Chủ đề: Nhập môn & Môi trường phát triển

1. Tổng Quan Về JavaScript

Hạng mục	Nội dung cốt lõi
Lịch sử	**1995**, bởi **Brendan Eich** (Netscape). Bản mẫu đầu tiên viết trong 10 ngày.
Tên gọi	Mocha → LiveScript → JavaScript.
Java vs JS	*"Java" và "JavaScript" không giống nhau* → Giống tên, nhưng khác hoàn toàn về bản chất.
Vai trò	**Frontend:** Xử lý tương tác, hiệu ứng, DOM. **Backend:** Node.js (Server, API). **Mobile/Desktop:** React Native, Electron.

2. Bộ Ba Cấu Thành Website

Mối quan hệ giữa JS và các ngôn ngữ khác trong trình duyệt.

Ngôn ngữ	Vai trò	Ví dụ ẩn dụ
HTML	**Cấu trúc (Structure):** Khung xương của trang web.	Bộ khung nhà
CSS	**Giao diện (Presentation):** Màu sắc, bố cục, trang trí.	Sơn, nội thất
JavaScript	**Hành vi (Behavior):** Xử lý logic, tương tác động.	Điện, nước, Smart home

3. Phân Loại Website (Static vs Interactive)

Đặc điểm	Static Website (Web Tĩnh)	Interactive Website (Web Động)
Nội dung	Cố định, ai vào cũng thấy giống nhau.	Thay đổi linh hoạt tùy theo người dùng.
Tương tác	Một chiều (Read-only).	Hai chiều (Read/Write/Respond).
Database	Không có.	Có lưu trữ dữ liệu.

Ví dụ	Portfolio, Landing page giới thiệu.	Facebook, Shopee, Web tin tức.

4. Công Cụ & Môi Trường (DevTools)

- **Nơi chạy JS:** Trình duyệt (Browser) hoặc Server (Node.js).
- **Mở DevTools:** Nhấn F12 hoặc Ctrl + Shift + I (Win) / Cmd + Opt + I (Mac).
- **Tab Console:** Nơi gõ lệnh JS để test nhanh kết quả.

Lệnh cơ bản cần nhớ:

// In thông tin ra màn hình console
console.log("Xin chào, tôi là [Tên]");

// Thay đổi tiêu đề trang web
document.title = "Tiêu đề mới";

// Hiển thị hộp thoại thông báo
alert("Thông báo!");

Ví dụ:

```
// In ra trong console
console.log('Hello JavaScript!')
// Hàm
function chao(ten) {
    console.log("Xin chao " + ten);
}
chao("Huan")
// DOM
document.title = "Demo JavaScript trong DevTools";
```

5. Cách Nhúng JavaScript Vào HTML

Có 3 cách để đưa JS vào trang web, trong đó **External** là cách chuyên nghiệp nhất.

Cách nhúng	Cú pháp	Ưu/Nhược điểm	Khuyên dùng
1. Inline (Nội tuyến)	<button onclick="alert('Hi')">	**Nhược:** Code lẫn lộn, khó quản lý.	Hạn chế

2. Internal (Nội bộ)	Viết trong thẻ <script>...</script>	**Ưu:** Tiện cho demo nhỏ. **Nhược:** Khó tái sử dụng.	Cân nhắc, nếu chức năng ít và không phức tạp thì nên dùng
3. External (Bên ngoài)	<script src="script.js"></script>	**Ưu:** Tách biệt code, dễ quản lý, tái sử dụng cao.	Chức năng nhiều, phức tạp và cần quản lý theo component riêng

Chủ đề: Cú pháp cơ bản & Thao tác DOM

1. Biến & Kiểu Dữ Liệu (Variables & Data Types)

Biến (Variable) là tên định danh cho một vùng nhớ dùng để lưu trữ dữ liệu.

Mẹo chọn "chiếc hộp" để chứa dữ liệu.

Từ khóa	Ý nghĩa	Khi nào dùng?	Ví dụ
const	Hằng số (Bất biến)	Dùng mặc định cho mọi biến không cần thay đổi giá trị.	const pi = 3.14;

let	Biến số (Có thể đổi)	Chỉ dùng khi giá trị cần thay đổi (ví dụ: điểm số, biến đếm).	let score = 0;
			
			score = 10;

var	Biến kiểu cũ	**Hạn chế dùng** (Dễ gây lỗi logic scope).	

- ***Block Scope*** là phạm vi được định nghĩa trong một cặp ngoặc nhọn {}
- ***Function Scope*** Khi bạn khai báo biến trong một hàm (Ví dụ: dùng var), biến đó sẽ có Function Scope. Nó chỉ "sống" và được sử dụng trong hàm đó.

Ví dụ:

```
// 1. Vấn đề khai báo lại (Redeclaration)
var x = 10;
var x = 20; // Không báo lỗi, x bị ghi đè. Rất nguy hiểm trong dự án lớn.
console.log(x); // 20

let y = 10;
// let y = 20; // LỖI NGAY: Identifier 'y' has already been declared. Giúp bảo vệ biến.

// 2. Vấn đề phạm vi (Block Scope)
if (true) {
    var nameVar = "CICT";
    let nameLet = "CTU";
}

console.log(nameVar); // output: "Hùng Var" -> var lọt ra ngoài phạm vi { }
// console.log(nameLet); // LỖI: nameLet is not defined -> let được bảo vệ bên trong { }
```

Kiểu dữ liệu phổ biến:

- **String (Chuỗi):** Dùng để lưu trữ văn bản, được bao quanh bởi dấu nháy đơn ‘ hoặc nháy kép “. Ví dụ: "Hello", 'Vietnam'
- **Number (Số):** Bao gồm số nguyên và số thập phân 10, 5.5, -2
- **Boolean (Logic):** Là giá trị logic và chỉ có 2 giá trị **true** hoặc **false**
- **Object (Đối tượng):** là 1 kiểu dữ liệu tự định nghĩa thể hiện dưới dạng các cặp Key : Value. Ví dụ: const NguyenVanA: { name: "A", age: 20 };

2. Toán Tử & Điều Kiện

- **Toán tử số học** (Phép toán căn bản): **+**, **-**, ***, **/**
- **Toán tử gán**: dấu “**=**” (dùng để đưa giá trị vào trong biến)**.**

Logic để máy tính ra quyết định.

- **Toán tử so sánh:**
    - **==** (So sánh thường): 5 == "5" → **True** (Chỉ so giá trị).
    - **===** (So sánh chặt chẽ): 5 === "5" → **False** (So cả giá trị & kiểu dữ liệu).

Câu lệnh điều kiện if / else: Đưa ra quyết định dựa trên một điều kiện nhất định.

```
if (age >= 18) {
   console.log("Đủ tuổi");
} else {
   console.log("Chưa đủ tuổi");
}
```

3.Hàm (Function) & Mảng (Array) & Vòng lặp

Hàm (Function) là một khối xử lý logic được thiết kế để thực hiện một nhiệm vụ cụ thể.

Ví dụ:

```
function tênHàm(thamSố1, thamSố2) {
  // Khối mã xử lý logic bên trong
  return kếtQuả; // (Tùy chọn) trả về giá trị sau khi xử lý
}
```

```
// Khai báo hàm "chào hỏi"
function xinChao(ten) {
  return "Xin chào, " + ten + "!";   // Logic: Cộng chuỗi để tạo lời chào
}

// Sử dụng hàm (Gọi hàm)
let thongBao = xinChao("CICT");
console.log(thongBao); // Kết quả: "Xin chào, CICT!"
```

Mảng (Array):

Mảng (Array) là một biến đặc biệt dùng để lưu trữ danh sách nhiều giá trị.

- **Cách lấy phần tử:** Dùng chỉ số (index) bắt đầu từ 0. Ví dụ: tenMang[0].
- **Cách thêm phần tử:** Dùng hàm **.push()** để thêm một giá trị mới vào cuối danh sách.

  Ví dụ:
  let tasks = ["Ăn", "Ngủ", "Code"];

  // Lấy phần tử (Index bắt đầu từ 0)
  console.log(tasks[0]); // -> "Ăn"

  // Thêm vào cuối
  tasks.push("Lặp lại");

Vòng lặp (Loop): dùng để lặp lại một khối mã cho đến khi điều kiện không còn đúng nữa.

Vòng lặp	Cú pháp

(Cổ điển): Kiểm soát chi tiết số lần lặp thông qua chỉ số i.	for (vị trí bắt đầu; điều kiện dừng; bước_nhảy) {
	// hành động
	}

(Hiện đại): Cách viết ngắn gọn, chuyên dùng để duyệt qua từng phần tử của mảng mà không cần quản lý biến i.	tenMang.forEach(function(phanTu) {
	// hành động
	});

Ví dụ:

```
let danhSach = ["Làm bài tập", "Đi chợ", "Nấu cơm"];

// Cách 1: Sử dụng vòng lặp for
for (let i = 0; i < danhSach.length; i++) {
  console.log("For: " + danhSach[i]);
}

// Cách 2: Sử dụng forEach (Ngắn gọn hơn)
danhSach.forEach(function(congViec) {
  console.log("forEach: " + congViec);
});
```

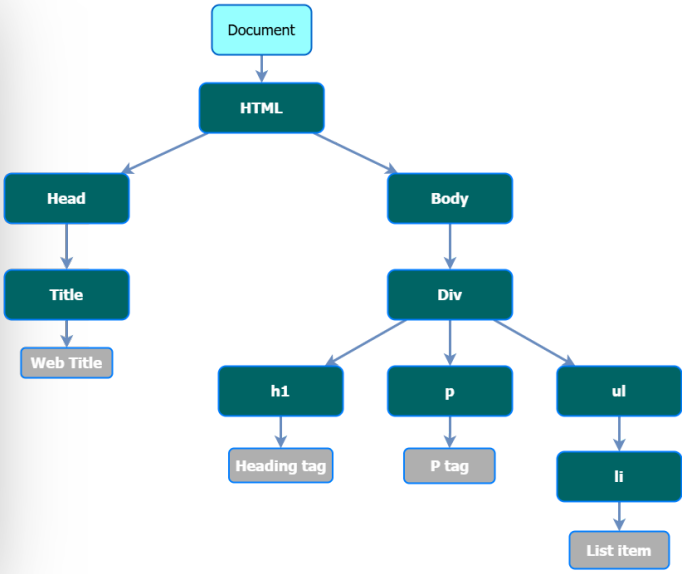
4. DOM (Document Object Model)

DOM (Document Object Model) đóng vai trò **trung gian giữa HTML và JavaScript.**

Lưu ý: JS không sửa file HTML gốc, JS dùng DOM để thay đổi những gì đang hiển thị trên màn hình (được lưu trong RAM).

Cấu trúc Cây DOM:

```html
<html>
  <head>
    <title>Web Title</title>
  </head>
  <body>
    <div>
      <h1>Heading tag</h1>
      <p>P tag</p>
      <ul>
        <li>List item</li>
      </ul>
    </div>
  </body>
</html>
```



5. Thao Tác DOM

A. Truy xuất (Lấy phần tử)

Lệnh	Ý nghĩa	Kết quả trả về
document.**getElementById**('id')	Chọn qua ID (Duy nhất).	1 Phần tử
document.**querySelector**('.class')	Chọn phần tử **đầu tiên** thấy.	1 Phần tử
document.**querySelectorAll**('.class')	Chọn **tất cả** phần tử khớp.	Danh sách (NodeList)
element.tagName	Kiểm tra xem phần tử đó là loại thẻ gì	Tên thẻ dạng chữ IN HOA
parentElement	Tìm cha của phần tử hiện tại	1 Phần tử

B. Thay đổi nội dung

- element.**innerText** = "Nội dung mới" → Chỉ sửa chữ hiển thị.
- element.**innerHTML** = "Đậm" → Sửa cả HTML (cẩn thận bảo mật).
- element.**textConten:** Lấy toàn bộ chữ (bao gồm display:none)
- document.**createElement**("tên_thẻ")
    - Công dụng: Dùng để "đẻ" ra một phần tử HTML mới hoàn toàn bằng JavaScript (nó mới chỉ nằm trong bộ nhớ, chưa hiện ra màn hình).
    - Ví dụ: const thongBao = document.createElement("p"); tạo ra một thẻ .
- parentElement.appendChild(con)

- Công dụng: Đưa phần tử vừa tạo ở trên vào trong một phần tử "cha" để nó hiển thị lên giao diện.
- Ví dụ: manHinh.appendChild(thongBao); sẽ đưa thông báo vào trong khung TV.

C. Thay đổi giao diện (Style & Class)

- **Style trực tiếp:** element.style.backgroundColor = "red"; (Lưu ý: dùng **camelCase**).
- **Quản lý Class (Khuyên dùng):**
  - element.classList.**add**('active') (Thêm)
  - element.classList.**remove**('active') (Xóa)
  - element.classList.**toggle**('active') (Bật/Tắt)

D. Bắt sự kiện (Event Listener)

Cú pháp chuẩn: element.**addEventListener**('tên_sự_kiện', hàm_xử_lý)

Nhóm sự kiện	**Tên sự kiện**	**Mô tả**
Chuột	click	Nhấn chuột trái (dùng nhiều nhất).
	dblclick	Click đúp
	contextmenu	Chuột phải
Phím	keydown	Nhấn xuống
	keyup	Nhả phím ra
Form	submit	Gửi form
	change	Giá trị thay đổi
	input	Gõ vào input

e.target:

Công dụng: Xác định chính xác vật thể nào đã bị click bên trong một vùng sự kiện lớn.
Ví dụ: Khi bạn đặt sự kiện click cho cả khung chứa đèn, e.target sẽ chỉ điểm đúng cái bóng đèn cụ thể mà bạn vừa chạm tay vào.