UNIT-I: Basics of Internet and Web

- **Internet**: A global network connecting millions of computers for data and resource sharing.
- **Uses**: Communication (email, chat), research, e-commerce, entertainment, education.

🌍 2. World Wide Web (WWW)

- **WWW**: A system of interlinked hypertext documents accessed via the Internet using web browsers.
- **Difference from Internet**: Internet is the infrastructure; WWW is a service over it.
- Uses protocols like **HTTP**, **HTTPS**.

📄 3. Web Page, Home Page, Website

- **Web Page**: A single document on the web written in HTML.
- **Home Page**: The main or introductory page of a website.
- **Website**: A collection of related web pages hosted under a domain name.

🔄 4. Static, Dynamic, and Active Web Pages

Type	Description	Technologies Used
Static	Fixed content, same for all users.	HTML, CSS
Dynamic	Content changes based on user input or server data.	PHP, ASP.NET, JavaScript
Active	Client-side interaction, runs scripts locally.	JavaScript, VBScript, ActiveX

Type	Description	Technologies Used

◆ 4. Static, Dynamic, and Active Web Pages

1. Static Web Pages

- **Definition**: A static web page displays the same content for every user and does not change unless manually edited by the developer.
- **Characteristics**:
  - Content remains **constant** unless changed in the source code.
  - Each page is saved as an **individual HTML file**.
  - No interaction with databases or external scripts.
  - Faster to load because they do not require processing by the server.
- **Technologies Used**:
  - **HTML (Hypertext Markup Language)** – for structure.
  - **CSS (Cascading Style Sheets)** – for styling and layout.
- **Example**: A company's About Us page or a personal portfolio without forms or data input.

---

2. Dynamic Web Pages

- **Definition**: A dynamic web page changes its content or layout based on user interactions, server-side processing, or data from a database.
- **Characteristics**:
  - Content is **generated in real-time**.
  - Can interact with **databases** to fetch or update content.
  - Personalized user experiences (e.g., login systems, shopping carts).
  - Requires **server-side scripting languages**.
- **Technologies Used**:
  - **PHP** – Server-side scripting language.

Type	Description	Technologies Used

  - **ASP.NET** – Microsoft’s web development framework.
  - **JavaScript** – Often used for dynamic interaction on the client-side.
  - **Databases** – MySQL, Oracle, SQL Server, etc.
- **Example**: Social media feeds, email inboxes, content management systems (CMS).

---

3. Active Web Pages

- **Definition**: A web page that includes **client-side scripts** allowing it to respond to user actions without reloading the entire page.
- **Characteristics**:
  - Runs scripts **locally** in the user's browser.
  - Provides a **rich and interactive** user experience.
  - May update part of the web page dynamically (e.g., AJAX calls).
  - Often uses **event-driven programming** (e.g., onClick, onHover).
- **Technologies Used**:
  - **JavaScript** – Primary language for client-side scripting.
  - **VBScript** – Supported only in Internet Explorer (obsolete now).
  - **ActiveX Controls** – Microsoft’s technology for interactive content (now deprecated).
- **Example**: Form validation, dropdown animations, interactive maps.

Comparison Table

Type	Description	Technologies Used
Static	Fixed content, same for all users	HTML, CSS

Type	Description	Technologies Used
Dynamic	Changes based on user or server-side data	PHP, ASP.NET, JavaScript, MySQL
Active	Responds to user interaction, runs scripts locally	JavaScript, VBScript, ActiveX (deprecated)

✅ Summary Points for Students

- **Static** = Simple, fast, no interactivity.
- **Dynamic** = Interactive, server-processed content.
- **Active** = Interactive in-browser experience with scripts.

📡 5. Overview of Internet Protocols

Important Protocols:

Protocol	Description
HTTP (Hypertext Transfer Protocol)	Used to access web pages.
SMTP (Simple Mail Transfer Protocol)	Sends emails.
POP3/IMAP	Receives emails.
FTP (File Transfer Protocol)	Transfers files between computers.
TFTP (Trivial FTP)	Simplified FTP for fast file transfers.
Telnet	Remote login to another computer.
Gopher	A text-based file retrieval protocol (obsolete).
SNMP (Simple Network Management Protocol)	Manages and monitors network devices.

Protocol	Description

5. Overview of Internet Protocols

◈ What is a Protocol?

- A **protocol** is a set of **rules and standards** that define how data is transmitted and received over a network.
- In networking, protocols are essential for **communication between devices** like computers, routers, and servers.

🌐 Important Internet Protocols

Protocol	Full Form	Purpose
HTTP	HyperText Transfer Protocol	Used to transfer web pages over the web.
HTTPS	HTTP Secure	A secure version of HTTP using encryption (SSL/TLS).
SMTP	Simple Mail Transfer Protocol	Sends emails from a client to a server or between servers.
POP3	Post Office Protocol version 3	Downloads email from the server to the client, then deletes it from server.
IMAP	Internet Message Access Protocol	Synchronizes and manages email across multiple devices.
FTP	File Transfer Protocol	Transfers files between two computers on a network.
TFTP	Trivial File Transfer Protocol	Simplified FTP; used for quick, small file transfers (e.g., boot files).
Telnet	--	Provides remote terminal access to another computer (unsecured).
Gopher	--	Early protocol for retrieving documents; menu-driven (now obsolete).

Protocol		Description
SNMP	Simple Network Management Protocol	Monitors and manages network devices like routers, switches, servers.

□ Detailed Explanation of Key Protocols

---

1. HTTP (HyperText Transfer Protocol)

- **Purpose**: Used by browsers to communicate with web servers and fetch web pages.
- Works over **port 80** (HTTP) or **port 443** (HTTPS).
- Transmits data in **plain text** (unless HTTPS is used).
- **Stateless**: Each request is treated independently.

2. SMTP (Simple Mail Transfer Protocol)

- Used to **send** emails from the sender's client to the mail server.
- Also transfers email between **mail servers**.
- Operates over **port 25**, or **587** for secure sending.
- Works with **POP3** or **IMAP** for receiving.

3. POP3 (Post Office Protocol v3)

- Downloads emails from the server and **deletes** them afterward.
- Useful for reading mail **offline**.
- Operates over **port 110** (or 995 for secure POP3).
- Limitation: Not suitable for accessing mail from multiple devices.

4. IMAP (Internet Message Access Protocol)

- Keeps emails on the **server** and syncs them across devices.
- Good for accessing email from **multiple devices**.
- Operates on **port 143** (or 993 for secure IMAP).

5. FTP (File Transfer Protocol)

- Transfers files between systems (e.g., client and server).
- Requires login (username/password).
- Two modes: **Active** and **Passive**.

Protocol **Description**

- Operates on **port 21**.

6. TFTP (Trivial File Transfer Protocol)

- Simpler version of FTP, with **no authentication**.
- Used for **fast, small transfers** (e.g., booting OS images).
- Operates on **port 69**.
- Not secure or reliable for critical transfers.

7. Telnet

- Allows **remote login** to another computer over the internet.
- Text-based interface (command-line).
- Operates on **port 23**.
- Lacks encryption – replaced mostly by **SSH (Secure Shell)**.

8. Gopher

- Early, **menu-driven** system for retrieving files.
- Text-based, predates the World Wide Web.
- Now **obsolete**, but historically important.

9. SNMP (Simple Network Management Protocol)

- Used by **network administrators** to manage network devices.
- Can collect statistics (like traffic, errors, uptime).
- Operates on **UDP ports 161 (agent) and 162 (manager)**.
- Devices like routers, switches, and firewalls support SNMP.

📌 Comparison Table

Protocol	Use Case	Secure?	Ports
HTTP	Web browsing	✗	80
HTTPS	Secure web	✓	443
SMTP	Sending email	✗ / ✓	25 / 587
POP3	Receiving email	✗ / ✓	110 / 995

Protocol	Description		
IMAP	Receiving email	✓	143 / 993
FTP	File transfer	✗ / ✓	21
TFTP	Simple file transfer	✗	69
Telnet	Remote login	✗	23
SNMP	Network monitoring	✗	161, 162

✓ Key Takeaways

- Protocols are **communication rules** for the internet.
- Each protocol serves a **specific purpose** (email, browsing, file transfer, etc.).
- Use **secure versions** where possible (HTTPS, IMAPS, etc.).
- Understanding protocols is essential for **networking and web development**.

🖥️ 6. Client-Server Computing Concepts

- **Client**: Requests services/data (e.g., browser).
- **Server**: Provides services/data (e.g., web server).
- Communication via **request-response** model.

◈ Introduction to Client-Server Computing

Client-Server computing is a **network architecture** in which **multiple clients** (users or devices) request and receive services from a **centralized server**.

It is the **foundation of the internet**, used in web browsing, email, databases, file sharing, etc.

◆ Key Components

Component	Role
Client	A device or application that **requests** data or services (e.g., browser).
Server	A powerful system or application that **provides** data or services (e.g., web server).

◆ Client

- A **software application** or **device** that initiates a request to the server.
- Examples:
  - Web browsers like Chrome, Firefox, Edge.
  - Email clients like Outlook or Thunderbird.
  - Apps like WhatsApp Web, Zoom, etc.
- Role:
  - Sends requests to the server using protocols like **HTTP**, **FTP**, or **SMTP**.
  - Waits for and processes the server's response.

◆ Server

- A **powerful computer or software** system that listens for and responds to client requests.
- Types of servers:
  - **Web Server**: Delivers web pages (e.g., Apache, Nginx).
  - **Mail Server**: Handles email sending and receiving.
  - **Database Server**: Provides access to database systems (e.g., MySQL, Oracle).
  - **File Server**: Allows clients to store or retrieve files.
- Role:
  - Handles multiple clients **simultaneously**.
  - Ensures **security**, **data integrity**, and **resource management**.

◆ How Client-Server Communication Works

✅*Request-Response Model (Basic Workflow):*

1. **Client sends a request** to the server (e.g., request for a web page).
2. **Server processes the request** and sends back a **response** (e.g., HTML page).
3. Client **receives and displays** the result.

◆ Features of Client-Server Architecture

- **Centralized Control**: Server manages all resources and data.
- **Scalability**: Can serve multiple clients at once.
- **Security**: Centralized server can enforce authentication and encryption.
- **Data Sharing**: All clients access shared data stored on the server.
- **Cost-Effective**: Clients don't need powerful hardware; server handles processing.

◆ Advantages

- Efficient resource sharing.
- Easier to maintain and update software centrally.
- Supports **multiple clients simultaneously**.
- Enhanced **data security** and **backup**.

◆ Disadvantages

- If the server fails, all client services may be interrupted.
- Requires a **high-performance server** to handle heavy traffic.
- More **complex** than peer-to-peer architecture.

◆ Real-Life Examples

Scenario	Client	Server
Accessing a Website	Browser (Chrome)	Web Server (Apache)
Sending an Email	Outlook	Mail Server
Accessing Online Banking	Banking App	Banking Server
Watching YouTube	YouTube App	YouTube Server

📝 Summary

- **Client-Server Computing** is essential for all modern web and app systems.
- It follows a **request-response model**.
- Clients initiate requests; servers provide responses and services.
- Scalable, manageable, and secure – ideal for large systems.

🌐 7. Web Client and Web Server

- **Web Client**: User device or browser that requests data.
- **Web Server**: Hosts websites, sends web pages to clients (e.g., Apache, IIS).

❖ Introduction

In web technology, two key components interact over the internet to display web content:

- **Web Client** – The device or software that makes the request.
- **Web Server** – The system that processes the request and returns the appropriate response (typically a web page).

This is a part of the **Client-Server Architecture** used for internet communication.

□💻 1. Web Client

✅ Definition:

A **web client** is an application (typically a **web browser**) or device used by the **end user** to request content (e.g., a website or image) from a **web server**.

✅ Common Web Clients:

- **Web browsers**:
  - Google Chrome
  - Mozilla Firefox
  - Microsoft Edge
  - Safari

- **Mobile browsers or apps** that access web services

✅ Functions:

- Sends **HTTP/HTTPS requests** to the server
- Receives **HTML, CSS, JavaScript** content in response
- Renders and displays content for the user
- May store temporary data in **cookies**, **cache**, or **local storage**

✅ How it works:

1. User enters a URL (e.g., `www.example.com`)
2. Web client sends a **request** to the server
3. Waits for the server's **response**
4. Displays the web page content received from the server

🖥 2. Web Server

✅ Definition:

A **web server** is a **computer system or software** that hosts websites and delivers web pages or content to clients over the internet.

✅ Examples of Web Server Software:

- **Apache HTTP Server** (open source)
- **IIS (Internet Information Services)** by Microsoft
- **Nginx** (high-performance)
- **LiteSpeed**, **Tomcat** (Java-based)

✅ Functions:

- Listens for incoming **HTTP/HTTPS requests**
- Processes these requests (e.g., retrieve files or run server-side scripts)
- Sends **responses** to the web client
- May support server-side languages like **PHP**, **ASP.NET**, **Python**, etc.
- Handles **security**, **load balancing**, **error handling**, etc.

✅ Resources Hosted by a Web Server:

- HTML pages
- CSS stylesheets
- JavaScript files
- Images and videos
- Server-side scripts and databases

🔄 Interaction Between Web Client and Web Server

✅ Request-Response Cycle:

1. **Client** sends a request to server using HTTP/HTTPS
2. **Server** receives the request and processes it
3. **Server** sends back a response (web page, image, data, etc.)
4. **Client** displays the content in a browser

🔁 Example: Accessing a Website

Step	Action
1	User types `www.example.com` in browser
2	Browser (client) sends HTTP request to the web server
3	Web server receives the request
4	Server sends HTML content as a response
5	Browser renders the page for the user

📝 Comparison Table

Feature	Web Client	Web Server
Role	Sends requests	Responds to requests
Initiated By	End-user	System administrator
Example Tools	Chrome, Firefox, Safari	Apache, IIS, Nginx
Location	User's device	Hosting server or cloud platform
Protocol Used	HTTP/HTTPS	HTTP/HTTPS

✅ Key Takeaways

- A **web client** is typically a **browser** that sends requests.
- A **web server** is software/system that **hosts and delivers** content.
- The two communicate over the internet using the **HTTP protocol**.
- Understanding their interaction is essential in **web development and deployment**.

🌎 8. Web Browsers

- Software to view and interact with web pages.
- Examples:
  - **Netscape Navigator** (historic)
  - **Internet Explorer** (now Edge)
  - **Mozilla Firefox**
  - **Google Chrome**
  - **Safari**

□ 9. Client-Side Scripting Languages

Used to create interactive websites; runs in browser.

Examples:

- **JavaScript**: Most popular scripting language.
- **VBScript**: Microsoft technology, mostly outdated.
- **ActiveX Controls**: Software components used to add functionality (e.g., multimedia), mostly deprecated due to security risks.

🔌 10. Plug-ins

- Software added to browsers to support additional content types (e.g., Flash, PDF viewers).
- Many modern browsers now integrate these features or use HTML5.

🏗 11. Web Server Architecture

- Handles client requests and serves web content.
- Components:
  - **Web Server Software** (Apache, Nginx)
  - **Application Server**
  - **Database Server**

🗺 12. Image Maps

- An image with clickable regions (hotspots).
- Used in navigation and interactive graphics.
- Defined using `<map>` and `<area>` tags in HTML.

💻 13. CGI (Common Gateway Interface)

- A standard protocol for web servers to execute external programs (scripts) and generate dynamic content.
- Languages: Perl, Python, Shell Script.

□ 14. API (Application Programming Interface)

- A set of functions and protocols that allow interaction between software applications.
- Used for:
  - Third-party integrations (e.g., payment gateways)
  - Data retrieval (e.g., weather data)

🗃 15. Web Database Connectivity

- Connects web applications to databases.

Common Methods:

- **ODBC** (Open Database Connectivity):
  Standard API for accessing DBMS.
- **JDBC** (Java Database Connectivity):
  API for connecting Java apps to databases.

✓ Summary Table

Concept	Key Points
Internet	Global network
WWW	Web pages over Internet
Web Page	HTML document
Web Browser	Tool to access web pages
Client-Server	Request-Response architecture
Protocols	HTTP, SMTP, FTP, SNMP etc.
Scripting	JavaScript, VBScript
Web Server	Hosts sites, delivers content
Connectivity	ODBC, JDBC for DB access