1. Đồng bộ và bất đồng bộ là như thế nào

Đồng bộ: Code được xử lí theo thứ tự từ trên xuống dưới. Giải thích code trong slide 5

Về bất đồng bộ: <https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Asynchronous/Introducing>

Định nghĩa: Bất đồng bộ là kĩ thuật cho phép chương trình bắt đầu một nhiệm vụ tốn nhiều thời gian, nhưng vẫn có khả năng phản ứng lại với các events khác trong khi nvu tốn thời gian đó vẫn chạy. Khi mà một task đó hoàn thành, chương trình sẽ đưa ra kết quả.

Đặc điểm nhận biết: Tốn nhiều thời gian để chạy mà ko nhất thiết phải trả về kết quả ngay

Ví dụ tại sao phải có async (phần ví dụ): Ajax call, truy cập user media và microphone hoặc hỏi user chọn file nào để upload

<https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Asynchronous/Introducing>

Demo về async bằng một bài toán fetch cơ bản. URL: <https://api.dictionaryapi.dev/api/v2/entries/en/hello>

Video tham khảo: <https://www.youtube.com/watch?v=64ASqMjj9_o&t=442>

1. Callback

Callback là một hàm được truyền làm một đối số của hàm khác, dùng để thực hiện một số tác vụ bên trong hàm

Ví dụ về callback như sau

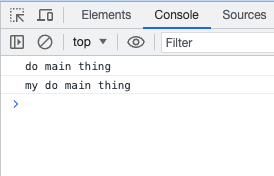
function myDisplayer(some) {  
  document.getElementById("demo").innerHTML = some;  
}  
  
function myCalculator(num1, num2, myCallback) {  
  let sum = num1 + num2;  
  myCallback(sum); 🡺 (2)  
}  
  
myCalculator(5, 5, myDisplayer); 🡺 (1)

Ở đây, khi (1) chạy, myDisplayer là tham chiếu, tương ứng sẽ map với myCallback, sau đó myCallback sẽ tiến hành thực thi (2), chỉ khi nào có dấu () thì hàm mới được tiến hành chạy.

Nhưng: nếu như hàm myCallback() lại định nghĩa thêm các hàm callback() nữa thì các hàm callback sẽ gọi nhau liên tục 🡺 callback hell.

Ví dụ: <https://blog.avenuecode.com/callback-hell-promises-and-async/await>

Ví dụ về callback: Làm một bài toán cơ bản về callback: in một chuỗi ra màn hình, sử dụng callback để tiếp tục in một chuỗi khác ra màn hình, kết quả như sau



Xử lí thực hành 1 trên LMS: <https://learn.codegym.vn/courses/web-app-building-with-js/lessons/ajax-va-xu-ly-bat-dong-bo/topic/thuc-hanh-trien-khai-xu-ly-bat-dong-bo-bang-callback>

Exercises 1: Viết một function atLeastTwo([input], callback). Kết quả của hàm này trả về **true** khi tối thiểu 2 elements trong mảng thỏa mãn điều kiện callback(), nếu không return **false**

**VD:**

* atLeastTwo([1, 2, 3, 4, 5], isOdd); 🡺 true (vì có số 1,3,5 thỏa mãn điều kiện số lẻ và số lượng số lẻ >2)
* atLeastTwo([2, 4, 6], isOdd); 🡺 false (vì không có số lẻ)
* atLeastTwo([1, 2, 3, 4, 5], (t) => t > 3); 🡺 true (vì có số 4,5 thỏa mãn điều kiện > 3 và số lượng >=2)

Bắt đầu bằng script sau

function atLeastTwo(array, callback) {

// Code ở đây

}

Thời gian làm bài: 15’

1. Fetch API để demo promise

Sử dụng fetch api để xem trạng thái (fulfilled, rejected, pending) của promise, demo tại demo/fetch.html

Url: <https://mdn.github.io/learning-area/javascript/apis/fetching-data/can-store/products.json>

Xử lí bắt lỗi thế nào (lỗi có nhiều nguyên nhân, ví dụ như: mất mạng, timeout, server trả về lỗi 404, 501,…): sử dụng catch, demo bằng cách đánh sai url

1. Promise

Demo “Demo/promise.html”

Thực hành promise ở Thực hành 2

Demo với bài toán cơ bản: Chia hết cho 2, nếu như ko chia hết cho 2 thì reject, còn nếu chia hết cho 2 thì resolve.

1. Async await

Demo API : https://mdn.github.io/learning-area/javascript/apis/fetching-data/can-store/products.json

Sử dụng fetch API kết hợp với Promise cùng với url như ở trên

Triển khai lại cách viết ajax bằng Async await “Demo/asyncawait.html”

Sử dụng axios để fetch data, sử dụng đúng url đã được đề cập ở trên bằng cả promise.then() và async await

Thực hành async await ở Thực hành 3 (nếu có thời gian)