1.Json  
 a. là một định dạng dữ liệu( chuỗi)

b. javascript object notation

b1. Notation: ký hiệu

c. JSON.stringify ( chuyển đỗi )

🡪 từ javascript type 🡪 JSON

/ JSON.parse ( trả lại ban đầu)

🡪từ JSON 🡪 javascript type

d. ngăn cách các phần tử trong mảng bằng dấu ",", các phần tử bọc bởi "", cuối phần tử không dược có dấu "," như bình thường

e. dạng chuỗi trong JSON 🡪 thêm cập dấu nháy vào "string"

f. typeof Null = object

/////

2.Promise:

A1( lý thuýet phỏng vấn) được sinh ra để giải quyết các thao tác bất đồng bộ

Trước khi ra đời promise chúng ta thường sử dụng callback 🡪 mà sử dụng callback thường xuát hiện callbackhell

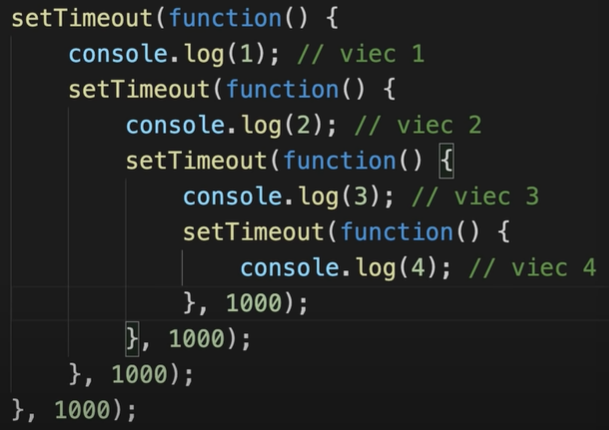
Đẻ sử dụng promise ta se sử dụng new promise,trong thằng constructor của nó truyền vào một executro function, trong executor function nhận vào hai tham số là resovle(gọi nó khi thao tác sử lí của chúng ta thành công) và reject ( gọi nó khi thao tác sử lí thất bại) . khi thằng promise được gọi ra thì chúgn ta sẽ sử dụng các phương thức như là .then() , .catch() , hai thằng này đều nhận một callback function , và nó sẻ đc thực hiên vào thằng then khi promise được resolve, nhân được vào thằng catch() khi mà promise dược reject

a. Sync / Async

setTimeOut, setInterval, fetch , XTMLHttpRequest, file reading , request animation fram

là các hàm bất đồng bộ ( Async)

Callback hell và cách dùng promise đẻ giải quyết



3. Promise constructor :

a. nhận đối số là một function

*var* promise =  new Promise(function(){

})

Function truyền vào được gọi là executor(người thi hành), lệnh trên được thực hiện thì function sẽ được thực hiện

Trong function truyền vào hai tham số là hai funciton resovle( thành công ) và reject (thất bại), trong promise luôn luôn gọi ít nhất 2 tham số trên

b.một promise (object) có ba phương thức :

b1. .Then()🡪 khi resolve được gọi, .catch()🡪 khi reject được gọi, .finally() 🡪 khi một trong hai được gọi, cả ba đều truyền vào một functioin

b2. Promise có ba trạng thái : +Pending( khi chưa có resovle hay reject được gọi,

+fulfilled : biết trạng thái hoạt động thành công ( resolve được gọi)

+rejected: biét trạng thái hoạt động thất bại ( reject được gọi )

🡪 dù resolve hay reject được gọi thì phương thức finally luộn được gọi

Chain 🡪 chuỗi ( tính chát chuỗi 🡪 nối nhau , thằng sau lấy dữ liệu từ thằng trước để lại)

Nếu thằng trước không return về một promise 🡪 thì nhanh chóng chạy thằng liền kề dưới của nó, còn thằng trên return về một thằng promise thì then đăng sau chính là then của promise được trả về ở trước ( tức là chờ cho promise ở đằng trước được giải quyết xong thì mới lọt được thằng đằng sau)

# 🡪Promise methods (resolve, reject, all)( bài 174)

# trong trường hợp trên đường có một thằng return về promise ( có funciton bên trong là reject vè reject về cái lỗi gì đó ) thì các thằng then đăng sau sẽ ngưng chạy , và bắt buộc lúc này phải có thằng catch() để bắt cái lỗi đó

*var* promise = Promise.reject('failure');

promise

    .then(function(course){

        console.log(course);

    })

    .catch(function(course){

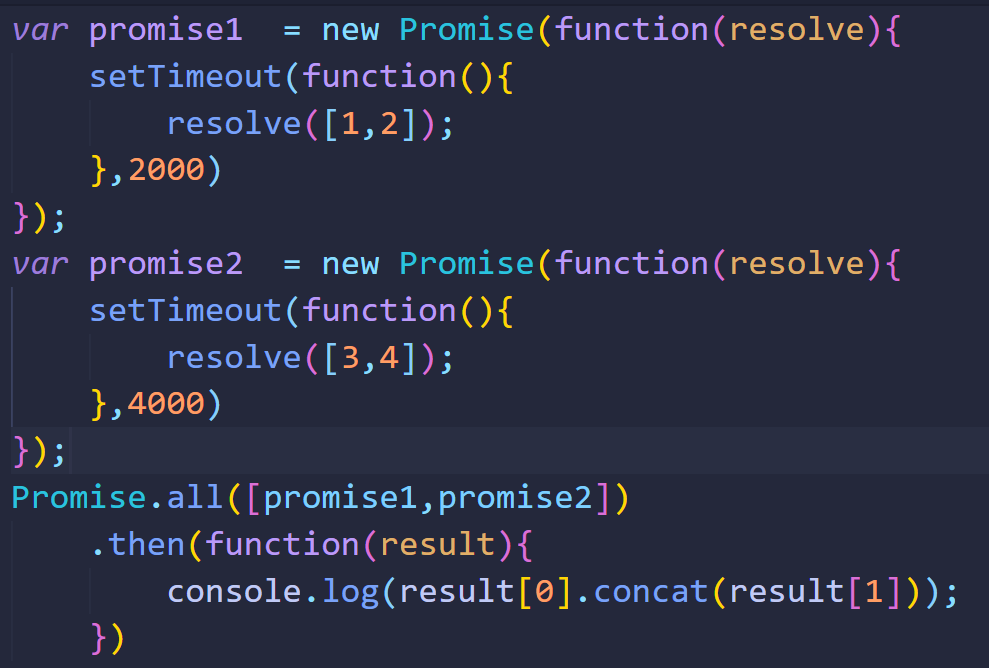
        console.log(course);

    });

*// --> OUTPUT: failure*

Promise.all

a. khi mà các promise dầu vào xong hết thì nó mới lọt vào cái thằng .then



🡪 result nhận kết quả từ hai promise kai là một array , nên nó cũng là một array

[[1,2],[3,4]]

Result[0] = [1,2]

Result[1] = [3,4]