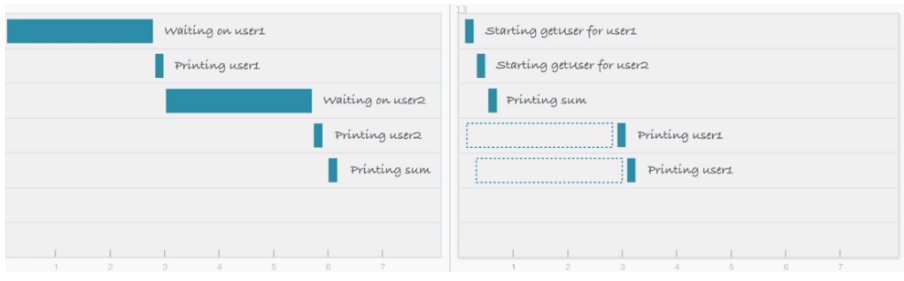
**Tổng quan nodeJs**

Khái niệm :

* Là môi trường chạy phía server, mã nguồn mở được xây dựng dựa trên V8 của Chrome - công cụ Javascript.
* Nó cung cấp thời gian chạy đa nền tảng (window, linux, mac,....)

Các tính năng chính của NodeJs :

* *Cực nhanh* : phần core NodeJs được xây dựng dựa trên C++ nên tốc độ xử lý và hiệu năng khá cao.
* *Đơn luồng* : NodeJs tuân theo mô hình đơn luồng với vòng lặp sử kiện.
* *Mã nguồn mở* : NodeJs có cộng đồng mã nguồn mở => đã tạo ra nhiều module tuyệt vời để bổ sung cho NodeJs.
* *Mô hình non-blocking I/O và theo hướng sự kiện* :
* Tất cả API của nodeJs là bất đồng bộ
* Input/Output có thể là đọc ghi file nội bộ hoặc request/response, tạo HTTP đến API và chúng khá tốn thời gian nên sẽ chặn các tính năng khác.
* Ví dụ : Chúng ta cần dữ liệu của user1 và user2



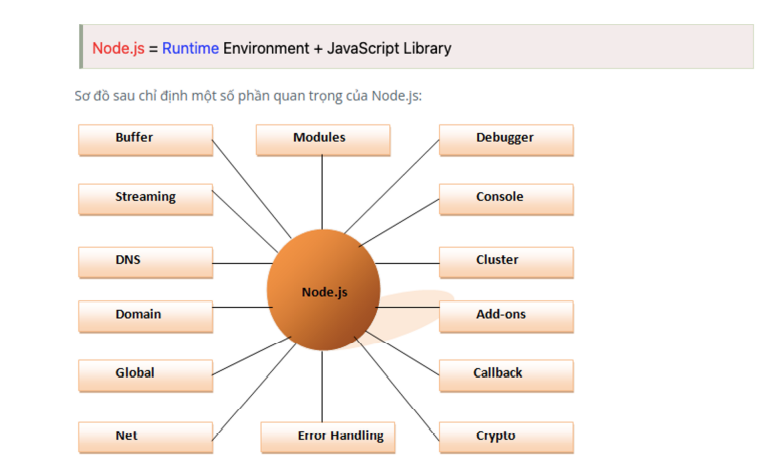
Trường hợp Blocking/IO : Dữ liệu của user2 sẽ không xuất hiện cho tới khi dữ liệu của user1 xuất hiện xong.

Trường hợp Non-blocking I/O: Có thể có dữ liệu từ user2 mà không cần phải đợi dữ liệu user1 xuất hiện.

*Ứng dụng NodeJs :*

* Có thể xây dựng các ứng dụng khác nhau như app, web, trò chuyện thời gian thực (real-time), REST Api server => chủ yếu được xây dựng các chương trình mạng như web server tương tự Java, PHP, …

*Sơ đồ thành phần NodeJs*



*NodeJs Module*

Khái niệm :

* Là các khối mã đóng gói giao tiếp với ứng dụng bên ngoài trên cơ sở, chức năng liên quan của chúng.
* Module là 1 tập hợp nhiều tệp, thư mục
* Chúng có khả năng tái sử dụng và chia nhỏ các đoạn mã phức tạp thành các đoạn mã có thể quản lý được

*Phân loại :*

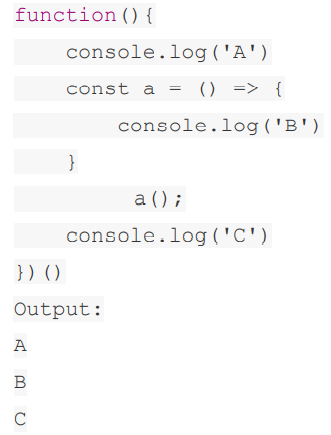
* *Core Module* : Là các module được tích hợp và đi kèm với cài đặt của NodeJs
* Có thể cài đặt vào chương trình bằng cách sử dụng require : const module = require(‘module’)
* Ví dụ : http, fs, …

* *Local Module* : Module cục bộ được tạo cục bộ trong ứng dụng của bạn và cung cấp ra bên ngoài thông qua export hoặc module.export và các tệp khác sử dụng chức năng của local module thông qua require().
* *Third - Party Modules :*
* Là các module có sẵn bằng cách sử dụng Node Package Manager (NPM)
* Có cộng đồng support lớn
* Cài đặt vào dự án bằng cách sử dụng : npm i name\_module
* Sử dụng bằng cách sử dụng require()
* Ví dụ : readline-sync, mysql, express,...

*Blocking và Non-blocking*

*Blocking (đồng bộ)*

* Đề cập đến việc ngăn chặn hoạt động tiếp theo cho đến khi hoạt động hiện tại kết thúc => các phương thức chặn được thực thi từng dòng một



* *Non-blocking (không đồng bộ) :* Đề cập đến việc chương trình không chặn việc thực hiện các hoạt động tiếp theo => Chúng không nhất thiết phải thực thi theo từng dòng một.
* Chương trình gọi hàm và chuyển đến thao tác tiếp theo mà không đợi nó trả về.

*Phân biệt Node Js và Browser*

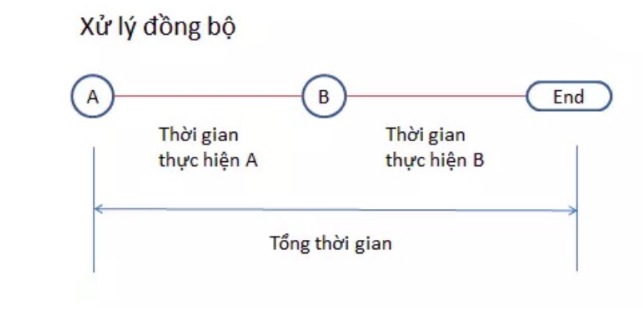
* Node Js : là 1 nền tảng phát triển độc lập được xây dựng ở trên javascript runtime của chrome mà chúng ta có thể xây dựng các ứng dụng mạng một cách nhanh chóng và dễ dàng.
* Trình duyệt web (browser) : phần mềm cho phép truy cập website biên dịch được mã html, css, javascript và lưu trữ vào cookie.
* Giống nhau : Cả 2 môi trường đều để thực thi javascript.

|  | NodeJs | Browser |
| --- | --- | --- |
| Vị trí trong mô hình web | Máy chủ (Server) | Client |
| Sử dụng DOM, LocalStorage, sessionStorage, … | Không | Có |
| Sử dụng Node Modules | Có | Không |
| Sử dụng từ khóa “require” | Có | Không |
| Giao diện | Không | Có |
| Quyền truy cập hệ thống người dùng | Có | Không |

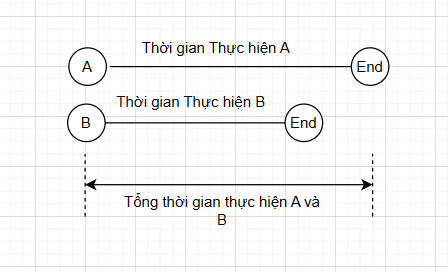
**Xử Lý Bất Đồng Bộ 1**

Các khái niệm :

* Đồng bộ :
* Là cơ chế tuần tự, các dòng code chạy tuần tự từ trên xuống dưới.
* Các hàm gọi nhau sẽ đợi kết quả hàm khác trả về rồi mới chạy tiếp.



* Bất đồng bộ :
* Code thực thi không tuần tự.
* Nhiều lệnh chạy cùng 1 lúc.

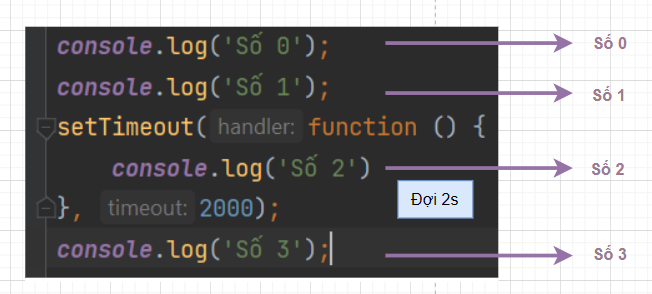


***Mô hình event loop***

Cho đoạn code :



Theo dự đoán :



Kết quả :



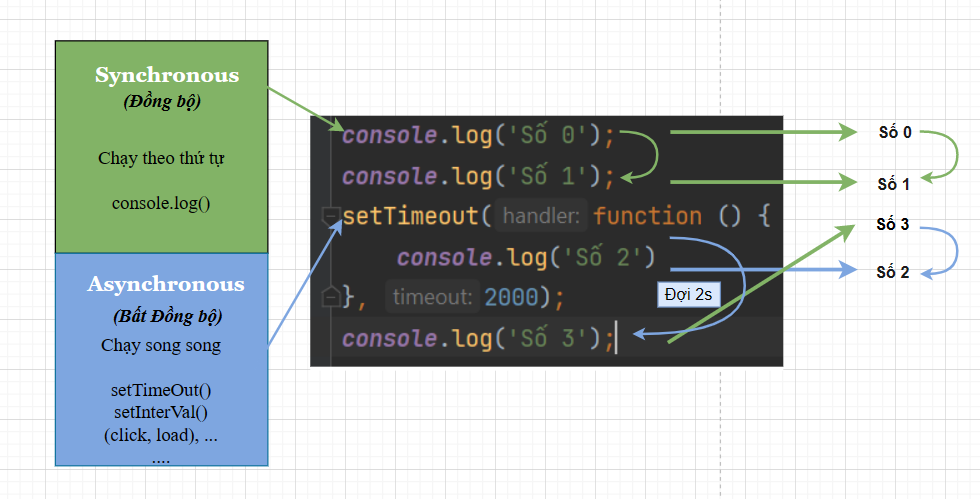
Thực tế :



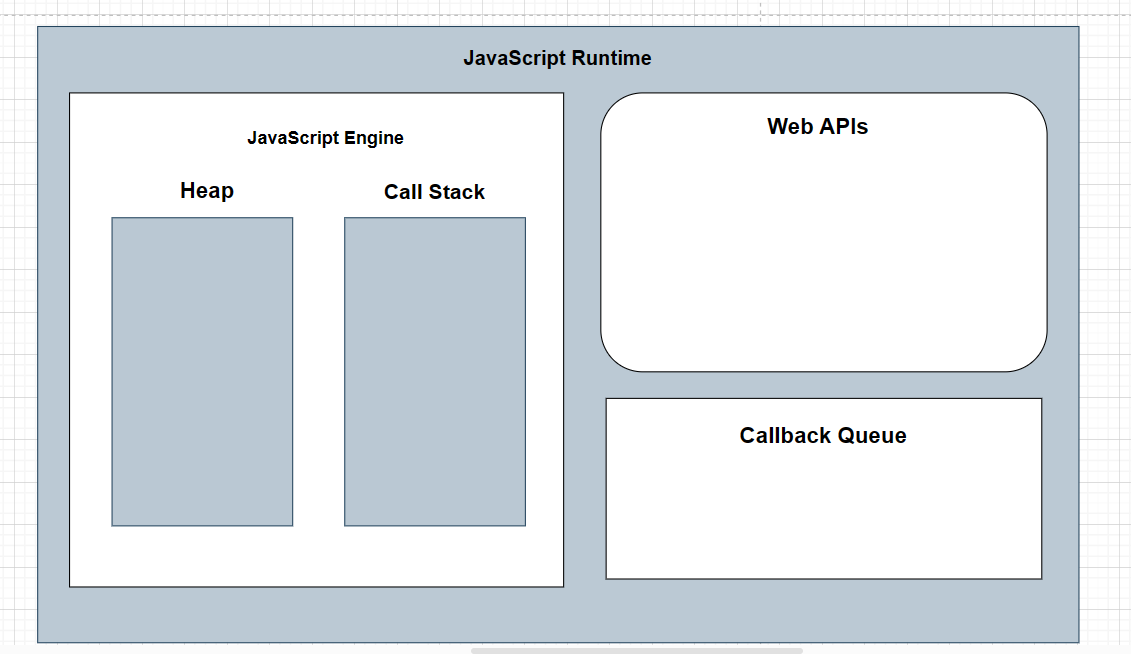
Tại sao lại không theo đúng thứ tự viết code ?

=> Do vấn đề đồng bộ và bất đồng bộ trong Javascript.

* Khi ta viết code Javascript nó sẽ chia làm 2 nhóm tác vụ cơ bản : đồng bộ (synchronous) và bất đồng bộ (asynchronous)



*Mô hình JavaScript Runtime*

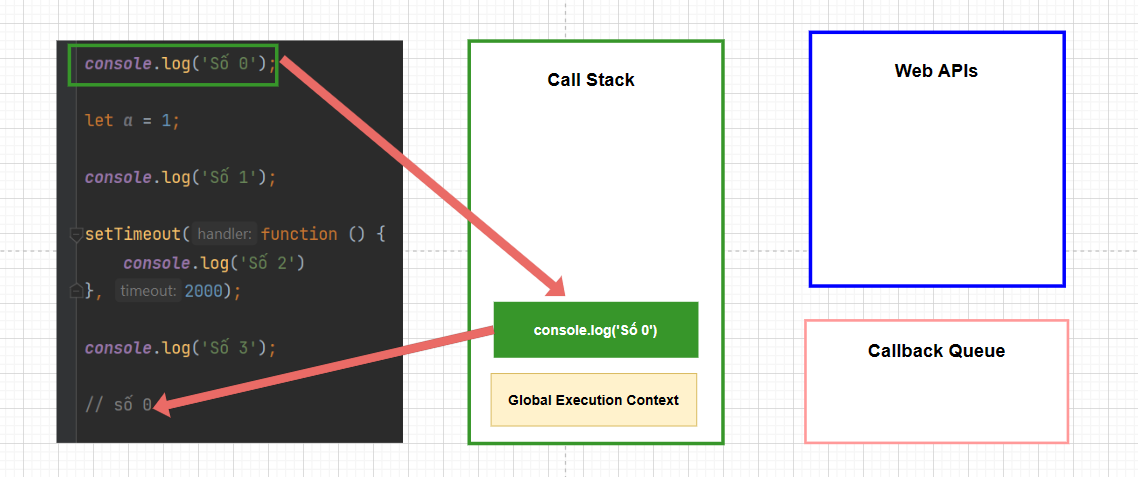


*Chú ý:*

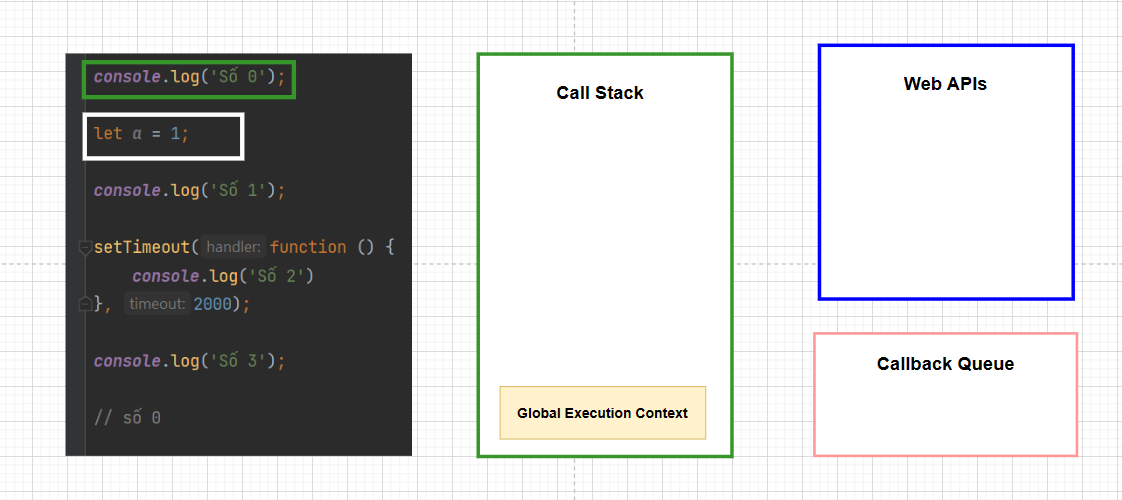
* Call stack: có thể hiểu là gọi luồng theo dõi tiến trình và chuyên xử lý tác vụ đồng bộ.
* Web apis : có thể hiểu là nơi chuyên để xử lý tác vụ bất đồng bộ.
* Callback queue : nhiệm vụ là trung chuyển các hàm đưa từ web apis tới call stacks.

*Thứ tự thực hiện :*

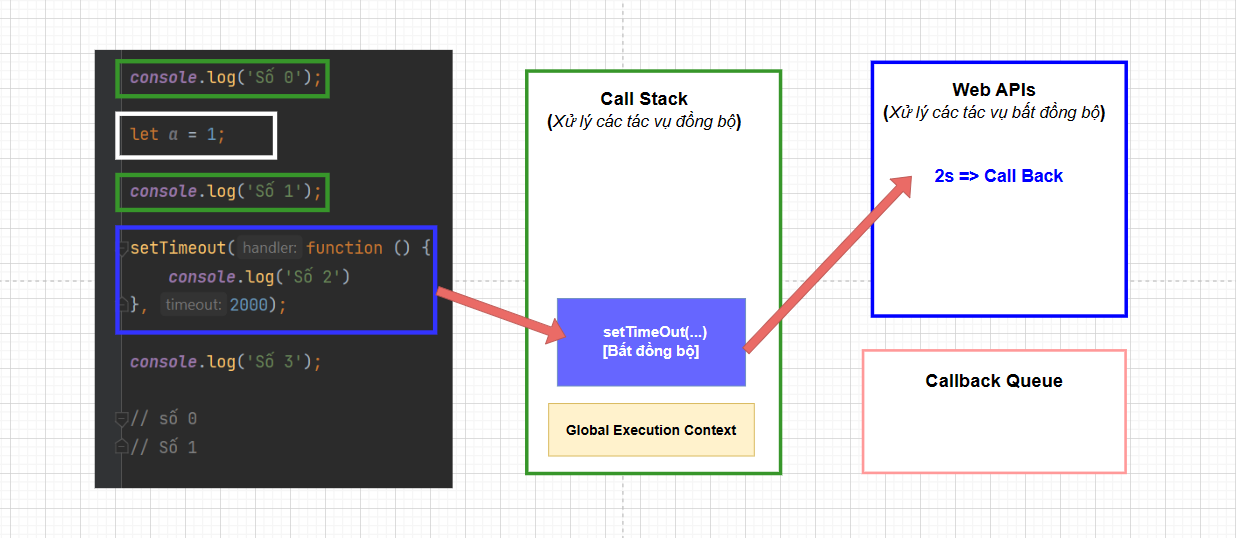
* Dòng đầu tiên gọi console.log(‘Số 0’) :
* Đưa console.log(‘Số 0’) vào call stack và thực hiện log ra ‘Số 0’.



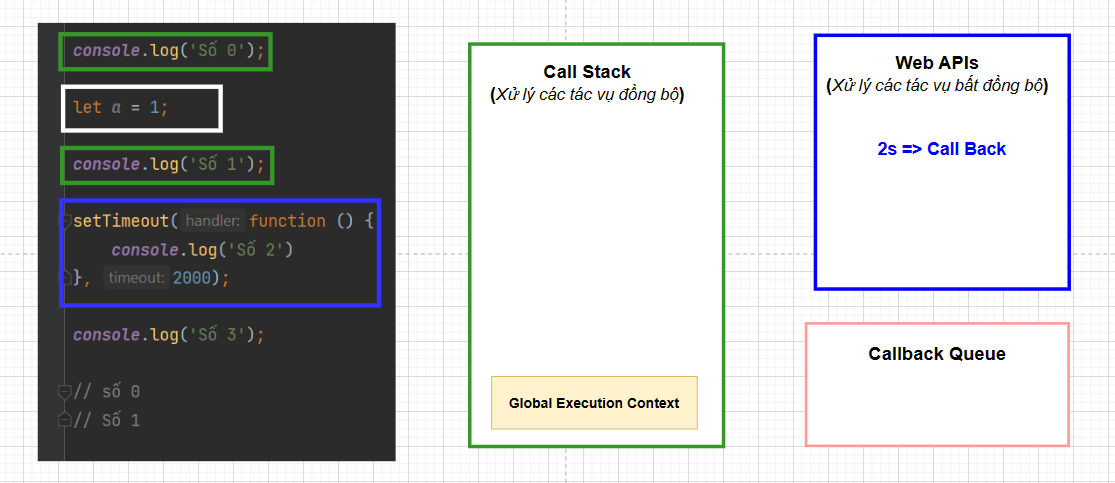
* Dòng tiếp let a = 1 : là cú pháp khai báo thông thường không gọi gì nên sẽ không mang vào call stack.



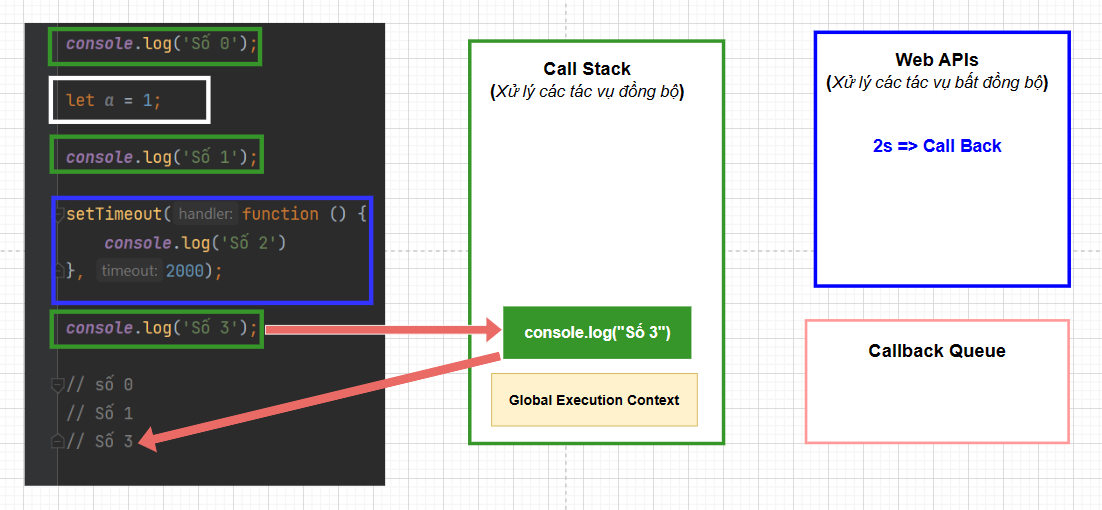
* Dòng tiếp là console.log(‘Số 1’) sẽ là tác vụ đồng bộ nên sẽ tương tự dòng đầu tiên.
* Tiếp đến setTimeOut(...) sẽ được đưa vào call stack và call stack nhận thấy đây là 1 hàm bất đồng bộ mà call stack chỉ xử lý các tác vụ đồng bộ => call stack ném qua web apis (chuyên xử lý tác vụ bất đồng bộ).

j

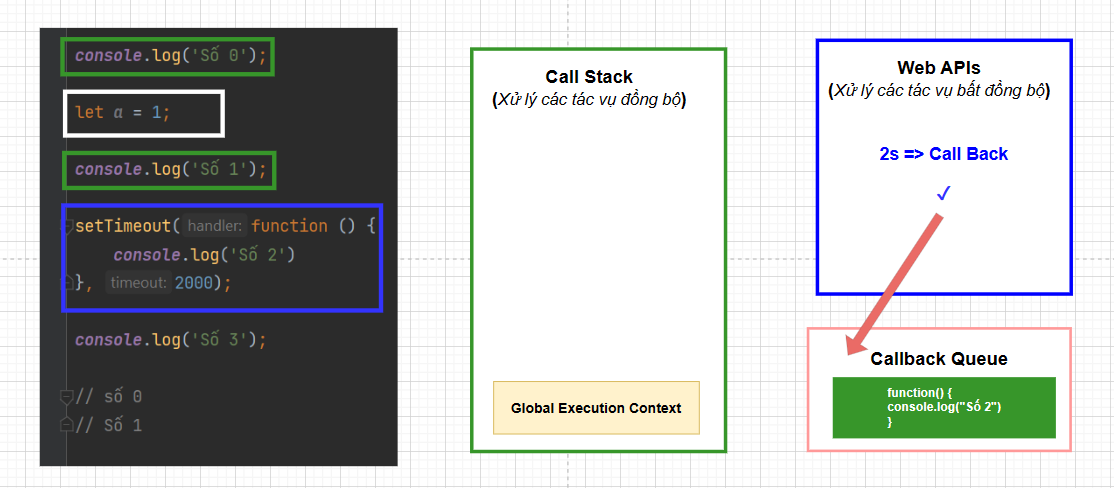
* Sau khi ném setTimeOut(...) cho web apis call stack sẽ trống và đợi tác vụ kế tiếp.



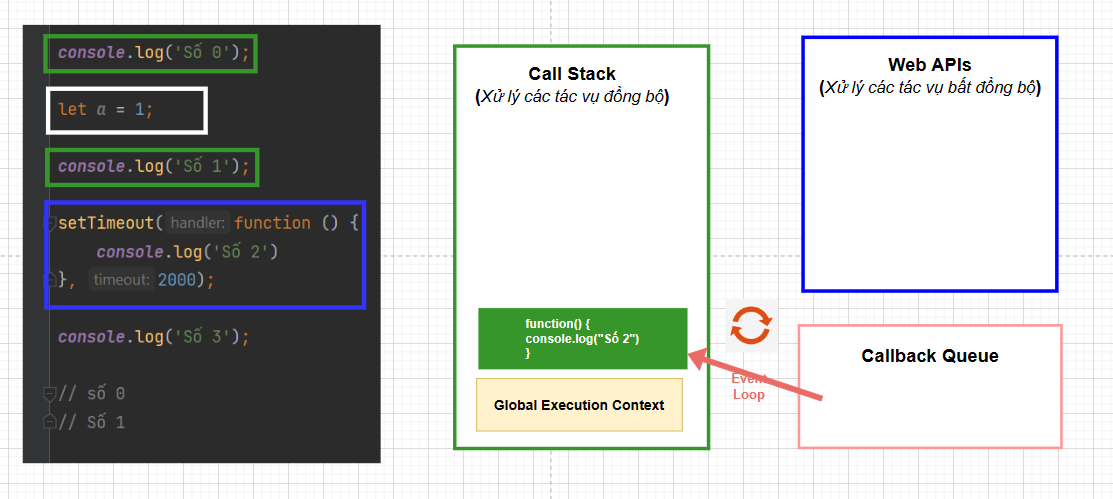
* Tiếp theo là console.log(“Số 3”) : tương tự như hai tác vụ đồng bộ bên trên nó sẽ thực hiện đưa vào call stack và log ra “Số 3”.



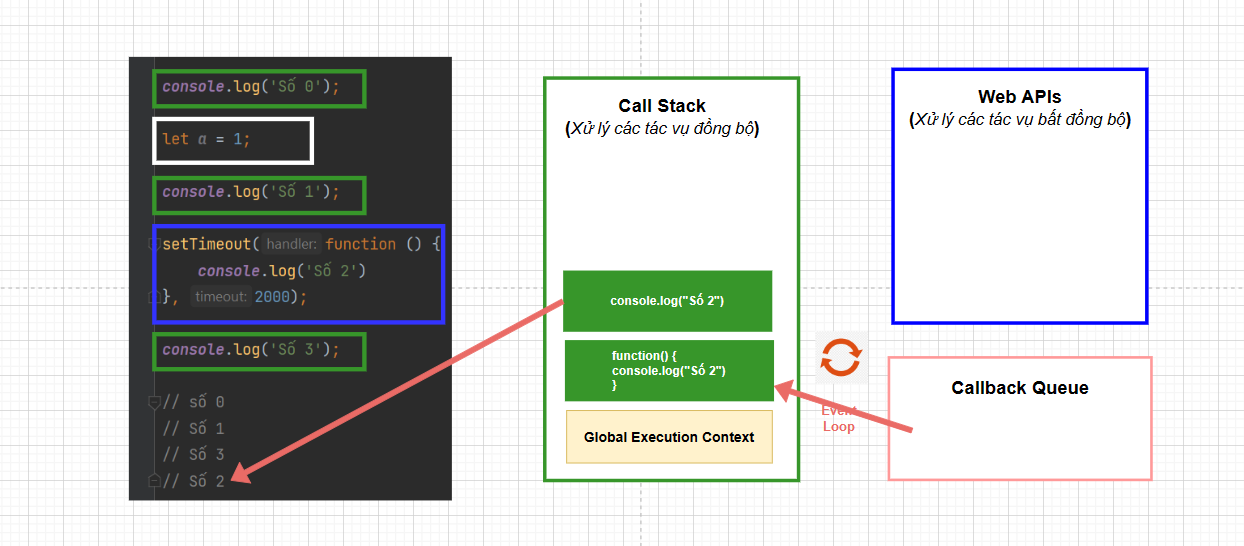
* Web Apis sau khi đợi đúng 2s sẽ trả ra một hàm gọi là callback (code nằm trong setTimeOut(..)) và callback đó sẽ chuyển xuống callback queue



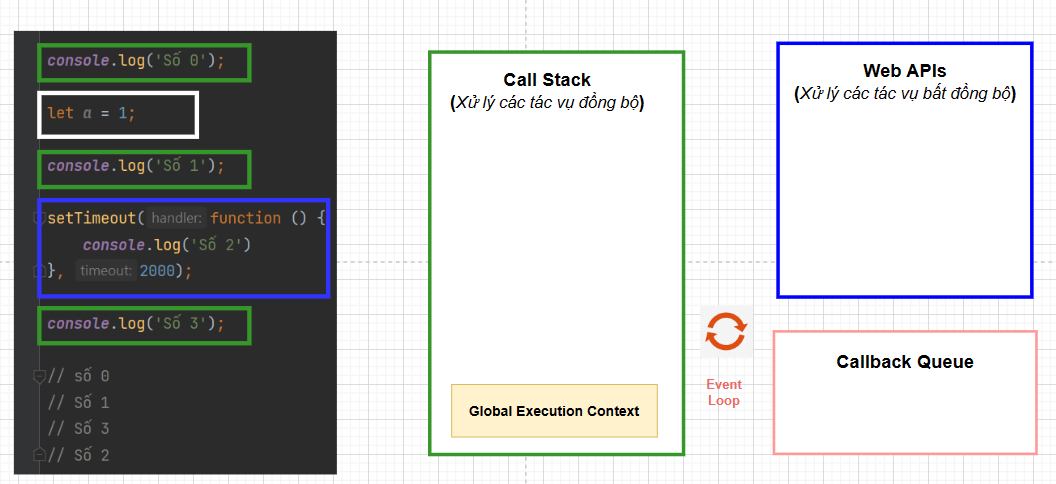
* Lúc này web apis đã hoàn thành.
* Để kết nối giữa callback queue và call stack : có một sự kiện là event loop và nhiệm vụ của nó là kiểm tra call stack đã trống hay chưa và còn tác vụ nào sẽ chạy vào call stack hay không.
* Khi trong call stack trống và bên ngoài không còn tác vụ nào chạy vô nữa => call stack sẽ rảnh rỗi => Tiếp tục event loop kiểm tra xem callback queue xem có hàm nào đang chờ không => Nếu có event loop chuyển hàm đó sang call stack đến thực thi.



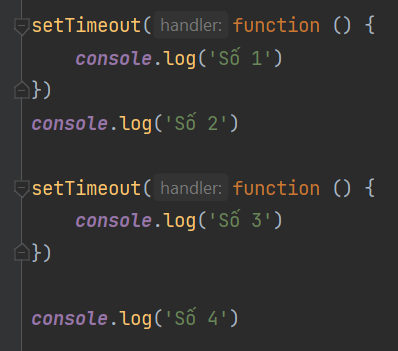
* Lúc này hàm được chuyển sang đó sẽ được thực hiện
* Trong function gọi console.log(“Số 2”) => đưa console.log(‘Số 2’) vào call stack để thực hiện và log ra “Số 2”

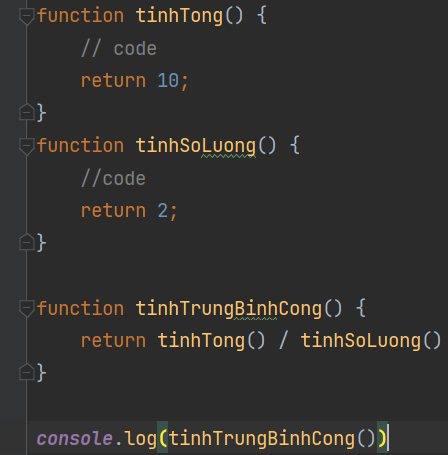


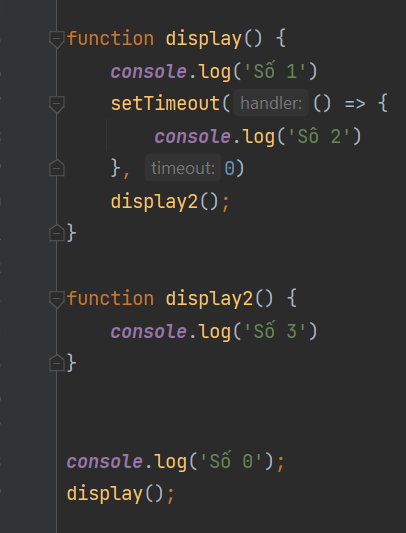
* Sau khi log ra số 2 sẽ xóa console.log(“Số 2”) ra khỏi call stack và function cũng đã hoàn thành => function cũng bị loại khỏi call stack.
* Cuối cùng kết quả ra được như đáp án.



*Bài tập :* Nhìn đoạn code bên dưới và thực hành vẽ theo mô hình event loop







***Xử lý bất đồng bộ***

*Xử lý bất đồng bộ là gì ?*

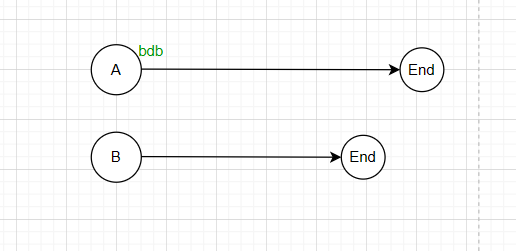
* Là việc đưa một hoặc một số câu lệnh bất đồng bộ trở lại chạy có thứ tự. Có 3 cách xử lý bất đồng bộ hay được sử dụng :
* Callback
* Promise
* Async/Await

*Vậy tại sao cần xử lý bất đồng bộ ?*

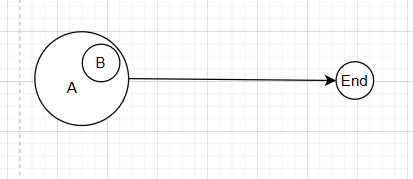
* Chính vì các câu lệnh trả về đồng thời nên các câu lệnh trả về sẽ không theo thứ tự nên sẽ rất khó kiểm soát và sẽ trả về kết quả không mong muốn.

*Callback* :

* Là một hàm sẽ sẽ được truyền vào một hàm khác dưới dạng đối số => thường được sử dụng để xử lý bất đồng bộ.
* Ta có : A là tác vụ bất đồng bộ và B là đồng bộ.



* Mà B cần dữ liệu của A mới kết thúc được, làm sao để B lấy dữ liệu của A ?



* Giải pháp : Đưa B vào làm 1 thành phần của A => Đưa A, B lại thực hiện tuần tự.

Ví dụ :



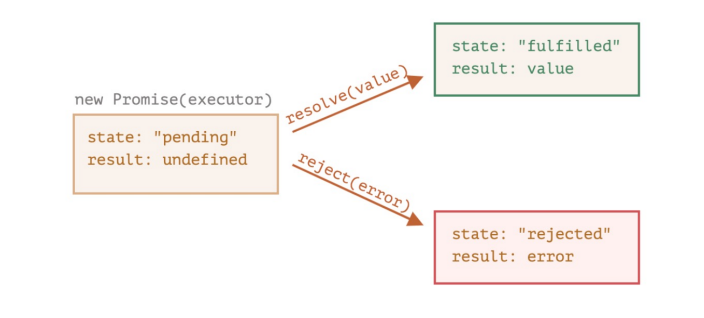


*Callback Hell :* là những trường hợp quá làm dụng callback



***Promise***

* Là 1 kỹ thuật để xử lý bất đồng bộ.
* Đối tượng Promise đại diện cho việc hoàn thành hoặc thất bại của một tác vụ không đồng bộ và giá trị kết quả của nó.

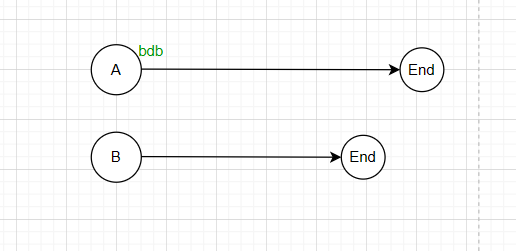


* let promise = new Promise (function (resolve, reject) {})

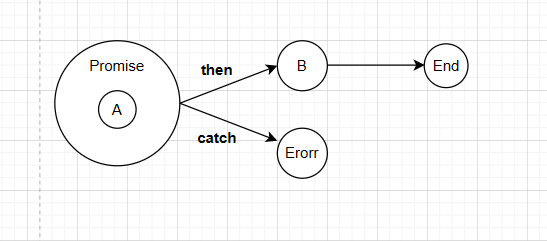
+ resolve là callback đại diện cho hành động thành công.

+ reject là callback đại diện cho hành động thất bại

* Ta có : A là tác vụ bất đồng bộ và B là đồng bộ.



* Mà B cần dữ liệu của A mới kết thúc được, làm sao để B lấy dữ liệu của A ?



* *Giải pháp sử dụng Promise* : đưa A là tác vụ bất đồng bộ đóng gói vào đối tượng Promise và dữ liệu của A sẽ được đưa vào resolve() và lỗi sẽ được đưa vào reject().
* Sử dụng đối tượng promise vừa lấy được .then(function (data) {}) để lấy được dữ liệu của A sau đó gọi đến B. (sử dụng .catch(function (error) {}) để bắt lỗi.

Ví dụ :



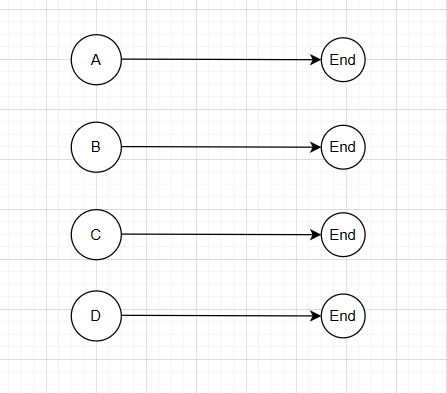


Chú ý :

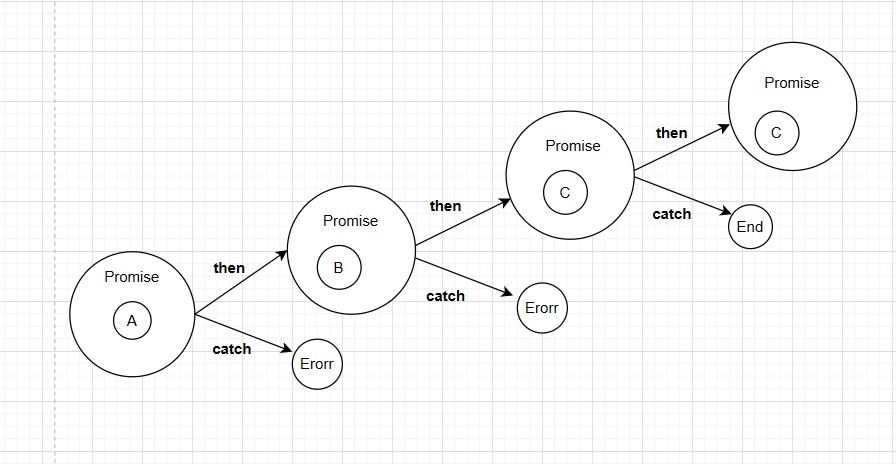
* Không sử dụng lặp với Promise.
* Không quên thêm .catch().
* Tránh kim tự tháp Promise.

**Async/Await**

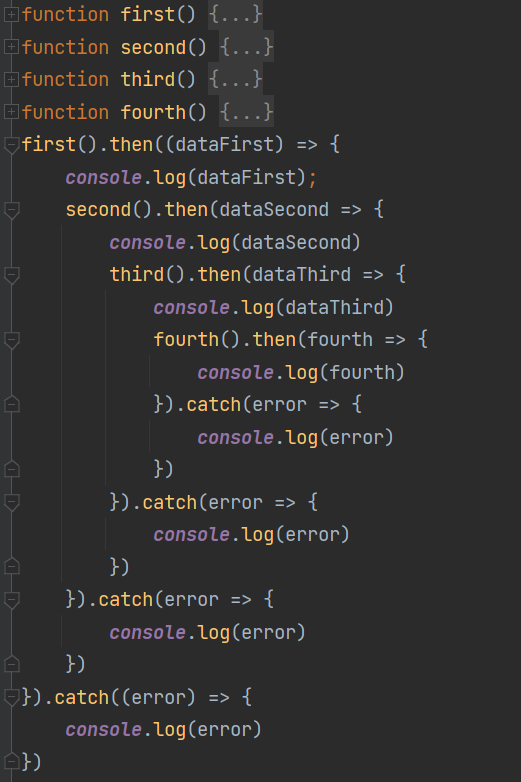
* Ta có A, B, C, D là các tác vụ bất đồng bộ.



* Bây giờ tác vụ A cần dữ liệu tác vụ B, B lại cần dữ liệu tác vụ C, tương tự C cần dữ liệu tác vụ D => Thử cách sử dụng Promise.



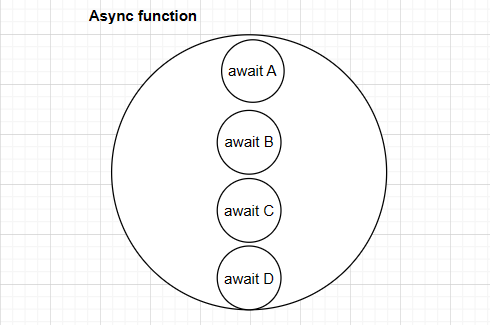
Ví dụ :

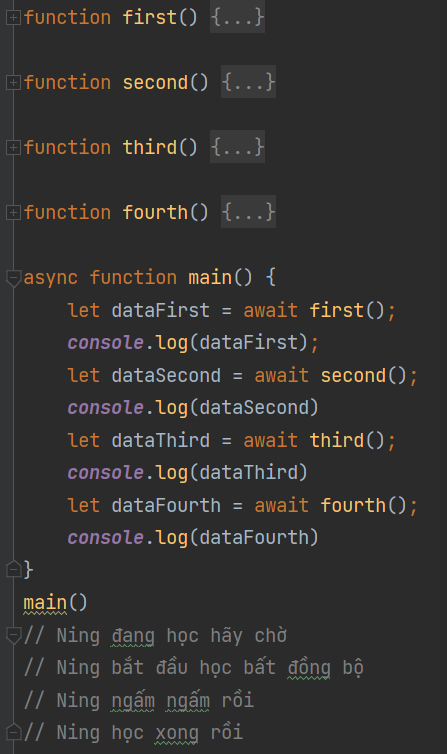


=> Như đã thấy chúng ta đang mắc phải “kim tự tháp” Promise

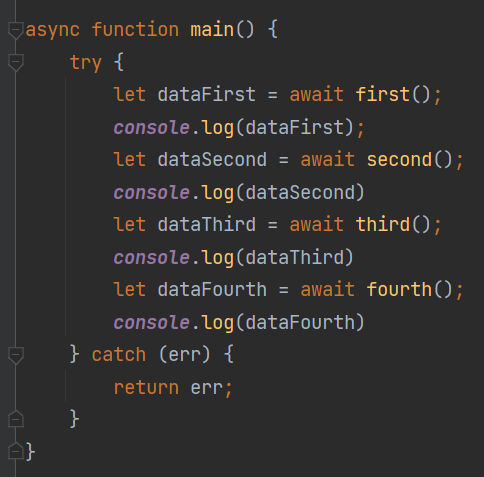
**Giải pháp : Sử dụng async/await;**

* *Khái niệm* : là 1 tính năng của Javascript giúp ta làm việc với các hàm bất đồng bộ 1 cách dễ dàng hơn.
* *Async* :
* Khai báo trước tên hàm (báo hiệu là 1 hàm bất đồng bộ).
* Hàm async trả về một promise.
* Giá trị trả về của async được đóng trong resove của promise.
* *Await* :
* Tạm dừng việc thực hiện các lệnh sau đó và đặt trước lời gọi hàm chờ promise.
* Bắt được data trong then() từ resolve() của promise.
* Chỉ làm việc trong async function và không làm việc với callback.



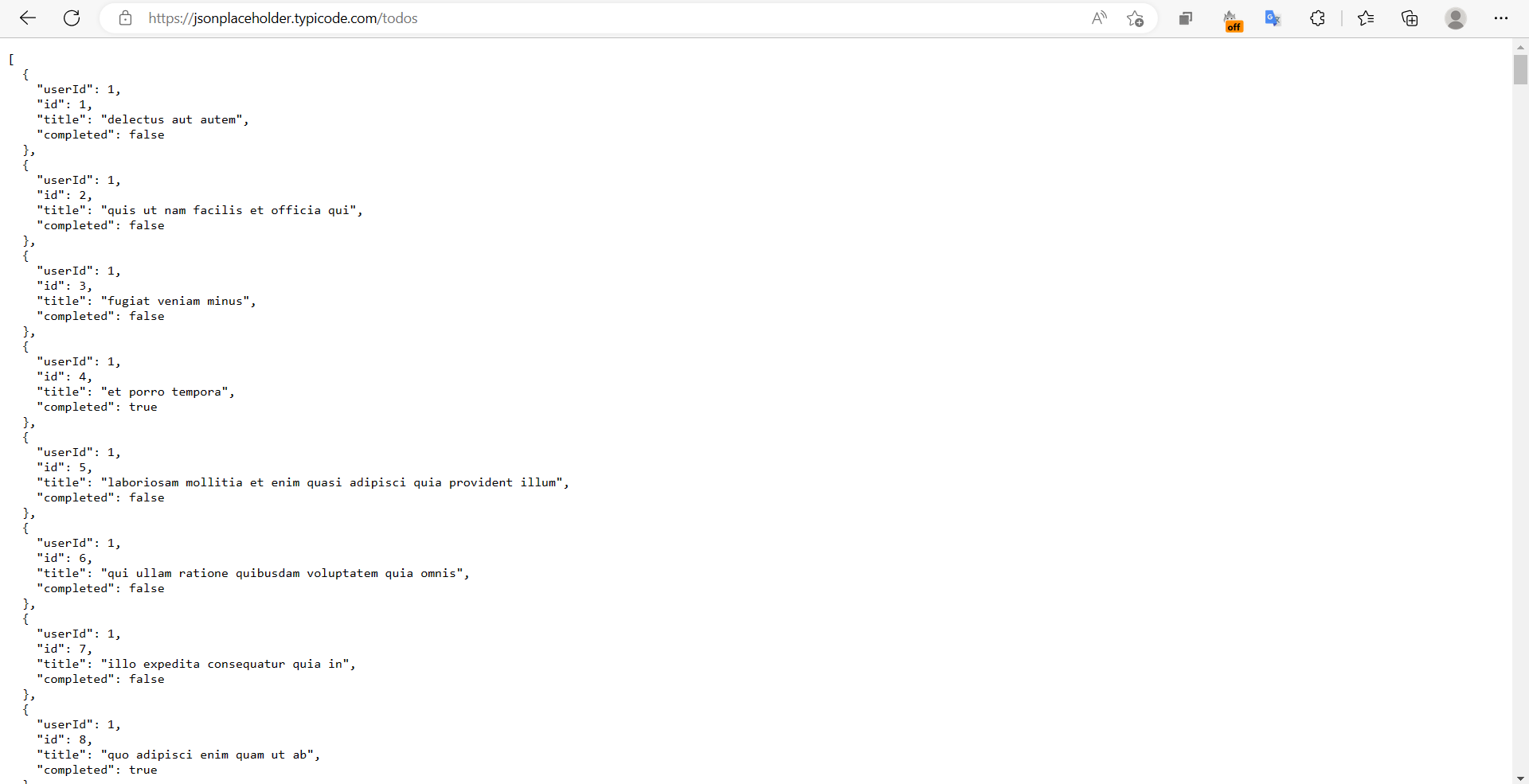


=>> Như đã thấy await chỉ bắt được data trong .then() của promise tương ứng và nếu promise đó lỗi thì đồng thời đoạn code đó cũng sẽ lỗi nên cần sử dụng try catch để bắt lỗi.

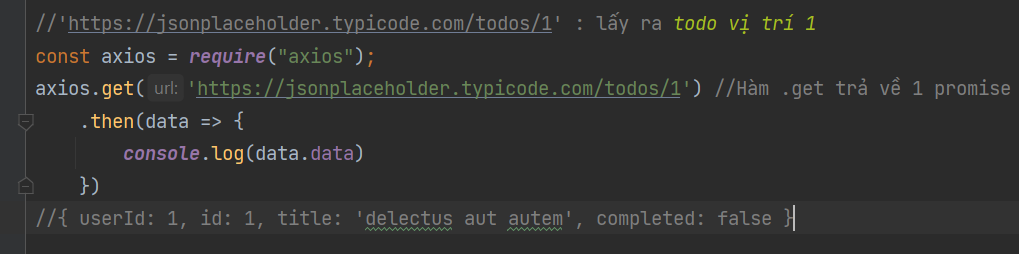


**Axios :**

* Khái niệm : Là một http client (có thể hiểu là 1 phần mềm, thư viện gửi các yêu cầu lấy, thêm, sửa, xóa dữ liệu đến máy chủ)
* Được viết dựa trên nền tảng Promise, hỗ trợ xây dựng các ứng dụng API.
* Ví dụ : Khi truy cập vào đường dẫn [https://jsonplaceholder.typicode.com/todos](https://jsonplaceholder.typicode.com/todos/1) ta thấy được :



* Giờ hãy thực hiện lấy data này về chương trình và log ra bằng axios :



*Bài tập:* Sử dụng đường dẫn [https://jsonplaceholder.typicode.com/todos](https://jsonplaceholder.typicode.com/todos/1) và thư viện axios dùng 2 cách promise và async/await để in ra số lượng các công việc đã hoàn thành (các đối tượng có thuộc tính completed: true).

**Tổng quan mô hình web**

- Web : Là 1 công nghệ trên internet thể hiện thông tin tương ứng với chủ đề của web đó và được thể hiện trên browser (trình duyệt).

* Web tĩnh: là web không có sự thay đổi theo thời gian ( thường là các trang giới thiệu sản phẩm và thông tin như landing page,...) => (chỉ bao gồm HTML , CSS , JS thuần)
* Web động : là web có nội dung, dữ liệu thay đổi theo thời gian ( do có người dùng tương tác) có thêm ngôn ngữ phía server như JAVA, PHP, hoặc node js , có nơi lưu trữ dữ liệu web như mysql , mongodb , sql server

- HTTP : là giao thức truyền tải siêu văn bản sử dụng www để truyền dữ liệu từ web server đến

các trình duyệt web và ngược lại => https : tương tự https nhưng sẽ có thêm chứng chỉ bảo mật

SSL

[HTTP protocol - YouTube](https://www.youtube.com/watch?v=SdcdneSdoV4&list=PL_-VfJajZj0VatBpaXkEHK_UPHL7dW6I3&index=2)

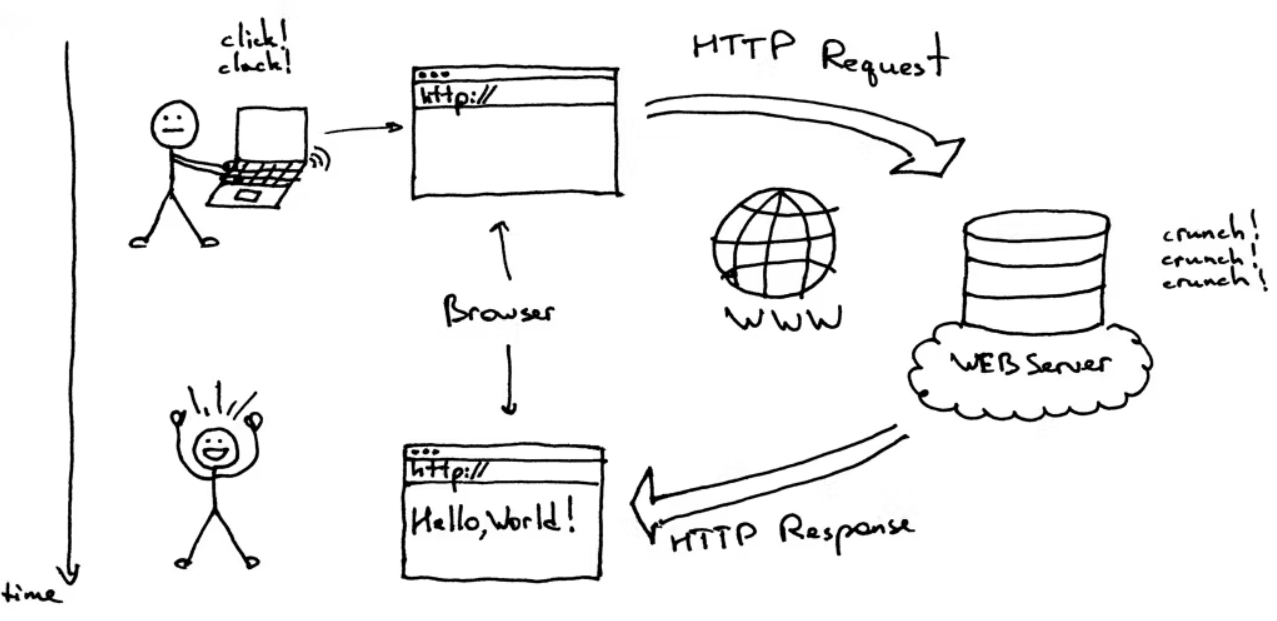
- Phân biệt khái niệm client và server :

* Client (Máy khách) : Đóng vai trò là người dùng và người dùng sẽ sử dụng tài nguyên mà bên khác cung cấp (vd : search google , facebook , ken , james…)
* Server (Máy chủ) : Đóng vai trò máy cung cấp tài nguyên cho những nơi khác sử dụng ( bất cứ máy tính nào cũng có thể là máy chủ ) (có máy chủ ảo hoặc máy chủ thật)
* Client và server sẽ giao tiếp nhau với nhau thông qua http requests và http response

- Http requests và Http response : request là những yêu cầu từ client gửi về server và response là những thông mà server trả về cho client.

Mô hình client-server

[Mô hình Client - Server | Mô hình toàn bộ các websites sử dụng? - YouTube](https://www.youtube.com/watch?v=zoELAirXMJY)



- Phân biệt GET và POST

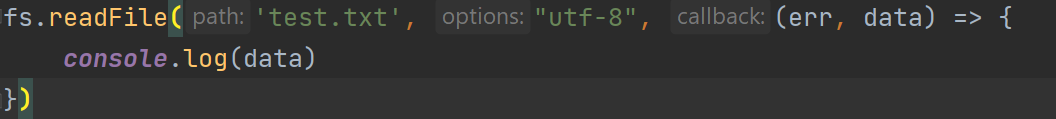
- Web server là server trong mô hình web xử lý các http requests và phản hồi lại cho client bằng http response.

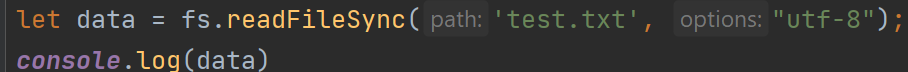
**Làm việc với file trong Node Js**

- Sử dụng thư viện fs - file system

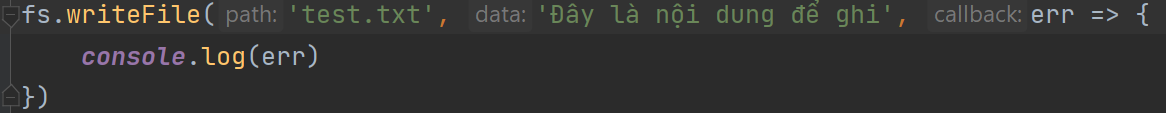
- Các thao tác với file : crud

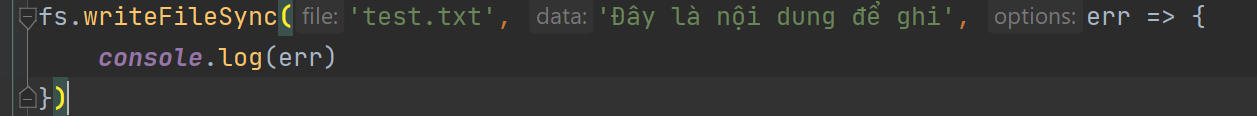
* Đọc file :





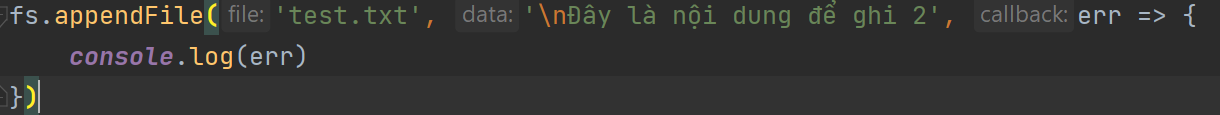
* Ghi file :





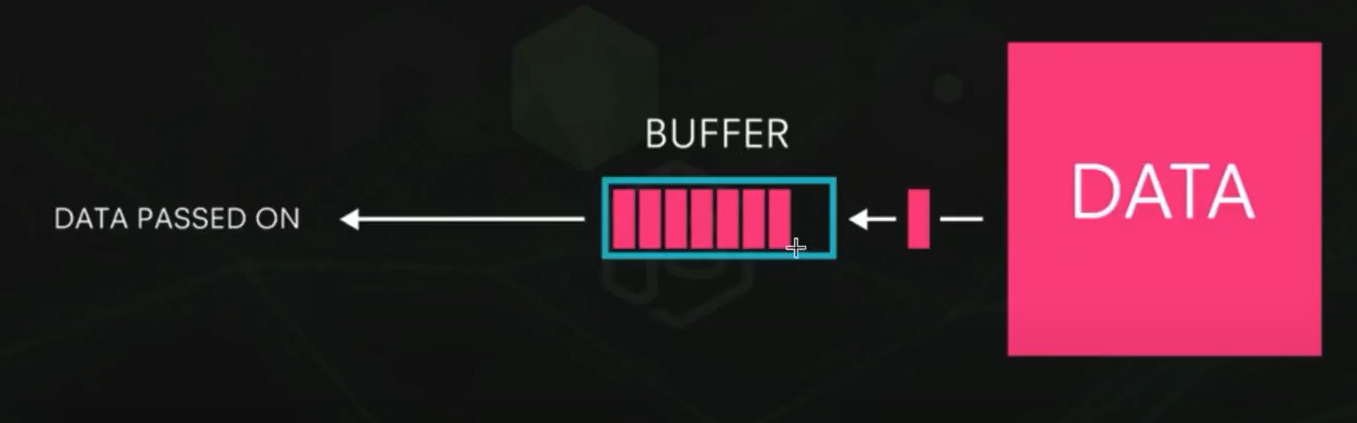
*Node : Có thể chỉ định là thay đổi nội dung của tệp hay không bằng 1 đối tượng chứa thuộc tính flag có thể có các value tương ứng là :*

* *r+ : mở tệp và đọc ghi*
* *w+ : mở tệp và đọc ghi , định vị ở đầu tệp*
* *a : mở tệp để ghi định vị luồng ở cuối tệp*
* *a+ : mở tệp để đọc và ghi định vị ở cuối tệp*
* Nối file:



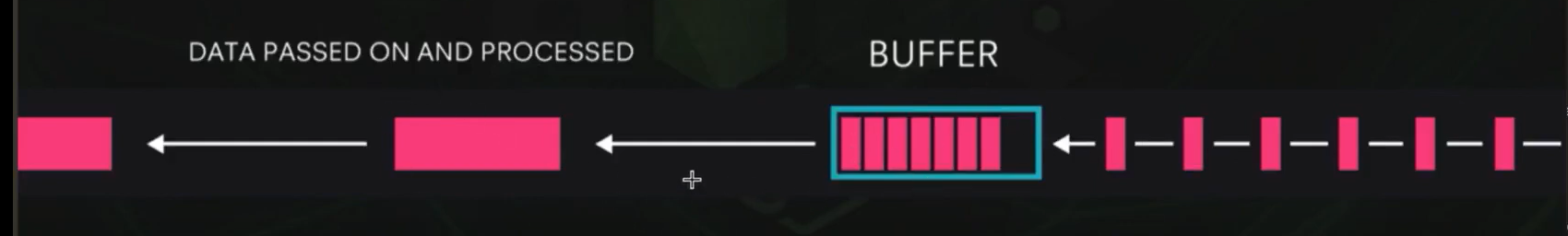
Buffers :

* Là điểm không gian lưu trữ các mẩu data nhỏ => chuyển đến nơi khác khi nó đầy.



- Stream là gì ?

* Khái niệm : là cách xử lý đọc ghi file , truyền thông mạng hoặc trao đổi thông tin đầu cuối 1 cách hiệu quả



* Theo cách thông thường : đọc 1 file , file đó sẽ được đọc và lưu vào bộ nhớ rồi sau đó xử lý nó còn stream sẽ đọc từng đoạn 1 mà không cần lưu vào bộ nhớ .
* Hiệu quả : về bộ nhớ và thời gian.
* Piping stream : cho phép ta lấy dữ liệu đầu ra của stream vào đầu vào cho 1 streams khác, nó cho phép chuyển dữ liệu giữa các file với nhau (

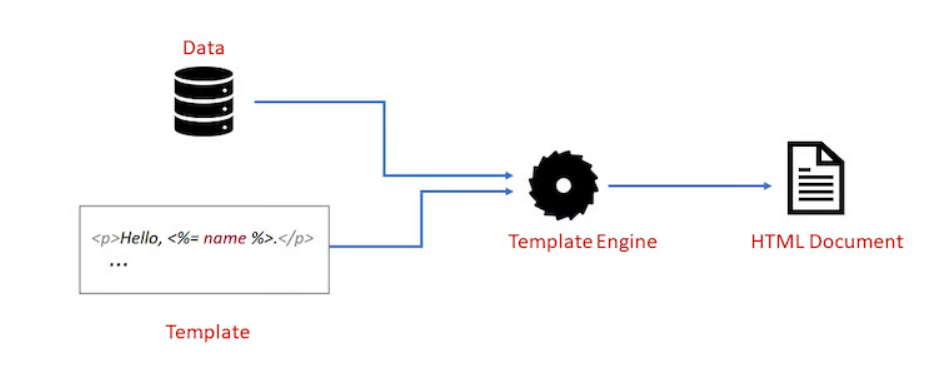
Ví dụ : readStream.pipe(writeStream) lấy data từ readStream viết vào writeStream)

**Template và Static file**

- Template là phần có khung giao diện được xây dựng sẵn có thể tái sử dụng lại được

- Template engine là bộ công cụ để xây dựng giao diện ( nó có thể đưa dữ liệu vào HTML => tạo ra HTML )

- 1 vài thư viện có sẵn : vash , ejs , dust ,...



Static file : là những file không thay đổi khi chương trình chạy

- Các loại file phổ biến : javascript và image và css

**Routing**

* Là đường dẫn giúp xác định các tài nguyên mà mình muốn truy cập vào ( tương ứng với các yêu cầu - request mà người dùng gửi tới server)
* Các thành phần của http-request : *Http Method + request-uri + http-version crlf*

+ Http-method: cách thức gửi tài nguyên (get, post, put, delete,...)

+ Request-uri : dùng để xác định tài nguyên

- URI : là viết tắt của uniform resource identifier => dùng để xác định tài nguyên

bởi chính xác nơi lấy nó hoặc tên nó hoặc cả 2

Ví dụ : ftp://sample/files/document.doc , urn:isbn:978-0132350884

*Phân biệt :*

- URL : là 1 dạng của uri chỉ ra cách thể hiện chính xác lấy tài nguyên về

(nó bao gồm giao thức , domain , name và path tới tài nguyên thể hiện nó)

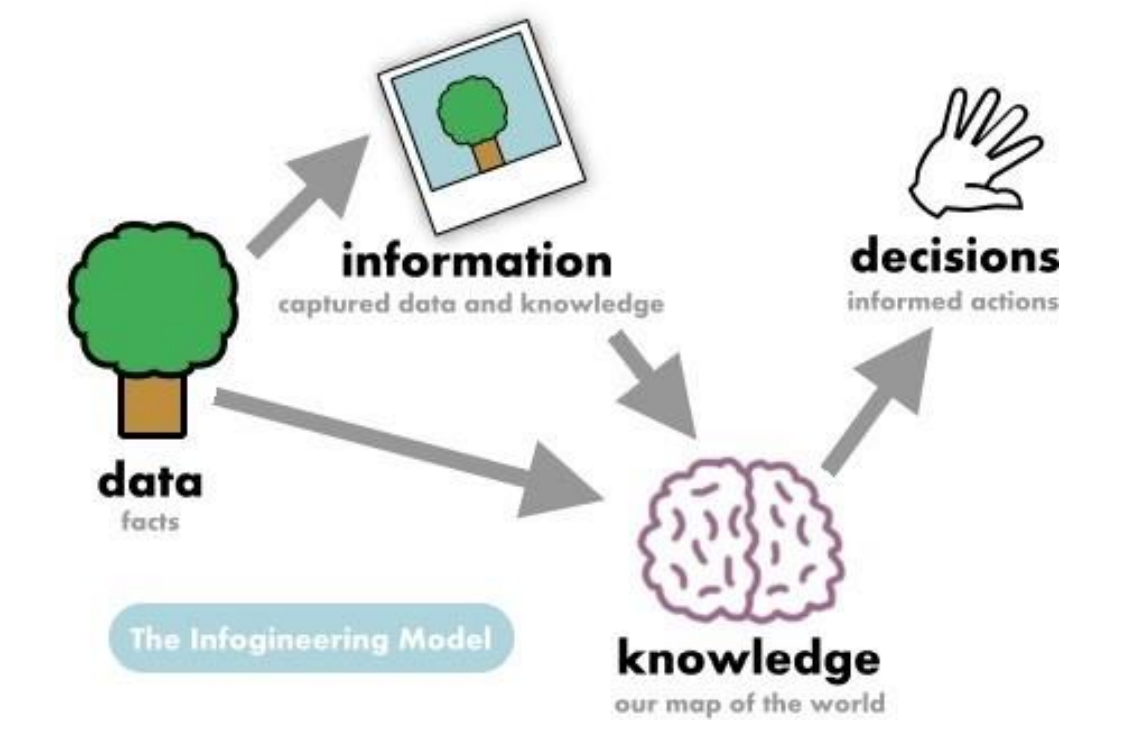
- URN : là 1 dạng của uri nó không chỉ ra chính xác sử dụng giao thức hay

cách để lấy tài nguyên mà sẽ cung cấp định danh tài nguyên trên mạng )

**Cơ sở dữ liệu quan hệ**

Các khái niệm :

* Dữ liệu (data) là các mô tả về sự vật, hiện tượng sự việc được biểu diễn dưới dạng các tài liệu văn bản, hình ảnh, âm thanh,.... => Dữ liệu là rất hữu ích.
* Vậy việc lưu trữ dữ liệu là rất quan trọng và có nhiều cơ chế khác nhau để lưu trữ dữ liệu.
* Khi xử lý dữ liệu chúng ta có thông tin.

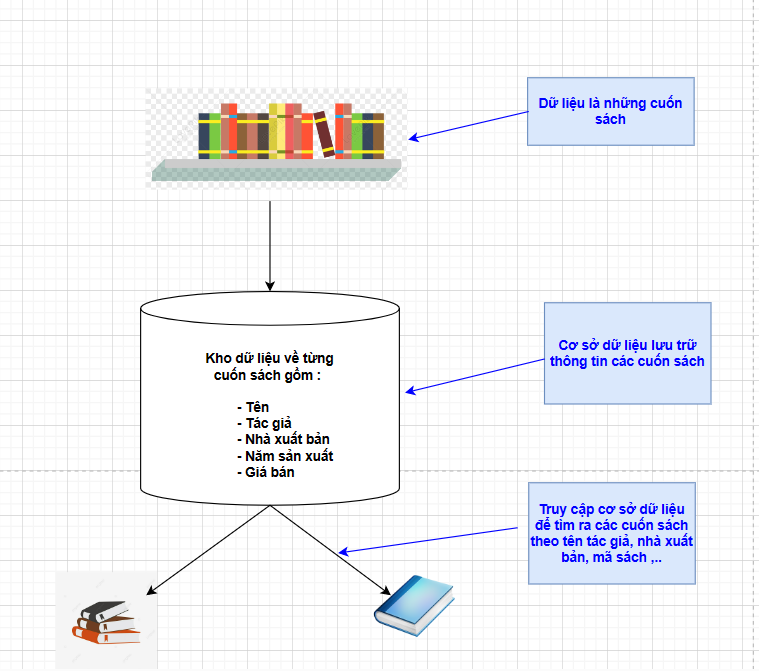


* Ví dụ : quản lý điểm: điểm môn 1, điểm môn 2
* Quản lý dữ sinh viên : sinh viên có tên, địa chỉ, ngày sinh,....

*Cơ sở dữ liệu :*

[*Bài 1: Tìm hiểu về cơ sở dữ liệu - Lý thuyết đơn giản nhưng quan trọng - YouTube*](https://www.youtube.com/watch?v=RHEKsSYCoQ0)

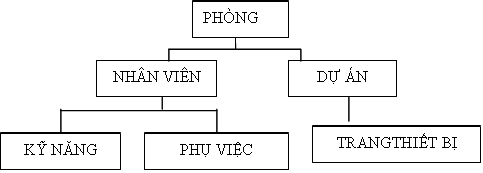
* Cơ sở dữ liệu là 1 tập hợp dữ liệu có cấu trúc và liên quan tới nhau, chúng được tổ chức trong máy tính.
* Cơ sở dữ liệu thì được thiết kế cho phép người dùng lưu trữ, truy xuất hoặc cập nhật dữ liệu.
* Cơ sở dữ liệu được lưu trữ có cấu trúc :
* Các dữ liệu được lưu trữ thành các bản ghi (record).
* Các trường dữ liệu (field).
* Và các dữ liệu có quan hệ (relation) với nhau.



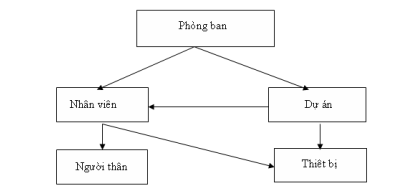
* *Một số loại cơ sở dữ liệu :*

[Lý thuyết chung về cơ sở dữ liệu (viblo.asia)](https://viblo.asia/p/ly-thuyet-chung-ve-co-so-du-lieu-3P0lPqyv5ox)

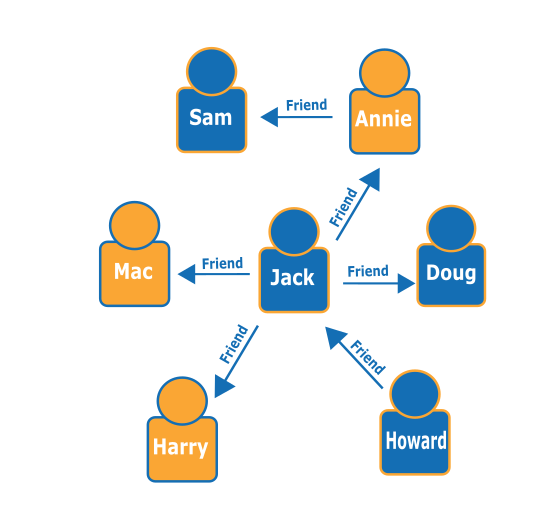
* Phân cấp :



* Mạng :



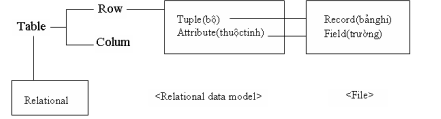
* Đồ thị :

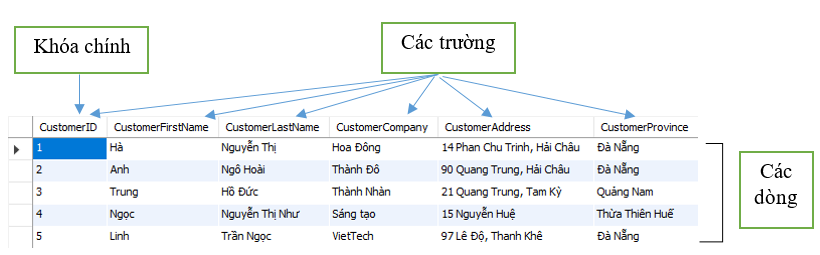


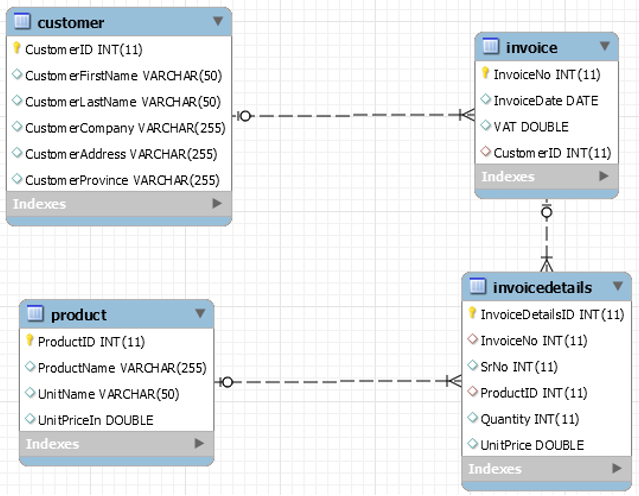
* Đối tượng :



* Quan hệ :







* *Quản lý dữ liệu :*
* Là quản lý một lượng lớn dữ liệu bao gồm và cung cấp thêm sửa xóa và truy vấn dữ liệu.

|  | **File** | **Cơ Sở Dữ Liệu** |
| --- | --- | --- |
| Lưu trữ | Lưu trữ dưới dạng file text | Lưu trữ file chứa dữ liệu theo dạng bảng, cây, đồ thị, đối tượng ,... |
| Truy xuất | Kém hiệu quả trong việc truy xuất ngẫu nhiên hoặc truy xuất đồng thời. | Có các câu lệnh để thực hiện truy vấn. |
| Bảo mật | Kém | Tốt |
| Cách thức lưu trữ | Rời rạc | Có cấu trúc |

*Tìm hiểu về cơ sở dữ liệu quan hệ*

* Là csdl được xây dựng dựa trên mô hình dữ liệu quan hệ (relational model of data)
* Dữ liệu được tổ chức dạng bảng (table)
* Các dòng gọi là các bản ghi (record)
* Các thuộc tính gọi là các trường (field)
* Một bảng thường lưu trữ dữ liệu của 1 thực thể (entity) nhất định ( vd : KhachHang, Order)

*Hệ quản trị cơ sở dữ liệu :*

* Là 1 ứng dụng máy tính dùng để quản lý dữ liệu.
* Lợi ích : giúp người dùng có thể tương tác với dữ liệu (thêm, sửa, xóa, cập nhật và truy vấn)
* Cung cấp giao diện truy cập cho người dùng.

*SQL :*

* Structured Query Language (Ngôn ngữ truy vấn có cấu trúc)
* Được sử dụng để thao tác với cơ sở dữ liệu

*Truy vấn đơn giản* : select, creat, drop, update, alter, drop

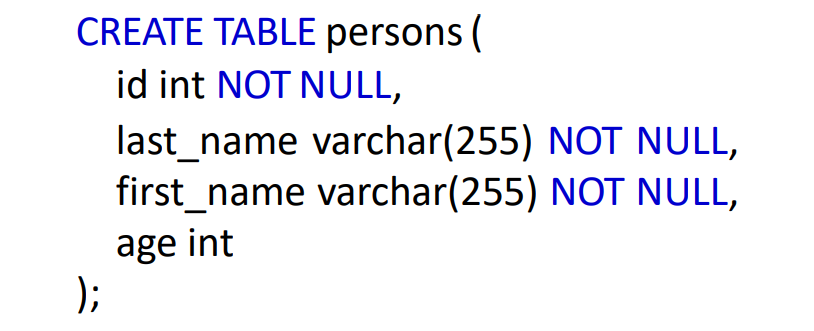
***Thiết kế và tạo cơ sở dữ liệu***

*Ràng buộc (Constrain) :*

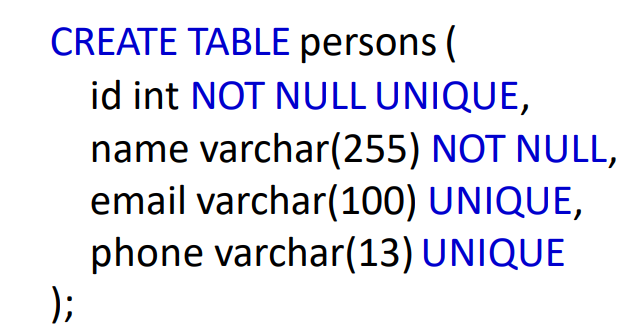
* Là các quy tắc được quy định cho bảng
* Dùng để hạn chế các dữ liệu đưa vào bảng => Giúp cho dữ liệu chính xác, toàn vẹn.
* Đưa dữ liệu vào trong bảng cần tuân thủ các ràng buộc.
* Có thể quy định cho bảng hoặc cột.

*Một số ràng buộc :*

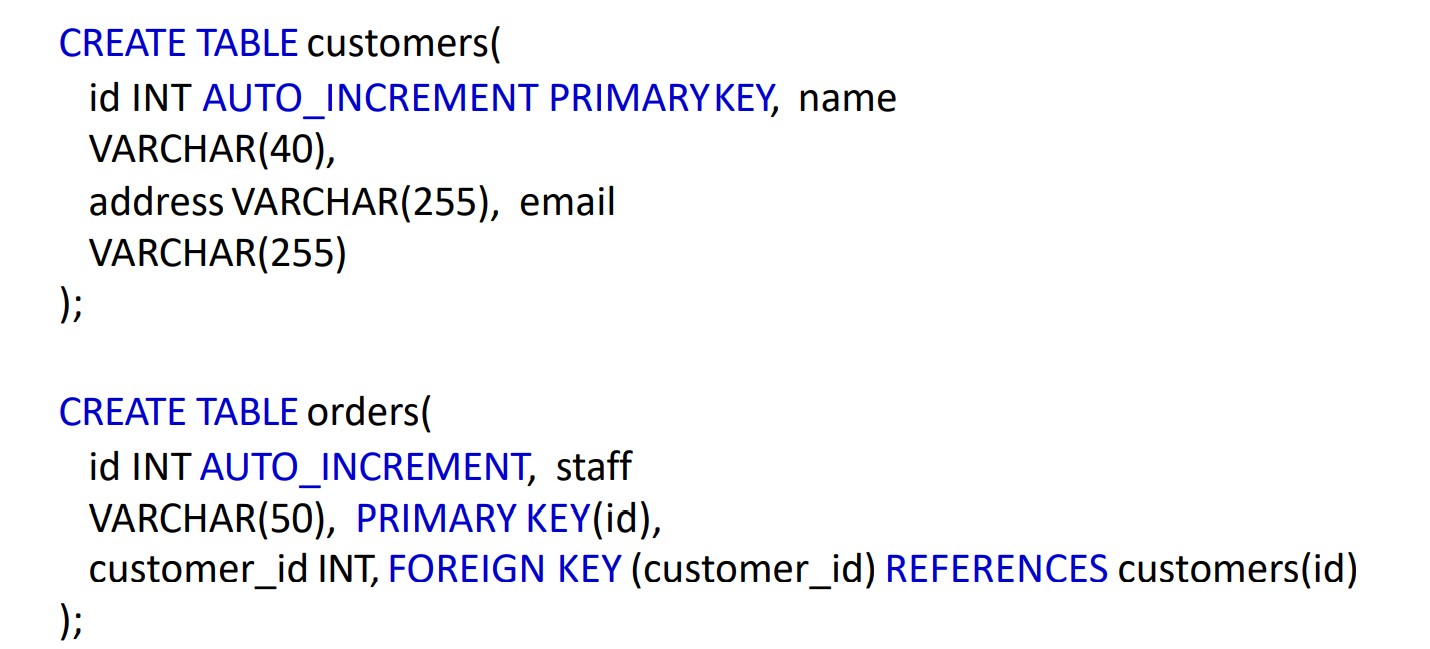
* Not null : không cho phép null.



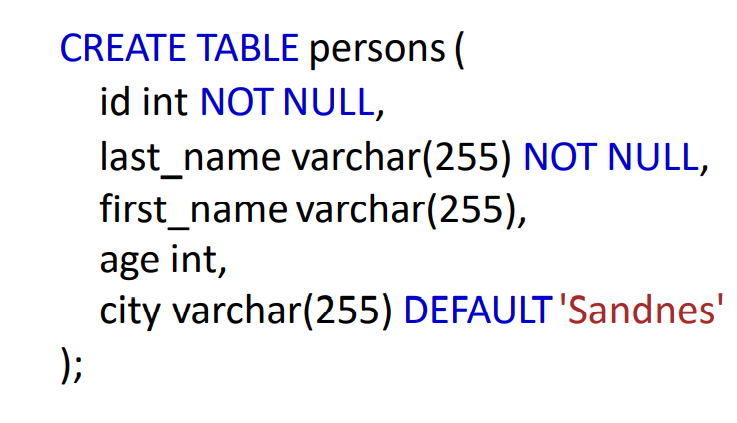
* Unique: biểu thị giá trị là duy nhất



* Foreign key : Khóa ngoại, tham chiếu sang bảng khác.



* Default : Quy định giá trị mặc định cho trường (nếu không có giá trị nào được nhập vào)



* Index: giúp tăng tốc độ truy vấn

***Phân tích và Thiết kế CSDL***

* Là các thao tác để tìm ra csdl trong một tình huống nhất định => Kết quả : Một bản thiết kế cơ sở dữ liệu => dựa vào bản thiết kế sử dụng câu lệnh tạo ra nó .
* Bản thiết kế của cơ sở dữ liệu thường được thiết kế dưới dạng lưu đồ Thực thể quan hệ.

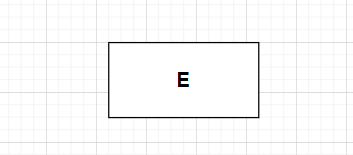
*Mô Hình Thực Thể Kết Hợp (Entity Relationship Diagram)*

* Dùng để thiết kế cơ sở dữ liệu ở mức khái niệm
* Biểu diễn mối quan hệ giữa các đối tượng và mối quan hệ giữa chúng.
* ERD bao gồm :
* Tập các thực thể / Thực thể ( Entity)
* Thuộc tính (Attributes)
* Mối quan hệ (Relationship)

*Thực thể :*

* Diễn tả các đối tượng trong thực thể.
* Có tên gọi riêng.
* Có thuộc tính mô tả đặc trưng thực thể
* Có mã khóa riêng.

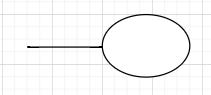
Ví dụ: 1 SinhVien, 1 Lop, … => Tập các thực thể là tập hợp các thực thể có tính chất giống nhau.

Kí hiệu :  , E : là danh từ hoặc cụm danh từ.

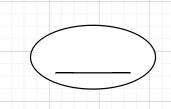
*Thuộc tính :*

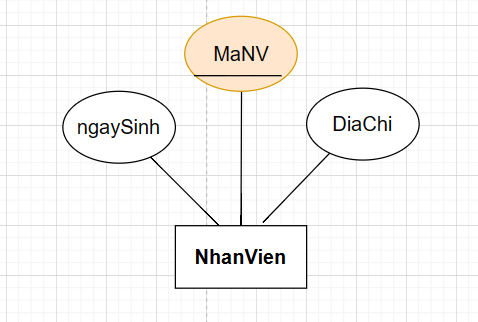
* Là những đặc tính riêng biệt của tập thực thể.
* Diễn tả các thuộc tính thành phần của thực thể hay mối kết hợp.
* Thuộc tính là những giá trị nguyên tố : kiểu chuỗi, số nguyên, số thực.
* Tên thuộc tính danh từ hoặc cụm danh từ.

Ví dụ : Thực thể NHANVIEN có thuộc tính : họ tên, ngày sinh, địa chỉ, quê quán,..

* Kí hiệu : 

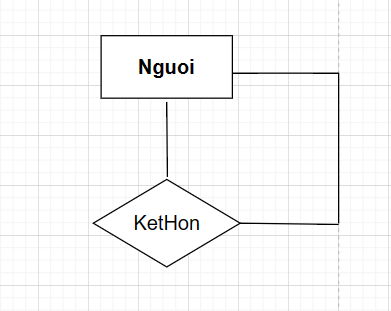
Note : Thuộc tính khóa chính : Các thực thể cần được phân biệt, mỗi thực thể PHẢI có khóa và có thể có 1 hay nhiều thuộc tính.

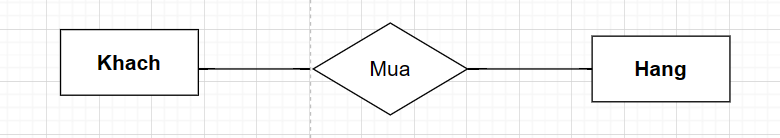
* Kí hiệu : 

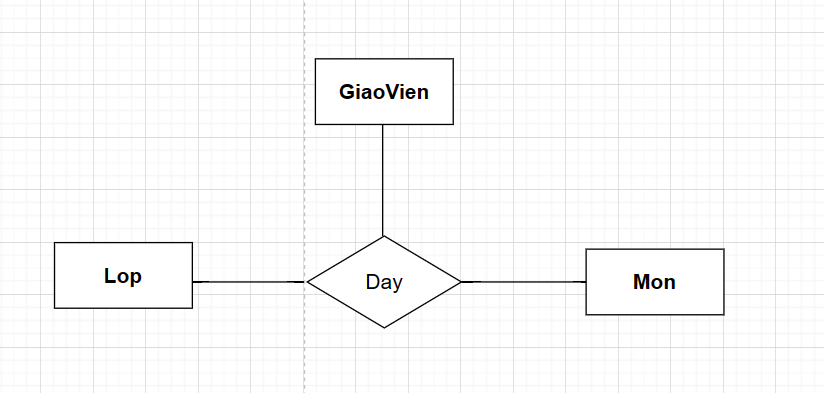
Ví dụ : 

***Mối kết hợp :***

* Diễn tả mối liên kế ít nhất giữa 2 thực thể khác nhau :
* Quan hệ giữa 2 thực thể
* Quan hệ giữa nhiều thực thể.
* *Bậc* : Mối quan hệ tham gia vào thực thể :
* Bậc 1 : Một thực thể tham gia vào mối quan hệ



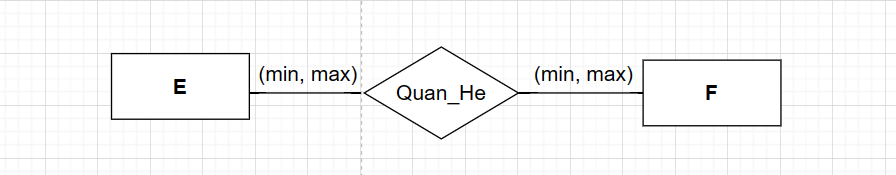
* Bậc 2 : Hai thực thể tham gia vào mối quan hệ 
* Bậc 3 : Ba thực thể tham gia vào mối quan hệ.



* Mối kết hợp có thể có thuộc tính riêng
* Phân loại mối kết hợp :
* Một - Một (1-1)
* Một - Nhiều (1 - n) hay Nhiều - Một (n - 1)
* Nhiều - Nhiều (n - n)
* Ví dụ : 1 Giáo Viên dạy 1 hoặc n lớp học , 1 Giáo Viên dạy 1 hay n môn học.

=> Những câu nói như vậy có thể giúp ta xác định được mối kết hợp giữa 2 thực thể.

*Bản số của mối kết hợp :*



=> (min, max) bên trái : chỉ định một thực thể của E tham gia ít nhất min lần và nhiều nhất max lần vào thực thể của F.

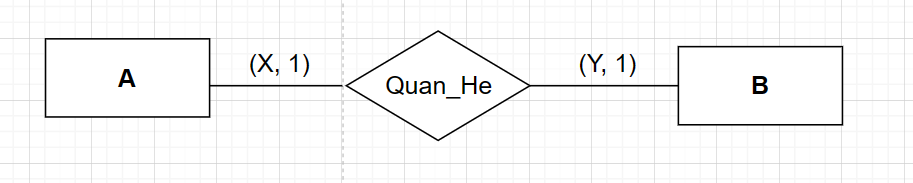
(min,max) ở bên phải : chỉ định một thực thể của F tham gia ít nhất min lần và max lần vào thực thể E.

Cách biểu diễn :



*Cách xác định mối kết hợp giữa các thực thể :*

Khi ta phân tích các bộ số được :

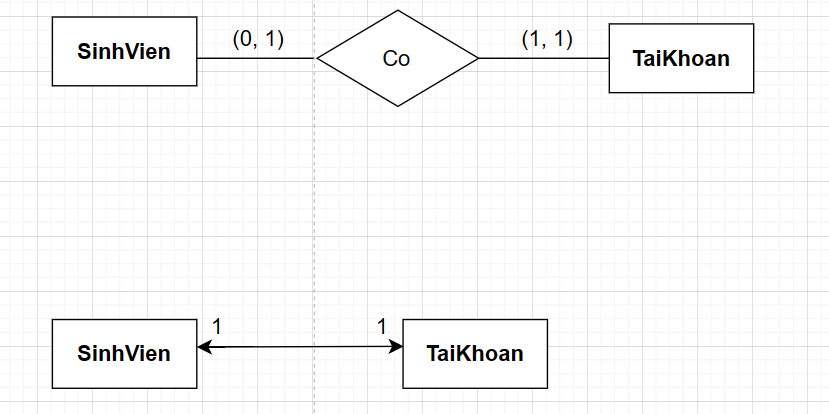


=> Lấy max hai bên ra được : 1 - 1



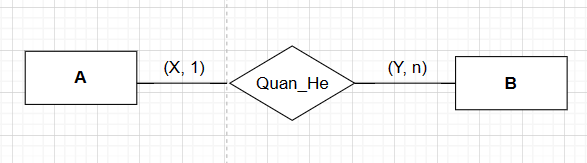
=> Khi thiết kế database thì có thể cho Khóa chính bảng A thành 1 thuộc tính bảng B hoặc ngược lại.

Ví dụ : Một Sinh Viên có 0 hoặc 1 tài khoản, 1 tài khoản phải gắn với 1 sinh viên nào đó.

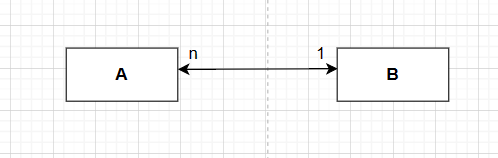


*Mối kết hợp 1 - n hoặc 1 - n:*

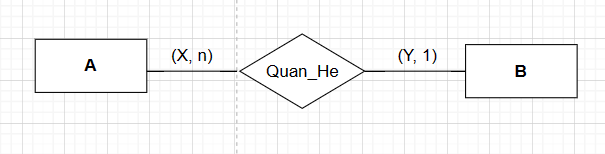
Khi ta phân tích các bộ số ta được :



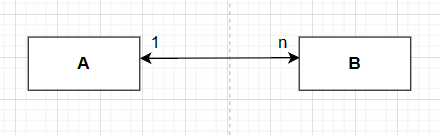
=> Lấy max 2 bên ta đảo chiều => n - 1



hoặc :

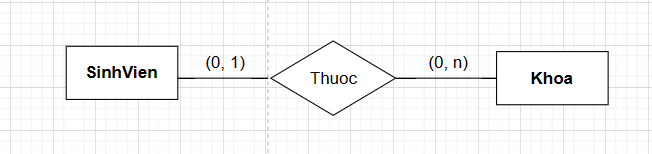


=>

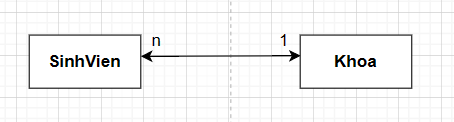


=> Khi thiết kế thì sẽ đặt Khóa Chính của thực thể bên 1 vào làm thuộc tính của bên n.

Ví dụ : Một sinh viên có thể thuộc về 0 hoặc 1 khoa nào đó, và Một khoa có thể có 0 hoặc nhiều sinh viên.



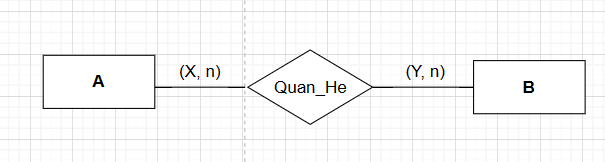
=> Sinh viên có quan hệ n - 1 với khoa



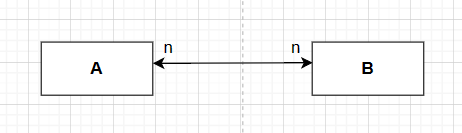
*Bài tập* : KhachHang co HoaDon => phân tích bộ số và tìm ra mối kết hợp

*Mối kết hợp n - n*

- Khi ta phân tích được



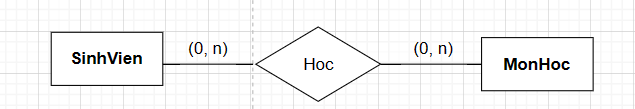
- Lấy max 2 bên ta được : n - n



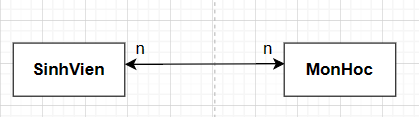
=> Khi thiết kế thì ta gặp quan hệ n - n : thì sẽ lấy Quan\_He tạo thành một bảng ở giữa lấy khóa chính 2 bảng A và B làm thuộc tính của bảng Quan\_He.

Ví dụ : Một sinh viên có thể học 0 hoặc n môn học

Một môn học có thể học bởi 0 hoặc n sinh viên.



=> Lấy max 2 bên :



*Quy trình xây dựng ERD :*

Bước 1 : Tìm hiểu các danh mục và nội dung hồ sơ.

Bước 2 : Chọn lọc thông tin đã lấy ra được từ bước 1.

Bước 3: Xác định các tập thực thể và thuộc tính của nó.

Bước 4: Vẽ sơ đồ thực thể liên kết

Bước 5: Chuẩn hóa, thu gọn.

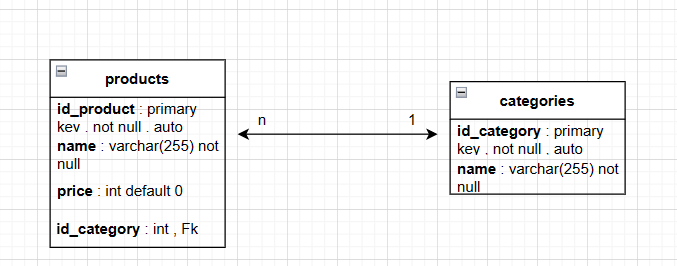
Bước 6 : Xây dựng ERD.

***Bài tập phân tích và thiết kế database ( Làm theo nhóm) :***

Xây dựng CSDL cho Bài toán quản lý bán hàng:

1. Quản lý sản phẩm: tên, giá, mô tả, số lượng, loại. 1 loại thì có nhiều sản phẩm, 1 sản phẩm chỉ có 1 loại.
2. Quản lý thông tin khách hàng: tên, tuổi, sđt, địa chỉ.
3. Quản lý thông tin hóa đơn: thời giao tạo, thông tin sản phẩm, thông tin khách hàng, tổng tiền.
   1. 1 khách hàng có thể có nhiều hóa hơn, 1 hóa đơn chỉ thuộc về 1 khách.
   2. 1 hóa đơn có nhiều sản phẩm, 1 sản phẩm có thể nằm trong nhiều hóa đơn.

=> Phân tích và thiết kế database, dựa vào ERD tạo ra được mô hình như dưới :



Sau đó dựa vào nó dùng các câu lệnh thiết kế database có tên là store

=> Đưa các câu lệnh vào file text + hình ảnh mô hình trên lên git => nộp.

Bài 3 : Các câu lệnh SQL

Câu lệnh Select

Cú pháp : select cot\_1, cot\_2, cot\_3, … from ten\_bang, ten\_bang2 where dieu\_kien.

* Các loại từ khóa trong MYSQL được sử dụng để điều kiện
* Toán tử >, <, >=, <=, <>, !=
* AND, OR
* Nằm trong between…and… (Ví dụ tuổi từ 18-60)
* Is not null, is null
* Like => Giống như là …=> dùng để kiểm tra chuỗi theo định dạng cho trước. Ví dụ : lấy ra học viên bắt đầu bằng chữ Q => where name like “Q%”
* In: kiểm tra dữ liệu của cột nằm trong 1 mảng có sẵn. Ví dụ cần lấy học viên từ Hà Nam, Nam Định => where in (‘Hà Nam’, ‘Nam Định’)
* Not in : ngược lại của in

Group by cot1, cot2,....

* Được sử dụng để gom nhóm dữ liệu `

Having

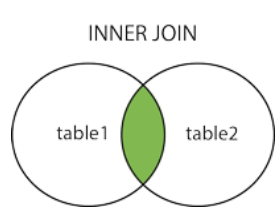
* Dùng để đặt điều kiện cho các hàm có sẵn của mysql

Order by :

* Dùng để sắp xếp asc tăng dần và desc là giảm dần

Limit => giới hạn số lượng bản ghi

**Join trong SQL :**



Lấy phần chung có mặt ở cả 2 bảng.

Student

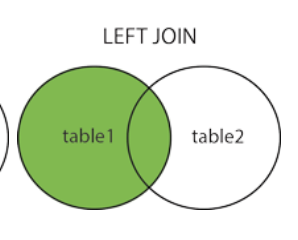
| Id | Name | ClassId |
| --- | --- | --- |
| 1 | Linh | 1 |
| 2 | Hùng | 2 |
| 3 | Đăng | null |

Class

| Id | Name |
| --- | --- |
| 1 | C0221H1 |
| 2 | C1222H1 |

*Cú pháp* : select cot\_1, cot\_2, … from Bang\_A join Bang\_B on dieu\_kien\_join

Ví dụ : select \* from student join class on student.ClassId = class.id



Lấy tất cả bản ghi ở bảng bên trái và phần chung giữa 2 bảng

Student

| Id | Name | ClassId |
| --- | --- | --- |
| 1 | Linh | 1 |
| 2 | Hùng | 2 |
| 3 | Đăng | null |

Class

| Id | Name |
| --- | --- |
| 1 | C0221H1 |
| 2 | C1222H1 |
| 3 | C0922G1 |

*Cú pháp* : select cot\_1, cot\_2, … from Bang\_A left join Bang\_B on dieu\_kien\_join

Ví dụ : select \* from student join class on student.classId = class.id

Advance Git

Git init : sao chép repo từ remote về local

Git add : Xác nhận sự thay đổi của file (đưa đến staging area)

Git commit -m : cập nhật thông tin thay đổi lên local repository

Git remote add : tạo remote trên server

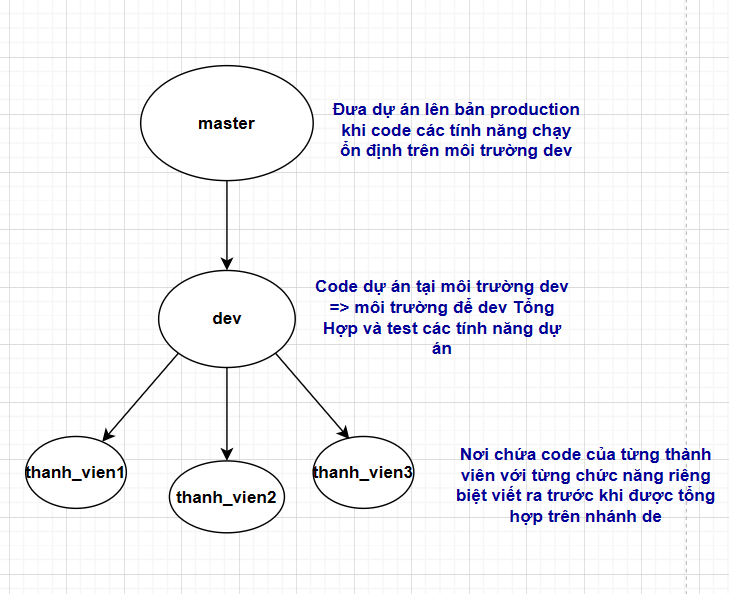
Git push : đưa code từ local lên remote

Git status : kiểm tra các file chưa được add và được add trong staging

Git pull : kéo code từ remote về local

Git reset –h: đưa code trở về commit gần nhất

Nhánh :



Git branch ten\_nhanh : tạo nhánh mới

Git branch -d ten\_nhanh : để xóa nhánh

Git checkout ten\_nhanh chuyển nhánh

* Ví dụ : git checkout dev => đổi sang nhánh dev

Git checkout -b ten\_nhanh: vừa để tạo nhánh vừa để chuyển nhánh (lệnh này chỉ được dùng khi nhánh đó chưa được tạo)

Git push origin ten\_nhanh : đẩy code từ local lên remote

Git pull origin ten\_nhanh : kéo code nhánh A về local

Git merge ten\_nhanh : trộn code của nhánh A tại local vào nhánh hiện tại mình đang đứng

* Vd: Hiện tại ta đang ở nhánh dev và muốn lấy code của thanh\_vien1 tại local => git merge thanhvien1 => lấy code thanh\_vien1 vào cho nhánh dev

.gitignore : file được sử dụng khai báo các file các thư mục không được add vào remote

*Quy trình làm việc nhóm :*

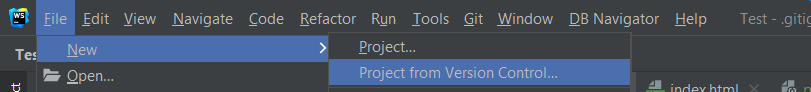
Bước 1 : Trưởng nhóm tạo repo và add các thành viên vào

Bước 2 : Trưởng nhóm tạo project

* Đầy đủ các thư viện cần sử dụng và khung các folder, các file cần thiết (mô hình database, file chứa các câu lệnh tạo database, model, view, controller,...)
* Sử dụng git init để tạo file .gitignore chứa:
* .idea
* node\_module
* package-log.json
* Dùng lệnh git add .
* Dùng lệnh git commit -m “first commit - tên - giờ/ngày”
* Dùng lệnh git push origin master => tạo nhánh master trên remote
* Dùng lệnh git branch dev => Tạo nhánh dev ở local
* Dùng lệnh git push origin dev => Tạo nhánh dev trên remote

Bước 3 : Thành viên clone dự án từ trên remote về máy tính cá nhân

Chọn new => Chọn Project from Version Control => Dán url



* Thành viên tự tạo nhánh dev ở máy mình : git branch dev
* Thành viên sẽ sử dụng lệnh git branch ten\_nhanh (tên thành viên) để tạo nhánh cho bản thân. => Thành viên sẽ code ở nhánh này (**Không được code ở dev, master**)

Bước 4: Thành viên code tính năng + tự test tính năng + test chéo

Bước 5: Sau khi test chéo không có lỗi => Thành viên yêu cầu nhóm trưởng merge code tính năng của mình vào nhánh dev trên remote.

Bước 6: Trưởng nhóm sẽ review lại đoạn code thành viên vừa yêu cầu => nếu bị conflict thì thực hiện fix => chạy thành công thì thực hiện đưa lên remote và yêu cầu các thành viên pull về nhánh dev local trên máy thành viên.

Đến hẹn lại lên

🙌 Nhận lớp NodeJs + React Js

🤔Lớp phù hợp với những bạn trang giấy trắng

Lộ trình 30 buổi :

- Nhập môn lập trình

- TypeScript

- Node Js + Express

- React Js

💻 Thời gian học 2 4 6

Học phí 88k/1b