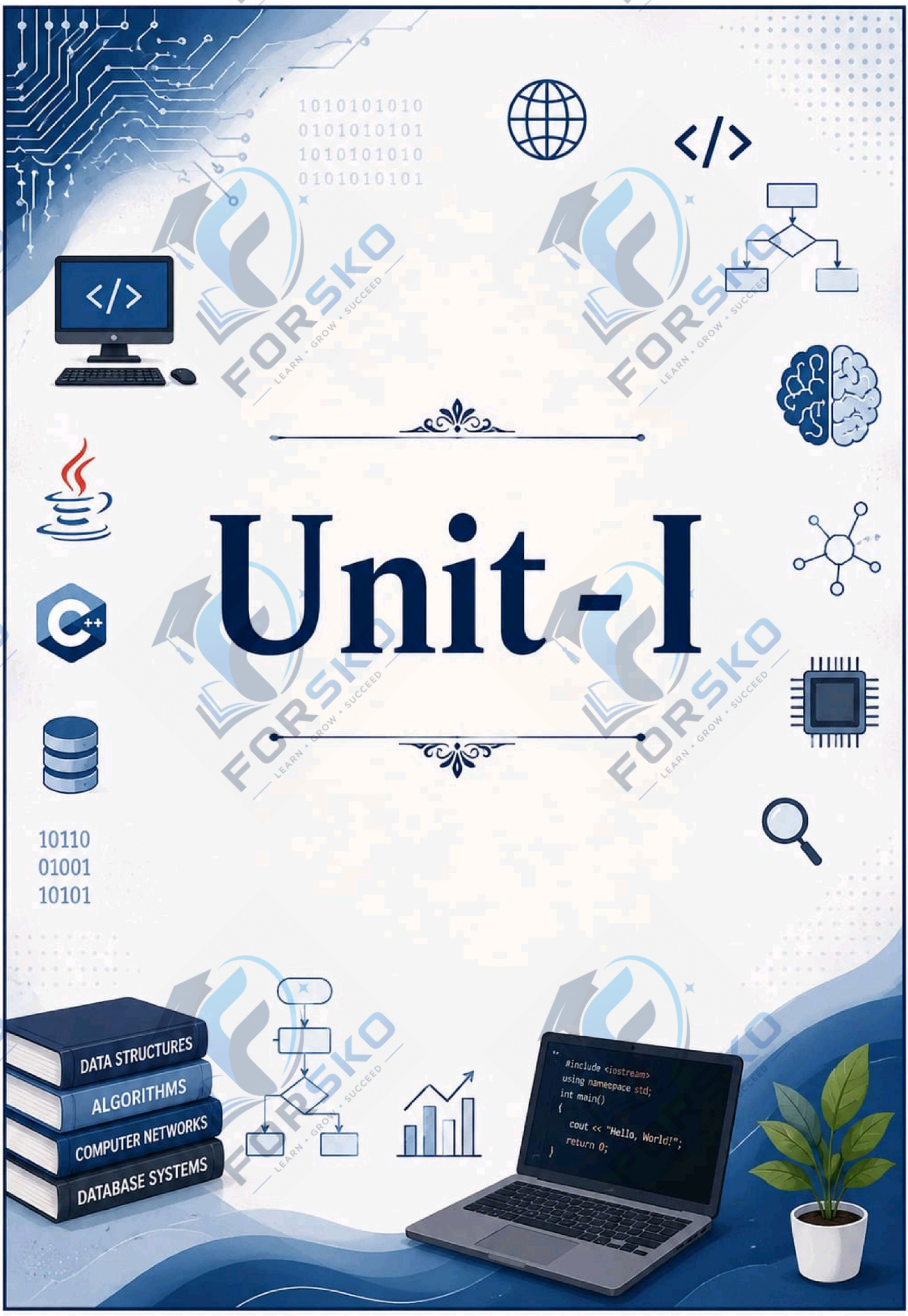

1010101010
0101010101
1010101010
0101010101
Notes
DATA STRUCTURES
ALGORITHMS
COMPUTER NETWORKS
DATABASE SYSTEMS
Good Notes
Better Understanding
Better Results

Unit-I


DATA STRUCTURES
ALGORITHMS
COMPUTER NETWORKS
DATABASE SYSTEMS
#include <iostream>
using nameepace std;
int main()
{
cout << "Hello, World!";
return 0;
}

INTERNET

The Internet is a global system of interconnected computer networks that communicate using standard protocols.

1. HISTORY OF THE INTERNET

1960s — The Beginning (1960s)

The Internet began as ARPANET in the USA, a project of the U.S. Department of Defense to connect research computers.

1970s — Development of TCP/IP (1970s)

Vint Cerf and Bob Kahn developed the TCP/IP protocol suite, the foundation of the modern Internet.

1980s — Expansion (1980s)

The Internet expanded globally. DNS (Domain Name System) was introduced in 1984 to make it easier to remember addresses.

1990s — The World Wide Web (1990s)

Tim Berners-Lee invented the World Wide Web in 1991. Web browsers made Internet access easy for everyone.

2000s — Growth & Popularity (2000s)

High-speed Internet, search engines, social media and online services became popular worldwide.

2010s to Present — The Modern Internet (2010s – Present)

Smartphones, cloud computing, IoT and AI are shaping the future of the Internet.

2. APPLICATIONS OF THE INTERNET

1. Education

E-learning, online courses, digital libraries and research materials.

2. Communication

E-mail, instant messaging, video calls and social networking.

3. E-Commerce

Online shopping, digital payments, banking and trading.

4. Healthcare

Telemedicine, online appointments, health information and remote monitoring.

5. News & Information

Latest news, weather updates and information on any topic.

6. Business & Work

Remote work, cloud storage, online meetings and project collaboration.

7. Entertainment

Movies, music, games, streaming and online videos.

8. Government Services

E-governance, online applications, forms, tax filing and public services.

9. Navigation & Travel

Maps, route planning, ticket booking and hotel reservations.

10. Smart Living (IoT)

Smart homes, connected devices, automation and real-time monitoring.

In Short: The Internet connects the world, makes information accessible anytime, anywhere and improves the way we live, learn, work and communicate.

WORLD WIDE WEB (WWW)

The World Wide Web (WWW) is a system of interlinked documents and resources on the Internet, accessed through web browsers using URLs.

WHAT IS IT?

The WWW is a collection of web pages and multimedia content (text, images, audio, video, etc.) stored on web servers around the world and connected through hyperlinks.

HOW IT WORKS

1. You type a URL in the browser.
2. The browser sends a request to the web server.
3. The server responds with the requested web page.
4. The browser displays the web page on your screen.

KEY FEATURES

Global Access
Access information from anywhere in the world.

Hyperlinks
Connects one web page to another.

Multimedia Support
Supports text, images, audio, video and more.

Browser Based
Accessed using web browsers like Chrome, Firefox, Edge, Safari.

EXAMPLES OF WWW

www.google.com · www.wikipedia.org · www.youtube.com · www.facebook.com

IN SHORT

The WWW makes the Internet user-friendly by organizing information in the form of web pages linked together.

WEB STANDARDS

Web Standards are a set of guidelines and best practices for designing and developing web pages and applications to ensure consistency, accessibility, compatibility and quality.

WHAT ARE WEB STANDARDS?

They are rules and specifications created by international organizations to ensure that websites work well on different browsers, devices and platforms.

NEED FOR WEB STANDARDS

- Ensure compatibility across different browsers and devices.
- Improve accessibility for all users, including people with disabilities.
- Promote clean, efficient and maintainable code.
- Reduce development time and cost.
- Future-proof websites with long-term support.

EXAMPLE
A website built using standards will look and work properly on Chrome, Firefox, Edge, Safari and mobile devices.

MAJOR WEB STANDARDS

HTML
Defines the structure and content of web pages.

CSS
Defines the style, layout and design.

JavaScript
Defines the behavior and interactivity.

W3C
World Wide Web Consortium (W3C) develops and maintains web standards.

BENEFITS

Consistency
Provides a consistent experience across platforms.

Accessibility
Makes the web usable for everyone.

SEO Friendly
Clean code helps search engines better understand websites.

Performance
Optimized code improves loading speed.

KEY ORGANIZATION – W3C

The World Wide Web Consortium (W3C) is an international community that develops open standards to ensure the long-term growth of the Web.

WEB STANDARDS PRINCIPLES

Universal Access · Interoperability · Quality · Long-term Stability

★ FOLLOWING WEB STANDARDS = BETTER WEBSITES FOR EVERYONE, EVERYWHERE. ★

BASICS IN WEB DESIGN

Understanding the foundation of web applications

WHAT IS WEB DESIGN?

Web design is the process of creating websites and web pages that are visually appealing, user-friendly and effective in delivering information.

KEY ELEMENTS OF WEB DESIGN

Layout
Structure and arrangement of content

Colors
Use of colors to create harmony and readability

Typography
Choosing the right fonts for better readability

Images
Use of images and graphics to enhance content

User Experience (UX)
Designing for easy navigation and interaction

PRINCIPLES OF GOOD WEB DESIGN

- **Simplicity** – Keep it clean and uncluttered.
- **Consistency** – Maintain uniformity in design.
- **Hierarchy** – Organize content by importance.
- **Contrast** – Highlight important elements.
- **Alignment** – Proper alignment for a professional look.
- **Repetition** – Reuse design elements for consistency.

COMMON WEB DESIGN TOOLS

Figma · Adobe XD · Photoshop · Canva · VS Code

GOAL OF WEB DESIGN

To create websites that are attractive, functional, accessible and responsive across all devices.

MULTITIER APPLICATION ARCHITECTURE

Multitier Architecture is a design approach where an application is divided into multiple layers (tiers). Each tier has a specific role and communicates with the adjacent tiers.

TIERS IN MULTITIER ARCHITECTURE

1. **Presentation Tier (Client Tier)**
   Handles the user interface and user interactions.
   Example: Web Browser
2. **Application Tier (Business Logic Tier)**
   Processes business logic, validates data and handles application functionality.
   Example: Web Server
3. **Data Tier (Database Tier)**
   Stores and manages data and performs database operations.
   Example: Database Server

HOW MULTITIER ARCHITECTURE WORKS

1. PRESENTATION TIER (Client Tier)
User interacts using browser (HTML, CSS, JS) and sends request.

Request →
← Response

2. APPLICATION TIER (Business Logic Tier)
Web Server processes the request, applies business logic and interacts with the database.

Request →
← Response

3. DATA TIER (Database Tier)
Database Server stores, manages and retrieves data as required.

ADVANTAGES

- Improved Scalability
- Better Security
- Maintainability
- Flexibility
- Separation of Concerns

EXAMPLES

- E-commerce websites
- Online banking systems
- Content management systems

CLIENT-SIDE

Client-side refers to everything that runs on the user's device (browser) rather than on the server. It is responsible for presentation and user interaction.

WHAT IS CLIENT-SIDE?

Client-side technologies are used to create the front-end of web applications. They run in the browser and handle the look, feel and behaviour of the website.

TECHNOLOGIES USED

HTML
Defines the structure and content of web pages.

CSS
Controls the styling, layout and design of web pages.

JavaScript
Adds interactivity and dynamic behavior to web pages.

FEATURES OF CLIENT-SIDE

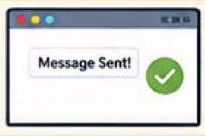
- Runs in the browser (no need for server execution).
- Provides fast and responsive user experience.
- Validates user input before sending data to server.
- Uses technologies like HTML, CSS, JavaScript.
- Interacts with users through events (click, keypress, etc.).

IMPORTANCE OF CLIENT-SIDE

- Enhances user experience
- Reduces server load
- Provides real-time feedback
- Essential for modern, interactive websites

EXAMPLE

When you click a button on a website and a message appears without reloading the page – that's client-side (JavaScript) in action!



SCRIPTING VERSUS SERVER-SIDE SCRIPTING

Two powerful approaches used in web development to create dynamic and interactive websites.

1. SCRIPTING (CLIENT-SIDE SCRIPTING)

Scripts run in the user's browser.
The server is not involved for execution.

HOW IT WORKS

FEATURES

- Runs on the client (browser).
- No server interaction required for basic tasks.
- Provides instant feedback to users.
- Can manipulate HTML, CSS and page content.
- Used for form validation, animations, events, etc.

EXAMPLES OF TECHNOLOGIES

HTML · CSS · JavaScript · TypeScript

EXAMPLE USE CASES

- Form validation before submission
- Image sliders and carousels
- Dynamic menus and dropdowns
- Real-time UI updates

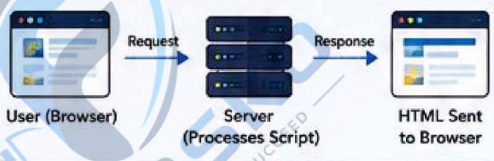
KEY DIFFERENCES

Scripting (Client-Side)		Server-Side Scripting
Runs in the user's browser	Where it runs	Runs on the web server
Not required (after page load)	Server involvement	Required for every request
Faster (response is instant)	Speed	Slower (due to server processing)
Limited access (no direct database access)	Access to resources	Full access to databases, files, APIs, etc.
Less secure (code is visible)	Security	More secure (code is hidden on server)
JavaScript (TypeScript)	Examples of languages	PHP, Python, Java, ASP.NET, Node.js, etc.
Enhances user experience	Use	Handles business logic & data processing

2. SERVER-SIDE SCRIPTING

Scripts run on the server.
The result (HTML) is sent to the browser.

HOW IT WORKS



FEATURES

- Runs on the server.
- Requires server for every request.
- Can access databases and server resources.
- Handles business logic, authentication, sessions.
- Generates HTML and sends it to the client.

EXAMPLES OF TECHNOLOGIES

PHP · Python · Java · Node.js

EXAMPLE USE CASES

- User login and authentication
- E-commerce order processing
- Database operations (CRUD)
- Generating reports and invoices

WORLD WIDE WEB CONSORTIUM (W3C)

The World Wide Web Consortium (W3C) is an international community that develops open standards to ensure the long-term growth of the Web.

ABOUT W3C

Founded in 1994 by Tim Berners-Lee, W3C brings together organizations and individuals to create standards for the Web that promote:

- Compatibility
- Accessibility
- Interoperability
- Internationalization
- Security and Privacy

MISSION

W3C's mission is to lead the Web to its full potential by developing protocols and guidelines that ensure the Web is:

- Universal
- Open
- Accessible to all
- Based on standards

WHAT DOES W3C DO?

- Develops Web standards and guidelines.
- Provides testing tools and validation services.
- Publishes specifications and best practices.
- Promotes the Web for everyone, everywhere.

KEY WEB STANDARDS DEVELOPED BY W3C

HTML	Defines the structure and content of web pages.
CSS	Defines the style and layout of web pages.
JavaScript	Defines the behavior and interactivity of web pages.
XML	Extensible Markup Language for storing and transporting data.
WAI	Web Accessibility Initiative (WAI) Guidelines to make the Web accessible to people with disabilities.

HOW W3C WORKS

01 **Collaboration**
Experts, organizations and developers from around the world collaborate.

02 **Discuss & Draft**
Ideas are discussed in Working Groups and draft specifications are created.

03 **Review**
The drafts are reviewed by members and the public.

04 **Recommendation**
Once approved, the specification becomes a W3C Recommendation.

BENEFITS OF W3C STANDARDS

- Ensures interoperability across browsers and devices.
- Improves accessibility for all users.
- Promotes innovation and new technologies.
- Enhances security and privacy on the Web.
- Supports a single Web for everyone.

GET INVOLVED

Anyone can participate in W3C activities: developers, companies, researchers, and interested individuals.

Learn more: www.w3.org

"W3C's vision: One Web for all, forever." | Open Standards · Global Community · Better Web

HISTORY OF HTML

HTML (HyperText Markup Language) is the standard markup language used to create and structure content on the World Wide Web.

MAJOR MILESTONES

1989 — Tim Berners-Lee's Vision
Tim Berners-Lee proposed the idea of the World Wide Web at CERN.

1991 — Birth of HTML
The first version of HTML was created by Tim Berners-Lee. It had simple features to share information.

1993 — HTML 2.0
The IETF published HTML 2.0 (RFC 1866) with more tags and support for forms.

1995 — HTML 3.2
W3C took over the development and released HTML 3.2 with improved features.

1997 — HTML 4.0
Introduced style sheets (CSS), scripting (JavaScript), and better support for multimedia.

2014 — HTML5
The latest and powerful version with new semantic tags, multimedia support, offline storage and more.

Present — The Future
HTML continues to evolve with new APIs and features for modern web applications.

KEY HIGHLIGHTS

- HTML is the foundation of every web page.
- It works with CSS (for styling) and JavaScript (for behavior).
- HTML is maintained by the W3C (World Wide Web Consortium).
- From simple text pages to modern web apps, HTML has evolved a lot.

DID YOU KNOW?

The first website in the world is still online!

URL: http://info.cern.ch

It was created by Tim Berners-Lee in 1991.

INTRODUCTION TO HTML TAGS AND ATTRIBUTES

HTML tags are used to structure content in a web page. Tags are written inside angle brackets < >. Most tags come in pairs: an opening tag and a closing tag.

COMMON HTML TAGS

Tag	Description	Example
<!DOCTYPE html>	Defines the document type and version	<!DOCTYPE html>
<html> </html>	Root element of an HTML document	<html>...</html>
<head> </head>	Contains meta-information about the document	<head>...</head>
<title> </title>	Sets the title of the web page	<title>My Page</title>
<body> </body>	Contains the content of the web page	<body>...</body>
<h1> to <h6>	Heading tags (h1 is highest level)	Main Heading
	Defines a paragraph	This is a paragraph.
<a> 	Defines a hyperlink	Link
<img>	Embeds an image	<img src="image.jpg" alt="Image">
 	Line break	Line 1 Line 2
<hr>	Horizontal rule	<hr>
	Unordered list	Item 1
	Ordered list	Item 1
	List item	Item
	Defines a section/division	Content
	Inline container	Text
<table> </table>	Defines a table	<table>...</table>
<form> </form>	Defines an HTML form	<form>...</form>

WHAT IS AN ATTRIBUTE?

Attributes provide additional information about HTML elements. They are written inside the opening tag and usually come in name="value" pairs.

Example:

```
<tagname attribute="value">Content</tagname>
```

Attribute — Value

COMMON ATTRIBUTES

Attribute	Description	Example
id	Unique identifier for an element	
class	Specifies one or more class names	
style	Adds inline CSS styles	
title	Provides extra information (tooltip)	Google
href	Specifies the URL for a link	Link
src	Specifies the source of media (img, script, etc.)	<img src="pic.jpg">
alt	Alternative text for an image	<img src="pic.jpg" alt="Photo">
width	Sets the width of an element	<img src="pic.jpg" width="300">
height	Sets the height of an element	<img src="pic.jpg" height="200">
target	Where to open the linked document	Link

NESTING OF TAGS

Tags can be placed inside other tags.

Example:

```
<p>This is a <b>bold</b> word in a paragraph.</p>
```

Opening tag — Nested tag — Closing tag

POINTS TO REMEMBER

- HTML tags are not case sensitive.
- Always close your tags.
- Proper nesting makes the structure correct.
- Use meaningful attributes for better accessibility and SEO.

SIMPLE HTML EXAMPLE

```
<!DOCTYPE html>
<html>
<head>
  <title>My First Page</title>
</head>
<body>
  <h1>Welcome to My Website</h1>
  <p>This is my first web page.</p>
  <a href="https://example.com" target="_blank">Visit Example</a>
</body>
</html>
```

★ HTML is the building block of the web. Learn it well and build the future! ★

Unit - II


DATA STRUCTURES
ALGORITHMS
COMPUTER NETWORKS
DATABASE SYSTEMS
#include <iostream>
using namespace std;
int main()
{
cout << "Hello, World!";
return 0;
}

HTML5: FEATURES

HTML5 is the latest version of HTML. It introduces powerful new features that make web development easier, faster and more capable.

KEY FEATURES OF HTML5

1. Simple & Clean Syntax
Easier to read, write and maintain code.

2. Semantic Elements
New semantic tags help to describe the structure of web pages clearly.

3. Multimedia Support
Native support for audio and video without using third-party plugins.

4. Graphics with Canvas
Draw graphics, charts, animations and games using the <canvas> element.

5. Scalable Vector Graphics (SVG)
Supports 2D vector graphics for better quality on any screen size.

6. Forms 2.0
New input types and attributes make forms more powerful and user-friendly.

7. Local Storage
Store data in the browser with better capacity and without expiration.

8. Offline Support (App Cache)
Web applications can work offline using application cache.

9. Geolocation
Get the user's location with the Geolocation API.

10. Cross-Platform & Device Independent
Works smoothly across different browsers, devices and platforms.

NEW SEMANTIC ELEMENTS IN HTML5

<header>	Defines the header of a document or section
<nav>	Defines navigation links
<section>	Defines a section in a document
<article>	Defines independent, self-contained content
<aside>	Defines content aside from the page content
<footer>	Defines the footer of a document or section
<main>	Specifies the main content of the document
<figure>	Specifies self-contained content like images, diagrams
<figcaption>	Defines a caption for a <figure> element
<mark>	Defines marked/highlighted text
<time>	Defines a date/time
<details>	Defines additional details that the user can open or close
<summary>	Defines a visible heading for a <details> element

OTHER IMPROVEMENTS

- Drag and Drop API
- Web Workers
- Web Sockets
- Microdata for better SEO
- Responsive images
- Improved accessibility

BROWSER SUPPORT

Chrome · Firefox · Edge · Safari · Opera

DID YOU KNOW?

HTML5 is not just HTML. It also includes CSS3, JavaScript APIs and many new technologies.

HTML5 EDITING

HTML5 documents can be created and edited using different tools and methods.

1. HOW TO CREATE AN HTML5 FILE

1. Open any text editor.
2. Write HTML5 code.
3. Save the file with .html extension. Example: index.html

2. BASIC HTML5 DOCUMENT STRUCTURE

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
  initial-scale=1.0">
  <title>My HTML5 Page</title>
</head>
<body>
  <h1>Hello, HTML5!</h1>
  <p>Welcome to my first HTML5 page.</p>
</body>
</html>
```

<!DOCTYPE html> tells the browser that this is an HTML5 document. It helps the browser to render the page in standards mode.

3. POPULAR HTML EDITORS

VS Code
Lightweight and powerful tool with extensions.

Sublime Text
Fast, simple and developer friendly.

Notepad++
Free editor for Windows with many features.

Brackets
Open source editor with live preview.

Atom
Hackable editor by GitHub.

Dreamweaver (Dw)
Professional web development tool by Adobe.

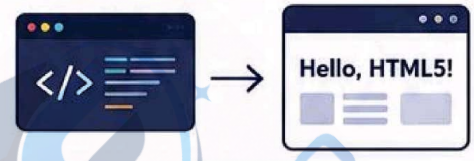
TIP

Choose the editor you are comfortable with. Practice regularly to write clean code.

4. LIVE PREVIEW IN BROWSER

After saving the HTML file:

- Open the file in any web browser.
- Or use the Live Server extension (in VS Code) to see changes instantly.



5. BEST PRACTICES

- Use semantic HTML5 elements.
- Keep your code clean and well-structured.
- Use meaningful names for ids and classes.
- Validate your code using W3C Validator.
- Test your pages in different browsers.

6. USEFUL TOOLS

- W3C Validator – Check and validate your HTML code.
- Can I use – Check browser support for features.
- MDN Web Docs – Learn HTML5 with examples.
- CSS-Tricks – Helpful guides and references.

HTML5 ESSENTIALS

Learn the basics of HTML5 with examples
Build the structure of web pages using simple tags.

1. FIRST HTML5 EXAMPLE

This is a simple HTML5 document. Save the file with .html extension and open it in a web browser.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>My First HTML5 Page</title>
</head>
<body>
    <h1>Welcome to HTML5!</h1>
    <p>This is my first HTML5 page.</p>
</body>
</html>
```

Output in Browser

Welcome to HTML5!

This is my first HTML5 page.

2. HEADINGS

HTML5 provides six levels of headings from <h1> (highest) to <h6> (lowest).

This is Heading 1
This is Heading 2
This is Heading 3
This is Heading 4
This is Heading 5
This is Heading 6

```
<h1>This is Heading 1</h1>
<h2>This is Heading 2</h2>
<h3>This is Heading 3</h3>
<h4>This is Heading 4</h4>
<h5>This is Heading 5</h5>
<h6>This is Heading 6</h6>
```

3. LINKING

Links are used to connect one page to another page or website.

a) Link to another website

```
<a href="https://www.example.com"
  target="_blank">Visit Example</a>
```

b) Link to another page in same website

```
<a href="about.html">About Us</a>
```

Output in Browser

Visit Example

About Us

4. IMAGES

Images are used to display pictures on a web page. The <img> tag is used to embed an image.

```
<img src="image.jpg" alt="Nature Image"
     width="400" height="250">
```

Output in Browser

5. LISTS

HTML5 supports Ordered, Unordered and Description Lists.

a) Unordered List

```
<ul>
  <li>Apple</li>
  <li>Banana</li>
  <li>Cherry</li>
</ul>
```

Output

- Apple
- Banana
- Cherry

b) Ordered List

```
<ol>
  <li>Step 1</li>
  <li>Step 2</li>
  <li>Step 3</li>
</ol>
```

Output

1. Step 1
2. Step 2
3. Step 3

c) Description List

```
<dl>
  <dt>HTML</dt>
  <dd>Hyper Text Markup Language</dd>

  <dt>CSS</dt>
  <dd>Cascading Style Sheets</dd>
</dl>
```

Output

HTML
Hyper Text Markup Language
CSS
Cascading Style Sheets

6. TABLES

Tables are used to display data in rows and columns.

```
<table border="1">
  <tr>
    <th>Roll No</th>
    <th>Name</th>
    <th>Course</th>
  </tr>
  <tr>
    <td>1</td>
    <td>Faizan</td>
    <td>B.Sc</td>
  </tr>
  <tr>
    <td>2</td>
    <td>Shahid</td>
    <td>B.Com</td>
  </tr>
</table>
```

Output in Browser

Roll No	Name	Course
1	Faizan	B.Sc
2	Shahid	B.Com

7. HTML-IFRAME

The <iframe> tag is used to embed another web page inside the current page.

Syntax:

```
<iframe src="URL" width="600" height="350"
        title="Page Title"></iframe>
```

Example:

```
<h3>Example of Iframe</h3>
<iframe src="https://www.example.com"
        width="600" height="350"
        title="Example Website"></iframe>
```

Output in Browser

Example Domain

This domain is for use in documentation examples without needing permission. Avoid use in operations.

Learn more

Common Attributes

Attribute	Description
src	Specifies the URL of the page to display in the iframe.
width	Sets the width of the iframe.
height	Sets the height of the iframe.
title	Provides a title for the iframe.
frameborder	Specifies border around the iframe (0 or 1).
allowfullscreen	Allows the iframe to be displayed in full screen.

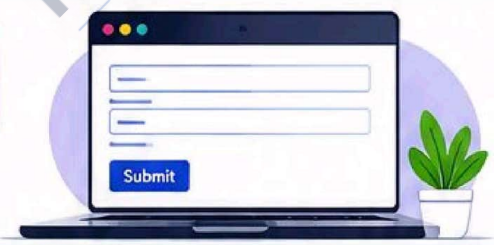
Tips

- Always include <!DOCTYPE html> at the top of your HTML5 document.
- Use semantic tags and proper structure for better websites.
- Validate your HTML code for error-free results.

HTML5 makes web pages more powerful, interactive and user-friendly!

HTML-FORM

HTML forms are used to collect user input and send it to a server for processing.



COMMON FORM ELEMENTS

1. <input>

abc

The <input> element is the most used form element.

- Used to create many types of input fields.
- type attribute defines the type of input.

Syntax:

```
<input type="type" name="name"
    value="value">
```

Examples:

```
<input type="text" name="username">
<input type="email" name="email">
<input type="password" name="pwd">
<input type="date" name="dob">
```

Use: Text fields, email, password, date, radio, checkbox, file upload, etc.

2. <textarea>

Defines a multi-line text input control.

- Users can enter multiple lines of text.
- Commonly used for messages, comments, address, etc.

Syntax:

```
<textarea name="message" rows="4"
   cols="30"></textarea>
```

Example:

```
<textarea name="address" rows="4"
   cols="30"></textarea>
```

Use: Comments, feedback, address, description.

3. <button>

Button

Defines a clickable button.

- Can be used to submit a form or trigger JavaScript.
- type can be "submit", "reset" or "button".

Syntax:

```
<button type="type">Button Text</button>
```

Examples:

```
<button type="submit">Submit</button>
<button type="reset">Reset</button>
<button type="button">Click Me</button>
```

Use: Submit form, reset form, or custom actions.

4. <select>

--Select--

Defines a drop-down list.

- Used to select one option from a list.
- <option> is used inside <select>.

Syntax:

```
<select name="country">
  <option value="">--Select--</option>
  <option value="in">India</option>
  <option value="us">USA</option>
</select>
```

Use: Country list, categories, choices, etc.

5. <label>

Label

Defines a label for an input element.

- Improves accessibility.
- When clicked, it focuses the associated input.

Syntax:

```
<label for="id_name">Label Text</label>
```

Example:

```
<label for="username">User Name:</label>
```

Use: Name, email, password, checkbox, radio, dropdown, etc.

INPUT TYPES (Commonly Used)

Type	Purpose	Example	Type	Purpose	Example
text	Single line	abc	hidden	Hidden field	
email	Email input	abc@xyz.com	submit	Submit form	Submit
password	Password input	••••••	reset	Reset form	Reset
number	Numeric input	123	button	Clickable button	Click Me
date	Date picker	dd-mm-yyyy	tel	Telephone input	+91 1234567890
radio	Choose one option		url	Website URL	https://example.com
checkbox	Choose multiple options		search	Search input	Search...
file	File upload	Choose File	color	Color picker	

COMPLETE FORM EXAMPLE

Registration Form

Full Name: Enter your name

Email: Enter your email

Password: Enter your password

Gender: Male Female Other

Country: -- Select Country --

About You: Write something about yourself...

I agree to the terms and conditions

Submit Reset

HTML CODE

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Registration Form</title>
</head>
<body>
  <h2>Registration Form</h2>
  <form action="#" method="post">
    <label for="name">Full Name:</label><br>
    <input type="text" id="name" name="name"
           placeholder="Enter your name"><br><br>
    <label for="email">Email:</label><br>
    <input type="email" id="email" name="email"
           placeholder="Enter your email"><br><br>
    <label for="password">Password:</label><br>
    <input type="password" id="password"
           name="password" placeholder="Enter your password"><br><br>
    <label>Gender:</label><br>
    <input type="radio" name="gender" value="male"> Male
    <input type="radio" name="gender" value="female"> Female
    <input type="radio" name="gender" value="other"> Other
    <br><br>
    <label for="country">Country:</label><br>
    <select id="country" name="country">
      <option value="">-- Select Country --</option>
      <option value="in">India</option>
      <option value="us">USA</option>
      <option value="uk">UK</option>
    </select><br><br>
    <label for="about">About You:</label><br>
    <textarea id="about" name="about" rows="4" cols="30"
           placeholder="Write something about yourself..."></textarea>
    <br><br>
    <input type="checkbox" id="agree" name="agree">
    <label for="agree">I agree to the terms and conditions</label>
    <br><br>
    <button type="submit">Submit</button>
    <button type="reset">Reset</button>
  </form>

</body>
</html>
```

TIPS

- Always use <label> with form controls for better accessibility.
- Use meaningful name and id for form elements.
- Use appropriate input types for better user experience.
- Validate form data using HTML5 validation and JavaScript.

BENEFITS OF HTML FORMS

- Collect user information easily.
- Improve user interaction.
- Send data to server for processing.
- Essential for login, registration, feedback, search, payment, etc.

Unit - III


DATA STRUCTURES
ALGORITHMS
COMPUTER NETWORKS
DATABASE SYSTEMS
#include <iostream>
using namespace std;
int main()
{
cout << "Hello, World!";
return 0;
}

CSS: BENEFITS OF CSS

CSS (Cascading Style Sheets) is used to style and layout web pages. It separates content (HTML) from presentation.

BENEFITS OF CSS

1. Separates Content from Design
CSS separates HTML content from its presentation, making code cleaner and easier to manage.

2. Consistent Look & Feel
CSS helps maintain a consistent design across multiple web pages of a website.

3. Saves Time and Effort
Change the style in one CSS file and it will be updated across all linked pages.

4. Improves Website Performance
CSS reduces HTML code and page size, which improves page load speed.

5. Better Responsiveness
CSS makes it easier to create responsive layouts that work well on different devices and screen sizes.

6. Easy Maintenance
Styles are written in separate CSS files, making websites easier to maintain and update.

7. Reusability
CSS classes and IDs can be reused across multiple elements and pages.

8. Enables Teamwork
Designers and developers can work separately on styles (CSS) and structure (HTML).

9. Improves Accessibility
CSS helps in creating accessible websites with better readability and user experience.

10. Print Friendly
CSS provides control over how web content looks when printed.

CSS VERSIONS HISTORY

CSS Level 1 (Recommendation) — 1996
- First official version of CSS.
- Released by W3C.
- Basic text, font, color, background, margin, padding, borders, etc.

CSS Level 2 (Recommendation) — 1998
- Released by W3C.
- Added positioning, z-index, floating, tables, advanced selectors, etc.

CSS Level 2.1 (Recommendation) — 2011
- Small update to Level 2.
- Corrected errors and improved browser compatibility.

CSS Level 3 (Module-based) — 1999 Onwards (Modules Released Gradually)
- Not a single document, but a collection of modules.
- Introduced rounded corners, shadows, gradients, transitions, animations, flexbox, media queries, etc.

CSS Level 4 (In Progress) — Future
- Next generation of CSS.
- More powerful selectors, layout features, responsive design improvements, and new values.

KEY TAKEAWAY

CSS has evolved from simple styling to a powerful language for building modern, responsive, and visually stunning websites.

QUICK COMPARISON

Version	Year	Type	Key Features	Status
CSS Level 1	1996	Recommendation	Basic styling (fonts, colors, backgrounds, text alignment)	Stable
CSS Level 2	1998	Recommendation	Positioning, floats, tables, advanced selectors	Stable
CSS Level 2.1	2011	Recommendation	Minor updates and bug fixes to Level 2	Stable
CSS Level 3	1999 onwards	Module-based	Round corners, shadows, gradients, animations, media queries, flexbox, grid, etc.	Ongoing
CSS Level 4	Future	Module-based	New selectors, layout features, and advanced capabilities	In Progress

CSS is continuously evolving to meet the demands of modern web design and deliver better user experiences.

CSS SYNTAX

CSS (Cascading Style Sheets) is used to style HTML elements on a web page.

BASIC SYNTAX

CSS rules are written using a simple syntax:

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

- **selector** — Selects the HTML element(s) to style.
- **property** — The style attribute you want to change.
- **value** — The value of the property.

EXAMPLE

HTML

```
<h1 class="title">
  Hello World!
</h1>
```

CSS

```
.title {
  color: blue;
  font-size: 24px;
}
```

OUTPUT IN BROWSER

Hello World!

WAYS TO WRITE CSS

1 Inline CSS

Used inside an HTML element using the style attribute.

```
<h1 style="color: red; font-size: 24px;">Hello</h1>
```

2 Internal (Embedded) CSS

Used inside a <style> tag in the <head> section.

```
<head>
  <style>
    p { color: green; font-size: 18px; }
  </style>
</head>
```

3 External CSS

Used in a separate .css file and linked to the HTML.

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

index.html

```
<html>
 ...
</html>
```

style.css

```
p {
  color: green;
}
```

IMPORTANT POINTS

- Each rule ends with a semicolon (;).
- Multiple properties are separated by semicolons.
- CSS is not case-sensitive.
- Spaces and line breaks are ignored, but improve readability.

EXAMPLE: MULTIPLE SELECTORS

```
h1, h2, p {
  color: navy;
  font-family: Arial;
  line-height: 1.5;
}
```

This rule will apply the same styles to all <h1>, <h2> and <p> elements.

EXAMPLE: ID AND CLASS

```
#header {
  background-color: lightblue;
  padding: 10px;
}
.btn {
  background-color: blue;
  color: white;
  padding: 8px 16px;
  border: none;
}
```

- **# for ID** (unique for one element)
- **. for Class** (can be used for multiple elements)

CSS PROPERTIES

CSS properties are used to control the appearance and layout of HTML elements.

1. TEXT PROPERTIES

Property	Description	Example
color	Sets the text color	color: blue;
font-family	Sets the font of text	font-family: Arial;
font-size	Sets the size of text	font-size: 20px;
font-weight	Sets the thickness of text	font-weight: bold;
text-align	Aligns the text	text-align: center;
text-decoration	Adds decoration to text	text-decoration: underline;
line-height	Sets the space between lines	line-height: 1.6;
text-transform	Controls capitalization	text-transform: uppercase;
letter-spacing	Sets the space between letters	letter-spacing: 2px;
word-spacing	Sets the space between words	word-spacing: 4px;
text-indent	Indents the first line	text-indent: 30px;
white-space	Controls whitespace	white-space: nowrap;

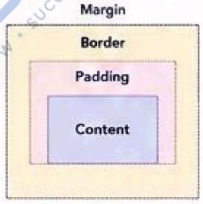
2. BACKGROUND & BORDER PROPERTIES

Property	Description	Example
background-color	Sets background color	background-color: #f0f0f0;
background-image	Sets background image	background-image: url(bg.jpg);
background-repeat	Repeats background image	background-repeat: no-repeat;
background-position	Sets position of background	background-position: center;
border	Sets border (width style color)	border: 1px solid black;
border-width	Sets border width	border-width: 2px;
border-style	Sets border style	border-style: dashed;
border-color	Sets border color	border-color: red;
border-radius	Rounds the corners	border-radius: 8px;

3. BOX MODEL PROPERTIES

Property	Description	Example
margin	Sets outer space	margin: 10px;
padding	Sets inner space	padding: 15px;
width	Sets width of element	width: 300px;
height	Sets height of element	height: 200px;
box-sizing	Defines how width & height are calculated	box-sizing: border-box;

THE BOX MODEL



4. DISPLAY & POSITION PROPERTIES

Property	Description	Example
display	Specifies display behavior	display: block;
visibility	Sets visibility of element	visibility: hidden;
position	Sets the positioning method	position: relative;
top	Moves element from top	top: 10px;
bottom	Moves element from bottom	bottom: 10px;
left	Moves element from left	left: 20px;
right	Moves element from right	right: 20px;
z-index	Sets stacking order	z-index: 1;

TIP Use CSS properties to make your web pages beautiful, responsive and user-friendly!

REMEMBER

Practice is the key to master CSS.
Experiment with different properties and values.

```
.box {
  color: blue;
  padding: 10px;
}
```

CSS

CSS SELECTORS

CSS selectors are used to "select" HTML elements and apply styles to them.

TYPES OF BASIC SELECTORS

1. UNIVERSAL SELECTOR

The universal selector (*) selects all HTML elements on the page.

Syntax

```
*
```

Example

```
* {
    margin: 0;
    padding: 0;
}
```

Example in HTML

```
<h1>Welcome</h1>
<p>This is a paragraph.</p>
<div>Box</div>
```

Output in Browser

All elements will have margin: 0; and padding: 0;

2. TYPE SELECTOR

The type selector selects all elements of a specific type.

Syntax

```
element { ... }
```

Example

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

Example in HTML

```
<h1>Welcome</h1>
<p>This is a paragraph.</p>
<p>Another paragraph.</p>
<div>Box</div>
```

Output in Browser

Welcome
This is a paragraph.
Another paragraph.
Box

3. ID SELECTOR

The ID selector selects the element with a specific id attribute. (IDs must be unique on a page.)

Syntax

```
#id { ... }
```

Example

```
#header {
    background-color: navy;
    color: white;
    padding: 15px;
}
```

Example in HTML

```
<div id="header">
  Website Header
</div>
<p>Welcome to our site.</p>
<div id="footer">Footer</div>
```

Output in Browser

Website Header
Welcome to our site.
Footer

4. CLASS SELECTOR

The class selector selects all elements with a specific class attribute. (A class can be used multiple times.)

Syntax

```
.class { ... }
```

Example

```
.box {
    border: 2px solid green;
    padding: 10px;
    margin: 10px 0;
}
```

Example in HTML

```
<div class="box">Box 1</div>
<p class="box">Box 2</p>
<span class="box">Box 3</span>
```

Output in Browser

Box 1
Box 2
Box 3

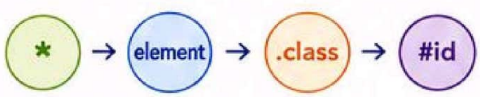
COMPARISON TABLE

Selector	Syntax	Selects	Example	Use When
Universal Selector	*	All elements on the page	* { color: red; }	You want to apply styles to all elements.
Type Selector	p	All elements of a specific type (e.g., all <p> tags)	p { color: blue; }	You want to style all elements of the same type.
ID Selector	#header	The element with a specific id (unique)	#header { background: navy; }	You want to style one unique element on the page.
Class Selector	.box	All elements with a specific class (can be multiple)	.box { border: 2px solid green; }	You want to style a group of elements.

KEY POINTS

- Universal selector (*) has the lowest specificity.
- Type selector is more specific than universal.
- Class selector is more specific than type selector.
- ID selector is the most specific among these four.

SPECIFICITY (Lowest → Highest)



QUICK EXAMPLE (All Together)

HTML

```
<div id="header" class="box">
   Welcome!
</div>
<p class="box">This is a paragraph.</p>
<span>Just a span.</span>
```

CSS

```
* { margin: 0; padding: 0; }
p { color: blue; }
#header { background: navy; color: white;
    padding: 10px; }
.box { border: 2px solid green; }
```

in Browser

Welcome!
This is a paragraph.
Just a span.

ID is used for one unique element.
Class is used for groups of elements.

CSS – WAYS TO APPLY STYLES

There are three ways to add CSS to an HTML document.

1 INLINE STYLES

Inline styles are applied directly to an HTML element using the `style` attribute.

Syntax

```
<tagname style="property: value;">Content</tagname>
```

Example

```
<p style="color: blue; font-size: 18px;
    text-align: center;">
    This is a paragraph.
</p>
```

How it works

- The style is written directly inside the HTML tag.
- It affects only that specific element.
- It has the highest priority.

Output in Browser

This is a paragraph.

Pros / Cons

- ✅ Quick and easy for small changes.

Cons

- ❌ Not reusable.
- ❌ Hard to maintain.
- ❌ Not recommended for large websites.

2 EMBEDDED STYLE SHEETS

Embedded styles are written inside the `<style>` tag within the `<head>` section of an HTML document.

Syntax

```
<style>
   selector { property: value; }
</style>
```

Example

```
<!DOCTYPE html>
<html>
<head>
  <style>
    p {
      color: green;
      font-size: 18px;
      text-align: center;
    }
  </style>
</head>
<body>
  <p>This is a paragraph.</p>
</body>
</html>
```

How it works

- The styles are written inside the <style> tag in the <head> section.
- It affects all matching elements in that single HTML page.
- It is useful when a single page has unique styles.

Output in Browser

This is a paragraph.

Pros / Cons

- ✅ Better than inline styles.
- ✅ Can style multiple elements.
- ✅ Easy to modify within the page.

Cons

- ❌ Styles cannot be reused in other pages.
- ❌ Not ideal for large websites with many pages.

3 EXTERNAL STYLE SHEETS

External styles are written in a separate `.css` file and linked to the HTML document using the `<link>` tag.

Syntax

```
/* style.css */
selector { property: value; }
```

Example

HTML File (index.html)

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet"
        href="style.css">
</head>
<body>
  <h1>Welcome!</h1>
  <p>This is a paragraph.</p>
</body>
</html>
```

CSS File (style.css)

```
h1 {
  color: darkblue;
  text-align: center;
}

p {
  color: #555;
  font-size: 18px;
  line-height: 1.6;
}
```

How it works

- CSS is written in a separate .css file.
- The file is linked to the HTML page using <link> tag.
- It affects all matching elements in all pages where the CSS file is linked.
- Changes in the CSS file reflect on all linked pages.

Output in Browser

Welcome!

This is a paragraph.

Pros / Cons

- ✅ Best for large websites.
- ✅ Reusable across multiple pages.
- ✅ Easy to maintain.
- ✅ Keeps HTML and CSS separate.

Cons

- ❌ Requires an extra request to load the CSS file (very small impact on performance).

COMPARISON SUMMARY

Method	Where Written	Affects	Reusability	Best Use	Priority
Inline Styles	Inside HTML tag (style attribute)	Single element	No	Small and quick changes	Highest (1)
Embedded Style Sheets	Inside <style> tag in <head>	All elements in the same page	No	Single page with unique styles	Medium (2)
External Style Sheets	Separate .css file linked using <link>	All linked pages	Yes	Large websites and multiple pages	Lowest (3)

TIP: Always prefer External Style Sheets for professional and scalable websites.

HTML → CSS →

Unit - IV


DATA STRUCTURES
ALGORITHMS
COMPUTER NETWORKS
DATABASE SYSTEMS
#include <iostream>
using namespace std;
int main()
{
cout << "Hello, World!";
return 0;
}

INTRODUCTION TO SCRIPTING:

JAVASCRIPT BASICS

JavaScript is a lightweight scripting language used to make web pages interactive and dynamic.

WHAT IS JAVASCRIPT?

- JavaScript is a scripting language.
- It is used to add behavior to web pages.
- It is an essential part of web development, along with HTML and CSS.
- It runs in the browser and makes web pages interactive.

WHY USE JAVASCRIPT?

- Makes web pages interactive
- Validates forms and user input
- Updates content without reloading
- Creates animations and effects
- Works with APIs and servers

WHERE DOES JAVASCRIPT RUN?

HTML + JS →

Browser reads HTML, uses JavaScript to add behavior and display interactive content.

JAVASCRIPT BASICS

1. JAVASCRIPT IN HTML

JavaScript is embedded to HTML using the `<script>` tag.

Syntax

```
<script>
   // JavaScript code
</script>
```

Example

```
<script>
   document.write("Hello, JavaScript!");
</script>
```

Output in Browser

```
Hello, JavaScript!
```

2. VARIABLES

Variables are used to store data values.

Syntax

```
let x = value;   // block-scoped
var y = value;   // function-scoped
const z = value; // constant (cannot be changed)
```

Example

```
let name = "Shahid";
var age = 19;
const pi = 3.14;
```

Output (via console)

```
name = Shahid
age = 19
pi = 3.14
```

3. DATA TYPES

JavaScript has different data types.

Common Data Types

Type	Example	Description
String	"Hello"	Text value
Number	25, 3.14	Numeric value
Boolean	true, false	True or False
Undefined	undefined	Variable declared but not assigned
Null	null	Intentional absence of value
Object	{ }	Collection of key-value pairs
Array	[1,2,3]	Ordered list

4. OPERATORS

Operators are used to perform operations.

Arithmetic Operators

Operator	Description	Example	Result
+	Addition	5 + 2	7
-	Subtraction	5 - 2	3
*	Multiplication	5 * 2	10
/	Division	5 / 2	2.5
%	Modulus	5 % 2	1

Example

```
let a = 10,  b = 3;
console.log(a + b);  // 13
```

5. COMMENTS

Comments are ignored by the browser.

Syntax

```
// This is a single-line comment
/*
  This is a
  multi-line comment
*/
```

Example

```
let x = 5; // x stores 5
/*
  This program shows
  JavaScript comments.
*/
```

6. OUTPUT METHODS

Used to display output to the user.

Methods

Method	Description	Example
alert()	Shows an alert box	alert("Hi!");
document.write()	Writes to HTML	document.write("Hi");
console.log()	Writes to console	console.log("Hi");
innerHTML	Displays in an HTML element	document.getElementById ("demo").innerHTML = "Hi";

Example (console.log)

```
console.log("Hello in console");
```

Output (via console)

```
Hello in console
```

KEY POINTS

- JavaScript is a scripting language used in web development.
- It makes web pages interactive and dynamic.
- It works together with HTML and CSS.
- It runs in the browser.
- Basic concepts include variables, data types, operators, comments and output methods.

JavaScript is the foundation of modern web applications and is used everywhere!

SIMPLE JAVASCRIPT EXAMPLE

HTML

```
<!DOCTYPE html>
<html>
<head>
  <title>My First JS</title>
</head>
<body>
  <h2 id="demo">Original Text</h2>
  <button onclick="changeText()">
    Click Me
  </button>

  <script>
    function changeText(){
      document. getElementById("demo").
        innerHTML = "Text Changed!";
      }
  </script>

</body>
</html>
```

JavaScript

OUTPUT IN BROWSER

My First JS

Original Text

Click Me

↓

My First JS

Text Changed!

Click Me

JAVASCRIPT BASICS
OPERATORS, DATA TYPES, POPUP BOXES

JavaScript provides various operators to perform operations, different data types to store values, and popup boxes to interact with users.

1. OPERATORS

Operators are symbols used to perform operations on values and variables.

Type	Operator	Description	Example	Result
Arithmetic Operators	+	Addition	5 + 3	8
	-	Subtraction	5 - 3	2
	*	Multiplication	5 * 3	15
	/	Division	10 / 2	5
	%	Modulus (Remainder)	10 % 3	1
	**	Exponentiation	2 ** 3	8
	++	Increment	x++	x = x + 1
	--	Decrement	x--	x = x - 1
Assignment Operators	=	Assign	x = 10	10
	+=	Add and assign	x += 5	x = x + 5
	-=	Subtract and assign	x -= 5	x = x - 5
	*=	Multiply and assign	x *= 2	x = x * 2
	/=	Divide and assign	x /= 2	x = x / 2
	%=	Modulus and assign	x %= 3	x = x % 3
Comparison Operators	==	Equal to	5 == '5'	true
	===	Strict equal to	5 === '5'	false
	!=	Not equal to	5 != 3	true
	!==	Strict not equal to	5 !== '5'	true
	>	Greater than	7 > 3	true
	<	Less than	2 < 5	true
	>=	Greater than or equal	7 >= 7	true
	<=	Less than or equal	4 <= 2	false
Logical Operators	&&	Logical AND	(5 > 3) && (2 < 3)	true
	||	Logical OR	(5 < 3) || (2 < 3)	true
	!	Logical NOT	!(5 > 3)	false
String Operators	+	Concatenation	"Hello" + " JS"	"Hello JS"
	+=	Concatenate and assign	str += " World"	str = str + " World"
Conditional (Ternary) Operator	?	Ternary operator	age >= 18 ? "Adult" : "Minor"	Adult or Minor

EXAMPLE

```
let a = 10, b = 5;
console.log(a + b);   // 15
console.log(a % b);   // 0
console.log(a > b && b < 10); // true
console.log(a > b ? "A is greater" : "B is greater");
// Output: A is greater
```

2.DATA TYPES

Data types define the type of value a variable can hold.

Type	Description	Example	Example Value
String	Represents text	let name = "Shahid";	"Shahid"
Number	Represents numbers (integer or float)	let age = 19;	19 3.14
BigInt	Represents large integers	let big = 1234567890123456789n;	1234567890... (BigInt)
Boolean	Represents true or false	let isStudent = true;	true / false
Undefined	Variable declared but no value assigned	let x;	undefined
Null	Represents no value or empty value	let y = null;	null
Symbol	Unique and immutable value	let id = Symbol("id");	Symbol(id)
Object	Collection of key-value pairs	let person = {name:"Ali", age:20};	{name:"Ali", age:20}
Array	Ordered list of values	let arr = [1, 2, 3];	[1, 2, 3]
Function	Block of code that performs a task	function add(a,b){ return a+b; }	add(a,b){}

EXAMPLE

```
let name = "Shahid";          // String
let age = 19;                 // Number
let isStudent = true;         // Boolean
let x;                        // Undefined
let y = null;                 // Null
let arr = [10, 20, 30];       // Array
letperson = {name: "Ali", age: 20}; // Object
function greet(){ return "Hello!"; } // Function
```

3. POPUP BOXES (DIALOG BOXES)

Popup boxes are used to display messages, take input or show confirmations to users.

1. ALERT BOX

Displays a message with an OK button.

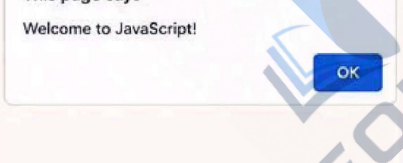
Syntax

```
alert("message");
```

Example

```
alert("Welcome to JavaScript!");
```

Output



Use

To show information or messages to the user.

2. CONFIRM BOX

Displays a message with OK and Cancel buttons.
Returns true if OK is clicked, otherwise false.

Syntax

```
confirm("message");
```

Example

```
let result = confirm("Do you want to proceed?");
console.log(result);
```

Output (if OK is clicked)

Output (if Cancel is clicked)

Use

To take user confirmation (Yes/No).

3. PROMPT BOX

Displays a message with an input field, OK and Cancel buttons.
Returns the input value if OK is clicked.

Syntax

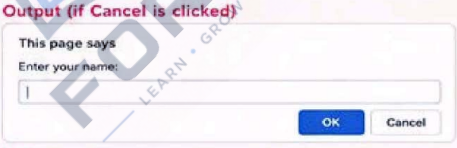
```
prompt("message", "default value");
```

Example

```
let name = prompt("Enter your name:", "Guest");
console.log(name);
```

Output (if user enters "Shahid")

Output (if Cancel is clicked)



Use

To take input from the user.

QUICK SUMMARY

OPERATORS
Used to perform operations on values and variables.

DATA TYPES
Define the type of data that can be stored.

POPUP BOXES
Used to display messages, take input or confirmation.

JAVASCRIPT CONTROL STRUCTURES
IF, IF-ELSE, SWITCH

Control structures allow you to make decisions and execute code blocks based on different conditions.

1. IF STATEMENT

The if statement executes a block of code only if the specified condition is true.

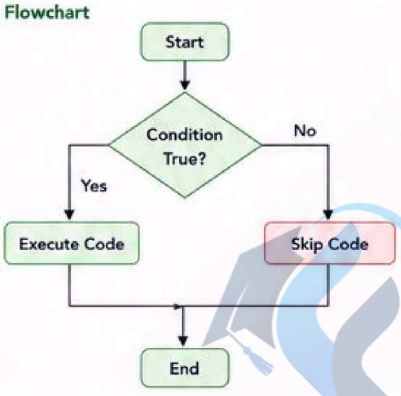
Syntax

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

Example

```
let age = 18;
if (age >= 18) {
   console.log("You are eligible to vote.");
}
```

Flowchart



Output (in console)

```
You are eligible to vote.
```

2. IF-ELSE STATEMENT

The if block executes if the condition is true, otherwise the else block executes.

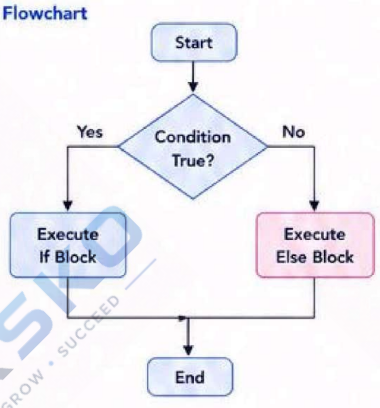
Syntax

```
if (condition) {
   // code if condition is true
} else {
   // code if condition is false
}
```

Example

```
let num = 7;
if (num % 2 == 0) {
   console.log("Even number");
} else {
   console.log("Odd number");
}
```

Flowchart



Output (in console)

```
Odd number
```

3. SWITCH STATEMENT

The switch statement is used to execute one block of code from many alternatives.

Syntax

```
switch (expression) {
   case value1:
      // code to be executed
      break;
   case value2:
      // code to be executed
      break;
   default:
      // code to be executed
}
```

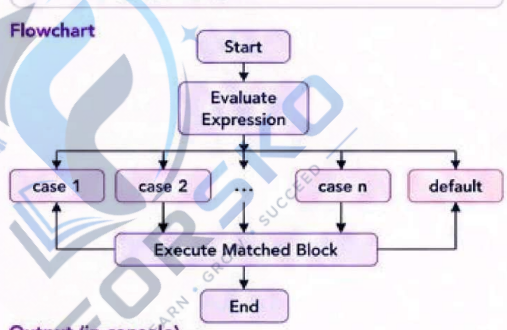
Example

```
let day = 3;
let dayName;

switch (day) {
   case 1: dayName = "Monday"; break;
   case 2: dayName = "Tuesday"; break;
   case 3: dayName = "Wednesday"; break;
   case 4: dayName = "Thursday"; break;
   case 5: dayName = "Friday"; break;
   default: dayName = "Weekend";
}

console.log(dayName);
```

Flowchart



Output (in console)

```
Wednesday
```

COMPARISON TABLE

Feature	If Statement	If-Else Statement	Switch Statement
Purpose	Executes code if condition is true.	Executes one block if true, another if false.	Selects one block from many choices.
Conditions	One condition	One condition (true/false)	Multiple possible values
Best Use	Simple condition check	Two possible outcomes	Many discrete choices
Supports Ranges	Yes (with operators)	Yes (with operators)	No (checks equality only)
Break Required	No	No	Yes (to avoid fall-through)
Default Option	No	No	Yes (default case)

EXAMPLES TO UNDERSTAND

IF Example

```
let marks = 85;

if (marks >= 50) {
   console.log("You passed!");
}
```

Output:
You passed!

IF-ELSE Example

```
let num = -4;
if (num > 0) {
   console.log("Positive");
} else {
   console.log("Not positive");
}
```

Output:
Not positive

SWITCH Example

```
let grade = 'B';
switch (grade) {
   case 'A': console.log("Excellent"); break;
   case 'B': console.log("Good"); break;
   case 'C': console.log("Average"); break;
   default: console.log("Try harder");
}
```

Output:
Good

KEY POINTS

- Use if for a single condition.
- Use if-else for two possible outcomes.
- Use switch when you have multiple fixed options.
- Always use break in switch to prevent fall-through.
- Control structures make your code smart and dynamic!

TIP Choose the right control structure based on the problem. It improves readability and performance of your code.

DECISION MAKES CODE SMARTER! ? → </> → ✓ → Better Programs

JAVASCRIPT LOOPING STRUCTURES
for, do-while, while

Loops are used to execute a block of code repeatedly as long as a condition is true.

1. FOR LOOP

The for loop is used when you know how many times the loop should run.

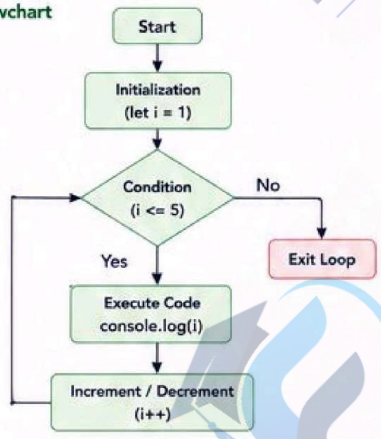
Syntax

```
for (initialization; condition; increment/decrement) {
   // code to be executed
}
```

Example

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

Flowchart



Output (in console)

```
1
2
3
4
5
```

2. DO-WHILE LOOP

The do-while loop executes the block of code at least once, then repeats if the condition is true.

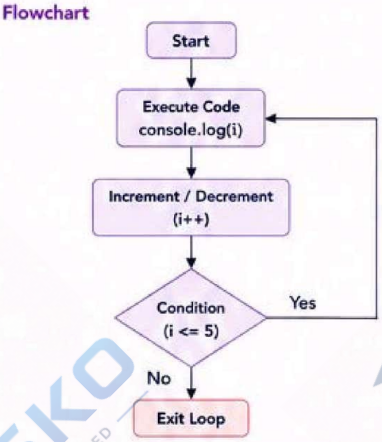
Syntax

```
do {
   // code to be executed
} while (condition);
```

Example

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

Flowchart



Output (in console)

```
1
2
3
4
5
```

3. WHILE LOOP

The while loop repeats the block of code as long as the condition is true.

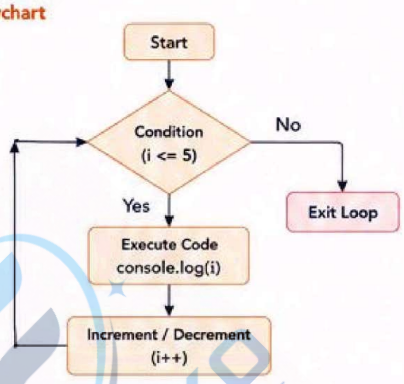
Syntax

```
while (condition) {
   // code to be executed
}
```

Example

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

Flowchart



Output (in console)

```
1
2
3
4
5
```

COMPARISON TABLE

Feature	for Loop	do-while Loop	while Loop
When to use	When number of iterations is known.	When at least one execution is required.	When number of iterations is unknown.
Checks condition	Before each iteration	After each iteration	Before each iteration
Executes atleast once?	No (may execute 0 times)	Yes (executes at least once)	No (may execute 0 times)
Initialization	In the loop statement	Before the loop	Before the loop
Increment / Decrement	In the loop statement	Inside the loop	Inside the loop
Best for	Counting / definite loops	Menu, input validation, etc.	Conditional / indefinite loops

PRACTICAL EXAMPLES

FOR LOOP – Sum of first 5 numbers

```
let sum = 0;
for (let i = 1; i <= 5; i++) {
   sum += i;
}
console.log("Sum =", sum);
```

Output

```
Sum = 15
```

DO-WHILE LOOP – Menu Example

```
let choice;
do {
   choice = prompt("Enter 1 to continue
      or 0 to exit: ");
} while (choice != 0);
console.log("You exited the menu.");
```

Output

```
You exited the menu.
```

WHILE LOOP – Countdown

```
let n = 5;
while (n >= 1) {
   console.log(n);
   n--;
}
console.log("Blast off!");
```

Output

```
5
4
3
2
1
Blast off!
```

KEY POINTS

- Use for loop when you know how many times to repeat.
- Use do-while when the block must run at least once.
- Use while loop when the number of iterations is not known.
- Choose the right loop to make your code efficient and readable.

{ } Loops save time, write less code, do more!

TIP: Infinite loops can crash your program! Always make sure your loop condition will eventually become false.

General Structure Reminder

for

```
for (init; condition; inc/dec) {
   // code
}
```

do-while

```
do {
   // code
} while (condition);
```

while

```
while (condition) {
   // code
}
```