URL

Know about HTTP URL

Each resource on the web has a **unique identifier** called a **URL** (Uniform Resource Locator).

https://www.example.com:8080/products/electronics?category=mobile&sort=price#reviews

URL Component	Example Part	Meaning / Purpose
Scheme / Protocol	https://	Defines **how** data is transferred between client and server. Common examples: • http → HyperText Transfer Protocol (unencrypted) • https → Secure HTTP (encrypted via SSL/TLS) • ftp → File Transfer Protocol • mailto → Email links
Host / Domain Name	www.example.com	Identifies the **server** hosting the resource. It can be: • A domain name (www.example.com) • A subdomain (blog.example.com) • Or an IP address (192.168.1.1)
Port (optional)	:8080	Specifies the **port number** on the server to connect to. Default ports are: • 80 for HTTP • 443 for HTTPS If omitted, the browser assumes the default.
Path	/products/electronics	Points to the **specific resource** or location on the server. Works like a folder/directory structure.
Query String (optional)	?category=mobile&sort=price	Used to send **parameters** or data to the server (often in GET requests). • Begins with ? • Each key–value pair is separated by = • Multiple pairs are joined with & → Example: ?key1=value1&key2=value2
Fragment / Anchor (optional)	#reviews	Refers to a **specific section** or element **within** the web page. It's handled by the **browser**, not the server. Example: #section2

		scrolls directly to the section with id="section2".

Client-Server Communication

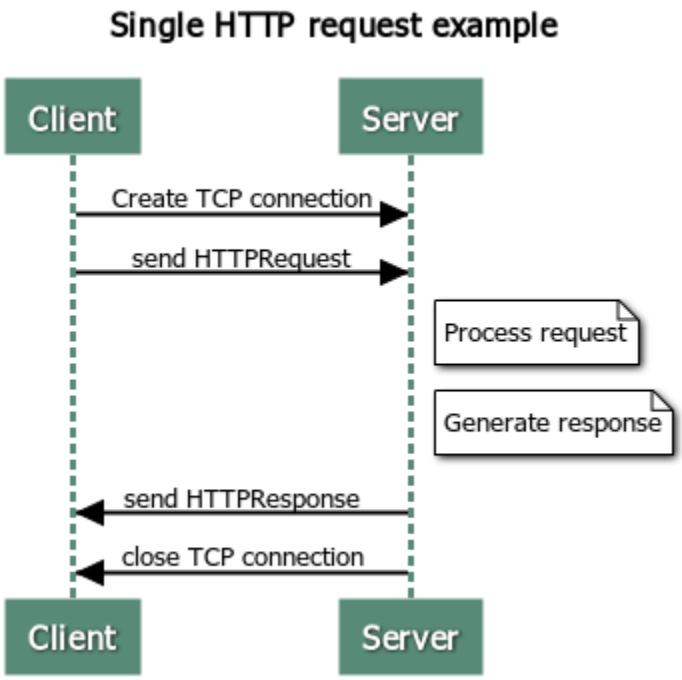
Basic concepts of web applications, how they work and the HTTP protocol

HTTP Protocol

The Hypertext Transfer Protocol is an application protocol for distributed, collaborative, hypermedia information systems that allows users to communicate data on the World Wide Web. HTTP clients generally use Transmission Control Protocol (TCP) connections to communicate with servers
https://www.tutorialspoint.com/http/index.htm
https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview

- Connectionless Protocol - Client theke server e request pathacchi, server response back kortese. Client theke server e ei request ta jay ekta TCP/IP connection diye. Request jawar somoy connection ta established hoy, ar response chole ashle closed hoye jay. That means server ta client ke chintese shudhu request/response er time tuku. Er baire server client keu kauke chine na.
- HTTP is stateless, but not sessionless- HTTP is stateless: there is no link between two requests being successively carried out on the same connection. This immediately has the prospect of being problematic for users attempting to interact with certain pages coherently, for example, using e-commerce shopping baskets. But while the core of HTTP itself is stateless, HTTP cookies allow the use of stateful sessions. Using header extensibility, HTTP Cookies are added to the workflow, allowing session creation on each HTTP request to share the same context, or the same state.
- Stateless Protocol - Meaning there is no link between two requests being successively carried out on the same connection (Ekta client er kach theke joto request ee pai, prottek ta request pawar por oi client ke ekta notun client hishebe handle korbe). This immediately has the prospect of being problematic for users attempting to interact with certain pages coherently, for example, using e-commerce shopping baskets. But while the core of HTTP itself is stateless, HTTP cookies allow the use of stateful sessions (will learn about it later).
- Application Layer Protocol - Jeno browser bujhte pare
- TCP/IP based
- Media independent - Response e jekono type media pathano jay. ‘Content type’ field e just media type ta boshay dite hobe.

How does connection establish between client & server?



1. **User enters the URL** of the website or file. The browser requests the DNS (Domain Name System) server for the corresponding IP address.
2. **DNS server provides the IP address** of the web server.
3. The **browser sends a TCP connection request** (also known as a **TCP handshake**) to the web server using the provided IP address. This establishes a reliable connection between the client (browser) and the server.
4. Once the TCP connection is established, the **browser sends an HTTP/HTTPS request** to the web server to request the necessary files.
5. The **server responds** with the website's files, and the browser renders the website.
6. Connection closes after the response.

From the connection request to the connection establishment is called one round trip time. Similarly, from the HTTP request to the HTTP response is one round trip.

HTTP Methods

https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods

Method	Typical Use Case (in Web Dev)	Has Request Body?	Idempotent?	Cacheable?

GET	Retrieve data or resource from the server (e.g., fetch a list of users, view a product page).	No	Yes	Yes
HEAD	Check if a resource exists or get metadata (like size or last-modified) without downloading the body.	No	Yes	Yes
POST	Send data to the server to create a new resource, submit a form, upload a file, or trigger an action (like login).	Yes	No	No
PUT	Replace an entire resource with new data. Used for full updates (client provides complete object).	Yes	Yes	No
PATCH	Partially update a resource — send only changed fields instead of the entire object.	Yes	Usually No	No
DELETE	Remove a resource from the server.	Optional	Yes	No
OPTIONS	Check which HTTP methods and operations the server supports for a specific endpoint. Common in CORS preflight requests.	Optional	Yes	No
CONNECT	Create a network tunnel, usually for HTTPS connections via a proxy. Rarely used directly by web developers. Used by **proxy servers** (not regular web developers) to **create a network tunnel** between the client and the target server, usually for **HTTPS (port 443)** connections.	Yes	No	No
TRACE	Debugging — echoes back the received request for testing the communication path. Not used in production.	No	Yes	No

HTTP Status Codes

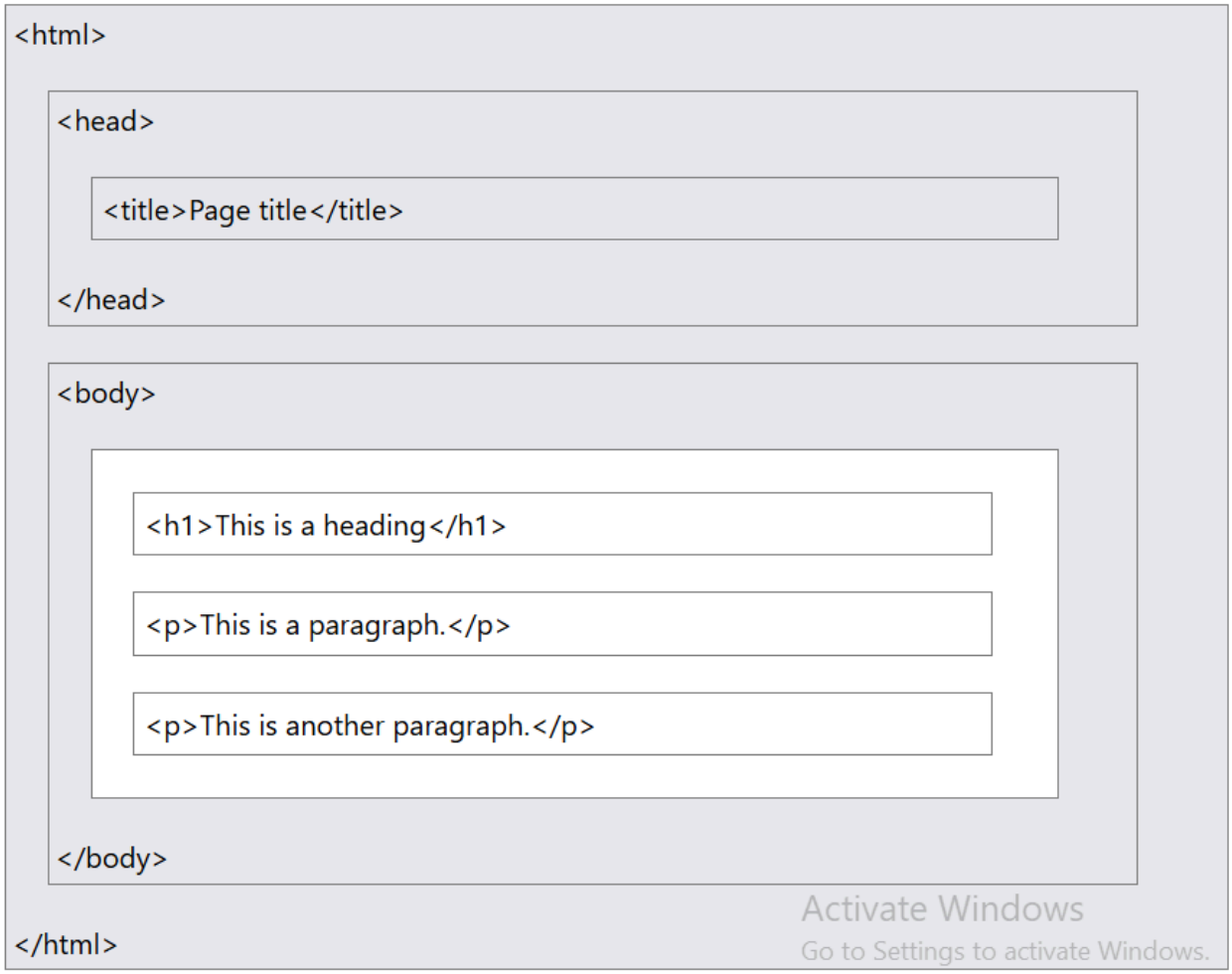
https://developer.mozilla.org/en-US/docs/Web/HTTP/Status (You do not need to memorize all the codes, just know the basics)

Class	Meaning / Purpose	Example Codes & Short Descriptions	Common Use Case / When Users See It
1xx – Informational	Indicates that the request was received and understood; client should continue or wait.	**100 Continue** – Client can keep sending data.**101 Switching Protocols** – Server switching to WebSocket or HTTP/2.	Rarely seen by end users; happens during protocol upgrades or data streaming.
2xx – Success	Request was successfully received, understood, and processed.	**200 OK** – Everything worked fine.**201 Created** – A new record or file was created.**204 No Content** – Successful but no response body.	When a webpage loads correctly, a form submits successfully, or an API call retrieves data.
3xx – Redirection	Client must take additional action (often redirect to another URL).	**301 Moved Permanently** – Resource moved to new address.**302 Found** – Temporary redirect.**304 Not Modified** – Use cached version.	When a website automatically redirects (`http → https`), or your browser loads a cached page for speed.
4xx – Client Error	The client sent a bad request or isn't allowed to access something.	**400 Bad Request** – Invalid data sent.**401 Unauthorized** – Login required.**403 Forbidden** – Access denied.**404 Not Found** – Page doesn't exist.	When a user enters a wrong URL (`404`), tries to access without logging in (`401`), or submits an invalid form (`400`).

5xx – Server Error	The server failed to handle a valid request — problem is on the server side.	**500 Internal Server Error** – Server crashed or misconfigured.**502 Bad Gateway** – Proxy received invalid response.**503 Service Unavailable** – Server overloaded or down for maintenance.	When a site crashes temporarily, or a high-traffic server is down — users see “500 Internal Server Error” or “Service Unavailable.”

HTML

HTML stands for Hyper Text Markup Language
HTML is the standard markup language for creating Web pages
HTML describes the structure of a Web page
HTML consists of a series of elements
HTML elements tell the browser how to display the content



intro.html

- The `<!DOCTYPE html>` declaration defines that this document is an HTML5 document
- The `<html>` element is the root element of an HTML page
- The `<head>` element contains meta information about the HTML page
- The `<title>` element specifies a title for the HTML page (which is shown in the browser's title bar or in the page's tab)
- Viewport: for responsiveness

- The `<body>` element defines the document's body, and is a container for all the visible contents, such as headings, paragraphs, images, hyperlinks, tables, lists, etc.
- The `<h1>` element defines a large heading
- The `<p>` element defines a paragraph

Element?

An HTML element is defined by a start tag, some content, and an end tag:

<tagname> Content goes here... </tagname>

Headings, Paragraph, Links, Image, Image with links:

html basic example.html

Path	Description
<img src="picture.jpg">	The "picture.jpg" file is located in the same folder as the current page
<img src="images/picture.jpg">	The "picture.jpg" file is located in the images folder in the current folder
<img src="/images/picture.jpg">	The "picture.jpg" file is located in the images folder at the root of the current web
<img src="../picture.jpg">	The "picture.jpg" file is located in the folder one level up from the current folder

List, Table

list table.html

Forms

Forms Intro.html

Blocks and Inline Element in HTML

WP-4-HTML.pptx
Div elements: HTML Div Element