INTRODUCTION TO WEB

WHAT IS WEB DEVELOPMENT

Web development is the process of creating and maintaining websites and web applications that are accessible on the internet. It involves a range of tasks, from designing the visual layout to writing the code that makes the site function. Basically, it's how websites are built and kept running smoothly.

TYPES OF WEB DEVELOPMENT

- **Frontend Development (Client-Side)**
- **Backend Development (Server-Side)**
- **Full-Stack Development**

FRONTEND DEVELOPMENT (CLIENT-SIDE)

What it is: Everything the user sees and interacts with in the browser.

Technologies:

- HTML (structure)
- CSS (design, layout, responsiveness)
- JavaScript (interactivity, dynamic content)
- Frameworks/Libraries: React, Angular, Vue, Svelte

Responsibilities:

- Designing responsive UIs
- Improving performance and accessibility
- Handling user interactions (buttons, forms, animations)

BACKEND DEVELOPMENT (SERVER-SIDE)

What it is: The logic, database, and server-side part of a web app that users don't see.

Technologies:

- Languages: Node.js, Python, Java, PHP, C#, Go, Ruby
- Frameworks: Express.js, Django, Flask, Spring Boot, Laravel, ASP.NET
- Databases: MySQL, PostgreSQL, MongoDB, Redis

Responsibilities:

- Processing client requests
- Managing databases and APIs
- Implementing authentication & authorization
- Ensuring security and scalability

FULL-STACK DEVELOPMENT

What it is: A combination of **frontend + backend**. Full-stack developers can build the complete web application.

Technologies:

- Frontend + Backend tech stacks (MERN, MEAN, PERN, LAMP, etc.)

Responsibilities:

- Building user interfaces
- Handling server logic and databases
- Deploying and maintaining applications

WEB PAGES

A web page is a single document that is displayed through a web browser. It is a part of a website and can include text, images, videos, links, and interactive elements. Web pages are primarily written in HTML (HyperText Markup Language), styled with CSS (Cascading Style Sheets), and have dynamic elements with JavaScript.

A web page shows text, images, and videos to provide information.

It includes elements like forms and buttons that users can click or fill out.

A web page has links that let users move between different pages or sections.

WEB SITES

- A website is a collection of interconnected documents, called web pages, that are stored on a web server and made accessible to people over the internet using a web browser. Web pages are connected with each other through hyperlinks.
- The first page of a website is usually the homepage, which acts like the main entrance or starting point, guiding visitors to other sections of the site.
- Websites can serve many purposes, such as providing information, facilitating communication, offering online shopping, delivering entertainment, supporting education, or enabling social networking.

For example: Wikipedia, BBC News .

WEB APPLICATIONS

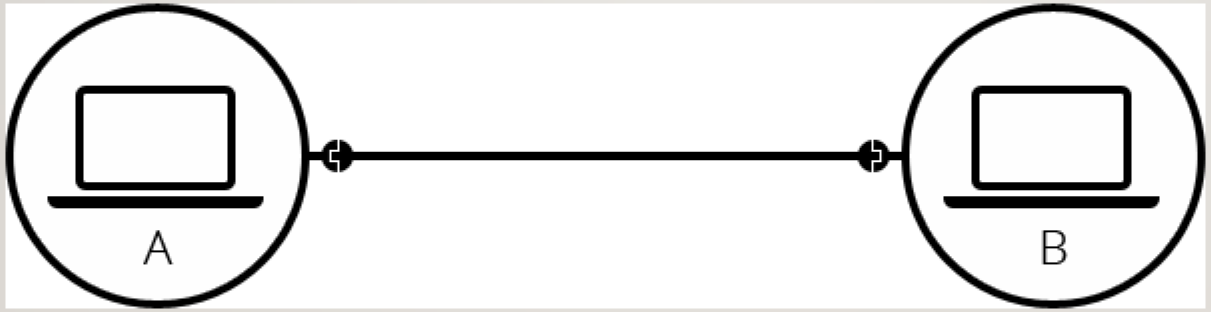
- A Web Application (Web App) is a software program or application that runs on a web server and is accessed by users through a web browser over the internet. Unlike traditional desktop applications that must be installed on each individual device, a web application runs within a web browser and can be used on multiple platforms and devices without installation.
- Web applications are interactive and dynamic in nature, meaning they allow collaborating in real time. They combine the structure and presentation of websites with the functionality of software applications.

Examples : Amazon, Email , Youtube.

Feature	Website	Web Application
Purpose	Provides information	Allows user interaction and functionality
Interactivity	Mostly static (read-only)	Dynamic and interactive
User Involvement	Users only view content	Users can input data and perform actions
Functionality	Displays text, images, videos, and links	Offers features like forms, authentication, and data processing
Examples	Blogs, news sites, company websites	Online banking, social media, e-commerce platforms
Development Complexity	Simpler, mainly uses HTML, CSS, and basic JavaScript	More complex, involving backend logic, databases, and APIs
Updates	Content updates occasionally	Requires regular updates for new features and improvements
Authentication	Usually not required	Often requires login and user authentication

NETWORK

- A network is a system of interconnected devices that communicate and share resources with each other either physically or wirelessly.



INTERNET

Internet = Interconnected Networks

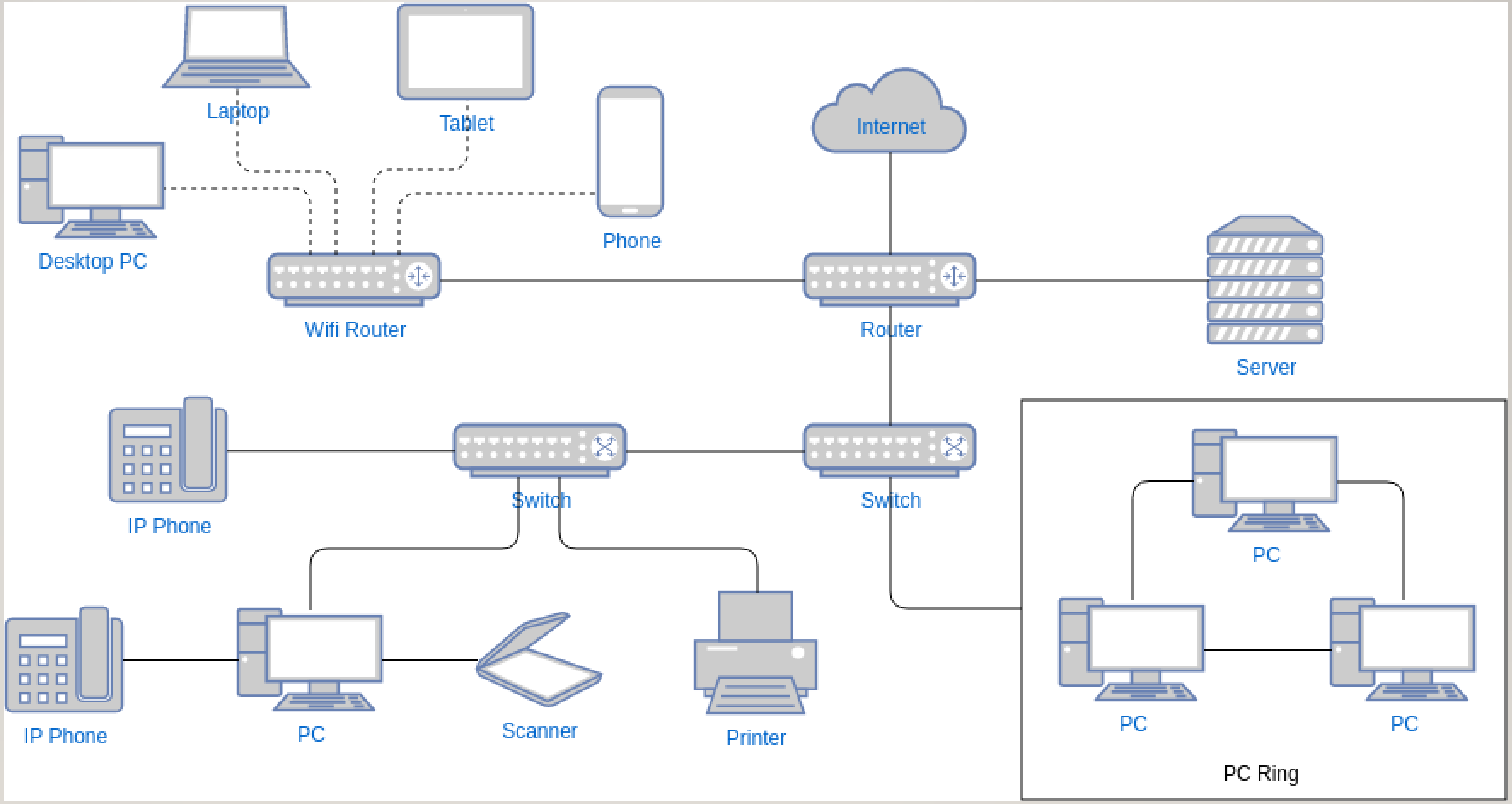
Inter → means connections between many different networks.

Net → short form of **network**. A group of interconnected computers or devices that share information.

The **Internet** is a **global system of interconnected computer networks** that connects billions of devices worldwide. It enables **communication, data exchange, and information sharing** across the globe.

It facilitates various online services like email, websites, social media, and more. The Internet's core is a system of interconnected networks that allow data to travel across vast distances.

INTERNET


Laptop
Tablet
Desktop PC
Phone
Internet
Wifi Router
Router
Server
IP Phone
Switch
Switch
PC
IP Phone
PC
Scanner
Printer
PC
PC
PC Ring

WORKING OF INTERNET

- Firstly, you'll be required to connect your system or PC with internet to establish a connection.
- When you open the browser and start typing something like "www.google.com", your system will push a query command to your ISP (Internet Service Provider) that is connected with other servers that store and process data.

- Now, the web browser will fetch the details in numeric format in their language to identify the address that you're trying to reach.
- Next, now your browser will start sending the HTTP request where you're trying to reach and send a copy of the website on the user system.

  Note : *The server will send data in the form of small packets (from the website to the browser)*
- Once all the data (of small packets) is received at the user's end (PC/Laptop), the browser will start arranging all those small packets and later will form a collective file (here, the browser will gather all the small packets and rearrange them just like a puzzle) and then you'll be able to see the contents of that website.

History of Internet

EARLY DEVELOPMENT (1960S – 1970S)

1960s:

- The idea of connecting computers together began.
- The U.S. Department of Defense created **ARPANET (Advanced Research Projects Agency Network)** in 1969.
    - ARPANET was the **first computer network** that allowed researchers to share information.

1970s:

- Development of **TCP/IP protocol** (Transmission Control Protocol / Internet Protocol).
- This became the **foundation of the Internet**, standardizing how computers communicate.

THE BIRTH OF THE INTERNET (1980S)

- ARPANET expanded and connected universities, research labs, and government organizations.
- In 1983, ARPANET officially switched to **TCP/IP**, marking the official **start of the Internet**.
- **Domain Name System (DNS)** introduced in 1984 → allowed easy names like www.example.com instead of long numbers (IP addresses).

THE WORLD WIDE WEB (1990S)

1989–1990: Tim Berners-Lee (a scientist at CERN) invented the **World Wide Web (WWW)**.

- Introduced **HTML, URLs, and HTTP**.
- First website went live in **1991**.

1993: Mosaic, the first popular **web browser**, launched.

Mid-1990s:

- Growth of websites, email, online chat.
- Companies like **Yahoo (1994), Amazon (1995), Google (1998)** emerged.

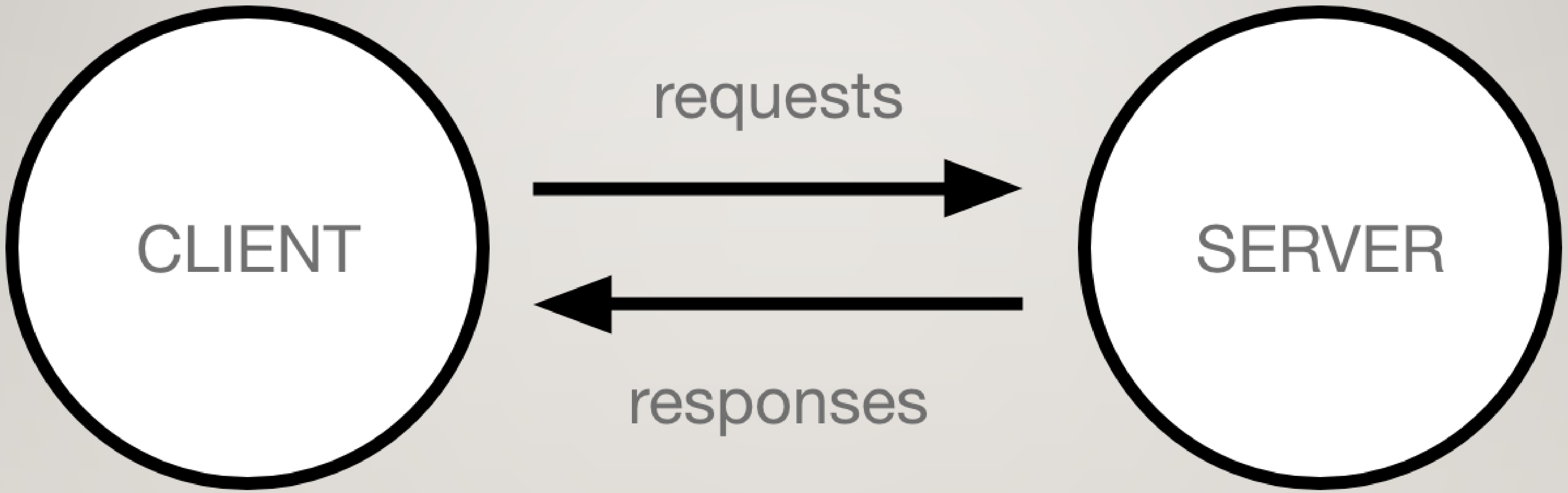
WEB 1.0 (1990S – EARLY 2000S)

- Known as the **"Static Web."**
- Websites were simple, read-only pages.
- Users could only read information, not interact much.
- Examples: early news sites, personal homepages.

WEB 2.0 (2000S – 2010S)

- Known as the **"Social Web."**
- Websites became interactive and dynamic.
- Users could create and share content → rise of **blogs, social media, YouTube, Facebook, Twitter**.
- Growth of **e-commerce** (Amazon, eBay) and **web applications** (Google Docs, Gmail).

CLIENT-SERVER ARCHITECTURE



CLIENT-SERVER ARCHITECTURE

- Computers connected to the internet are called **clients** and **servers**.
- Clients and servers exchange messages in a request–response messaging pattern. The client sends a request, and the server returns a response to received request.
- Clients are the typical web user's internet-connected devices (for example, your computer connected to your Wi-Fi, or your phone connected to your mobile network) and web-accessing software available on those devices (usually a web browser like Firefox or Chrome).
- Servers are computers that store webpages, sites, or apps. When a client wants to access a webpage, a copy of the webpage code is downloaded from the server onto the client machine to be rendered by the browser and displayed to the user.

BROWSER

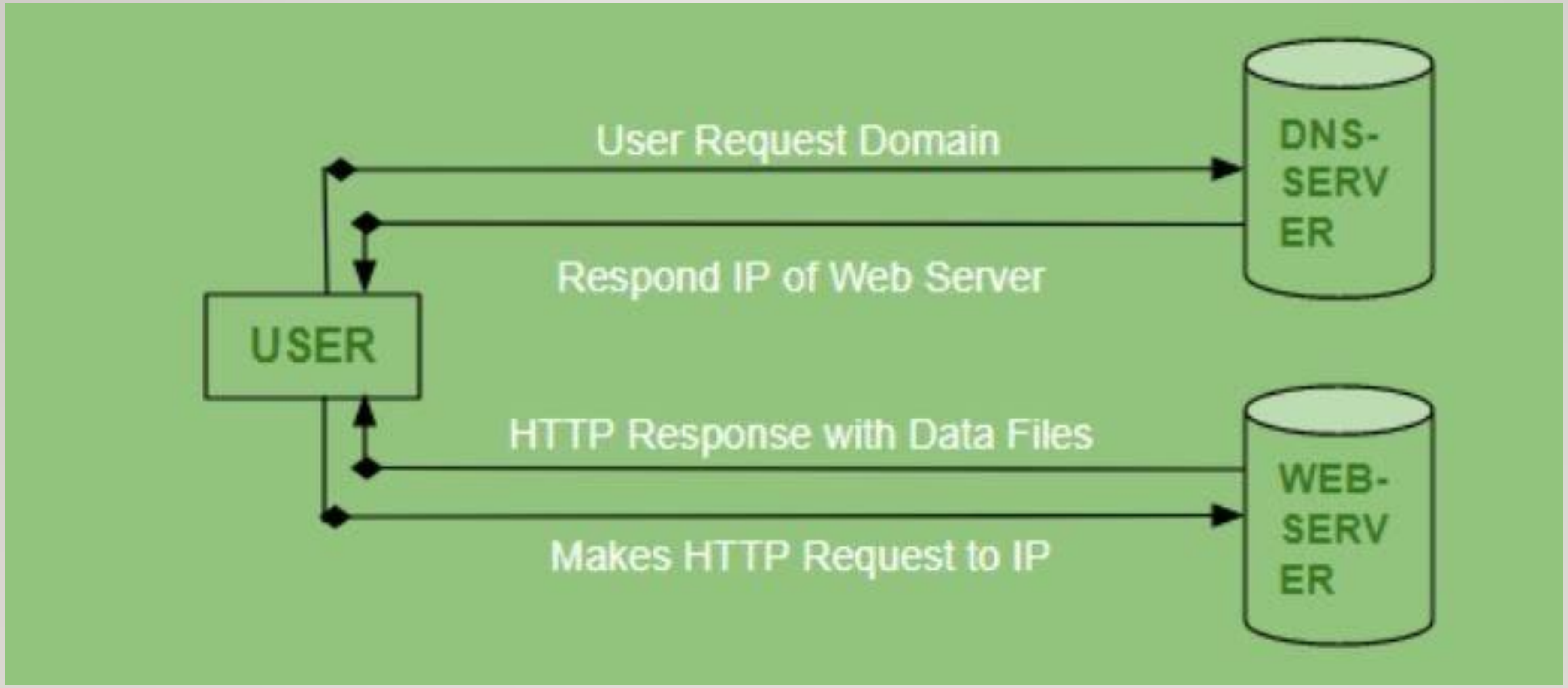
A **web browser** is a software application (e.g., Google Chrome, Mozilla Firefox, Microsoft Edge, Safari) that retrieves, interprets, and displays webpages written in HTML, CSS, and JavaScript. It acts as the bridge between the user and web content, turning code into a visual and interactive experience.

HOW BROWSER WORKS

- **User Requests Page**
  - The user enters a URL (e.g., example.com), clicks a link (e.g., <a href="example.com">Click</a>), or opens a local HTML file (e.g., index.html).
- **Fetch HTML**
  - The browser requests the HTML file from a server (via HTTP/HTTPS) for online pages or reads it directly for local files.
- **Parse HTML**
  - The browser reads the HTML and builds the **Document Object Model (DOM)**, a tree structure of tags like <h1>, <p>.

- **Load Resources**
  - The browser fetches additional files referenced in the HTML, like images (<img src="photo.jpg">), CSS, or JavaScript.
- **Render Page**
  - The browser combines the DOM with styles (e.g., from CSS) to decide how elements look and where they go, then displays them.
- **Enable Interactivity**
  - The browser runs JavaScript (if included) to add dynamic features, like responding to clicks.
- **Display Webpage**
  - The browser shows the final webpage and handles user actions, like clicking links to load new pages.

WHAT HAPPENS WHEN YOU VISIT A WEBSITE



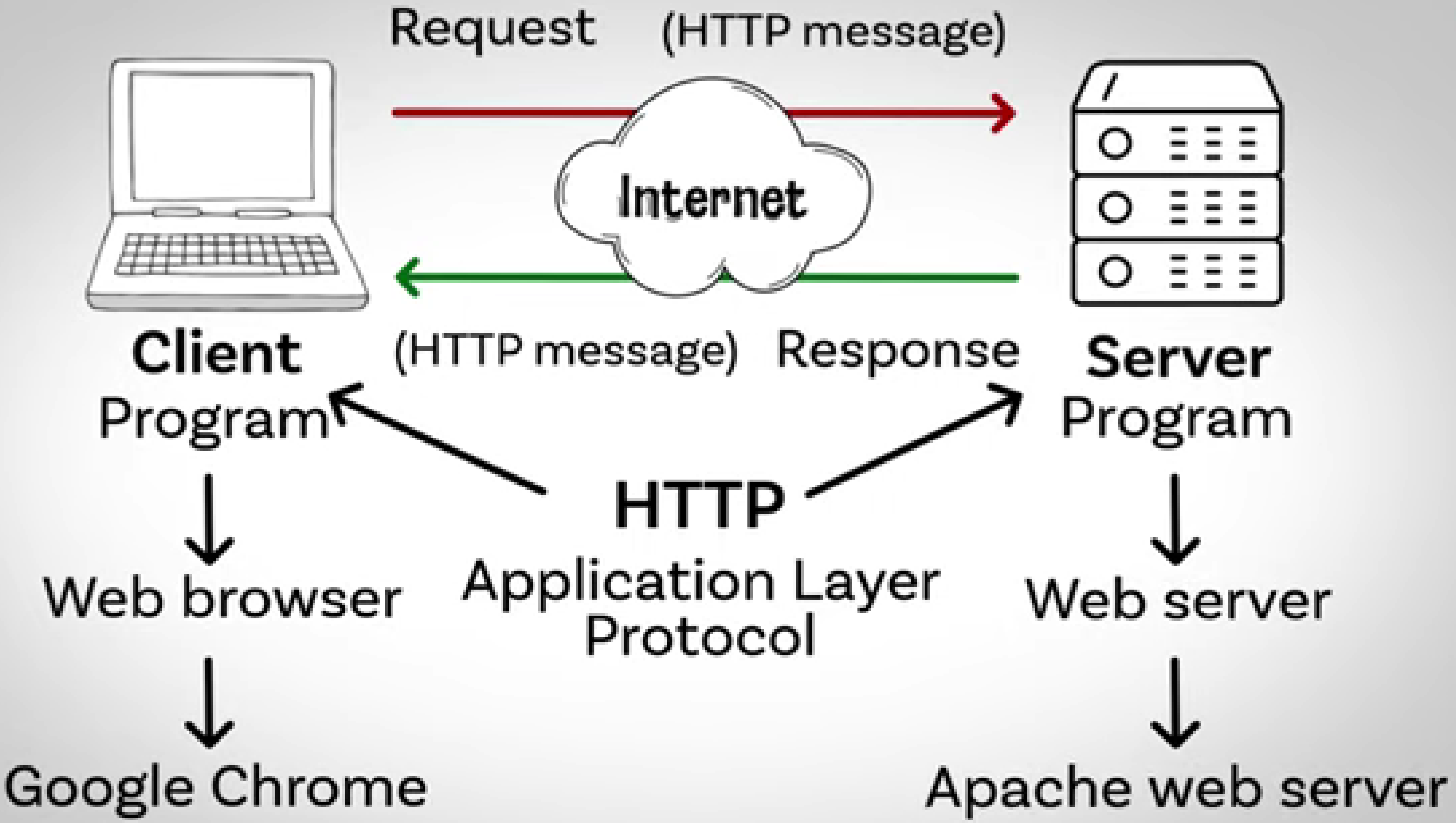
WHAT HAPPENS WHEN YOU VISIT A WEBSITE

- Enter URL or Click Link User types a URL or clicks a hyperlink to request a webpage. The browser interprets this as a command to load specific content.
- Resolve Domain Name Browser goes to the DNS server to find the real IP address of the server hosting the website. The IP address (e.g., 93.184.216.34) locates the server for the domain.
- Send Request to Server Browser sends an HTTP request message to the server via TCP/IP, asking for the website's HTML file. TCP/IP ensures reliable data transfer across the internet connection.
- Server Responds with "200 OK" and Data Packets If approved, the server sends a "200 OK" message to confirm the request's success. It then sends the website's files as small chunks called data packets.

- Parse HTML Browser reassembles data packets and reads the HTML code.
- Fetch Additional Resources Browser downloads files like images, CSS, or JavaScript via TCP/IP. These resources are sent as data packets from the server.
- Render the Page Browser combines HTML with styles to determine the layout and appearance. It paints text, images, and links onto the screen.
- Execute JavaScript Browser runs JavaScript, if included, to add interactivity. This enables features like button clicks or dynamic updates.
- Display and Interact Browser assembles packets into a complete webpage and displays it. It responds to user actions like clicking links or submitting forms.

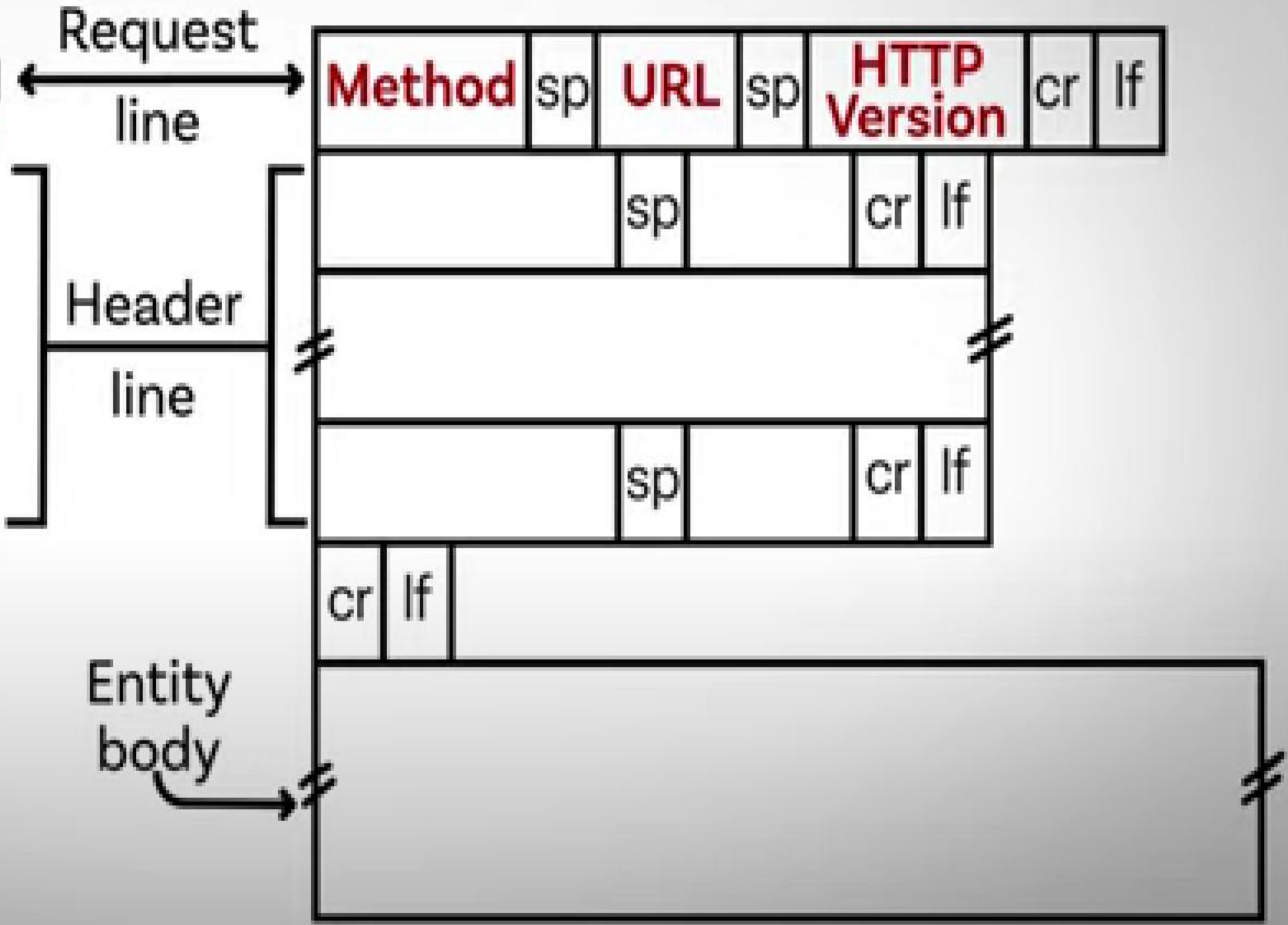
HTTP

- **HTTP (Hypertext Transfer Protocol) is a fundamental protocol** of the Internet, enabling the transfer of data between a client and a server. It is the foundation of data communication for the World Wide Web.
- HTTP provides a standard between a web browser and a web server to establish communication. It is a set of rules for transferring data from one computer to another. Data such as text, images, and other multimedia files are shared on the World Wide Web. Whenever a web user opens their web browser, the user indirectly uses HTTP.


Request
(HTTP message)
Internet
Client
Program
(HTTP message)
Response
Server
Program
HTTP
Application Layer
Protocol
Web browser
Google Chrome
Web server
Apache web server

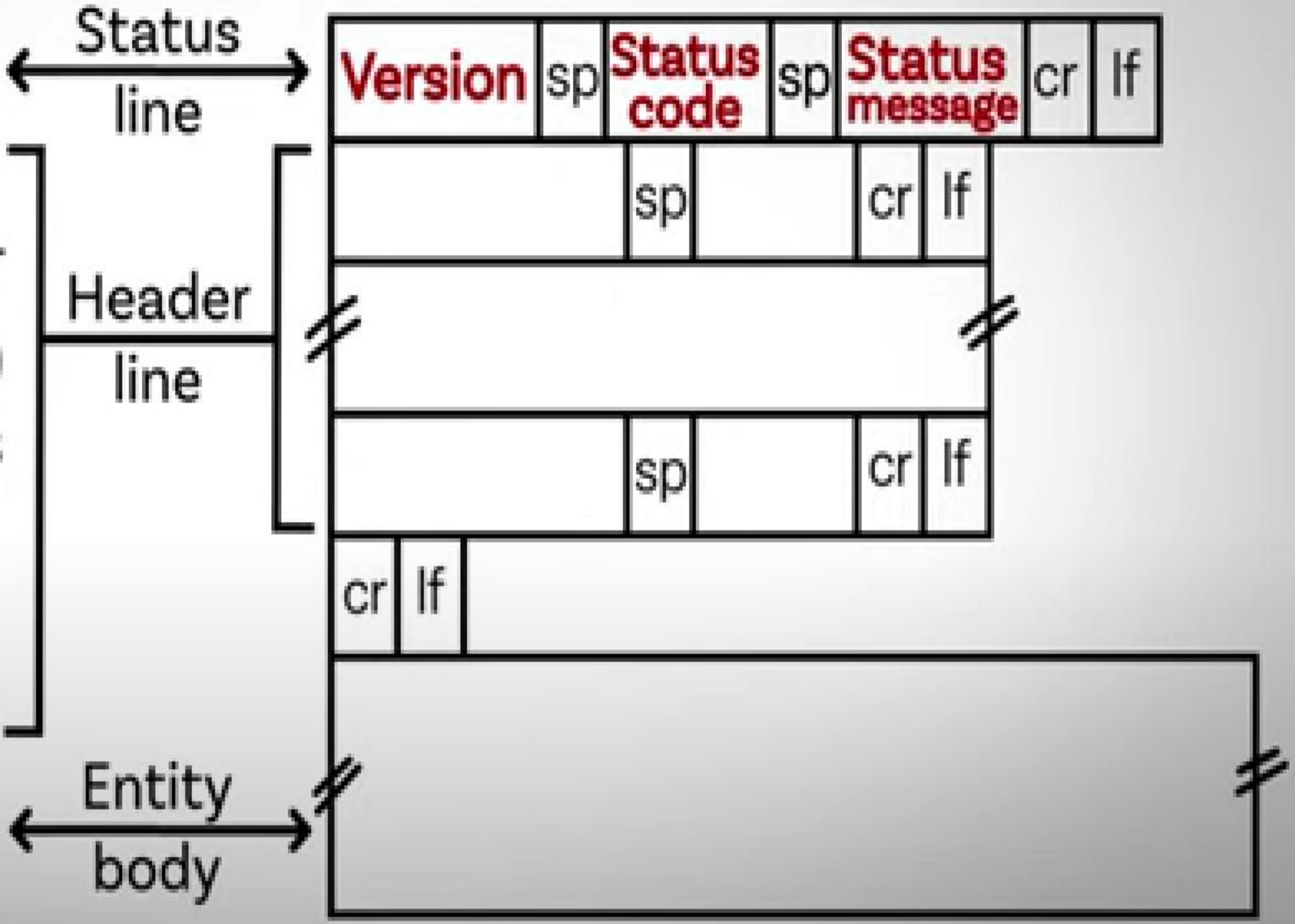
HTTP Request Message

```
GET /dir/page.html HTTP/1.1
Host: www.xyz.com
Connection: close
User-agent: Mozilla/5.0
Accept-language: fr
```



HTTP Response Message

HTTP/1.1 200 OK
Connection: close
Date: Tue, 18 Aug 2023 15:44:04 GMT
Server: Apache/2.2.3 (CentOS)
Last Modified: Tue, 18 Aug 2023 15:11:03 GMT
Content-Length: 6821
Content-Type: text/html
(data data data data ....)



HTTP REQUEST/RESPONSE:

- HTTP is a request-response protocol, which means that for every request sent by a client (typically a web browser), the server responds with a corresponding response. The basic flow of an HTTP request-response cycle is as follows:
- **Client sends an HTTP request**: The client (usually a web browser) initiates the process by sending an HTTP request to the server. This request includes a request method (GET, POST, PUT, DELETE, etc.), the target URI (Uniform Resource Identifier, e.g., a URL), headers, and an optional request body.

- **Server processes the request**: The server receives the request and processes it based on the requested method and resource. This may involve retrieving data from a database, executing server-side scripts, or performing other operations.
- **Server sends an HTTP response:** After processing the request, the server sends an HTTP response back to the client. The response includes a status code (e.g., 200 OK, 404 Not Found), response headers, and an optional response body containing the requested data or content.
- **Client processes the response**: The client receives the server's response and processes it accordingly. For example, if the response contains an HTML page, the browser will render and display it. If it's an image or other media file, the browser will display or handle it appropriately.

METHODS OF HTTP

- **GET**: Used to retrieve data from a specified resource. It should have no side effects and is commonly used for fetching web pages, images, etc.
- **POST**: Used to submit data to be processed by a specified resource. It is suitable for form submissions, file uploads, and creating new resources.
- **PUT**: Used to update or create a resource on the server. It replaces the entire resource with the data provided in the request body.
- **PATCH**: Similar to PUT but used for partial modifications to a resource. It updates specific fields of a resource rather than replacing the entire resource.
- **DELETE**: Used to remove a specified resource from the server.

FEATURES

- **Stateless:** Each request is independent, and the server doesn't retain previous interactions' information.
- **Text-Based:** Messages are in plain text, making them readable and debuggable.
- **Client-Server Model:** Follows a client-server architecture for requesting and serving resources.
- **Request-Response:** Operates on a request-response cycle between clients and servers.
- **Request Methods:** Supports various methods like GET, POST, PUT, DELETE for different actions on resources.

Status Codes

Status Code	Description
1xx: Informational	It means the request has been received and the process is continuing.
2xx: Success	It means the action was successfully received, understood, and accepted.
3xx: Redirection	It means further action must be taken in order to complete the request.
4xx: Client Error	It means the request contains incorrect syntax or cannot be fulfilled.
5xx: Server Error	It means the server failed to fulfill an apparently valid request.

- **102 Processing** → This code indicates that the server has accepted the request for processing, but the processing has not yet been completed.
- **200 OK** → Request succeeded, and the server returned the expected data
- **301 Moved Permanently** → Resource has been permanently moved to a new URL.
- **400 Bad Request** → Client sent an invalid request (e.g., malformed syntax).
- **404 Not Found** → Requested resource could not be found on the server.
- **500 Internal Server Error** → Generic server error when something goes wrong.

IP ADDRESS

- An IP address is a unique numerical label assigned to each device on a network.
- It identifies both the device and its location on the internet.
- It allows devices to communicate with each other over the internet.
- Every website and computer connected to the internet has an IP address.
- Example: 192.168.1.1.

DOMAIN NAME

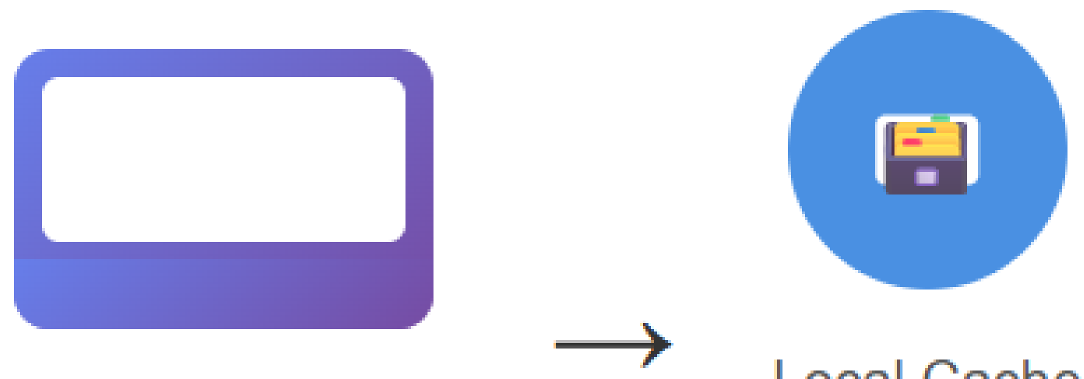
- A **domain name** is the human-friendly address of a website.
- It is used instead of numerical IP addresses.
- Example: typing **google.com** instead of 142.250.192.46.
- Domain names are unique for each website on the internet.
- They make it easier to find and access websites.

DNS (DOMAIN NAME SYSTEM)

- DNS is a hierarchical and distributed naming system that translates domain names into IP addresses. When you type a domain name your browser, DNS ensures that the request reaches the correct server by resolving the domain to its corresponding IP address.
- Without DNS, we'd have to remember the numerical IP address of every website we want to visit, which is highly impractical.

Working of DNS

Step 01: User requests and local cache is checked



Computer browser requests to visit **https://example.org**

Local Cache is checked for requested address

Step 02: Host Files are checked

If IP address not found in cache

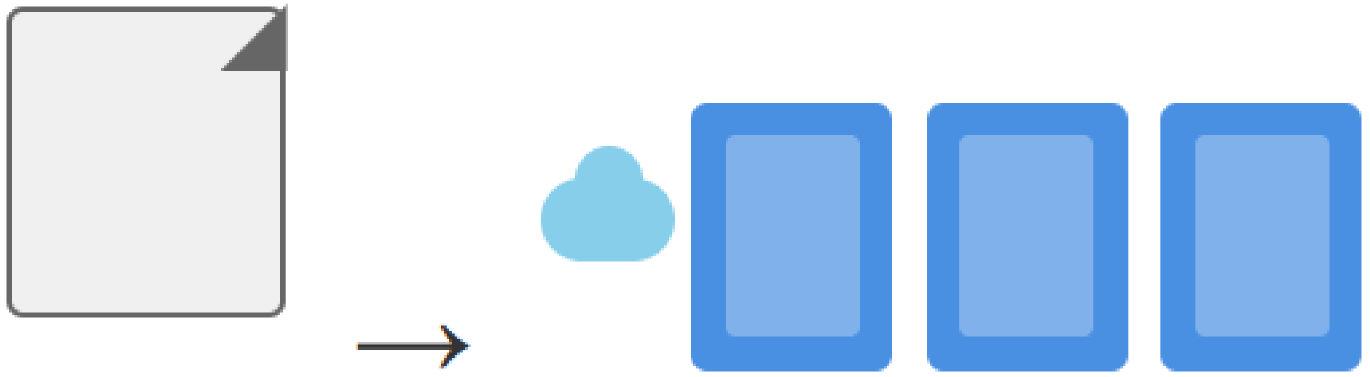
→

Checking Hosts File

Hosts File

The hosts file is a system file that maps domain names to IP addresses locally

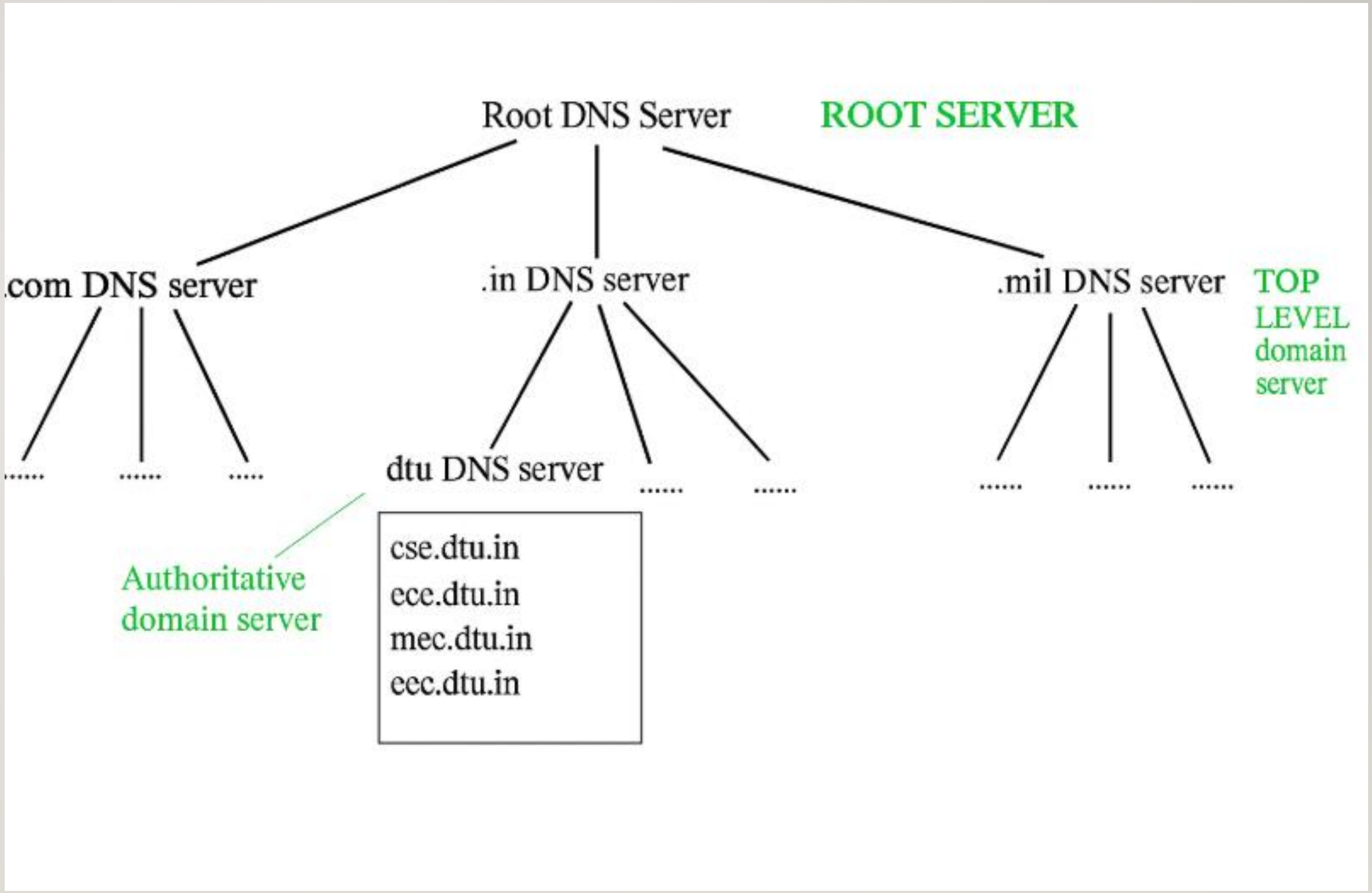
Step 03: Query DNS Resolver



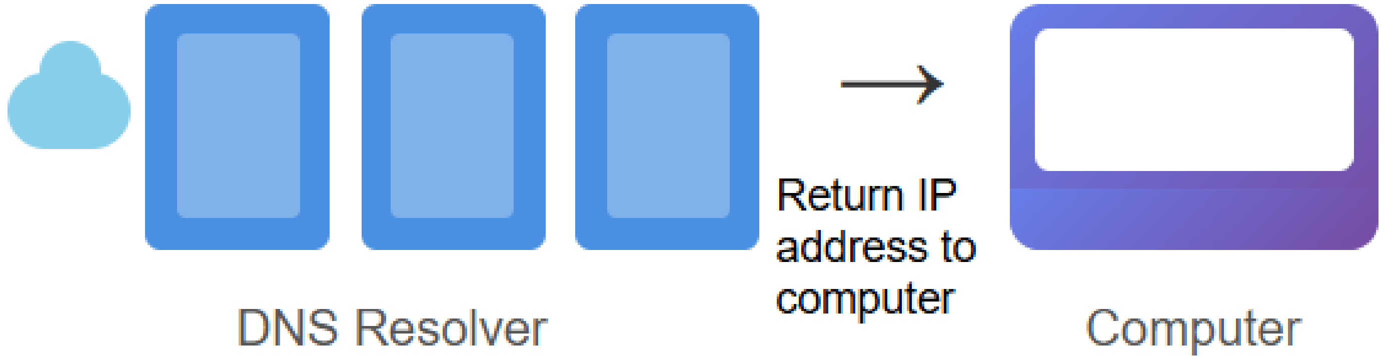
If IP address not found locally

DNS Resolver is a server provided by ISP

Step 04: DNS Search by DNS Resolver



Step 05: Return IP address to Computer



- **User Input:** You enter a website address (for example, www.example.com) into your web browser.
- **Local Cache Check:** Your browser first checks its local cache to see if it has recently looked up the domain. If it finds the corresponding IP address, it uses that directly without querying external servers.
- **DNS Resolver Query:** If the IP address isn't in the local cache, your computer sends a request to a DNS resolver. The resolver is typically provided by your Internet Service Provider (ISP) or your network settings.

- **Root DNS Server:** The resolver sends the request to a root DNS server . The root server doesn’t know the exact IP address for www.example.com but knows which Top-Level Domain (TLD) server to query based on the domain’s extension (e.g., .com).
- **TLD Server:** The TLD server for .com directs the resolver to the authoritative DNS server for example.com.
- **Authoritative DNS Server:** This server holds the actual DNS records for example.com , including the IP address of the website’s server. It sends this IP address back to the resolver.
- **Final Response:** The DNS resolver sends the IP address to your computer, allowing it to connect to the website’s server and load the page.

HOSTING

- **Hosting** is a service that provides storage space, server resources, and network connectivity so that your website/application can be accessible on the Internet 24/7.

HOW HOSTING WORKS

- You build a website on your local computer.
- You purchase a **domain name** (example: www.example.com)
- You rent **hosting space** from a provider (like AWS, Google Cloud).
- You upload your website/app files to the hosting server.
- When users type your domain, the **DNS** system points them to your hosting server's IP address.
- The hosting server delivers the files/webpage to the user's browser.