**Kiến thức chung**

IDE: môi trường phát triển tích hợp, là 1 phần mềm giúp các ltv phát triển pm, gồm trình soạn thảo, biên dịch, trình gỡ lỗi, và 1 số công cụ tùy vào IDE

Bộ nhớ: khá giống java

code

“use strict” : khắt khe về code, nếu chưa khai báo vẫn chạy đc nhưng về sau sẽ lôi, nếu thêm này vào sẽ báo lỗi. giống ngoại lệ. Hiện đại đã tự động dùng

biến

tên k trùng từ khóa, không bắt đầu bằng số, k dùng kí tự đặc biệt ngoại trừ $ \_

‘aa’ “aa” giống nhau

var - function scope/global scope - khai báo trong vòng for thì ra ngoài vẫn còn. (thông tin vô dụng: có thể khai báo lại cùng phạm vi, khai báo trước khi dùng mà k lỗi, let vs const k có cái này)

let - block scope

const - block scope , k thể đổi giá trị giống final bên java. Dùng cho hằng số

block scope : khai báo trong if rồi ra ngoài sẽ mất;

func scope: khai báo trong if rồi ra ngoài vẫn còn(chì trong block if, còn trong func thì ra ngoài vẫn mất như java)

KDL

number:

string: đặc biệt `${name}` : sẽ ra giá trị của biến name

bigint: số + n, 123n

boolean

null

undefined

symbol:

Toán tử

khá giống java

khác vs java gồm

\*\* vs ^: lũy thừa

khi tính toán string thì nó xem là number nếu là dạng number “2” - “1”

trừ dấu + thì nó ghép string, nếu là bigInt thì phải chung bigInt nếu k sẽ lỗi..

So sánh

khá giống java

khác so sánh bằng có thêm so sánh nghiêm ngặt

1==”1” true js tự hiểu “1” cũng là 1

1===”1” false khác KDL là false

tương tự != vs !==

this

arrow function,callback, anonymous function(settimeout,…), event handle k có this, nó sẽ gọi “object global” -window.

Object

1. let o = { key1 : value, key2 : value}

2. function Person(name, age) { this.name = name;

this.age = age; this.greet = function() { console.log('Hello!'); }; }

*Khởi tạo từ khai báo bt, hàm khởi tạo, từ khóa new Object*

// tạo method cho object

o.sayHello = function () { console.log("Hello!"); };

Delete o.key1/o[“keyString”]

console.log("name" in user); // true/false

\_.isEqual(object1;object2); // so sánh giá trị

**Prototype** (là 1 thuộc tính đặc biệt)giúp kế thừa bởi các đối tượng tạo ra( thay vì thiết kế function trong lúc định nghĩa thì mình tạo ở ngoài vẫn kế thừa được)

writable: nếu là true thì giá trị value của thuộc tính có thể thay đổi. Ngược lại, thuộc tính gọi là read-only (thuộc tính chỉ đọc).

enumerable: nếu là true thì thuộc tính xuất hiện trong vòng lặp, ngược lại thì không.

configurable: nếu là true thì thuộc tính có thể xóa và các cờ khác có thể thay đổi, ngược lại thì không.

Method có thể là biến, thuộc tính của 1 obj, tham số (hàm callback)

Các loại hàm

default: function a() {return b;}

hàm khởi tạo object - ở trên mục KIẾN THỨC CHUNG ->Object

hàm biểu thức: let a = function (){return b;}

hàm biểu thức dùng arrow: **() => {} ,** *nó kế thừa this từ hàm chứa nó.*

const a = (v,u) => {return b+2;}

const a = (v,u) => b + 2;

**Regex** – công cụ để xử lý, tìm kiếm, chỉnh sửa văn bản như kiểm tra có phải email hay sđt hợp lệ, kiểm ra là số hay là chữ,..

**Javascript web worker**

là (API) công nghệ tạo luồng xử lý chạy song song với luồng chính, nó giúp xử lý các tác vụ năng hoặc lâu dài mà k ảnh hưởng(làm chậm trễ) đến giao diện trình duyệt, tăng trải nghiệm người dùng

SCSS:

2. lồng các quy tắc

3. tái sử dụng (có khai báo biến để sử dụng khi cần, tái sử dụng quy tắc, thuộc tính cụ thể hoặc từ class khác,function tính toán)

5. chia nhỏ và import vào 1 file chính

**Array**

**Spread (...) :** sao chép, ghép mảng hoặc đối tượng

const array1 = [1, 2, 3];

const array2 = [4, 5, 6];

const combinedArray = [...array1, ...array2];

console.log(combinedArray); // Output: [1, 2, 3, 4, 5, 6]

**Rest (...):** thu thập các tham số thành 1 mảng (biến)

function myFunction(a, b, ...rest) {

console.log(a); // Output: 1

console.log(b); // Output: 2

console.log(rest); // Output: [3, 4, 5]

}

myFunction(1, 2, 3, 4, 5);

**Map/Set**

giống java, nhưng khác tên method

**WeakMap/WeakSet**

giống với Map/Set

khác ở

1. key phải là Obj

2. k phải là iterable obj nên k thể duyệt qua các phần tử(kể cả dùng for), do các cặp key-value k được giữ trong bộ nhớ, vì vậy sẽ được xóa nếu k còn sử dụng

**DOM** là giao diện(API) lập trình cho tài liệu HTML hoặc XML

Các hàm phổ biến

const result = confirm("Bạn muốn tiếp tục sử dụng chương trình?");

alert(result);

const name = prompt("Nhập vào tên của bạn:", "Anonymous");

alert(name);;

Cho phép js truy cập và chỉnh sửa nội dung, cấu trúc, và kiểu của các thành phần trong html, cũng như trên trang web.

Xử lý sự kiên trên trang web như click, double click, di chuyển chuột,...

ví dụ: thay đổi nội dung của 1 thẻ <p>, can thiệp thuộc tính css như màu, kích cỡ

JQUERY: 1. đơn giản hóa cú pháp

2. Xử lý sự kiên : click,di chuột,...

3.dễ dàng thực hiện các yêu cầu AJAX

const element = document.getElementById('myId');

const element = document.getElementById('myId');

const elements = document.getElementsByTagName('div');

const element = document.querySelector('.myClass');

const elements = document.querySelectorAll('div.myClass');

const parent = document.getElementById('parent');

const child = document.getElementById('child');

parent.removeChild(child);

const parent = document.getElementById('parent');

const newChild = document.createElement('p');

const oldChild = document.getElementById('oldChild');

parent.replaceChild(newChild, oldChild);

document là đối tượng toàn cục trên trang web

Nó là một phần của DOM (Document Object Model) và cung cấp các phương thức và thuộc tính để tương tác với nội dung của trang web.

**Bất đồng bộ (callback,promise,async/await)**

các hàm callback phổ biến:

**reduce()** - duyệt qua các phần tử từ trái sang phải thành 1 giá trị đơn, bằng cách (gọi lại 1 hàm callback, hàm đó sẽ thực hiện tính toán, còn hàm reduce() sẽ thực hiện vòng lặp và kết nạp giá trị trả về) thực hiện các phép tính phức tạp, như tính tổng mảng, tìm giá trị tối đa,gộp các thuộc tính thành các nhóm...

call back là 1 hàm được truyền vào đối số của 1 hàm khác, sẽ được thực thi sau khi thực thi hàm chứa nó

Phần lớn là dùng tham số cho các phương thức có vòng for

*(mục đích: bên java cần đổi cách thức hoạt động của phương thức có sẵn thì cần đổi tên đối với phép tích phức tạp như vừa for vừa tính toán (vd: reduceSum, reduceSubtract ), còn bên js thì chỉ cần đổi hàm callback (vd arr.reduce((acc, val) => acc + val) hoặc arr.reduce((acc, val) => acc - val) ) rất tiện lợi.*

*numbers.sort((a, b) => a - b) tăng dần*

*vs numbers.sort((a, b) => a - b) giảm dần*

*bên java phải overide lại phương thức compareTo() giữ nguyên tên và tham số của hàm sort. bên js này chỉ đổi tham số chính là hàm callback.*

*\* giải thích: sort sẽ tạo vòng lặp, tham số sẽ là 1 func, trong đó a,b là thuộc tính, nó sẽ được gán cho vị trí 2 ptu liên tiếp của number thông qua "this". (let array = this; giả sử có 1 array gọi nó) Sau đó nó return ra a-b. Nếu tham số là func trả về 1 phần tử âm ( a < b) thì nó xử lý trong sort là xếp a đứng trước b, xong chạy vòng for tiếp.*

*Tức là khi gọi sort nó sẽ chạy vòng for, mỗi vòng nó lấy giá trị từ hàm callback (ở đây là a-b giá trị sô) để xử lý, hoặc đơn giản là nó xử lý hàm callback bằng dữ liệu là tham số a,b (ví dụ nó lấy result của callback gom lại thành 1 list rồi trả về func chính)*

**Design pattern**

là mẫu thiết kế, là 1 kỹ thuật trong LTHĐT, cung cấp giải pháp chung để giải quyết 1 cách tối ưu các vấn đề thường gặp trong thiết kế hướng đối tượng.

giúp tăng tính linh hoạt và tái sử dụng mã nguồn, dễ bảo trì và mở rộng

Singleton - là một mẫu thiết kế (design pattern) trong lập trình phần mềm nhằm đảm bảo rằng chỉ có một instance duy nhất của một lớp được tạo ra trong toàn bộ ứng dụng., cung cấp một phương thức để truy cập instance đó

cơ chế : tạo ra một thuộc tính static để lưu trữ instance duy nhất của lớp đó. -> có 1 pt kiểm tra nếu chưa có thì tạo mới hoặc trả về 1 instance nếu đã tồn tại.

Inversion of Control (IoC): là nguyên tắc thiết kế, Spring sử dụng IoC để chuyển giao trách nhiệm quản lý các phụ thuộc từ các lớp đến Spring Container. DI chính là phương pháp

DI: giúp tiêm các phụ thuộc giữa các thành phần trong ứng dụng, giảm sự phụ thuộc của các bên liên quan, tăng tính bảo trì và dễ dàng mở rộng khi thay đổi 1 bên thì bên kia sẽ k bị ảnh hưởng.

**ES6 - tiêu chuẩn hóa của js, cải tiến về mặt code -học**

* let, const (biến final, k thể đổi) giúp dễ bảo trì.
* hỗ trợ cú pháp classes
* hỗ trợ modules có thể import/export file js, sử dụng biến từ file khác.
* Hỗ trợ đối tượng Promises thực hiện bất đồng bộ 1 cách linh hoạt ( tạo ra 1 Promises rồi Promises.then() )
* template literal: `${name}` nhúng biến và biểu thức vào chuỗi bằng 2 dấu backticks ``
* Arrow function (dữ ngữ cảnh của ‘this’ trong phạm vi chứa nó)

// Function thông thường

function add(a, b) { return a + b; }

// Arrow function

const addArrow = (a, b) => a + b;

* lấy các giá trị từ mảng hoặc đối tượng và gán chúng vào các biến một cách dễ dàng.

**NOTE:**

**code**

[1, 2].forEach((value) => console.log(value));

mảng, vòng lặp each

console.log(name ?? "Người dùng ẩn danh"); *// Người dùng ẩn danh*

trả về giá trị khác null/undefined, ưu tiên vế 1

console.log(length || 10); *// 10*

console.log(length ?? 10); *// 0*

trả về giá trị true (khác rỗng/0/””/null/undefined)