**1. Cách web hoạt động**

* **Mô hình Client-Server là gì? Ví dụ?**

Mô hình Client-Server là mô hình trong đó Client (máy khách) gửi yêu cầu và Server (máy chủ) xử lý rồi phản hồi lại.

VD: Truy cập Facebook ở trình duyệt và nhấn đăng xuất, client gửi yêu cầu đăng xuất đến máy chủ của facebook, rồi máy chủ Facebook xử lí đăng xuất tài khoản ra khỏi máy

* **Client và Server khác nhau thế nào?**

Client:

Gửi yêu cầu

Hiển thị kết quả

Sever:

Nhận yêu cầu

Xử lí, lưu trữ dữ liệu

* **Khi nhập một URL vào trình duyệt, chuyện gì xảy ra?**

Trình duyệt gửi yêu cầu đến DNS (Domain Name System) ánh xạ tên miền thành IP

Kết nối đến máy chủ thông qua giao thức truyền tải VD HTTP, HTTPS,…

Sever nhận, xử lí yêu cầu và phản hồi dữ liệu

Trình duyệt nhận và hiển thị trang web

**2. Domain, Hosting, DNS, IP Address**

* **Domain là gì?**

Domain name (tên miền) là địa chỉ trang web bạn nhập vào trình duyệt

* **Hosting là gì?**

Hosting là nơi lưu trữ dữ liệu của trang web trên máy chủ để mọi người có thể truy cập.

* **DNS hoạt động như thế nào?**

DNS (Domain Name System) chuyển đổi tên miền thành địa chỉ IP để trình duyệt có thể truy cập đúng máy chủ.

**Các bước:**

**B1**: kiểm tra DNS cache cục bộ (trên máy tính) xem có bản ghi IP của tên miền đó không: có -> lập tức truy cập

Không-> chuyển sang bước tiếp

**B2**: Hỏi DNS Resolve: DNS Resolver chịu trách nhiệm tìm ra địa chỉ IP cho tên miền.

**B3**: Truy vấn đệ quy: Nếu Resolver không có kết quả, nó sẽ thực hiện truy vấn đến các máy chủ khác:

* **3.1. Root DNS server**: Trả về địa chỉ của máy chủ DNS cấp cao hơn (.com, .org, v.v.)
* **3.2. TLD DNS server (Top-Level Domain)**: Trả về máy chủ quản lý tên miền cấp hai (ví dụ: example.com).
* **3.3. Authoritative DNS server**: Đây là nơi chứa bản ghi IP chính xác của tên miền.

**B4**: Nhận IP từ kết quả của bước 3 rồi lưu vào cache

**B5**: Dùng IP để kết nối đến website.

* **IP Address là gì?**

Viết tắt của Internet Protocol Address: là 1 địa chỉ dung để định danh 1 thiết bị trong mạng máy tính toàn cầu Internet.

**Gồm 2 loại chính**

**+ IPv4**

Gồm 4 số, mỗi số từ 0 – 255, cách nhau bởi dấu chấm

Dạng IP phổ biến nhất hiện nay

**+ IPv6**

Dài hơn và được tạo ra để giải quyết thiếu IP trong IPv4

**3. HTTP & HTTPS**

* **HTTP là gì?**  
  HTTP (HyperText Transfer Protocol / Giao Thức Truyền Tải Siêu Văn Bản) là giao thức truyền tải dữ liệu giữa Client và Server.

Cổng mặc định: 80

* **HTTPS khác gì so với HTTP?**

HTTPS (HTTP Secure) sử dụng mã hóa SSL/TLS để bảo mật dữ liệu

Cổng mặc định 442

* **Tại sao nên dùng HTTPS thay vì HTTP?**

Bảo mật dữ liệu tránh bị đánh cắp

Được google ưu tiên xếp hạng SEO

SEO: là vị trí của một trang web trên trang kết quả tìm kiếm của Google hoặc các công cụ tìm kiếm khác. Khi ai đó tìm kiếm một từ khóa liên quan, trang web của bạn có thể xuất hiện ở vị trí cao hoặc thấp tùy thuộc vào mức độ tối ưu hóa SEO của nó

* **HTTP Method là gì?**

HTTP Method là giao thức giao tiếp giữa Client và Sever. 1 só HTTP Method thường gặp

+ GET: Lấy dữ liệu từ sever

+ POST: Gửi dữ liệu lên sever

+PUT: Cập nhật dữ liệu

+DELETE: Xóa dữ liệu

+PATCH: Giống PUT nhưng chỉ cập nhật 1 phần dữ liệu

+HEAD: Giống GET nhưng chỉ nhận về phần header…

* **HTTP Status Code là gì?**

Là mã phản hồi từ server cho biết kết quả yêu cầu.

VD:

200: OK : Thành công

404: Not Found : Không tìm thấy tài nguyên yêu cầu

500: Internal Sever Error: Lỗi máy chủ

**4. Cookies & Sessions**

* **Cookies là gì?**

Cookies là dữ liệu nhỏ lưu trên trình duyệt của người dùng, thường dùng để ghi nhớ thông tin đăng nhập, cài đặt trang web

* **Sessions khác gì so với Cookies?**

Session là một khoảng thời gian mà người dùng tương tác với ứng dụng hoặc website. Nó giúp lưu trữ dữ liệu tạm thời về người dùng trên server để duy trì trạng thái giữa các yêu cầu HTTP (HTTP là stateless, tức là không nhớ thông tin giữa các request).

Khác nhau với cookies:

Lưu trữ: Trên sever

Dung lượng: Ít

Bảo mật: An toàn hơn

Thời gian tồn tại: Thời gian quy định hoặc khi đóng Browser

* Khi nào nên sử dụng Cookies, khi nào nên dùng Sessions?

Dùng Cookies**:** Khi cần lưu trữ dữ liệu lâu dài trên trình duyệt (VD: Ghi nhớ đăng nhập).

Dùng Sessions**:** Khi cần bảo mật dữ liệu hơn và chỉ cần lưu trữ tạm thời (VD: Thông tin giỏ hàng khi mua sắm).

**5. API là gì**

API (Application Programming Interface) là giao diện lập trình ứng dụng, cho phép các phần mềm hoặc hệ thống khác nhau giao tiếp với nhau. Nó định nghĩa cách các phần mềm có thể yêu cầu dịch vụ hoặc tài nguyên từ nhau.

API có thể là một tập hợp các hàm, phương thức, hoặc các yêu cầu mà một ứng dụng hoặc dịch vụ cung cấp để các phần mềm khác có thể sử dụng. Ví dụ, một ứng dụng có thể sử dụng API của một dịch vụ web để lấy dữ liệu, gửi yêu cầu, hoặc thực hiện các tác vụ cụ thể mà không cần phải biết cách thực hiện các thao tác bên trong dịch vụ đó.

Ví dụ, nếu bạn muốn xây dựng một ứng dụng và muốn nó hiển thị thông tin thời tiết, bạn có thể sử dụng API của một dịch vụ thời tiết để lấy dữ liệu về nhiệt độ và dự báo thời tiết.

**6. SSR & CSR**

* **SSR (Server-Side Rendering) là gì?**

SSR (Server-Side Rendering) là một kỹ thuật trong lập trình web, trong đó nội dung của một trang web được tạo ra trên máy chủ thay vì trên trình duyệt của người dùng. Khi người dùng yêu cầu trang, máy chủ sẽ tạo ra nội dung HTML hoàn chỉnh và gửi trả về trình duyệt

Chi tiết: Khi người dùng yêu cầu một trang web, server sẽ xử lý tất cả các logic và dữ liệu, sau đó trả về một trang HTML hoàn chỉnh. Điều này giúp trình duyệt hiển thị nội dung ngay lập tức, mà không cần phải tải JavaScript để xây dựng DOM

* **CSR (Client-Side Rendering) là gì?**

CSR (Client-Side Rendering) là một kỹ thuật trong lập trình web, trong đó quá trình render (hiển thị) nội dung của trang web diễn ra trên trình duyệt của người dùng thay vì trên máy chủ. Khi người dùng yêu cầu một trang web, máy chủ sẽ gửi về một file HTML cơ bản (thường là trống hoặc chứa cấu trúc cơ bản) và mã JavaScript. Sau đó, trình duyệt sẽ tải JavaScript và thực hiện render nội dung của trang ngay trên client

* Khác biệt giữa SSR và CSR là gì?

A screenshot of a computer

AI-generated content may be incorrect.

* Khi nào nên dùng SSR, khi nào nên dùng CSR?

A screenshot of a black screen

AI-generated content may be incorrect.

**BONUS KIẾN THỨC:**

**PORT:**

**1. Khái niệm về Port**

Port là một điểm kết nối trong giao tiếp mạng, giúp xác định chính xác loại dịch vụ hoặc ứng dụng mà dữ liệu được gửi tới. Mỗi dịch vụ hoặc ứng dụng đang chạy trên một máy tính đều được gán một số port cụ thể để phân biệt.

Trong giao tiếp mạng, **địa chỉ IP** giúp xác định máy tính (hoặc thiết bị) nhận và gửi dữ liệu, trong khi **port** giúp chỉ rõ ứng dụng hoặc dịch vụ trên máy tính đó mà dữ liệu sẽ được chuyển đến.

**2. Cấu trúc của một Địa chỉ IP và Port**

Khi một máy tính gửi một yêu cầu qua mạng, nó không chỉ cần biết địa chỉ IP của máy chủ, mà còn cần phải biết port mà dịch vụ của máy chủ đang lắng nghe. Một địa chỉ IP kết hợp với port tạo thành một "socket" (ví dụ: 192.168.1.1:8080).

**3. Các loại Port**

Ports được chia thành ba loại chính:

* **Well-known Ports (Port nổi tiếng)**: Đây là các port có số từ 0 đến 1023. Những port này thường được sử dụng cho các dịch vụ mạng phổ biến như:
  + HTTP (Port 80)
  + HTTPS (Port 443)
  + FTP (Port 21)
  + SMTP (Port 25)
  + DNS (Port 53)
  + SSH (Port 22)
* **Registered Ports (Port đã đăng ký)**: Các port từ 1024 đến 49151 được sử dụng bởi các ứng dụng hoặc dịch vụ không phải hệ thống, nhưng chúng được đăng ký và quy định bởi IANA (Internet Assigned Numbers Authority). Ví dụ, port 3306 là của MySQL, port 5432 là của PostgreSQL.
* **Dynamic and Private Ports (Port động và riêng tư)**: Các port từ 49152 đến 65535 thường được sử dụng cho các kết nối tạm thời hoặc bởi các ứng dụng cụ thể trong thời gian chạy.

**4. Vai trò của Port trong giao tiếp mạng**

Khi bạn truy cập một trang web, chẳng hạn, trình duyệt của bạn sẽ gửi một yêu cầu đến một máy chủ web qua địa chỉ IP của máy chủ đó và port 80 (cho HTTP) hoặc 443 (cho HTTPS). Máy chủ nhận yêu cầu này thông qua port tương ứng và phản hồi lại trình duyệt.

**5. Port và Tường lửa (Firewall)**

Port cũng đóng vai trò quan trọng trong bảo mật mạng. Các tường lửa (firewalls) có thể chặn hoặc mở các port nhất định để kiểm soát lưu lượng mạng. Điều này giúp bảo vệ các dịch vụ mạng khỏi các mối đe dọa từ bên ngoài.

Ví dụ: nếu bạn muốn chạy một ứng dụng web trên máy chủ của mình, bạn cần đảm bảo rằng port 80 (HTTP) hoặc port 443 (HTTPS) mở và có thể nhận kết nối từ bên ngoài. Nếu không, người dùng không thể truy cập vào ứng dụng web của bạn.

**6. Tính chất của Port trong Lập trình Web**

* Khi bạn tạo một ứng dụng web, bạn cần phải cấu hình server của mình để lắng nghe trên một port cụ thể (thường là port 80 cho HTTP hoặc port 443 cho HTTPS).
* Nếu bạn sử dụng Node.js, một server Express.js có thể lắng nghe trên một port bạn chỉ định, ví dụ: app.listen(3000), để ứng dụng web của bạn chạy trên port 3000.

**7. Port và Proxy**

Một proxy server là một máy chủ trung gian giúp chuyển tiếp yêu cầu giữa client và server đích. Proxy server có thể thay đổi hoặc ẩn đi port thực sự mà client đang kết nối đến, điều này giúp ẩn đi cấu trúc mạng thực sự của hệ thống.