**QUY CHUẨN ĐẶT TÊN:**

1. PascalCase 🡪 Tên lớp. Vd: UserClass, CategoryClass…
2. camelCase 🡪 Tên hàm, tên phương thức, tên biến. Vd: getUser, …
3. UPPER\_CASE 🡪 Tên hằng số. Vd: DISCOUNT\_PERCENT, …
4. snake\_case 🡪 Tên bảng, tên cột CSDL SQL. Vd: company\_name, …
5. kabab-case 🡪 Tên file, tên UsRL . Vd: style-mobile.css, …

\* Chú ý:

- Không đặt tên biến bắt đầu bằng số.

- Không chứa ký tự đặc biệt.

- Không dùng trùng function mặc định của ngôn ngữ.

**CÁC PHÍM TẮT QUAN TRỌNG TRONG VISUAL CODE:**

1. F12 🡪 Đi đến hàm đang dùng.
2. Shift + F12 🡪 Đi đến những nơi hàm này được dùng.
3. Ctrl + ` 🡪 Mở Terminal.
4. Ctrl + / 🡪 Comment.
5. Alt + < 🡪 Trở về
6. Ctrl + p 🡪 Tìm mở file.
7. Ctrl + p + > 🡪 Tìm command của Visual Code hoặc của Extension.
8. Ctrl + p + # 🡪 Tìm hàm
9. Ctrl + p + : 🡪 Tìm hàng
10. Ctrl + Shift + k 🡪 Xóa 1 dòng.
11. Alt + Click 🡪 Chọn nhiều con trỏ sửa cùng lúc.
12. Alt + Up/Down🡪 Di chuyển lên trên hoặc xuống dưới.
13. Alt + Shift + Up/Down 🡪 Copy lên trên hoặc xuống dưới.
14. Alt + Shift + I 🡪 Chọn nhiều con trỏ ở cuối hàng.
15. Ctrl + d 🡪 Tìm từng đoạn bôi đen giống nhau.
16. Ctrl + Shift + l 🡪 Tìm tất cả đoạn bôi đen giống nhau.
17. F2 🡪 Đồng bộ đổi tên hàm, biến.
18. Window + . 🡪 Thêm emoji.
19. Alt + z 🡪 Tự động xuống dòng (Wrap text).

**CÁC LỆNH CƠ BẢN CỦA TERMINAL, POWER SHELL, CMD, …**

1. code . 🡪 mở visual code trong thư mục.
2. cd *tên-thư-mục* 🡪 mở thư mục con.
3. cd .. 🡪 trở về thư mục cha.
4. Clear 🡪 xóa nội dung terminal.

**GIT CƠ BẢN**

- Lên trang gitHub/gitLab tạo 1 Remote Repository. (Lưu ý: tên Repo === tên project folder)

1. git –version 🡪 Kiểm tra phiên bản và git đã được cài chưa.
2. git init 🡪 Tạo lcal repo trong máy tính (file .git). (Nếu clone về thì ko cần init)
3. git status 🡪 Kiểm tra tình trạng code thay đổi so với repo.
4. git add *tên-file* 🡪 File đã thay đổi vào trong repo.
5. git add . 🡪 Thêm tất cả thay đổi.
6. git reset 🡪 Gỡ code thay đổi ra khỏi repo.
7. git commit -m *”Ghi-chú”* 🡪 Lưu code đã add.
8. git commit 🡪 Lưu code khi giải quyết conflict.
9. git log 🡪 Xem lại tất cả commit.
10. git log –oneline 🡪 Hiện gọn hơn (id commit ở đầu tiên).
11. git checkout *idcommit* 🡪 Chuyển đổi giữ các commi.
12. git branch -b new*branchname* 🡪 Tạo branch mới.
13. git branch -d *branchname* 🡪 Xóa branch.
14. git branch 🡪 hiện tất cả branch.
15. git checkout *branchname* 🡪 Chuyển đổi giữa các branch (mặc định: main).
16. git merge *branch1* 🡪 Gộp branch1 vào branch hiện tại.
17. git push 🡪 Đẩy code từ local repo lên remote repo. (Nếu push lần đầu cần upstream)
18. git clone *<git repo link>* 🡪 Lấy remote repo từ trên server về máy. (Nếu init thì ko cần clone về)
19. git pull 🡪 Kéo code từ remote repo về local repo.
20. git stash 🡪 Recover lại commit đã lưu

**HOW TO CODE**

1. Have a clear GOAL.
2. DON’T copy code.
3. DO tasks/ project/ practices.
4. Create your own style.
5. Calmdown.
6. Learn what needed. (Học cái gì mới phải biết ích lợi để làm gì)
7. Multitasking >< Asynchronous \*

**SEO SKILL**

1. Back links 🡪 Link từ nơi khác.
2. Slug key 🡪 [www.webname.com/***slug-slug-slug***/](http://www.webname.com/slug-slug-slug/)
3. H1 key 🡪 Tối đa 1 H1 trong page để làm tiêu đề.
4. Content is king.
5. Key được lập lại nhiều lần trong page.
6. Alt key, src name key….

**GLOSSARY / TERMINOLOGY**

1. Attributes 🡪 Thuộc tính (Dùng trong CSS)
2. Properties 🡪 Đặc tính (Object.Property = value)
3. Methods 🡪 Phương thức (function của object - Object.Method())
4. Elements 🡪 Phần tử (id, class, tag HTML – DOM Js)
5. Parameter 🡪 Tham số
6. Argument 🡪 Đối số
7. Variable scope 🡪 Phạm vi biến – Global, Public, Private,…. Toàn cục, nội bộ…
8. API 🡪 Application Programming Interface.
9. Encode >< Decode.
10. Stringify >< Parse.
11. Callback Hell 🡪 Cần điều kiện của hàm trước đó để chạy hàm phía sau.
12. Chain 🡪 Object.Property.Method().Method().Property.Property…..

**OOP (Object Oriented Programming)**

1. Abstraction 🡪 Tính trừu tượng: Ẩn các chi tiết không cần thiết, cho cái nhìn tổng thể về object.
2. Enscapsulation 🡪 Tính đóng gói: Tạo các properties hoặc methods riêng cho object. (private)
3. Inheritance 🡪 Tính kế thừa: Tái sử dụng các properties hoặc methods của object cha cho con.
4. Polymorphism 🡪 Tính đa hình: Ghi đè lên các methods của object cha.
5. Instance 🡪 this
6. Prototype 🡪 Là các properties và methods được kế thừa
7. Constructor Function 🡪 new Function()
8. Private Properties 🡪 properties bị chặn không cho sửa trực tiếp. (#)

**HTML**

1. ID > Class > Tag

**CSS**

1. Selector 🡪 Cú pháp thứ tự để chon
2. Inline > Internal > External

**JAVASCRIPT**

1. Closure (js) 🡪 Hàm có thể truy cập biến của hàm khác.
2. DOM (js) 🡪 API liên kết giữa Js và browser html

**DOM**

1. *property:* document.querySelector(‘*Tên .class/#id/selector*’).*element*
2. *property:* document.querySelectorAll(‘*Tên .class/#id/selector*’).*element*
3. *property*: document.createElement(‘Tên *tag*’) – thêm một thẻ tag mới
4. *element*: textContent – nội dung text trong thẻ tag html
5. *element*: innerHTML – nội dung code html trong thẻ tag html­­­­­
6. *element*: insertAdjacentHTML – thêm nối tiếp nội dung code html
7. *element*: classlist – thêm/xóa class.
8. *element*: prepend/append – di chuyển vào đầu/cuối trong thẻ tag code html
9. Dom tree: nodes
10. NodeList: No update. Vd: document.querySelectorAll
11. HTMLCollection: Update. Vd: document.getElementsByTagName

**ARRAY**

1. Methods: slice, splice, reverse, concat, join.
2. Methods: foreach, map, filter, reduce.

**FUNCTION**

1. Regular/Declaration: functionName(parameter1, parameter2, parameter3){ *code* }
2. Expression: const fuctionName = function(parameter1, parameter2, parameter3) { *code* }
3. Arrow: functionName = (parameter1, parameter2, parameter3) => { code }
4. Callback function: Hàm được gán làm tham số của hàm khác

**CLASS (OOP)**

1. Object Class
2. Function Class
3. CS6 Class

**ASYNCRONOUS**

1. Callback.
2. Promise.
3. Async and Await.
4. Pending / Fulfilled >< Rejected. (3 types)
5. Resolve >< Rejec. (Promise)
6. Catch >< Then / Final (Promise)
7. Try >< Catch (Async Await)
8. Promise.all (Promise / Async Await)