Web Fundamentals & Intro to HTML

- Q1. What makes up the foundation of Web Development?
  - HTML

    It is the backbone of web development, just like our spine supports our structure. It provides a basic structure which we technically call the '**Markup**' that is used in creating the webpages.
  - CSS
    1. It allows us to style or format the content. Every website (Netflix, Flipkart, Amazon etc.) has almost similar structure/markup (all websites have heading, images, content, links etc.), yet they look very different!
    2. Why? Because they have been styled differently using CSS
  - JavaScript
    1. JavaScript or Client Side Scripting is the primary language for the browser.
    2. It is used for adding **interactivity** to the website, or to create **dynamic** websites.
    3. You can basically interact, play around and actually *do stuff* with the website which is not just READ-ONLY
- Q2. What has given superpower to websites?

  - JavaScript, with the way it has evolved over the years, helps in adding many features and functionality to any website

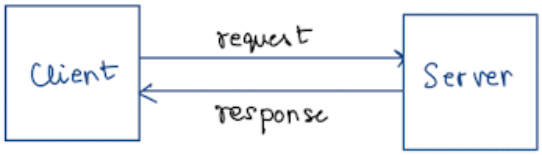
  - Many Frameworks and Libraries in JS do a lot of 'heavy lifting' for us, and with little/less code we can achieve more and more. (React, Angular, Vue etc.)
- Q3. How does JavaScript make full-stack development possible?

  1. JS is not only limited to the front-end. We can use it in the backend with nodejs for server side development, making full stack development possible with only one language.

How Does the Web Work?

- Q4. How do two systems talk?

  - The two systems are called 'client' and 'server'

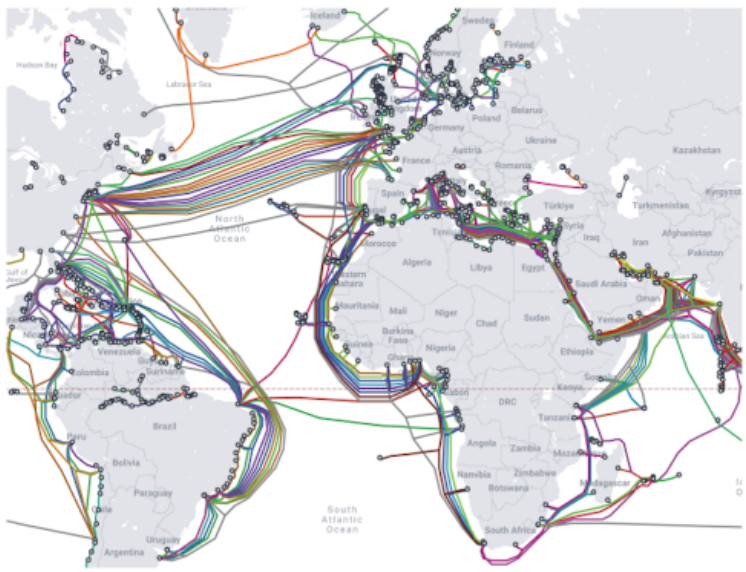


- ▼ All the computers or systems which are connected to the the internet are called *clients* or *servers*.
- ▼ **Client**
  - ▼ Anything which initiates a **request**. When we enter a URL, we are 'requesting' for the contents of that webpage
- ▼ Server
  - ▼ Anything, that **serves** the request.
  - ▼ These are systems that have the required data/information and issue **responses**.

▼ Q5. What is an internet-connection?

→ We can say that internet connection is network of networks

↳ We say so because, all the countries in the world, they are connected via fibre optic cables running at the sea bed where the signals travel from one end of the cable to the other nearly at the speed of light



This is why we are able to connect to other parts of the world almost instantly!

▼ Trivia

→ There are people like Elon Musk who have been trying to move towards satellites instead of these cables! (Starlink)

→ This is because some times ships / aquatic animals can damage these cables. Some countries in warfare as well have been reported damaging these cables to break / sabotage internet

▼ Q6. What is a Protocol ?

A set of rules to be followed

▼ Q7. What is TCP/IP ?

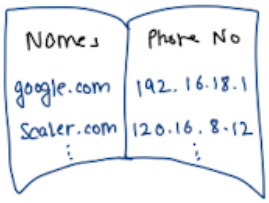
TCP/IP are some communication protocols that define how data should travel across the internet

→ Just like when we are out on the road, there are some protocols that everyone should follow, for proper movement of traffic

▼ Q8. What is DNS ?

In the olden days, there used to be a yellow page telephone directory which had the contact information of all the important people of that place, like doctors. etc.

DNS (Domain Name System) is like a directory, much like the



telephone book in the olden days, which consists of mapping between the websites (name) and their respective IP Address

→ Every website is mapped to an IP address, which the browser needs to figure out whenever you name-drop that website.

→ It's much like when you call your 'friend' from contact list, your mobile needs to figure out what the actual contact number of your 'friend' is.

It is this IP address to which we send requests and receive responses

▼ Q9. What is HTTP ?

HTTP (Hyper Text Transfer Protocol) is a set of protocols/rules for transferring 'hyper text'

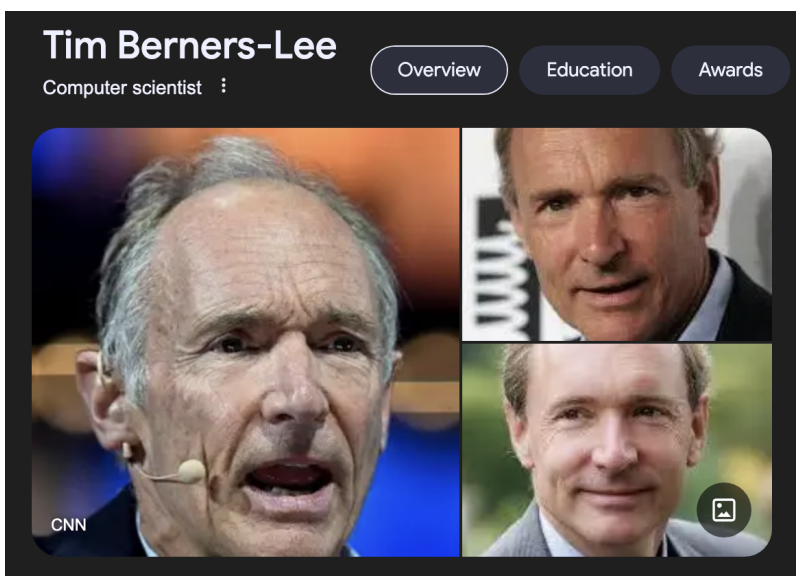
↓

The HTML documents that we're transferring from client to server

→ HTTP is the language of the browser for communication b/w client & server

This protocol is the common language for clients and servers

▼ Trivia: Who is Tim Berners Lee ?



Tim Berners Lee is a computer scientist, and is known as the father of the world wide web 👨🏻

▼ 🔥 How did the Web began ?

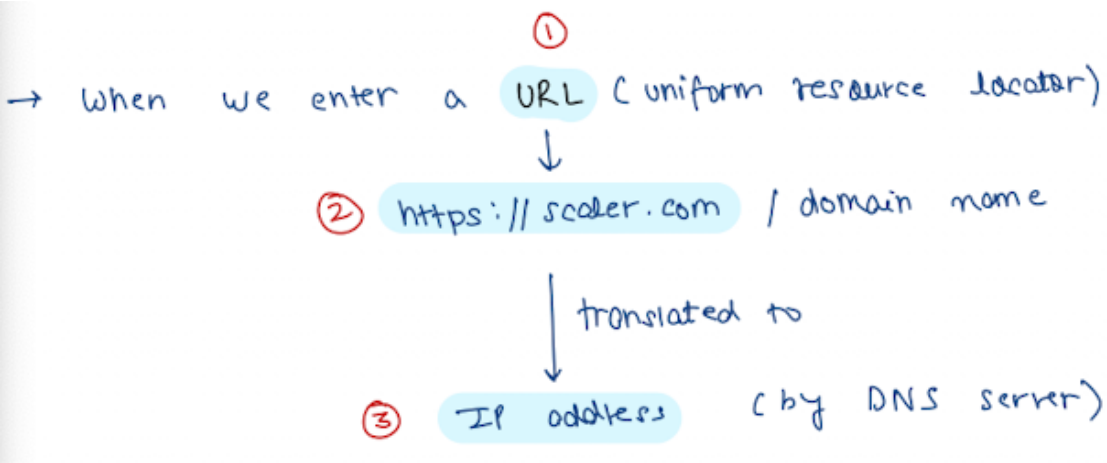
https://www.home.cern/science/computing/birth-web/short-history-web

He was working on a project, where he was trying to transfer a file from one system to another

Putting It All Together! (A Movie 🍿)

Plot: We have a **hero**(client) and a **heroine**(server) separated with each other, and the heroine is stranded on a far away island 🏝️ We need to ensure communication between these two ❤️

- How does an end to end communication take place ?
  - Providing the URL/Domain Name



Browser also figures out what is the protocol mentioned for fetching the domain name (HTTP or HTTPS)

  - IP lookup Optimisation
    1. After providing domain name, it needs to be translated to an IP Address
    2. There are different iterations involved in which the IP address is retrieved

→ But the browser will not repeat the same architecture every time.

→ If we open a URL very frequently, it won't hit DNS server each time to fetch IP.

→ Instead, there are optimizations in place, for eg the browser/router will save the frequently visited URL's (caching)

our OS, ISP's like Airtel also does some caching

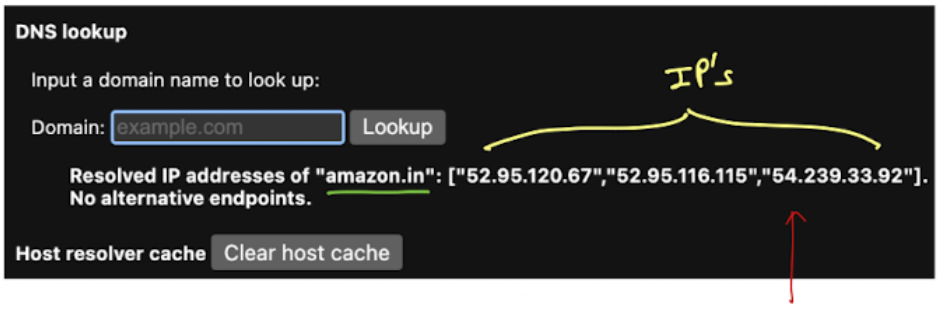
→ If the browser cache is not present, browser will talk to OS cache, if not even there, it goes to ISP's cache. Ultimately if not even there, then it hits the DNS server

→ This DNS server is configured as soon as we connect to the WiFi, such that the browser knows how to contact

▼ DNS Lookup feature in browser's cache!

For a domain-name provided, your browser might have cached the corresponding IPs of multiple servers where it is hosted! And you can look those up

Chrome has a DNS lookup feature, where you can see the IP's for the server's domain name provided



Any company might have multiple server's, hence IP's and depending upon which server is available to respond faster, we get the returned IP's (like a building with multiple entry gates)

→ Over the period of time, these listed IP's were received by the browser, so it cached those

▼ Dialling the number ☎️

Now, finally our hero has been able to get the phone number (IP Address) of our heroine.

The next step is actually dialling the number 😂 And for that purpose

④ HTTP messages will now be sent from client to server, basically an exchange where the client sends a request to the server asking it send a copy of the website (HTML/CSS/JS)

So the exchange of data that happens is basically the HTTP messages which get exchanged

▼ Additional Read 📖 (Recommended) - Is HTTP data exchange protocol only limited to HTML files given the “HyperText” Transfer Protocol ?

▼ Short Answer

Short Answer
HTTP may have "hypertext" in its name, but it can be used to transfer *any type* of file—not just HTML. In practice, web servers send HTML, CSS, JavaScript, images, videos, PDFs, and much more over HTTP. The key is that the server tells the client (browser) how to interpret the file by sending the correct *Content-Type* header.

▼ It's not just limited to Hyper Text!

- **Origins of the Name**: When the Web was first created, its primary goal was sharing hypertext documents (HTML). Hence the name *HyperText Transfer Protocol (HTTP)*.
- **Modern Reality**: Today, HTTP has evolved into a general-purpose application-layer protocol. It runs on top of TCP/IP and can transfer just about *anything*—HTML, CSS, JavaScript, images (PNG, JPEG), videos (MP4), PDFs, etc.

▼ Support for Different Files (Headers make all the difference🔥!)

2. How Different File Types Fit In

When your browser (the *client*) requests a resource (e.g., `index.html`, `styles.css`, `app.js`, `image.png`), here is what happens on a high level:

1. **Client Request**
   - The browser sends an HTTP request (e.g., `GET /styles.css`) to the server.
2. **Server Response**
   - The server locates the requested file (`styles.css`) and sends it back with a response header that looks like:

Copy

```
HTTP/1.1 200 OK
Content-Type: text/css
Content-Length: 1234

[file content...]
```

   - The `Content-Type` header tells the browser how to interpret the file (in this case, as CSS).
3. **Browser Interpretation**
   - Seeing `Content-Type: text/css`, the browser knows it's a CSS file and applies it to the rendered web page.

This same process works for JavaScript, images, audio, video, or any other file. The server just needs to specify the correct `Content-Type` so the client knows what to do with the received data.

▼ MIME Types 🤨

3. MIME Types

- **Why MIME Types?**

Multipurpose Internet Mail Extensions (MIME) were originally for email, but have been adopted by HTTP to label the content being transferred (e.g., `text/html`, `text/css`, `image/png`, `application/pdf`, etc.).

- **Any File Format**

As long as a server can produce a valid MIME type for the file, and the client can handle that file type (e.g., a browser knows how to display images, PDFs, etc. or your computer has a suitable application), *any* file format can be transmitted over HTTP.

▼ Putting it all Together and Key Takeaways!

4. Putting It All Together

- **HTTP is Just the Transport**: It's a set of rules for how requests and responses are structured, but it doesn't limit what the actual *payload* (file content) can be.
- **Browser Support**: Your browser might not *display* or *execute* every file type natively, but the file can still be *transferred* over HTTP. If the browser or an associated plugin knows how to handle the file, then you'll see or interact with it. Otherwise, it'll prompt you to download it.

Key Takeaways

1. **HTTP can send *any* file type.**
2. **The browser and server communicate using HTTP request/response pairs.**
3. **The** `Content-Type` **header is crucial** for letting the browser know how to handle or render a file.
4. **"HyperText"** in HTTP is historical and doesn't restrict it to HTML alone.

Overall, the Web stacks on top of HTTP to serve a variety of resource types, not just hypertext documents. So, you're completely correct in concluding that HTML, CSS, and JavaScript aren't the only files transferred—virtually *any* file format can be sent over HTTP!

▼ Exchange of Information(TCP/IP) ℹ️

Whenever there is a traffic, there has to be a rule 🤞🏻

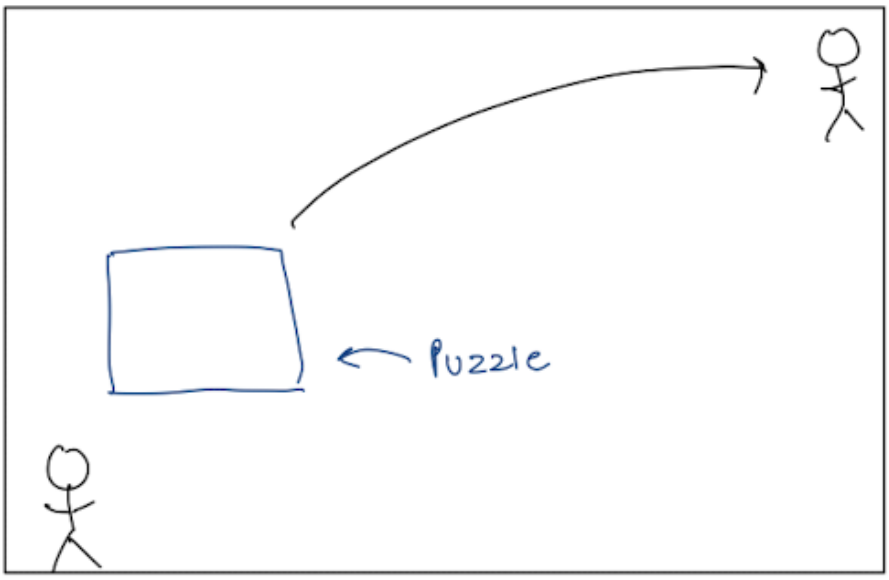
Whenever there is some network traffic/exchange that is going to happen between two systems, we need to have a protocol

(5) All this exchange / communication between two or multiple systems, we need to have a protocol → TCP / IP

↳ The interacting computers / systems need to know ahead-of-time how they are expected to communicate

- how to start the communication
- how will system know that message sent was transferred correctly or not
- how will the communication end?

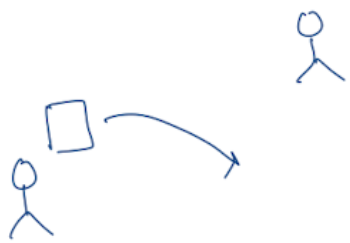
Lets understand this with a scenario 😐



Description → You and your friend are standing very far apart from each other in a large room.

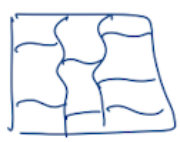
→ You have to pass a puzzle which you have to your friend

→ Since it's a very big room, there are high chances that it might fall somewhere in between

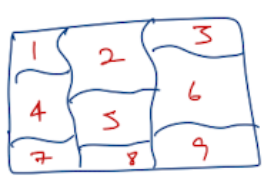


It was never mentioned it was a *very interesting* game but let's just assume it's that kind of party. So any solution comes to mind?

→ You do an interesting thing. Since it's a puzzle, it has smaller pieces in it. You break it into smaller pieces → First optimization



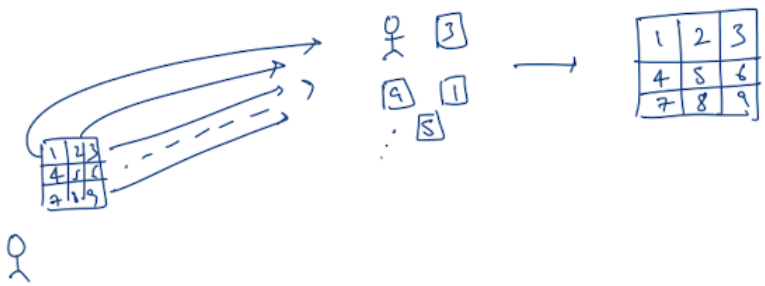
→ Next, you number the individual pieces → second optimization



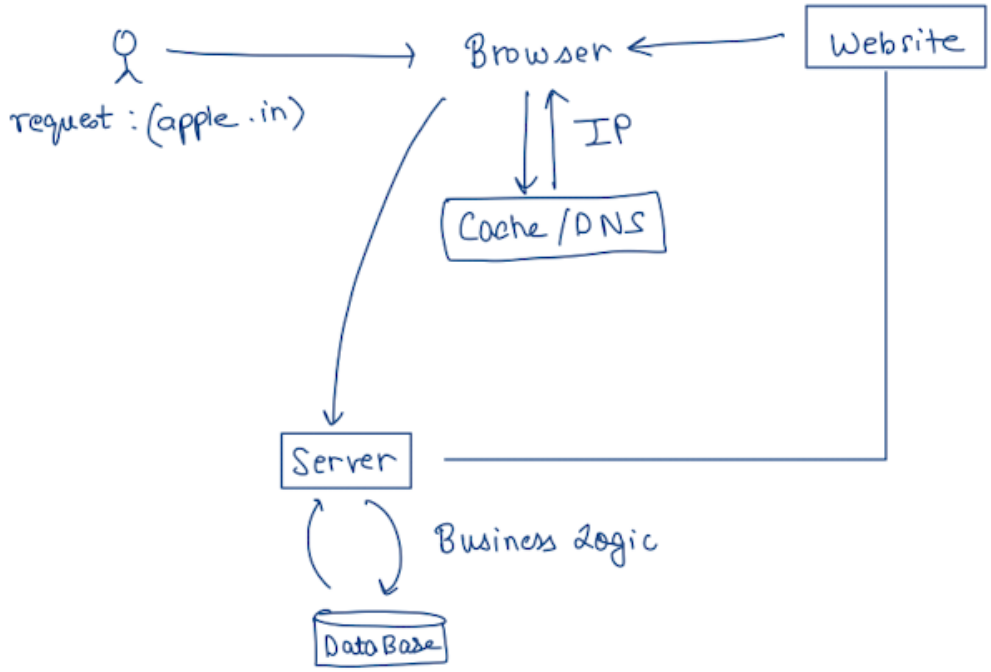
→ Now, you start throwing these pieces individually, not in any necessary order.

→ Since these are all smaller pieces, you can throw easily and will be able to transmit easily

→ Your friend on the other hand, receives them and based on the number, picks them up and recreates what you wanted to send in the first place



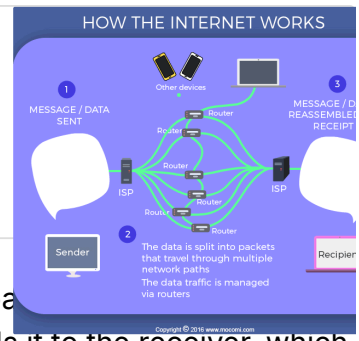
→ This is what takes place by the virtue of TCP/IP



What is Internet and How it Works? - Gifographic | Mocomi Kids

The Internet is a global collection of networks of computers connected to each other and data transferred from one place to another. But how it works?

https://www.google.com/url?sa=i&url=https%3A%2F%2Fmocomi.com%2Fhow-the-internet-works%2F&psig=AOvVaw2hlbmqxE_-WLuWyHz7O1JH&ust=1737199393608000&source=images&cd=vfe&opi=89978449&ved=0CBMQjRxqFwoTCNiR7ofS_IoDFQAAAAAdAAAABAE



In a nutshell, your message is broken down into smaller pa using different routes (whichever is easily available) sends it to the receiver, which will recreate it how it was sent originally

You might like it! (Read This)

▼ What is the difference between HTTP and TCP/IP ? Aren't they protocols after all ?

▼ Short and Crisp Answer✨

• **TCP/IP** is a *transport* and *network* protocol suite (operating at lower layers) that ensures data can get from one point on the Internet to another reliably.

• **HTTP** is an *application-level* protocol (operating at a higher layer) that specifies how clients and servers communicate to transfer web resources (HTML, CSS, JSON, etc.).

They both govern data exchange, but at *different layers* of the networking stack. HTTP *relies on* TCP/IP to do the actual data delivery.

▼ Want a look at Networking Layers 👀 ?

To see how HTTP and TCP/IP differ, let's place them in the context of the **OSI (Open Systems Interconnection)** or the **TCP/IP model**:

1. **Application Layer**
   - Protocols: HTTP, SMTP, FTP, DNS, etc.
   - Purpose: Define rules for *how data is formatted and exchanged* between applications.
2. **Transport Layer**
   - Protocols: **TCP (Transmission Control Protocol)** or UDP (User Datagram Protocol)
   - Purpose: Provide *end-to-end* communication; e.g., *TCP ensures reliable data transfer*, re-transmits lost packets, manages flow control, etc.

3. **Network Layer**
   - Protocol: **IP (Internet Protocol)**
   - Purpose: Addresses and routes data packets across different networks.
4. **Link/Physical Layer**
   - Protocols vary (Ethernet, Wi-Fi, etc.)
   - Purpose: Handles the actual transmission of raw bits and frames over a physical medium.

How They Stack

HTTP is at the *application layer*, while TCP is at the *transport layer* and IP is at the *network layer*.

- When you make an HTTP request (e.g., "GET /index.html"), HTTP uses TCP to establish a reliable connection.
- TCP (transport layer) breaks your data into segments and relies on IP (network layer) to figure out the best route across the Internet to the destination.

▼ Key Takeaways 🔑

1. **They Operate at Different Layers**

- TCP/IP is the *foundation* for reliable data transfer across networks.
- HTTP is built *on top of* TCP/IP to define how web-related data is requested and delivered.

2. **They Solve Different Problems**

- **TCP** ensures no packet loss, in-order delivery, and manages congestion and flow control.
- **IP** finds routes between networks.
- **HTTP** structures the *content* and *semantics* of requests/responses at the application layer.

3. **HTTP Needs TCP/IP**

- HTTP by itself doesn't know how to get data from one machine to another. It relies on the underlying TCP/IP suite to handle those lower-level details.

4. **TCP/IP Doesn't Care About the Content**

- TCP/IP will transfer any binary data. It doesn't parse or interpret whether the data is HTML, images, or streaming video.

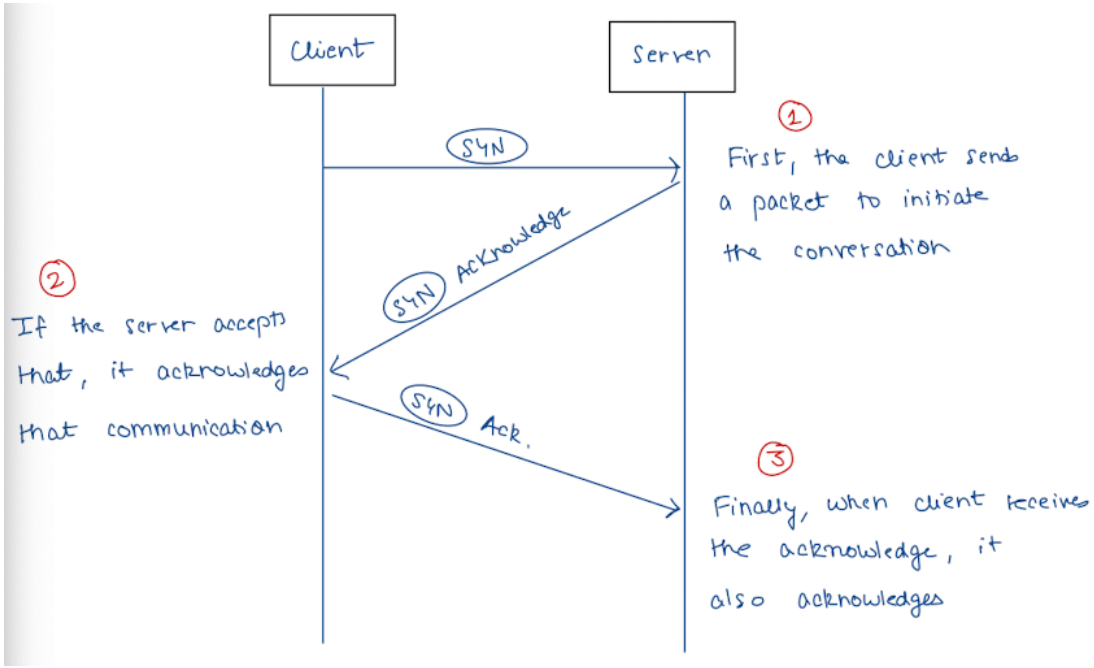
  In other words, **TCP/IP** is the *highway system* that moves trucks around reliably, and **HTTP** is the *language* your cargo (in this case, web content) is labeled and packaged in, so the recipient knows how to unpack and use it.

▼ The Server Responded!🔥

In General, the server sends the HTML, CSS and JS files in the response

▼ How does the communication start between the client and server ?

→ Before any real communication takes place between the client and server, a three way handshake is done, which is also known as the TCP/IP handshake.



If you were to draw parallels with it in real life, you might like this example

Hero: Hi Darling! (SYN)

Heroine: Hi Honey! Can you hear me? (SYN ACK)

Hero: Yes, I can hear you! (ACK)

So, TCP/IP plays a great role in initiating the conversation or setting up the connection, as well as actually transferring data packets once the connection is established

The Three Way Handshake

1. **Client → Server: SYN**

- The client sends a packet with the SYN (synchronize) flag set, indicating it wants to open a connection and specifying an initial sequence number.

2. **Server → Client: SYN + ACK**

- The server responds with a packet that has both SYN (it also wants to open a connection) and ACK (acknowledging the client's SYN) flags set, and a sequence number of its own.

3. **Client → Server: ACK**

- The client then sends back an ACK to acknowledge the server's SYN.

After these three steps, the TCP connection is *established*, and the two machines can now send data back and forth reliably.

▼ How many times does this handshake happen 🤝?

Once this handshake completes, **multiple** HTTP requests can reuse the *same* TCP connection (if "keep-alive" is enabled). So, **the handshake does not happen before *every* request**—only when a new TCP connection is established.

HTML/CSS (Building a Shopping Mall)

▼ Q1. What should be the first step before starting to build ?

1. The first approach should be to have a **blueprint** or a **plan** on what you want to design
2. Then accordingly we can plan our efforts, how many storeys we want, parking lot etc.

In Web Development, we have **Wireframes**(Figma) which serve as blueprints for us

1) Blueprint (Wireframes)

Figma → these tools are used to create wireframes, for how the website will eventually look

↓

design is done pre-coding by UX engineers

→ planning of interface

→ what will be the flow?

→ how will login flow look like?

→ what are the different options & screens?

→ to buy a product, from selecting till checkout, what are the different screens that user encounters?

▼ Q2. What is the second step involved ?

We would now be putting in the actual work that goes into building the **structure**

2) Structure (Brick - Mortar / Cementing)

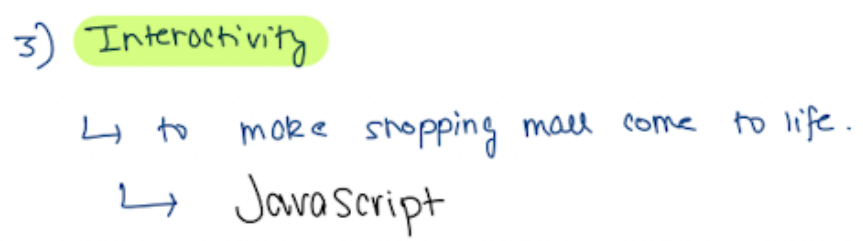
→ building bit by bit / brick by brick (skeleton that will support the structure)

→ this is where HTML creation happens

→ when the structure is ready, we start painting it, make it beautiful → CSS comes into picture

▼ Q3. What is the final act ?

The final step would involve making the mall interactive and adding features like lifts, escalators, ATM's that people actually interact with



▼ Q4. How to create a boilerplate code for HTML5 ?

Using Emmet Abbreviation **!**

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body></body>
</html>
```

▼ What does `DOCTYPE` mean ?

To inform the browser we are using HTML5

▼ What does `lang="en"` do ?

It specifies that 'English' will the main language used in the contents of this webpage

▼ What is the significance of the `<head>` ?

Just like our heads, it is the **brain** of the HTML document.

It is *not visible* on the actual page, but contains important information

▼ What does `charset="UTF-8"` signify ?

UTF-8 is a character set. By mentioning this, we make sure that our page can render and display a wide range of characters from different languages

For example, you might be developing a website for a middle-east client, and might have some arabic words involved.

If you did some have text like this, and did not mention this encoding, your browser might not render it properly and display some garbled/question mark characters.

- ▼ What does `name="viewport”` mean ?

  This makes our webpage compatible/look good with mobile devices like smartphones, tablets etc.

  Our webpage size would adjust according to the screen on which it being viewed on

- ▼ What does the `body` contain ?

  It contains the contents actually visible on the UI

- ▼ Q5. Why are `div` tags used ?

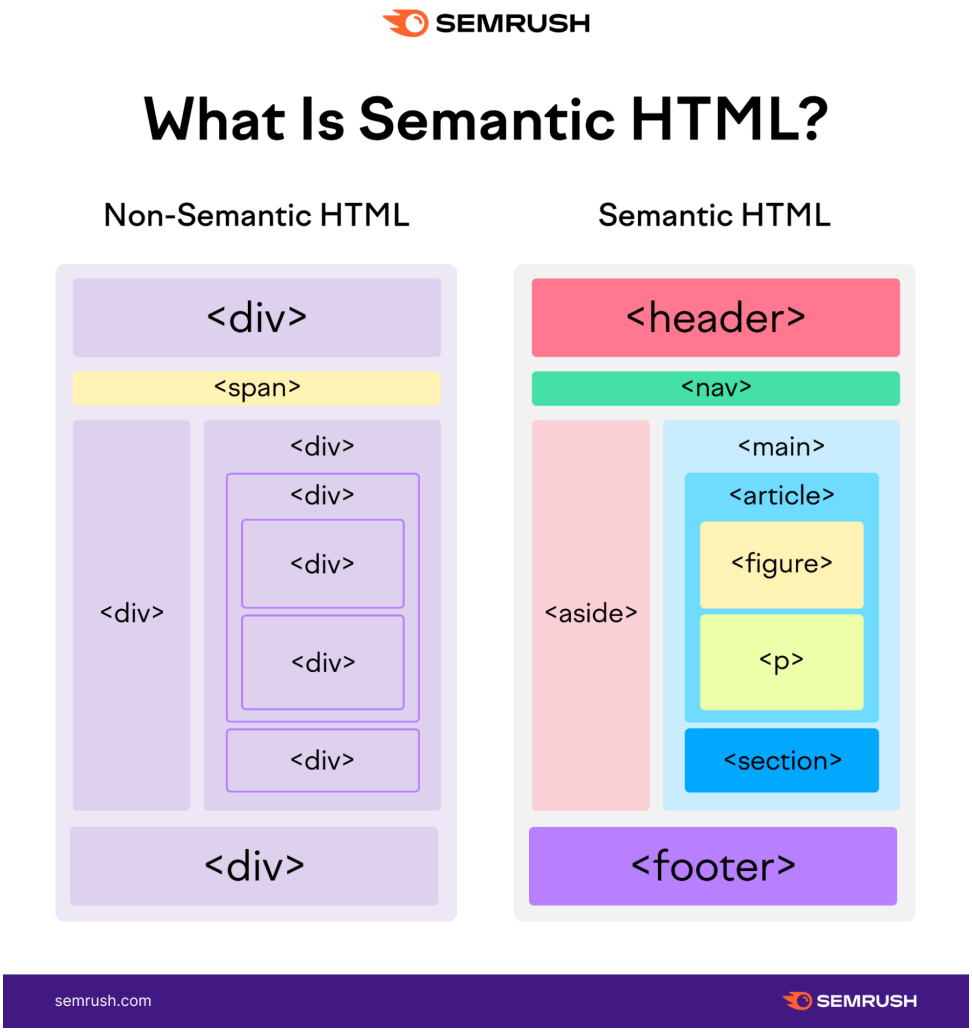
  `<div>` tags are used to logically group contents of your website

  For example, I may want all my headings to fall under one `div` and all paragraphs to be under another `div`

  Also, one **purpose** is to make it easier to design/style later!

- ▼ Q6. What is the difference between tags and semantic tags ?

  With the embark of HTML5, semantic tags came into existence



Imagine a book, that has no index, marking, chapter etc. just text all the way from start to finish

In reality, we have meaningful segregations as we need a good reading experience

Semantic tags were introduced to give more meaning and structure to the HTML web pages.

The `div` and `h1` tags do their jobs, but they are not very **intuitive** about the contents/website in general

The idea was to have more **meaningful** tags

▼ Q7. What is SEO ?

SEO or Search Engine Optimisation is a technique

→ It's a technique, which affects the ranking for a website in the website results displayed when a query is given

→ semantic tags play a role in the SEO for that website

→ All this is done to make the search engine understand the website better, at the same time making the markup more readable and intuitive.