**Lab 10. Hàm callback**

**10.1 Callback là gì?**

* **Callback** là một **hàm** được **truyền vào** hàm khác như một **tham số**, rồi được **gọi lại** (callback) khi hàm “cha” hoàn thành công việc.
* Mục đích: **đảm bảo** một đoạn mã chỉ chạy **sau** khi tác vụ bất đồng bộ (API, timer, I/O…) xong.

**10.2 Khi nào dùng callback?**

1. **Xử lý sự kiện (Event)**
2. button.addEventListener("click", () => {
3. alert("Bạn vừa bấm nút!");
4. });
5. **Đợi thời gian (Timer)**
6. setTimeout(() => {
7. console.log("Xong 5 giây!");
8. }, 5000);
9. **Duyệt mảng**
10. [1,2,3].forEach(n => console.log(n\* n));
11. **Gọi API**
12. fetch(url)
13. .then(res => res.json())
14. .then(data => console.log(data));
15. **Tự định nghĩa logic**
16. function process(arr, fn) {
17. return arr.map(fn);
18. }
19. console.log(process([1,2,3], x=>x\*2)); // [2,4,6]

**10.3 Ưu – Nhược của callback**

* **Ưu**
  + Dễ hiểu với thao tác đơn giản, không cần chờ đợi.
* **Nhược**
  + **Callback hell**: lồng quá nhiều → code loằng ngoằng.
  + Khó **quản lý lỗi** khi lồng sâu.

👉 **Thay thế**: dùng **Promise** hoặc **async/await** để code rõ ràng, dễ bảo trì hơn.

**10.4 Ví dụ thực hành với Express (Node.js)**

const express = require('express');

const app = express();

const port = 9000;

// Khi có GET "/" → callback trả text về

app.get("/", (req, res) => {

res.send("Chào ctk!");

});

// Khởi động server → callback in ra thông báo

app.listen(port, () => {

console.log(`Server chạy trên cổng ${port}`);

});

* app.listen chạy ngay, in thông báo trước.
* app.get chờ người dùng truy cập mới chạy.