TÀI LIỆU KHÓA HỌC

Tự Học JavaScript từ số 0

Tác giả: Hỏi Dân IT & Eric

Version: 1.0

Chapter 1+ 2: Giới thiệu và thiết lập môi trường thực hành

Cài đặt & chuẩn bị môi trường thực hiện dự án

#1. JavaScript là gì

1.Javascript là gì ?

JavaScript (JS) là một ngôn ngữ lập trình đa năng (multi-paradigm) được thiết kế ban đầu để tạo các trang web tương tác trên trình duyệt.

Ngày nay, JavaScript đã phát triển thành một ngôn ngữ toàn diện có thể chạy ở mọi nơi: từ trình duyệt đến máy chủ, ứng dụng desktop, mobile, và thậm chí IoT.

Một số đặc điểm chính:

- Ngôn ngữ thông dịch (interpreted) – Không cần biên dịch, có thể chạy ngay.
- Dạng script động – Có thể thay đổi nội dung HTML/CSS theo thời gian thực.
- Hỗ trợ lập trình hướng đối tượng (OOP), hàm (functional), mô-đun (modular).
- Là ngôn ngữ chính của Frontend Web và được hỗ trợ bởi mọi trình duyệt.

2. Ứng dụng thực tế của JavaScript

1. Phát triển Website (Frontend)

Dùng để điều khiển giao diện và trải nghiệm người dùng với thư viện/framework nổi tiếng như: React, Vue.js, Angular, Svelte

Ví dụ: Tạo Single Page Applications (SPA) như Facebook, Gmail...

2. Phát triển phía Server (Backend)

Dùng với Node.js để xây dựng server/API

Framework phổ biến: Express.js, NestJS, Fastify

3. Phát triển Mobile App

Sử dụng JavaScript để tạo ứng dụng di động đa nền tảng với: React Native, Ionic, Expo

4. Phát triển Game

Tạo game 2D/3D đơn giản với HTML5 + JS, dùng thư viện như Phaser, Three.js cho đồ họa

5. Ứng dụng Desktop

Viết app cho Windows/Mac/Linux bằng Electron.js

Ví dụ: Visual Studio Code, Slack, Discord đều viết bằng JavaScript

6. Ứng dụng AI, IoT, Automation

Xử lý mô hình học máy với TensorFlow.js

Giao tiếp thiết bị IoT qua JS runtime trên vi điều khiển (Espruino)

Viết script tự động thao tác trình duyệt (Puppeteer, Playwright)

#2. Cài đặt môi trường code JavaScript

Editor/IDE (Integrated Development Environment) cần đáp ứng các tiêu chí:

- Giá cả hợp lý (ưu tiên mã nguồn mở - miễn phí)
- Dễ sử dụng
- Hỗ trợ cú pháp coding (gợi ý code, phát hiện lỗi)
- Hỗ trợ debug
- Cộng động sử dụng rộng rãi (community support)
- Hỗ trợ đa nền tảng: Windows, MacOS, Linux

1.Các công cụ phổ biến để code JavaScript:

Dùng miễn phí:

- VSCode: rất phổ biến nhất
- Sublime Text, Atom
- Notepad 👌

Dùng trả phí: WebStorm (JetBrains IDEs)

Trong khóa học này, mình sử dụng Visual Studio Code (VSCode), vì miễn phí, hỗ trợ tốt cho việc code JavaScript/TypeScript

2. Cài đặt Visual Studio Code (VSCode)

Link download: https://code.visualstudio.com/download

3. Cấu hình VSCode

Format Code

Setup Format on Save

Mục đích: Mỗi lần nhấn Ctrl + S , code sẽ được auto format trông cho đẹp/dễ nhìn

Các extensions cài đặt thêm:

- **Code Spell Checker** : hỗ trợ check chính tả khi đặt tên tiếng anh
- **Auto Complete Tag** : hỗ trợ code nhanh HTML

#3. Hello World với JavaScript

Bước 1: Tạo folder chứa source code
Lưu ý về tên dự án thực hành và đường dẫn (path) nơi lưu trữ code, không dùng ký tự đặc biệt và tiếng việt có dấu

Nên dùng: D://my-project/js-ts-hoidanit
Không nên dùng: D//dự án/học javascript

Đặt tên dự án thực hành là: js-ts-hoidanit

Bước 2: Tạo file html, js
//todo: viết hello world

Sử dụng script tag trong file html

Bước 3: Chạy với Google Chrome
//todo

#4. Sử dụng Git để quản lý mã nguồn

Mục đích: công cụ để quản lý mã nguồn và sử dụng source code của dự án (tránh tình trạng máy tính hư bị mất hết code)

Nếu bạn chưa biết gì về Git, xem nhanh tại đây (miễn phí)
Khóa học Git trả phí, tham khảo tại đây

- Sử dụng Git theo nguyên tắc:
1. Học xong video nào, commit đẩy lên Github/Gitlab
=> tạo cơ hội để thực hành câu lệnh của Git, ví dụ:
git add
git commit
git push...

2. Git là công cụ "mặc định bạn phải biết" khi đi làm phần mềm
=> điều 1 ở trên giúp bạn thực hành

3. Thói quen học xong video nào, đẩy code lên Git, giúp bạn tạo ra bản "backup" cho project của bạn

Ví dụ máy tính bạn bị hỏng đột xuất/bị mất
=> vẫn còn code, chỉ cần pull về code tiếp mà không phải code từ đầu.

=> Mục đích sử dụng git ở đây là : backup code + thực hành công cụ đi làm mà bạn "phải biết" nếu muốn đi thực tập/đi làm.

Chapter 3: Biến, Kiểu dữ liệu và Toán tử

Tìm hiểu và nắm vững cách khai báo cú pháp cơ bản và việc sử dụng biến số của JavaScript

#5. Google Chrome Devtool (Extra)

Google Chrome DevTools (Developer Tools) là bộ công cụ được tích hợp sẵn trong trình duyệt Chrome, cho phép lập trình viên kiểm tra, sửa lỗi, phân tích hiệu năng, và tối ưu hóa website hoặc ứng dụng web ngay trong trình duyệt.

Lưu ý: bạn nên sử dụng Google Chrome, thay vì FireFox, Opera, Cốc Cốc… để đảm rằng thao tác của bạn và video là giống nhau.

Cách mở Chrome DevTools:

Nhấn **F12** hoặc Ctrl + Shift + I (Windows/Linux) / Cmd + Option + I (Mac)

Hoặc: Click chuột phải vào một phần tử trên trang → chọn “Inspect” (Kiểm tra)

Các tab chính của DevTools:

1. Elements : hiển thị DOM Tree và CSS đang có trên website

2. Console: cung cấp giao diện thực thi lệnh JavaScript trực tiếp, hiển thị các lỗi JavaScript.

3. Sources: hiển thị cấu trúc file JS/CSS…., hỗ trợ đặt breakpoint để debug.

4. Network: theo dõi tất cả các request được gửi từ trình duyệt

5. Application: quản lý dữ liệu lưu trữ:

Local Storage / Session Storage/ Cookies

IndexedDB/ Cache / Service Worker

6. Performance

Đo hiệu năng trang web: Thời gian load, render, paint.

Phân tích các thao tác nặng gây giật, delay.

Sử dụng để tối ưu frontend.

#6. Biến trong JavaScript (var, let, const)

//todo: tạo thư mục src để lưu source code theo từng bài học

Để khai báo biến (variable), cú pháp chung là <Từ khóa> <tên biến> = <giá trị>;

Ví dụ:

```
var name = "Alice";
let age = 25;
const country = "Vietnam";
```

Lịch sử ra đời:

https://www.w3schools.com/js/js_versions.asp

Phân biệt nhanh var/let

Yêu cầu bắt buộc:

- **Ưu tiên dùng let và const** thay vì var.

- Dùng **const** khi giá trị không cần thay đổi.

#7. Kiểu dữ liệu nguyên thủy (Primitive Data Types)

Mục tiêu:

- Hiểu được các kiểu dữ liệu nguyên thủy trong JavaScript
- Biết cách sử dụng typeof để kiểm tra kiểu dữ liệu

Định nghĩa: Kiểu dữ liệu nguyên thủy (primitive) là những kiểu dữ liệu cơ bản, không có phương thức hay thuộc tính, và được lưu trực tiếp giá trị trong vùng nhớ của biến

7 kiểu dữ liệu nguyên thủy trong JavaScript

Kiểu dữ liệu	Ví dụ giá trị	Ghi chú
Number	10; 3.14	Gồm cả số nguyên và số thực
String	hoidanit	Chuỗi ký tự đặt trong nháy đơn hoặc nháy kép
Boolean	true, false	Chỉ có 2 giá trị
Undefined	undefined	Khai báo biến số, và không gán giá trị cho biến số
Null	null	Khi khai báo biến, giá trị bằng null. Ám chỉ rằng, giá trị của biến số “nothing”/”empty”
BigInt	123456789n	Dùng cho số rất lớn (thêm n ở cuối), tối đa 2^53 - 1
Symbol	Symbol("id")	Dùng để tạo giá trị duy nhất

Lưu ý:

JavaScript không phân biệt số nguyên hay số thực, chỉ gọi chung là **number**

Có thể kiểm tra **datatype** bằng cách sử dụng: typeof

```
console.log(typeof 10);          // "number"
console.log(typeof "hello");     // "string"
```

#8. Kiểu dữ liệu tham chiếu (Object Data Types)

Định nghĩa: Dữ liệu không được lưu trực tiếp trong biến, mà biến chỉ chứa một địa chỉ tham chiếu trong bộ nhớ.

Tên kiểu dữ liệu	Ví dụ	Ghi chú
Object	{ name: "hoidanit" }	Dạng key-value, linh hoạt
Array	[1, 2, 3]	Danh sách các phần tử
Function	function() {}	
Date	new Date()	Xử lý thời gian, ngày tháng
RegExp	/abc/	Biểu thức chính quy
Map	new Map()	Dạng key-value
Set	new Set()	Lưu giá trị duy nhất, không trùng lặp

Tập trung vào 2 loại data chính: Object và Array

#9. Toán tử trong JavaScript

1. Toán tử số học (Arithmetic Operators)

Toán tử	Ý nghĩa	Ví dụ	Kết quả
+	Cộng	5 + 2	7
-	Trừ	5 - 2	3
*	Nhân	5 * 2	10
/	Chia	10 / 2	5
%	Chia lấy dư	5 % 2	1
**	Lũy thừa	2 ** 3	8
++	Tăng 1	let a=10; a++	11
- -	Giảm 1	let a=10; a--	9

//so sánh a++ và ++a. Lấy ví dụ

2. Toán tử so sánh (Comparison Operators)

Toán tử	Ý nghĩa	Ví dụ	Kết quả
>	Lớn hơn	5 > 3	true
<	Nhỏ hơn	2 < 1	false
>=	Lớn hơn hoặc bằng	5 >= 5	true
<=	Nhỏ hơn hoặc bằng	4 <= 3	false
==	So sánh bằng (lỏng)	5 == '5'	true
===	So sánh bằng (chặt)	5 === '5'	false
!=	Khác nhau (lỏng)	5 != '5'	false
!==	Khác nhau (chặt)	5 !== '5'	true

Luôn dùng === và !== nếu muốn so sánh datatype (kiểu) và giá trị (value)

Ngoài ra, còn có thể sử dụng:

Toán tử logic

```
let isAdult = true;
let hasID = false;

console.log(isAdult && hasID); // false : toán tử và (AND)
console.log(isAdult || hasID); // true : toán tử hoặc (OR)
console.log(!hasID);           // true: toán tử khác/phủ định (NOT)
```

#10. Câu lệnh console trong JavaScript

//todo: hướng dẫn cách comment code

Console là câu lệnh đã được khai báo sẵn trong môi trường javascript (tương tự bạn có hàm printf khi học Java/C)

1.Các câu lệnh thường gặp

console.log( ) – In giá trị ra màn hình console

console.error( ) – Hiển thị lỗi

console.warn( ) – Hiển thị cảnh báo

//phân biệt error và warning

2.Các tip hay dùng

Sử dụng dấu phẩy thay vì cộng chuỗi trong console.log()

```
let name = "hoidanit";
let age = 20;

// Không nên: cộng chuỗi dễ sai
console.log("Name: " + name + ", Age: " + age);

// Nên: dùng dấu phẩy (tự động thêm dấu cách)
console.log("Name:", name, "Age:", age);
```

Tô màu cho console.log() bằng %c

```
console.log("%cCảnh báo!", "color: red; font-weight: bold;");
```

#11. Template String (Extra)

Tài liệu:

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Template_literals

1. Template String là gì?

Template String (còn gọi là Template Literals) là cách khai báo chuỗi bằng **dấu backtick** (`) thay vì nháy đơn ' ' hay nháy kép " ".

Chúng hỗ trợ:

- Chèn biến trực tiếp bằng ${ }
- Xuống dòng không cần \n
- Chèn biểu thức JS vào trong chuỗi

Cách code cũ:

```
let name = "hoidanit";
let age = 20;
console.log("Tên: " + name + ", Tuổi: " + age);
```

Sử dụng template string:

console.log(`Tên: ${name}, Tuổi: ${age}`);

#12. Bài Tập Lab 01

Yêu cầu:

Tạo các biến sau với const hoặc let (chọn phù hợp):

fullName: tên đầy đủ (string)

birthYear: năm sinh (number)

isStudent: true/false

Tính tuổi hiện tại dựa trên birthYear.

Gợi ý: tính năm hiện tại, sử dụng:

```
const today = new Date();
const currentYear = today.getFullYear();
```

In ra console theo format:

Tên: [fullName]

Tuổi: [calculatedAge]

Sinh viên: [Đúng/Sai]

#13. Chữa Bài Tập Lab 01

//todo

Chapter 4: Cấu Trúc Điều Khiển & Hàm (Function)

Sử dụng câu điều kiện if-else, switch-case, vòng lặp for/while và function với JavaScript

#14. Câu lệnh điều kiện – if / else / else if

//todo: chia source code theo tên chapter

1.Khái niệm

Câu lệnh điều kiện cho phép chương trình ra quyết định. Tùy thuộc vào điều kiện đúng hoặc sai, ta có thể thực hiện những đoạn mã khác nhau.

Cú pháp:

```
if (điều_kiện) {
  // khối lệnh nếu điều_kiện đúng (true)
}
```

Hoặc:

```
if (điều_kiện) {
  // nếu đúng
} else {
  // nếu sai
}
```

#15. Câu lệnh switch / case

switch là một cấu trúc điều kiện giúp kiểm tra giá trị của một biểu thức và thực thi các khối lệnh tương ứng với từng **case (trường hợp cụ thể)**.

Sử dụng **switch** giúp mã nguồn **gọn gàng hơn** khi cần kiểm tra nhiều giá trị khác nhau của cùng một biến.

Cú pháp:

```
switch (biểu_thức) {
  case giá_trị_1:
    // khối lệnh nếu biểu_thức === giá_trị_1
    break;

  case giá_trị_2:
    // khối lệnh nếu biểu_thức === giá_trị_2
    break;

  ...

  default:
    // khối lệnh nếu không khớp bất kỳ case nào
}
```

#16. Vòng lặp (for, while, do...while)

Vòng lặp giúp **tự động hóa việc lặp lại các khối lệnh**, thay vì phải viết lặp thủ công nhiều lần. Chỉ cần thiết lập điều kiện và hành động lặp lại là xong.

1.Vòng lặp for

Cú pháp:

```
for (khởi_tạo; điều_kiện; cập_nhật) {
  // khối lệnh lặp
}
```

- khởi_tạo: Gán giá trị ban đầu cho biến đếm.
- điều_kiện: Kiểm tra trước mỗi vòng lặp. Nếu đúng thì chạy tiếp.
- cập_nhật: Tăng/giảm biến đếm.

2.Vòng lặp while

Cú pháp:

```
while (điều_kiện) {
  // khối lệnh lặp
}
```

- Kiểm tra điều kiện trước mỗi vòng lặp.
- Nếu điều kiện đúng thì tiếp tục chạy, sai thì dừng.

3.Vòng lặp do...while

Cú pháp:

```
do {
  // khối lệnh lặp
} while (điều_kiện);
```

Chạy ít nhất 1 lần, sau đó mới kiểm tra điều kiện.

//todo:
So sánh vòng lặp while và do…while ?

#17. Câu lệnh điều khiển luồng – break, continue

Trong quá trình lặp, đôi khi bạn không muốn thực hiện toàn bộ số vòng lặp, mà chỉ muốn:

- Thoát ra hoàn toàn khỏi vòng lặp → dùng **break**
- Bỏ qua vòng hiện tại và tiếp tục vòng sau → dùng **continue**

1.Câu lệnh break

Dùng để **thoát khỏi vòng lặp ngay lập tức,** kể cả khi điều kiện chưa sai.

Ví dụ:

```
for (let i = 1; i <= 10; i++) {
  if (i === 5) break;
  console.log(i);
}
```

2.Câu lệnh continue

Dùng để **bỏ qua vòng lặp hiện tại và chuyển sang vòng tiếp theo**.

Khối lệnh sau continue sẽ không được thực thi trong vòng hiện tại.

Ví dụ:

```
for (let i = 1; i <= 5; i++) {
  if (i === 3) continue;
  console.log(i);
}
```

#18. Hàm (Function) trong JavaScript

1.Hàm là gì?

Hàm là một **khối mã có tên**, thực hiện một công việc cụ thể.

Hàm giúp **tái sử dụng mã code,** chia chương trình thành các phần nhỏ dễ quản lý hơn.

Cú pháp:

```
function tenHam() {
  // Khối lệnh
}
```

Ví dụ:

```
function greet() {
  console.log("Xin chào!");
}

greet(); // Output: Xin chào!
//Gọi hàm bằng cách dùng tên hàm kèm dấu ngoặc ()
```

2.Tham số và đối số (Parameters & Arguments)

```
function greet(name) {
  console.log("Xin chào, " + name + "!");
}

greet("hoidanit"); // Output: Xin chào, hoidanit!
```

- **Tham số**: biến khai báo trong hàm (name).
- **Đối số**: giá trị truyền vào khi gọi hàm ("hoidanit").

#19. Arrow function (Extra)

Tài liệu:

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/Arrow_functions

//todo

Về Immediately Invoked Function Expression / anonymous function:

https://developer.mozilla.org/en-US/docs/Glossary/IIFE

#20. Keyword Return

1. return là gì?

return là từ khóa dùng để **kết thúc hàm và trả về một giá trị** ra ngoài.

Khi JavaScript gặp **return**, hàm sẽ **dừng thực thi ngay lập tức.**

Cú pháp:

```
function tenHam() {
  return giaTriTraVe;
}
// nếu không return giá trị thì sao ?
```

#21. Phạm vi biến (Scope)

1. Scope là gì?

Scope là phạm vi truy cập của biến – tức là nơi biến đó có thể được sử dụng.

Trong JavaScript, có 3 loại phạm vi chính:

- Phạm vi toàn cục (**Global** Scope)
- Phạm vi hàm (**Function** Scope)
- Phạm vi khối (**Block** Scope)

2.Phạm vi toàn cục – Global Scope

Biến được khai báo ngoài mọi hàm hoặc khối lệnh. Có thể **truy cập từ bất kỳ đâu** trong chương trình.

Ví dụ:

```
let globalVar = "Tôi là biến toàn cục";
function show() {
  console.log(globalVar); // Truy cập được
}
show();
console.log(globalVar); // Truy cập được
```

3.Phạm vi hàm – Function Scope

Biến khai báo b**ên trong một hàm** chỉ dùng được **bên trong hàm đó**.

Ví dụ:

```
function sayHi() {
  let name = "hoidanit";
  console.log("Hi " + name);
}

sayHi();
console.log(name); // ❌ Lỗi: name is not defined
```

4. Phạm vi khối – Block Scope (ES6 trở lên)

Biến khai báo bằng **let** và **const** trong cặp dấu {} chỉ dùng được bên trong khối đó.

Ví dụ:

```
if (true) {
  let x = 10;
  const y = 20;
  console.log(x, y); // ✅ Truy cập được
}

console.log(x); // ❌ Lỗi: x is not defined
```

#22. Bài Tập Lab 02

Yêu cầu:

1.Tạo hàm **tinhTrungBinh**(toan, van, anh)

- Hàm nhận vào 3 điểm số (sử dụng tham số).
- Trả về điểm trung bình (sử dụng return).

2.Tạo hàm **xepLoai**(diemTB)

Dựa vào điểm trung bình, xếp loại học sinh:

Từ 9 → "Xuất sắc"

Từ 8 và nhỏ hơn 9 → "Giỏi"

Từ 6.5 và nhỏ hơn 8→ "Khá"

Còn lại → "Trung bình"

Dùng if / else if / else để thực hiện

Output:

Cho trước 3 biến:

const diemToan = 9;

const diemVan = 8;

const diemAnh = 7;

Gọi các hàm trên để tính điểm trung bình và **in ra kết quả sau**:

Điểm trung bình: 8.0

Xếp loại: Giỏi

#23. Chữa Bài Tập Lab 02

//todo

Chapter 5: Mảng (Array) và Đối Tượng (Object)
Làm việc với các dạng dữ liệu phức tạp, bao gồm xử lý array và object

#24. Giới thiệu về Mảng (Array)

1. Mảng là gì?

Mảng (array) là một kiểu dữ liệu đặc biệt trong JavaScript, cho phép bạn **lưu nhiều giá trị trong một biến duy nhất.**

Ví dụ: thay vì sử dụng:
let color1 = "red";
let color2 = "green";
let color3 = "blue";

//sử dụng array: let colors = ["red", "green", "blue"];

2. Cách khai báo mảng

Dùng dấu ngoặc vuông [ ]

Ví dụ:
let fruits = ["apple", "banana", "orange"];

JavaScript **không giới hạn kiểu dữ liệu** bên trong mảng. Bạn có thể lưu chuỗi, số, boolean, object, thậm chí cả mảng khác.

Ví dụ:
let mixed = [42, "hello", true, null, [1, 2, 3]];

#25. Truy cập, chỉnh sửa, thêm và xóa phần tử của mảng

1.Truy cập phần tử trong mảng
Để truy cập một phần tử, bạn sử dụng cú pháp: **array[index]**
Lưu ý: Chỉ số (index) bắt đầu từ 0

Ví dụ:
```
let colors = ["red", "green", "blue"];

console.log(colors[0]); // "red" (phần tử đầu tiên)
console.log(colors[1]); // "green"
console.log(colors[2]); // "blue"
```

Nếu truy cập phần tử không tồn tại → kết quả là **undefined**

2.Chỉnh sửa phần tử trong mảng
Ví dụ:
```
let fruits = ["apple", "banana", "orange"];
fruits[1] = "grape"; // Thay "banana" bằng "grape"
console.log(fruits); // ["apple", "grape", "orange"]
```

3. Thêm, xóa

Thêm phần tử:
vào cuối mảng với **push()**
vào đầu mảng với **unshift()**

Xóa phần tử:
cuối mảng với **pop()**
đầu mảng với **shift()**

#26. Duyệt mảng với vòng lặp và forEach

1.Duyệt mảng bằng vòng lặp for

Ví dụ:

```
let scores = [80, 90, 70];

for (let i = 0; i < scores.length; i++) {
  console.log("Phần tử thứ " + i + ": " + scores[i]);
}
```

2.Duyệt bằng forEach( )

```
let names = ["Alice", "Bob", "Charlie"];

names.forEach(function(name, index) {
  console.log(index + ": " + name);
});

//todo: sử dụng arrow function
```

#27. Biến đổi mảng với map( )

Tài liệu:
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/map

1.map( ) là gì?
Phương thức **map( )** được dùng để duyệt qua từng phần tử trong mảng, **tạo ra một mảng mới** mà mỗi phần tử là kết quả biến đổi của phần tử tương ứng trong mảng ban đầu.

map() không thay đổi mảng gốc.

Cú pháp:
```
let newArray = array.map(function(element, index, array) {
  // return giá trị mới
});
```

element: phần tử hiện tại

index (tuỳ chọn): chỉ số phần tử

array (tuỳ chọn): chính mảng đang được duyệt

return: chính là phần tử mới của mảng kết quả

//todo: sử dụng arrow function

#28. Lọc phần tử mảng với filter( )

Tài liệu:
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/filter

1. filter( ) là gì?

filter( ) là một phương thức của mảng, cho phép bạn **lọc ra các phần tử thỏa mãn điều kiện**, và trả về một mảng mới chỉ chứa các phần tử đó.

Mảng gốc không bị thay đổi

Cú pháp:

```
let newArray = array.filter(function(element, index, array) {
  return điều_kiện_đúng; // true giữ lại, false loại bỏ
});
```

element: phần tử hiện tại

index (tuỳ chọn): chỉ số phần tử

array (tuỳ chọn): mảng gốc đang được duyệt

//todo: sử dụng arrow function

#29. Giới thiệu về Đối tượng (Object)

1.Đối tượng là gì?

Trong JavaScript, **đối tượng (object)** là kiểu dữ liệu dùng để l**ưu trữ thông tin có cấu trúc** dưới dạng cặp tên **thuộc tính** (key) và **giá trị** (value).

Nếu mảng lưu danh sách giá trị, thì đối tượng lưu thông tin mô tả rõ từng giá trị.

Mảng (Array)	**Đối tượng (Object)**
["hoidanit", 25]	{ name: "hoidanit", age: 25 }
Dùng chỉ số (index)	Dùng tên thuộc tính (key)

2. Cách khai báo đối tượng

```
let obj_name = {
        key: value
};
```

#30. Truy cập và cập nhật thuộc tính của Object

1.Truy cập thuộc tính trong Object

Cách 1: Dùng dấu chấm (.)

```
let user = {
  name: "hoidanit",
  age: 25
};

console.log(user.name); // "hoidanit"
console.log(user.age);  // 25
```

Cách 2: Dùng dấu ngoặc vuông ([ ])

```
console.log(user["name"]); // "hoidanit"
```

2.Cập nhật và thêm thuộc tính

```
user.age = 30;            // Cập nhật
user["city"] = "Hanoi";    // Thêm mới


delete user.age;    //xóa thuộc tính
```

#31. Lặp Object sử dụng for...in và for...of

for...in duyệt qua thuộc tính
for...of duyệt qua giá trị

Cú pháp:

```
for (let key in object) {
  console.log(key, object[key]);
}
```

```
for (let value of iterable) {
  console.log(value);
}
```

#32. Bài Tập Lab 03

Yêu cầu:

Hãy tạo một **mảng product**s chứa danh sách 5 sản phẩm, mỗi sản phẩm là một **object** có các thuộc tính:

- **name** (tên sản phẩm)
- **price** (giá sản phẩm)
- **inStock** (true/false – còn hàng hay không)

Ví dụ:

```
{
  name: "T-shirt",
  price: 200,
  inStock: true
}
```

Hãy thực hiện:

1. In ra **tên của sản phẩm đầu tiên**.
2. **Thay đổi giá sản phẩm thứ hai** thành 150 và in ra danh sách tất cả sản phẩm
3. **Thêm một sản phẩm mới** vào cuối mảng và in ra danh sách tất cả sản phẩm
4. **Xoá sản phẩm cuối cùng** ra khỏi danh sách và in ra danh sách tất cả sản phẩm
5. **Dùng forEach( )** để in ra tất cả **tên sản phẩm**.
6. **Dùng map( )** để tạo mảng mới chỉ chứa **giá sản phẩm**.
7. **Dùng filter( )** để lấy các **sản phẩm còn hàng** (inStock = true).
8. **Dùng for...in** để duyệt qua thuộc tính của sản phẩm đầu tiên.

#33. Chữa Bài Tập Lab 03

//todo

Chapter 6: DOM và Sự kiện (Event)

Hiểu và thao tác với DOM trong JavaScript. Biết cách truy cập, thay đổi nội dung HTML và xử lý sự kiện tương tác bằng JavaScript

#34. DOM là gì? Giới thiệu cơ bản về DOM

1. DOM là gì?

DOM (viết tắt của Document Object Model) là một mô hình dạng cây đại diện cho cấu trúc của một trang web.

- Khi trình duyệt tải trang HTML, nó phân tích (parse) nội dung HTML và tạo ra một mô hình dạng cây DOM.
- Mỗi thẻ HTML trong tài liệu sẽ được chuyển thành một "đối tượng" (object) mà JavaScript có thể truy cập và thao tác.

DOM trông như thế nào?

Giả sử bạn có đoạn HTML sau:

```
<!DOCTYPE html>
<html>
 <head>
  <title>Trang web của hoidanit</title>
 </head>
 <body>
  <h1>Chào bạn!</h1>
  <p>Đây là một đoạn văn bản.</p>
 </body>
</html>
```

DOM sẽ biểu diễn như một cây phân cấp:

```
Document
└── html
    ├── head
    │   └── title
    └── body
        ├── h1
        └── p
```

2.DOM cho phép làm gì?

Với DOM, bạn có thể dùng JavaScript để:

✅ Truy cập và đọc nội dung của các phần tử HTML

✅ Thay đổi nội dung hoặc thay đổi kiểu hiển thị (CSS)

✅ Thêm, xóa phần tử HTML

✅ Lắng nghe sự kiện người dùng: click chuột, nhập dữ liệu, rê chuột, v.v.

#35. Truy cập phần tử HTML trong DOM

Khi bạn muốn thay đổi nội dung, thêm màu sắc, ẩn/hiện một phần tử, hoặc xử lý khi người dùng tương tác (như click, nhập dữ liệu…), bạn cần lấy được phần tử HTML đó thông qua DOM.

Một số cách phổ biến để truy cập phần tử

✅ **document.getElementById(id)**

Dùng để lấy phần tử có id cụ thể.

Trả về duy nhất một phần tử (hoặc null nếu không tìm thấy).

html:

```
<p id="intro">Chào mừng!</p>
```

js:

```
const para = document.getElementById("intro");
console.log(para); // in ra phần tử <p>
```

✅ **document.querySelector(selector)**

Dùng để **tìm phần tử đầu tiên khớp** với CSS selector (id, class, tag, v.v.).

```
const firstNote = document.querySelector(".note");
console.log(firstNote); // chỉ in ra phần tử đầu tiên có class="note"
```

✅ **document.querySelectorAll(selector)**

Dùng để **lấy tất cả phần tử khớp với selector,** trả về NodeList (giống mảng).

```
const allNotes = document.querySelectorAll(".note");
console.log(allNotes.length);
```

#36. Giới thiệu xử lý sự kiện (Events)

Tài liệu:
https://developer.mozilla.org/en-US/docs/Web/Events

1.Sự kiện là gì?

Sự kiện (event) là hành động do người dùng hoặc trình duyệt tạo ra khi tương tác với trang web, ví dụ click, cuộn chuột…

=> Khi sự kiện xảy ra, JavaScript có thể thực hiện một đoạn mã để phản hồi lại sự kiện đó.

Một số event thường gặp:

click Người dùng nhấn chuột
input Khi người dùng gõ vào ô nhập liệu
change Khi giá trị input hoặc select thay đổi
keydown Khi nhấn phím bất kỳ trên bàn phím
submit Khi gửi form

Gắn sự kiện trực tiếp trong HTML (cách làm basic)

```
<button onclick="sayHello()">Chào</button>

<script>
  function sayHello() {
    alert("Xin chào!");
  }
</script>
```

Ưu điểm: Dễ viết, dễ hiểu cho người mới.
Nhược điểm: Không tách riêng JavaScript và HTML, khó bảo trì khi trang web lớn.

#37. Lắng nghe sự kiện với addEventListener
Tài liệu:
https://developer.mozilla.org/en-US/docs/Web/API/EventTarget/addEventListener

1.addEventListener( ) là gì?
addEventListener( ) là phương thức tiêu chuẩn và hiện đại để gắn sự kiện vào phần tử HTML.

Khi sự kiện xảy ra (người dùng click, nhập, rê chuột...), trình duyệt sẽ gọi hàm xử lý bạn đã chỉ định.

Cú pháp: element.addEventListener("eventName", function);

- **element**: phần tử HTML bạn muốn gắn sự kiện.
- "**eventName**": tên sự kiện, ví dụ "click", "input", "mouseover".
- **function**: hàm xử lý khi sự kiện xảy ra

Ví dụ:
```
<button id="myBtn">Bấm vào đây</button>

<script>
  const button = document.getElementById("myBtn");

  button.addEventListener("click", function () {
    alert("Bạn vừa nhấn nút!");
  });
</script>
```

#38. Thay đổi nội dung của phần tử HTML

innerText cho phép bạn lấy hoặc thay đổi phần nội dung dạng chữ mà người dùng thực sự nhìn thấy trên trang web (có tính đến CSS)

Ví dụ:

```
<p id="intro">Chào bạn!</p>

<script>
  const p = document.getElementById("intro");
  p.innerText = "Chào mừng đến với khóa học @hoidanit!";
</script>
```

innerHTML cho phép bạn lấy hoặc thay đổi nội dung HTML bên trong phần tử, bao gồm cả thẻ lồng bên trong.

```
<p id="greeting">Xin chào!</p>

<script>
  const p = document.getElementById("greeting");
  p.innerHTML = "<strong>Chào bạn,</strong> <em>học lập trình với hoidanit!</em>";
</script>
```

#39. Thay đổi CSS bằng JavaScript

1.Sử dụng style

Mỗi phần tử HTML trong DOM đều có thuộc tính .style để bạn thay đổi trực tiếp các thuộc tính CSS.

Cú pháp:

element.style.tênThuộcTínhCSS = "giá trị mới";

Lưu ý: Tên thuộc tính CSS dạng kebab-case như background-color sẽ được chuyển sang camelCase trong JavaScript: backgroundColor.

Ví dụ:

```
<p id="title">Xin chào!</p>
<button onclick="changeColor()">Đổi màu chữ</button>

<script>
  function changeColor() {
    const p = document.getElementById("title");
    p.style.color = "red";
  }
</script>
```

2. Sử dụng classList

Thêm hoặc xoá class bằng .classList

Thay vì thay đổi từng thuộc tính .style, bạn có thể thêm class CSS định nghĩa sẵn.

```
element.classList.add("ten-class");
element.classList.remove("ten-class");
```

#40. Alert (Extra)

1.Alert là gì?

alert( ) là **hàm có sẵn** trong JavaScript dùng để **hiển thị một hộp thoại thông báo** đơn giản (popup) cho người dùng.

Khi xuất hiện, hộp thoại sẽ **chặn toàn bộ trang web** cho đến khi người dùng nhấn nút "OK".

Ví dụ:

```
alert("Nội dung bạn muốn hiển thị");
```

#41. Local Storage (Extra)

1.Local Storage là gì?

Local **Storage** là một phần của Web Storage API, cho phép bạn lưu trữ dữ liệu ngay trong trình duyệt của người dùng.

Đặc điểm:

- **Dữ liệu không bị mất** khi reload hoặc tắt trình duyệt.
- Lưu trữ dạng **key-value** (khóa-giá trị), kiểu chuỗi (string).

Cú pháp:

Phương thức	**Mô tả**
localStorage.**setItem**(key, value)	Lưu dữ liệu (value phải là chuỗi)
localStorage.**getItem**(key)	Lấy dữ liệu theo key
localStorage.**removeItem**(key)	Xóa dữ liệu theo key
localStorage.**clear**()	Xóa toàn bộ dữ liệu đã lưu trong Local Storage

//todo: convert object => json (**stringify**/**parse**)
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON/stringify

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON/parse

#42. Bài Tập Lab 04
Yêu cầu: Tạo form đăng nhập đơn giản

Tạo một **form html** gồm:
- **Input nhập username**
- **Input nhập password**
- **Button "Đăng nhập"**

Khi người dùng nhấn nút “Đăng nhập”:

- Lấy giá trị từ hai ô nhập liệu

- So sánh với username/password đã định nghĩa sẵn trong JavaScript (hardcode)
  username: hoidanit@gmail.com
  password: 123456

Nếu đúng:

- Hiện thông báo "Đăng nhập thành công!" bằng alert()

- **Redirect** tới trang html (success.html)

Nếu sai:

- Hiện alert("Tài khoản hoặc mật khẩu sai")

- Đổi màu viền ô input thành đỏ (dùng .style.borderColor)

#43. Chữa Bài Tập Lab 04
//todo

Chapter 7: Xử Lý Bất Đồng Bộ (Asynchronous)
Sử dụng Callback, Promise và async await để xử lý bất đồng bộ trong javascript

#44. Cài đặt môi trường Nodejs
Mục đích: môi trường để chạy code Javascript ở phía backend (server)
Tài liệu: https://nodejs.org/en

1.Nodejs là gì ?
Nodejs không phải là thư viện (library), không phải framework của JavaScript.

Nodejs là môi trường để bạn thực thi code javascript, tại browser và server.

Bạn học Node.js, về bản chất, là học các thư viện/framework (viết bằng JavaScript), nên bạn cần cài đặt môi trường Nodejs để có thể thực thi code JavaScript

Điều này tương tự với:

Bạn học cách sử dụng Microsoft Excel (javascript)
Bạn cần cài hệ điều hành Windows để có thể học nó (nodejs)

2. Cài đặt Nodejs
Sai lầm của beginners, là không quan tới tới version của phần mềm. Nên nhớ, công nghệ nó thay đổi theo thời gian, vì vậy, để hạn chế tối đa lỗi tối đa, bạn nên dùng version phần mềm như khóa học hướng dẫn.

Điều này tương tự với:
Bạn đang chơi 1 con game rất ngon trên Windows 7, bạn vác lên Windows 10 để chạy, có điều gì để đảm bảo rằng "sẽ không có lỗi xảy ra" ?

Trong khóa học này, mình sử dụng **version Node.js là 22.13.0**
Vì vậy, để hạn chế tối đa lỗi có thể xảy ra, bạn vui lòng cài đặt chính xác version nodejs ở trên

Khi code giống nhau, môi trường thực thi code giống nhau (version nodejs), thì rất hiếm khi lỗi xảy ra.
Nếu đây là lần đầu tiên bạn học (coding) một dự án với Node.js, mình khuyến khích sử dụng duy nhất 01 version Node.js (dễ quản lý)

Chỉ sử dụng nhiều version Node.js, khi và chỉ khi, trên bạn có nhiều dự án Node.js, và mỗi dự án yêu cầu một version Node.js khác nhau. (hướng dẫn tại mục 3 bên dưới)

Link tải nodejs v22.13.0:
https://nodejs.org/download/release/v22.13.0/

Sau khi cài đặt xong, kiểm tra bằng cách gõ câu lệnh:

node -v

3. Trường hợp dùng nhiều version Nodejs

Lưu ý: bạn cần gỡ nodejs trước khi cài nvm

//áp dụng cho windows
https://github.com/coreybutler/nvm-windows

//áp dụng cho macos
Video hướng dẫn cài nvm cho mac, xem tại đây

https://dev.to/ajeetraina/how-to-install-and-configure-nvm-on-mac-os-5fgi

#45. Setup Dự Án Backend

Bước 1: Download/clone dự án thực hành tại đây

Bước 2: Cài đặt thư viện cần thiết
npm i

Bước 3: Cấu hình port (nếu muốn)
//json-server.json

Bước 4: Chạy dự án backend
npm run dev
http://localhost:8000/

#46. Xử Lý Bất Đồng Bộ (Asynchronous) là gì ?

1.Đồng Bộ (Synchronous) là gì?

Trong mô hình đồng bộ, các đoạn mã sẽ **thực hiện tuần tự**.

Mỗi dòng lệnh **phải hoàn thành xong** trước khi dòng tiếp theo được chạy.

Ví dụ:

```
console.log("1");
console.log("2");
console.log("3");
```

2.Bất Đồng Bộ (Asynchronous) là gì?

Trong mô hình bất đồng bộ, một số tác vụ có thể **chạy nền (non-blocking)**, cho phép chương trình **tiếp tục thực hiện các đoạn mã khác** mà không phải chờ đợi hoàn tất.

Đặc biệt quan trọng trong các thao tác như:

- Chờ API phản hồi
- Đọc/ghi file
- Lấy dữ liệu từ server
- Chờ sự kiện người dùng

Ví dụ:

```
console.log("1");
setTimeout(() => {
  console.log("2");
}, 1000);
console.log("3");
```

#47. Promise: Lời hứa từ tương lai

Tài liệu:
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise#examples

1.Promise là gì?

Trong JavaScript, Promise là một đối tượng đại diện cho một giá trị sẽ có trong tương lai, thường là kết quả của một tác vụ bất đồng bộ như:

- Gọi API
- Đọc file
- Truy vấn cơ sở dữ liệu

Nói cách khác: Promise giống như một lời hứa:
"Tôi sẽ cung cấp kết quả sau – hoặc thành công, hoặc thất bại!"

3 trạng thái của Promise

Trạng thái	**Mô tả**
pending	Chưa có kết quả – đang xử lý
fulfilled	Thành công – trả về kết quả (**resolve**)
rejected	Thất bại – trả về lỗi (**reject**)

//todo: Sử dụng Promise: .then() và .catch()

#48. Gọi API với Fetch

1. API là gì?

API (Application Programming Interface) là giao diện giúp các ứng dụng giao tiếp với nhau. Trong web, bạn thường gọi API để lấy dữ liệu từ server – ví dụ: danh sách sản phẩm, thông tin người dùng, bài viết,...

Ví dụ: Gọi API từ https://jsonplaceholder.typicode.com/users để lấy danh sách người dùng giả lập.

Hiểu đơn giản: API là 1 url (được backend tạo ra), frontend sẽ gọi tới URL ấy để lấy dữ liệu (data)

2. Fetch là gì?

fetch() là một hàm tích hợp sẵn trong JavaScript, dùng để gửi yêu cầu HTTP đến một URL (API).

- Được giới thiệu từ ES6
- Trả về **Promise**
- Hỗ trợ đầy đủ các phương thức HTTP: GET, POST, PUT, DELETE,...

Cú pháp:

```
fetch(url)
  .then(response => response.json())
  .then(data => {
    console.log(data);
  })
  .catch(error => {
    console.error("Có lỗi xảy ra:", error);
  });
```

url: Địa chỉ API
response.json( ) : Chuyển dữ liệu từ JSON sang Object
.then( ): Xử lý kết quả
.catch( ): Bắt lỗi nếu có vấn đề

#49. Xử lý lỗi với try/catch/finally

1. Lỗi (Error) là gì?

Lỗi là những vấn đề xảy ra trong quá trình thực thi chương trình, khiến chương trình bị dừng hoặc chạy sai.

Ví dụ:

- Truy cập biến không tồn tại
- Gọi API thất bại
- Chia cho 0 (trong một số ngôn ngữ)
- Cú pháp sai

2.Sử dụng try/catch

```
try {
  // code có thể gây lỗi

} catch (error) {
  // xử lý lỗi ở đây

} finally {
  // luôn chạy sau cùng (nếu cần)
}
```

#50. Callback và vấn đề Callback Hell

1.Callback là gì?

Trong JavaScript, một **callback** là **một hàm được truyền vào như một đối số** cho một hàm khác, và sẽ được **gọi lại** (callback) sau khi hàm kia thực hiện xong công việc của nó.

Callback thường được dùng trong các tác vụ bất đồng bộ (asynchronous) như:

- Đọc file
- Gọi API
- Chờ đợi sự kiện

Ví dụ:

```
function greet(name, callback) {
  console.log("Hi " + name);
  callback();
}

function sayGoodbye() {
  console.log("Goodbye!");
}

greet("hoidanit", sayGoodbye);
```

2.Callback Hell là gì?

Callback Hell (địa ngục callback) là thuật ngữ mô tả **việc lồng quá nhiều callback bên trong nhau**, khiến code trở nên khó đọc, khó bảo trì..

- Mã lồng nhau nhiều tầng (“pyramid of doom” – hình kim tự tháp)
- Khó kiểm soát luồng dữ liệu
- Khó bắt lỗi
- Gây mệt mỏi cho lập trình viên khi bảo trì

Ví dụ về callback hell:

```
loginUser("hoidanit", function (user) {
  getUserProfile(user.id, function (profile) {
    getUserSettings(profile.id, function (settings) {
      updateUI(settings, function () {
        console.log("Hoàn thành!");
      });
    });
  });
});
```

#51. Async/Await: Cú pháp cho code sạch đẹp

1.Async/Await là gì?

async/await là cú pháp hiện đại nhất trong JavaScript để xử lý bất đồng bộ, được giới thiệu từ ES2017 (ES8).

Nó là một cách viết gọn gàng hơn cho Promise, giúp code đọc giống như code đồng bộ, dễ hiểu và dễ bảo trì hơn.

2. So sánh với Promise

Promise:

```
getUser()
  .then(user => getProfile(user))
  .then(profile => getSettings(profile))
  .then(settings => console.log(settings))
  .catch(err => console.error(err));
```

async/await

```
async function showSettings() {
  try {
    const user = await getUser();
    const profile = await getProfile(user);
    const settings = await getSettings(profile);
    console.log(settings);
  } catch (err) {
    console.error(err);
  }
}
```

#52. Bài Tập Lab 05

Yêu cầu: xây dựng table users với data từ backend

Bước 1: Bạn cần chạy backend
npm run dev

Kiểm tra backend đã hoạt động chưa:
http://localhost:8000/

Kiểm tra API đã hoạt động chưa:
http://localhost:8000/users

Bước 2: Thực hiện gọi API với fetch (lấy danh sách users)
//sử dụng async await

Bước 3: Sử dụng javascript render table

Render động dữ liệu của table html dựa vào javascript và data của api

Tham khảo template table html tại đây

Gợi ý cách sử dụng javascript để chèn động nội dung cho table:

```
const tbody = document.querySelector("#myTable tbody");
tbody.innerHTML += `<tr><td>data 1</td><td> data 2</td></tr>`;
```

#53. Chữa Bài Tập Lab 05
//todo

Chapter 8: Dự Án Thực Hành JavaScript
Luyện tập các kiến thức đã học về JavaScript thông qua các dự án thực hành

#54. Bài Tập Thực Hành 01
Yêu cầu: tạo ứng dụng todo list, kết hợp lưu trữ data với Local Storage

Bước 1: tạo mới todo
Tạo file **create.todo.html, với nội dung sau:**
form html để tạo mới todo, bao gồm:

- Input để nhập todo
- Button để submit

Khi nhấn nút submit, sẽ cần xử lý:

1. Lấy dữ liệu (tên todo) tại input
2. Lưu todo vào localStorage theo định dạng object { id, name}
   Id của todo sẽ được random ngẫu nhiên (cần tham số này, để sau này xóa todo)

   Ví dụ: data todo khi lưu: { id: 1, name: "learn javascript"}
   LocalStorage sẽ lưu dưới định dạng array: localStorage.setItem(todo, data)
   [
   { id: 1, name: "learn javascript"},
   { id: 2, name: "learn typescript"}
   ...
   ]

3. Lưu todo thành công, redirect về file **video70.html**

Bước 2: Hiển thị danh sách todo

Hiển thị danh sách todo dưới dạng table, data lấy từ localStorage, tương tự :

Id	Name	Action
1	Learn javascript	Button Xóa
2	Learn typescript	Button Xóa

Bước 3: Xử lý hành động xóa todo
Khi nhấn vào button xóa trong table, sẽ tiến hành xóa todo. Sử dụng JavaScript để:

1. Lấy id của todo cần xóa
   Gợi ý:
   //với html, gán data attribute
   <td><button class="**delete-btn**" **data-id="1"**>Xóa</button></td>

   ```
   // với javascript gán sự kiện click cho tất cả nút có class "delete-btn"
   document.querySelectorAll('.delete-btn').forEach(function(button) {
     button.addEventListener('click', function() {
       const id = button.getAttribute('data-id');
        });
   });
   ```

2. Sử dụng hàm filter để xóa data localStorage

3. Xóa thành công, reload lại website để thấy kết quả

#55. Chữa Bài Tập Thực Hành 01 - Part 1

Mục tiêu: create todo

```
//random number
function getRandomInt(min, max) {
  min = Math.ceil(min);
  max = Math.floor(max);
  return Math.floor(Math.random() * (max - min + 1)) + min;
}
```

#56. Chữa Bài Tập Thực Hành 01 - Part 2

Mục tiêu: display todo

```
//todo
```

#57. Chữa Bài Tập Thực Hành 01 - Part 3

Mục tiêu: delete a todo

```
//todo
```

#58. Bài Tập Thực Hành 02

Yêu cầu: xây dựng table blog với data từ backend

Bước 1: Bạn cần chạy backend
npm run dev

Kiểm tra backend đã hoạt động chưa:
http://localhost:8000/

Kiểm tra API đã hoạt động chưa:
http://localhost:8000/blogs

Bước 2: Render tables
Thực hiện gọi API với fetch (lấy danh sách blogs)
//sử dụng async await

Id	Title	Author	Content	Action
1	...	...	...	Button Xóa
2	...	...	...	Button Xóa

Bước 3: Thêm mới blog
Tạo form thêm mới blog phía trên table (cùng page html), bao gồm
- 3 ô input: title, author, content
- 1 button submit

Khi nhấn nút submit thì:
Tạo mới blog bằng cách sử dụng fetch gọi API với method POST
https://stackoverflow.com/a/29823632

POST /blogs
body truyền lên object { title, author, content}

Sau khi tạo mới xong:

- Thêm row cuối của table với data vừa thêm mới

Gợi ý:

```
const tableBody = document.querySelector('#myTable tbody');

// Tạo phần tử dòng mới
const newRow = document.createElement('tr');

// Gán HTML cho dòng
newRow.innerHTML = `
   <td>Người mới ${currentId}</td>
  <td><button class="delete-btn" data-id="${currentId}">Xóa</button></td>
`;

// Thêm dòng vào cuối bảng
tableBody.appendChild(newRow);
```

Bước 4: Xóa blog

1. Lấy id của blog cần xóa

Gợi ý:

```
//với html, gán data attribute
<td><button class="delete-btn" data-id="1">Xóa</button></td>

// với javascript gán sự kiện click cho tất cả nút có class "delete-btn"
document.querySelectorAll('.delete-btn').forEach(function(button) {
  button.addEventListener('click', function() {
    const id = button.getAttribute('data-id');
    });
});
```

2. Gọi api để xóa blog

Xóa 1 blog

DELETE /blogs/:id

3. Dùng javascript để xóa row

```
document.querySelectorAll('.delete-btn').forEach(function(button) {
  button.addEventListener('click', function() {
    const row = this.closest('tr');
    row.remove();

     });
});
```

#59. Chữa Bài Tập Thực Hành 02 - Part 1

Mục tiêu: Hiển thị danh sách blog
//todo

#60. Chữa Bài Tập Thực Hành 02 - Part 2
Mục tiêu: Thêm mới blog
//todo

#61. Chữa Bài Tập Thực Hành 02 - Part 3
Mục tiêu: Xóa blog
//todo

#62. What's next với JavaScript

JavaScript có thể giúp bạn làm được website (frontend, backend), app mobile và nhiều ứng hơn nữa...

Cách đơn giản, và phổ biến nhất của javascript (tương tự như cách khóa học đã hướng dẫn), sử dụng javascript sẽ giúp kiểm soát nội dung HTML/CSS một cách linh động.

Các định hướng phát triển tiếp theo để hiểu rõ và nắm vững JavaScript hơn, là **học chuyên sâu về các thư viện (library) và framework của javascript**.

Gợi ý các định hướng phát triển:

Khuyến khích chủ động nâng cấp/upgrade skill với ngôn ngữ TypeScript
Tham khảo khóa học TypeScript tại đây

Định hướng 1: Phát triển **frontend** của website với JavaScript
Bạn có thể học các thư viện, framework như React.js, Angular, Vue... Đây là những công cụ chuyên nghiệp, giúp bạn giảm thiểu thời gian phát triển giao diện website

Tham khảo khóa học React.js của Hỏi Dân IT tại đây

Định hướng 2: Phát triển **backend** của website với Node.js

Bạn có thể học các framework làm website phổ biến như Express, Nestjs...

Tham khảo khóa học Node.js Pro của Hỏi Dân IT tại đây

Định hướng 3: Phát triển **mobile app**
Bạn có thể tìm hiểu về React Native
Tham khảo khóa học React Native của Hỏi Dân IT tại đây

Định hướng 4: Các ứng dụng khác

JavaScript có thể giúp bạn làm desktop app, các ứng dụng IoT...
Tùy vào mục đích sử dụng, bạn cần tìm hiểu các thư viện/framework hỗ trợ lĩnh vực ấy