**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

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, có thể khai báo lại cùng phạm vi, khai báo trước khi dùng mà k lỗi

let - block scope, k thể khai báo lại cùng phạm vi (chỉ khai báo trong phạm vi khác như trong 1 hàm, nhưng ra ngoài thì cái khai báo trong hàm sẽ mất)

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*

o.sayHello = function () { console.log("Hello!"); }; // tạo method cho object

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

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: **() => {} ,** *k có bối cảnh thực thi,k thể tạo construs, k có ‘this’, nó kế thừa this từ bên ngoài*

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 - học 4.6 -4.11 &4.49 &4.20 - xem cái nâng cao ở cuối**

**destructuring assignment**: là cú pháp giúp 1. gán nhiều biến từ nhiều giá trị trong mảng/obj. 2. gán giá trị mặc định cho thuộc tính trong obj

**reduce()** - duyệt qua các phần tử từ trái sang phải thành 1 giá trị đơn, bằng cách thực hiện các phép tính phức tạp, như tính tổng mảng, tìm giá trị tối đa,gộp các thuộc tính thành các nhóm...

**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 - xem sơ**

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

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)- học**

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.

**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
* template literal: `${name}` nối string 1 cách dễ dàng
* 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;

* có thể truyền giá trị mặc định cho tham số
* *classes*
* *modules có thể xuất/nhập các biến từ file khác vào*
* *hỗ trợ bất đồng bộ*
* *mở rộng mảng/đối tượng*
* *trích xuất/gán biến vào mảng*
* *iterator/Generator*

**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)

* Spring Core Container: Quản lý các bean và phụ thuộc.
* Spring Data Access/Integration: Cung cấp các công cụ để làm việc với cơ sở dữ liệu và ORM.
* Spring Web: Hỗ trợ phát triển ứng dụng web và RESTful.
* Spring Security: Cung cấp bảo mật cho ứng dụng.
* Spring Boot: Tự động cấu hình và quản lý các ứng dụng Spring.
* Spring Data: Hỗ trợ các công cụ và API cho các loại cơ sở dữ liệu khác nhau.