

JavaScript Tham Khảo Nhanh

ES6+, DOM, sự kiện, Fetch API, async/await

Cơ Bản

Biến

```
let name = "Alice"; // reassignable
const PI = 3.14; // constant
var old = "avoid"; // function-scoped (legacy)
```

Kiểu Dữ Liệu

string	Chuỗi ký tự: "hello" hoặc 'hello'
number	Số nguyên hoặc thực: 42, 3.14
boolean	true / false
null	Giá trị rỗng có chủ đích
undefined	Đã khai báo nhưng chưa gán giá trị
object	Cặp key-value: { a: 1 }
array	Danh sách có thứ tự: [1, 2, 3]
symbol	Định danh duy nhất

Kiểm Tra & Chuyển Đổi Kiểu

```
typeof "hello" // "string"
typeof 42 // "number"
Number("42") // 42
String(100) // "100"
parseInt("3.9") // 3
parseFloat("3.14") // 3.14
```

Chuỗi

Template Literal

```
const name = "Alice";
`Hello, ${name}!` // Hello, Alice!
`Total: ${2 + 3}` // Total: 5
`Multi
line string`
```

Phương Thức Chuỗi

s.length	Số ký tự
s.toUpperCase()	Bản sao CHỮ HOA
s.toLowerCase()	Bản sao chữ thường
s.trim()	Xóa khoảng trắng đầu/cuối
s.split(",")	Tách thành mảng
s.includes("x")	Kiểm tra chứa → bool
s.indexOf("x")	Chỉ số đầu tiên (-1 nếu không có)
s.slice(1, 4)	Chuỗi con theo chỉ số
s.replace(a, b)	Thay thế kết quả đầu tiên
s.replaceAll(a, b)	Thay thế tất cả kết quả
s.startsWith(x)	Kiểm tra tiền tố → bool
s.endsWith(x)	Kiểm tra hậu tố → bool
s.padStart(n, c)	Thêm ký tự vào đầu đến độ dài n

Mảng

Tạo & Truy Cập

```
const fruits = ["apple", "banana", "cherry"];
fruits[0] // "apple"
fruits.length // 3
fruits.at(-1) // "cherry"
```

Phương Thức Thay Đổi Mảng

arr.push(x)	Thêm vào cuối
arr.pop()	Xóa & trả về phần tử cuối
arr.unshift(x)	Thêm vào đầu
arr.shift()	Xóa & trả về phần tử đầu
arr.splice(i, n)	Xóa n phần tử tại chỉ số i
arr.sort()	Sắp xếp tại chỗ (theo từ điển)
arr.reverse()	Đảo ngược tại chỗ

Phương Thức Không Thay Đổi Mảng

arr.map(fn)	Biến đổi từng phần tử
arr.filter(fn)	Giữ phần tử mà fn trả về true
arr.reduce(fn, init)	Tích lũy thành một giá trị
arr.find(fn)	Kết quả đầu tiên hoặc undefined
arr.findIndex(fn)	Chỉ số kết quả đầu tiên (-1)
arr.includes(x)	Kiểm tra chứa → bool
arr.slice(a, b)	Sao chép nông một phần
arr.join(",")	Nối thành chuỗi
arr.forEach(fn)	Duyệt qua (không có giá trị trả về)
[...a, ...b]	Nối mảng (spread)

Object

Tạo & Truy Cập

```
const user = { name: "Alice", age: 20 };
user.name // "Alice"
user["age"] // 20
user.gpa = 3.85; // add/update
```

Destructuring & Spread

```
const { name, age } = user;
const copy = { ...user, age: 21 };
```

Phương Thức Object

Object.keys(o)	Mảng các key
Object.values(o)	Mảng các value
Object.entries(o)	Mảng các cặp [key, value]
Object.assign(t, s)	Sao chép thuộc tính s → t
"k" in obj	Key tồn tại? → bool
delete obj.k	Xóa thuộc tính
Object.freeze(o)	Làm bất biến (nông)

Luồng Điều Khiển

if / else if / else

```
if (score >= 90) {
  grade = "A";
} else if (score >= 80) {
  grade = "B";
} else {
  grade = "C";
}
```

Toán Tử Ba Ngôi & Nullish Coalescing

```
const status = score >= 60 ? "pass" : "fail";
const name = user.name ?? "Anonymous";
```

switch

```
switch (color) {
  case "red": stop(); break;
  case "green": go(); break;
  default: wait();
}
```

Vòng Lặp

for / for...of / for...in

```
for (let i = 0; i < 5; i++) { }

for (const item of ["a", "b"]) { } // arrays

for (const key in obj) { } // object keys
```

while / do...while

```
while (count < 10) { count++; }

do { count++; } while (count < 10);
```

break & continue

```
for (let i = 0; i < 10; i++) {
  if (i === 5) break; // stop loop
  if (i % 2 === 0) continue; // skip
}
```

Hàm

Khai Báo Hàm & Arrow Function

```
function greet(name) {
  return `Hello, ${name}!`;
}
const greet = (name) => `Hello, ${name}!`;
const square = x => x * x; // single param
```

Tham Số Mặc Định & Rest

```
function greet(name = "World") { }

function sum(...nums) {
  return nums.reduce((a, b) => a + b, 0);
}
```

Callback

```
[1, 2, 3].map(x => x * 2); // [2, 4, 6]
[1, 2, 3].filter(x => x > 1); // [2, 3]
setTimeout(() => console.log("done"), 1000);
```

Class

```
class Dog {
  constructor(name, breed) {
    this.name = name;
    this.breed = breed;
  }
  bark() { return `${this.name} says Woof!`; }
}

class Puppy extends Dog {
  constructor(name, breed, toy) {
    super(name, breed);
    this.toy = toy;
  }
}
```

Xử Lý Lỗi

```
try {
  JSON.parse("bad json");
} catch (err) {
  console.error(err.message);
} finally {
  console.log("always runs");
}
```

Ném Lỗi

```
throw new Error("Something went wrong");
```

DOM

Chọn Phần Tử

```
document.querySelector(".cls") // first match
document.querySelectorAll("li") // all matches
document.getElementById("main")
```

JavaScript Tham Khảo Nhanh

Chỉnh Sửa Phần Tử

```
el.textContent = "new text";
el.innerHTML = "<b>bold</b>";
el.style.color = "red";
el.classList.add("active");
el.classList.toggle("hidden");
el.setAttribute("data-id", "42");
```

Sự Kiện

```
btn.addEventListener("click", (e) => {
  console.log(e.target);
});
```

Tạo Phần Tử

```
const li = document.createElement("li");
li.textContent = "New item";
ul.appendChild(li);
el.remove(); // remove element
```

Fetch API

Yêu Cầu GET

```
fetch("https://api.example.com/data")
  .then(res => res.json())
  .then(data => console.log(data))
  .catch(err => console.error(err));
```

Yêu Cầu POST

```
fetch(url, {
  method: "POST",
  headers: { "Content-Type": "application/json" },
  body: JSON.stringify({ key: "value" }),
});
```

Async / Await

```
async function loadData() {
  try {
    const res = await fetch(url);
    const data = await res.json();
    return data;
  } catch (err) {
    console.error(err);
  }
}
```

Yêu Cầu Song Song

```
const [users, posts] = await Promise.all([
  fetch("/users").then(r => r.json()),
  fetch("/posts").then(r => r.json()),
]);
```

Module

Named Export

```
// math.js
export const PI = 3.14;
export function add(a, b) { return a + b; }

// main.js
import { PI, add } from "./math.js";
```

Default Export

```
// logger.js
export default function log(msg) { }

// main.js
import log from "./logger.js";
```