3. HTTP, HTTPS, DNS, hosting, domains

Core Web Infrastructure: HTTP, HTTPS, DNS, Hosting & Domains

HTTP — HyperText Transfer Protocol

Definition:
HTTP is the communication protocol that defines how browsers (clients) and web servers talk to each other.

It's stateless, meaning each request is independent — the server doesn't remember previous ones unless cookies or sessions are used.

Workflow:
When you visit a website, your browser sends an HTTP request:

GET /about.html HTTP/1.1
Host: www.example.com

The server replies with an HTTP response:
HTTP/1.1 200 OK
Content-Type: text/html

Then the HTML page is sent back and rendered in your browser.

Key Concepts:

- **Methods:** GET, POST, PUT, DELETE, PATCH → Used for CRUD operations.
- **Status Codes:** 200 (OK), 404 (Not Found), 500 (Server Error).
- **Headers:** Contain meta info like Content-Type, Authorization, Cache-Control.

HTTPS — Secure HTTP

Definition:

HTTPS = HTTP + SSL/TLS encryption.
It ensures all communication between browser and server is **private, authentic, and tamper-proof.**

How It Works:

- Browser connects to server on port 443 (not 80).
- Server presents an SSL certificate (issued by a trusted Certificate Authority).
- Browser verifies the certificate and starts encryption using TLS (Transport Layer Security).
- Now all HTTP traffic is encrypted — passwords, cookies, data cannot be intercepted.

Without HTTPS:

Attackers can perform man-in-the-middle (MITM) attacks, reading or altering unencrypted traffic.

With HTTPS:

- Data confidentiality (encrypted)
- Integrity (cannot be modified in transit)
- Authenticity (verifies you're talking to the real site)

Developer Note:

You can get SSL certificates via:

- **Free:** Let's Encrypt
- **Paid:** Namecheap, GoDaddy, Cloudflare Pro, etc.

DNS — Domain Name System

Definition:

- DNS is the Internet's phonebook.

- It translates human-readable domain names (like google.com) into machine-readable IP addresses (like 142.250.190.78).

Process (Simplified):

- Browser checks its local cache.
- If not found, it queries a recursive DNS resolver (your ISP or Google DNS 8.8.8.8).
- Resolver checks:
  - Root Server → "Who manages .com domains?"
  - TLD Server (.com) → "Ask the authority for google.com."
  - Authoritative Server → "Here's the IP address."
- Browser caches the IP and connects directly to it.

Common Record Types:

Record	Purpose	Example
A	Maps domain → IPv4	example.com → 192.168.1.5
AAAA	Maps domain → IPv6	example.com → ::1
CNAME	Alias → another domain	www → example.com
MX	Mail server record	mail.example.com
TXT	Verification or security data	SPF, DKIM, etc.

Hosting — Where Your Website Lives

Definition:

Web hosting is the physical or virtual server space where your website's files and code are stored and served to users over the internet.

Types of Hosting:

Type	Description	Example Use
Shared Hosting	Many sites share one server	Small static or portfolio sites
VPS (Virtual Private Server)	Shared hardware, isolated resources	Small businesses
Dedicated Server	Full physical server for one client	Large enterprises
Cloud Hosting	Scalable virtual servers	AWS, Google Cloud, DigitalOcean
Serverless / Managed	You deploy code, provider runs it	Vercel, Netlify, Render, Cloudways

Developer Note:

- Frontend-only sites can be hosted on GitHub Pages, Netlify, or Vercel.
- Dynamic PHP apps often run on cPanel or Cloudways.
- Node apps often deploy via Render, Railway, or AWS EC2.

Domains — Your Web Identity

Definition:

A domain name is your website's human-friendly address, registered through a domain registrar (like Namecheap or GoDaddy)

Example;

https://www.example.com

Component	Meaning
.com	Top-Level Domain (TLD)
example	Second-Level Domain
www	Subdomain

Domain Lifecycle:

1. You **register** it with a registrar
2. Registrar links it to **DNS** records (pointing to your host).
3. Browser uses DNS to find your hosting IP
4. Site becomes reachable globally

Key DNS Config for Hosting:

Record	**Target**	**Purpose**
A	Server IP	Main website
CNAME	Hosting alias	For sbdomains (e.g., www)
MX	Mail server	For custom emails

How They All Work Together

1. Users enters: https://www.myshop.com
2. DNS: Translates myshop.com => 203.0.113.5
3. Browser: Initiates TCP + TLS handshake => secure connection
4. HTTP/HTPPS: Sendes a Get / request to that IP
5. Hosting server: Runs PHP/ Node app => generates response
6. Domain Name: Ties it all together as your human-facing brand
7. Browser renders the returned HTML page.

End result: Your website loads

Developer-Level Perspective

Concept	Developer Focus	Tools / Examples
HTTP/HTTPS	API design, secure communication	Postman, SSL, CORS
DNS	Map domain to server	Cloudflare, Namecheap DNS
Hosting	App deployment, scaling	cPanel, Cloudways, Vercel
Domains	Branding & routing	Registrar dashboard
Integration	end -to-end workflow	Git + CI/CD deployment

Summary Mindmap

User -> Domain -> DNS-> IP -> Hosting -> Server -> HTTP/HTTPS -> Response -> Browser

Each component acts like a link in a chain.
Break one (bad DNS confic, no SSL, wrong hosting setup) - your site breaks.