# Week 02 — JS Basic 2 — Lý thuyết

## 1) Cấu trúc điều kiện: if/else, switch

• if/else dùng cho điều kiện phức tạp; dùng guard clause để thoát sớm khi dữ liệu không hợp lệ.

• switch phù hợp lựa chọn theo giá trị rời rạc; nhớ break để tránh rơi xuyên (fall-through).

• So sánh nghiêm ngặt (===) để tránh ép kiểu; dùng toLowerCase()/trim() trước khi so sánh chuỗi.

// Guard clause + phân loại tuổi  
function classifyAge(age){  
 if (!Number.isFinite(age) || age < 0) return 'Invalid';  
 if (age < 6) return 'Mầm non';  
 if (age <= 11) return 'Tiểu học';  
 if (age <= 15) return 'THCS';  
 if (age <= 18) return 'THPT';  
 return 'Khác';  
}  
  
// switch cho menu  
function handleMenu(choice){  
 switch(choice){  
 case 1: return 'Thêm';  
 case 2: return 'Sửa';  
 case 3: return 'Xoá';  
 case 0: return 'Thoát';  
 default: return 'Không hợp lệ';  
 }  
}

## 2) Vòng lặp: for, while, do..while; break/continue; for..of

• for: biết trước số lần lặp; while: lặp khi điều kiện còn đúng; do..while: chạy ít nhất 1 lần.

• break dừng vòng lặp; continue bỏ qua phần còn lại của vòng lặp hiện tại.

• for..of duyệt giá trị trong Iterable (Array, String); tránh dùng for..in cho mảng (duyệt key + kế thừa).

// Tổng 1..n & tổng chẵn/lẻ  
function sumStats(n){  
 if (!Number.isInteger(n) || n < 1) return 'Invalid';  
 let total=0, even=0, odd=0;  
 for(let i=1;i<=n;i++){  
 total += i;  
 if (i%2===0) even += i; else odd += i;  
 }  
 return { total, even, odd };  
}  
  
// Duyệt mảng bằng for..of  
function print(arr){  
 let out = '';  
 for(const x of arr){ out += x + ' '; }  
 return out.trim();  
}

## 3) Mảng: tạo, truy cập, các phương thức hay dùng

• Thao tác cơ bản: push/pop (cuối), shift/unshift (đầu), indexOf/includes, slice (không mutate), splice (mutate).

• join để nối thành chuỗi; concat/spread để gộp mảng; sort cần comparator để đúng thứ tự số.

• Sao chép mảng: [...arr] (nông). Tránh mutate khi không cần (dễ sinh bug).

const a = [3,1,2];  
a.push(5); // [3,1,2,5]  
a.pop(); // [3,1,2]  
a.unshift(0); // [0,3,1,2]  
a.shift(); // [3,1,2]  
  
[1,10,2].sort(); // [1,10,2] (sort theo chuỗi)  
[1,10,2].sort((x,y)=>x-y); // [1,2,10] (đúng cho số)  
  
const b = a.slice(0,2); // sao chép không mutate  
const c = [...a, 9]; // spread copy  
  
// Min / Max / Average không dùng Math.min(...arr) khi mảng quá lớn  
function stats(nums){  
 let min=Infinity, max=-Infinity, sum=0;  
 for(const n of nums){ if(n<min)min=n; if(n>max)max=n; sum+=n; }  
 return { min, max, avg: nums.length? sum/nums.length : NaN };  
}  
  
// Mode (giá trị xuất hiện nhiều nhất)  
function mode(arr){  
 const freq = new Map();  
 for(const x of arr) freq.set(x, (freq.get(x)||0)+1);  
 let best=null, cnt=0;  
 for(const [k,v] of freq.entries()) if(v>cnt){ cnt=v; best=k; }  
 return { value: best, count: cnt };  
}

## 4) Chuỗi: thao tác cơ bản

• length, charAt, slice/substring, indexOf/lastIndexOf, includes, trim, toLowerCase/toUpperCase.

• split để tách chuỗi; join để ghép lại. Khi so sánh nên chuẩn hoá (trim + lower).

const s = " JavaScript cơ bản ";  
s.trim(); // "JavaScript cơ bản"  
s.toLowerCase(); // " javascript cơ bản "  
"hello world".includes("lo"); // true  
"abcabc".lastIndexOf("ab"); // 3  
  
"1,2,3".split(","); // ["1","2","3"]  
["a","b","c"].join("-"); // "a-b-c"  
  
// Đảo chuỗi  
function reverseStr(str){  
 let out=''; for(let i=str.length-1;i>=0;i--) out+=str[i];  
 return out;  
}

## 5) Một số hàm thuật toán nhỏ

• Kiểm tra số nguyên tố; đếm số nguyên tố trong mảng; kiểm tra palindrome.

function isPrime(n){  
 if(n<2) return false;  
 for(let i=2;i\*i<=n;i++) if(n%i===0) return false;  
 return true;  
}  
function countPrimes(arr){ return arr.filter(isPrime).length; }  
  
function isPalindrome(s){  
 const t = s.replace(/\W+/g,'').toLowerCase();  
 return t === reverseStr(t);  
}

## 6) Menu điều khiển (switch) — logic lõi

• Viết các chức năng thành hàm nhỏ (add/remove/list), menu chỉ gọi hàm; dễ test & tái sử dụng.

const todo = [];  
function addItem(text){ if(text && text.trim()) todo.push(text.trim()); }  
function removeAt(i){ if(Number.isInteger(i) && i>=0 && i<todo.length) todo.splice(i,1); }  
function listItems(){ return todo.map((t,i)=>`${i}. ${t}`).join('\n'); }  
  
function runCommand(cmd, payload){  
 switch(cmd){  
 case 'add': addItem(payload); return 'OK';  
 case 'rm': removeAt(Number(payload)); return 'OK';  
 case 'ls': return listItems();  
 default: return 'Unknown command';  
 }  
}

## 7) Trò chơi đoán số — tách logic khỏi I/O

• Tách phần "sinh số bí mật & so sánh" khỏi nhập/xuất để dễ test:

function compare(secret, guess){  
 if(guess === secret) return 'Đúng!';  
 return guess < secret ? 'Lớn hơn' : 'Nhỏ hơn';  
}  
// UI/console có thể gọi compare(secret, userGuess) nhiều lần mà không dính vào logic

## 8) Lỗi thường gặp & mẹo

• Off-by-one (quên i<=n hoặc i<n).

• Quên ép kiểu khi nhập từ form (chuỗi) — dùng Number(), parseInt(...,10).

• Dùng for..in với Array làm lộ key không mong muốn — hãy dùng for thường hoặc for..of.

• sort số nhưng quên comparator → kết quả sai (so sánh chuỗi).

• mutate mảng gốc (splice/sort) khi không muốn — hãy copy trước (slice/spread).