Javascript

Style guide

Any fool can write code that a computer can understand
Good programmers write code that humans can understand.
“Martin Fowler”

Bất cứ thằng ngu nào cũng có thể viết code cho máy tính có thể hiểu. Lập trình viên giỏi là người viết code để cho người khác có thể hiểu.

What is problems with dirty code?

- Khó đọc
- Khó bảo trì
- Khó debug
- Khó phát triển (thêm, bớt tính năng …)
- Code xấu lôi kéo code xấu => chất lượng dự án đi xuống

Naming

50% of code quality

Rule 1: Chi tiết - rõ nghĩa

```
// bad
for (let i = 0; i < matrix.length; i++) {
    for (let j = 0; j < matrix[i].length; j++) {
        // do some thing
    }
}
// say what you mean - mean what you say
for (let col = 0; col < matrix.length; col++) {
    for (let row = 0; row < matrix[i].length; row++) {
        // do some thing
    }
}
const list = []; // bad name
const symbolList = []; // better
const symbols = []; // best
```

Rule 2: Không viết tắt

```
const d = 1; //// don't do that
const duration = 1;

let s = d / t; //// don't do that
let speed = distance / time;
```

Rule 3: Dễ phát âm ⇒ dễ nhớ , dễ search

```
let tutorialMgr; // bad
let tutorialManager;

let uiMgr; // bad
let uiManager;
```

Rule 4: Tên biến nên đủ ngắn

```
class Reel {
    constructor() {
        this._stackWildType = "C";
        this._stackWildSize = 7;
        this._firstIndexStackWild = 0;
        this._lastIndexStackWild = 0;
        this._stackWildNudge = 6;
        this._nudgeStep = 0;
    }
}
```

```
// using context to make it shorter
class Reel {
    constructor(){
        this._stackData = {
            type: "C",
            size: 7,
            firstIndex: 0,
            lastIndex: 0,
            nudgeStep: 0,
            nudge: 6
        };
    }
}
```

Dùng camelCase cho tên biến, hàm

```
let firstIndex = 0;
function doSomething() {
   // code
}

// don't do that
let SecondIndex = 1;
function DoTheOtherThing() {
        // code
}
```

Dùng danh từ với PascalCase cho tên class

```
class Card {};
// "Card.js"
class Item {};
// file have the same name with main
class
"Item.js"
// using camel Case for singleton file
"globalNetwork.js" //
```

Dùng UPPER_CASE cho biến constant

```
// using Upper Case for constants value
const MAX_STEP = 1000;

// always using ";" after statement
const EVENT_HIDE = 'event-hide';
```

Using right format

```
// open bracket "{" in the same line
function doThis() {
    // ...
}
// that is for "C", "C++", "C#"
function dontDoThatInJS()
{
    // ...
}
// you won't know what is wrong here
```

```
// you won't know what is wrong here
function getUser(id) {
    return // undefined
        {
        userName: "user166"
    };
}
if (isValid) {
    // else right after "}"
} else {

}
```

Chỉ dùng let khi có gán lại giá trị cho biến

```
// only using let if you will reassign it
let count = 0;
// and always using const for the others
const duration = 1;
let car = { type: "Fiat", model: "500", color: "white" };
const cars = ["Volvo", "Saab", "Fiat"];
```

Bắt đầu bằng is, can, has, have cho biến boolean

```
let isFinished = false;

let canFire = true;

let hasIceSkill = false;
```

Nên khai báo giá trị khởi tạo Khi khai báo biến

```
let firstName = “”;
let price = 0;
const myArray = [];
const myObject = {};
```

Function

Function

Luôn bắt đầu tên hàm bằng
một động từ

```
// always start by a verb
function doSomeThing() {
    // ...
}
```

Hàm chỉ làm một
và chỉ một việc duy nhất

```
// * do something or answer something, but not both.
// always have return value with "get..."
function getSomeThing() {
    // ...
    return { id: 1 };
}
// but do not return anything with "set..."
function setSomeThing() {
    // ...
}
```

Xử lý lỗi ở đầu hàm

```
playMusic(id, loop, volume) {
    if (this.sfxMap[id]) {
        let soundObj = this.sfxMap[id];
        if (soundObj.audioSource.node) {
            soundObj.audioSource.play();
            soundObj.audioSource.loop = loop;
            if (typeof volume === "number") {
                soundObj.audioSource.volume = volume;
            }
        }
        return soundObj.audioSource;
    } else {
        console.log(`do not have sfx: ${id}`);
        return null;
    }
}
```

```
playMusic(id, loop, volume) {
    const soundObj = this.sfxMap[id];
    if (!soundObj) {
        warn(`SOUND playSfx, do not have sfx: ${id}`);
        return null;
    }
    if (soundObj.audioSource.node) {
        soundObj.audioSource.play();
        soundObj.audioSource.loop = loop;
        if (typeof volume === "number") {
            soundObj.audioSource.volume = volume;
        }
       return soundObj.audioSource;
    }
}
```

Hàm nên hạn chế dưới 3 params

```
playParticleFly(startPos, endPos, duration = 1, easing, delay) {
        //
    }

playParticleFly(startPos, endPos, options) {
    let { duration = 1, easing, delay } = options;
    // easier to pass param now
}
```

Học cách dùng early return

```
function getSprite(num) {
    let spriteFrame;
    spriteFrame = this.normalLine;
    if (num === -1) {
        spriteFrame = this.line;
    } else if (num === -2) {
        spriteFrame = this.bigLine;
    } else if (num === 0) {
        spriteFrame = this.normalLine;
    } else if (num === 1) {
        spriteFrame = this.line;
    } else if (num === 2) {
        spriteFrame = this.bigLine;
    }
    spriteFrame = "default"
    return spriteFrame;
}
```

```
function getSprite(num) {
    if (num === 0) return this.normalLine;
    if (Math.abs(num) === 1) return this.line;
    if (Math.abs(num) === 2) return this.bigLine;
}
```

Chú ý không được che business logic

```
properties =  {
    listUi: [cc.Node]
}

function forceResetUI() {
    this._activeUI();
    this.listUi[2].active = false;
    // what the h*** you just hiding here?
}
```

```
properties = {
    exitBtn: cc.Node,
    infoBtn: cc.Node,
    settingBtn: cc.Node,
}

function forceResetUI() {
    this._activeUI();
    this.infoBtn.active = false;
}
```

Comment

We don’t really need that

Thay vì comment, tách code thành hàm nhỏ hơn

```
function playCoinFly(data) {
    const { value, startPos, endPos, delay, duration } = data;

    //init coin
    const coin = new Coin();
    coin.opacity = 0;
    coin.value = value;
    coin.zIndex = this.coins.length + 4;
    // ...

    //play coin fly
    //...
    tween(coin).fromTo(duration, { startPos }, { endPos });
}
```

```
function playCoinFly(data) {
    const { value } = data;
    const coin = this._initCoin(value);
    this._moveCoin(coin, data)
}
```

Bản thân code nói rõ nó làm gì

```
// check xem có đi tăng 2 được không?
if ((money > 15000000) && (dev.gender !== "Female")) {
    // ...
}

if (coTheDiTang2(dev)) {
    // ...
}
function coTheDiTang2(dev) {
    return money > 15000000 && (dev.gender !== "Female");
}
```

Comment chấp nhận được

```
// cung cấp thông tin
results.forEach(result => {
    //"2;2;2000004:2000000;4;BIG_"
    const dataSplit = result.split(";");
    const symbolId = dataSplit[0];
})
// cảnh báo anh em
this.reels.forEach((reel, col) => {
    //! // do not reverse matrix here.
    // already did it inside reel
});
```

Comment không nên

```
// ĐỪNG DÙNG COMMENT
// KHI CÓ THỂ DÙNG HÀM HOẶC BIẾN

/**
 * docs chỉ giành cho thư viện,
  hoặc file interface
 * @return the day of the month.
 */
getDayOfMonth() {
    return dayOfMonth;
}
```

```
// code cũ
// đã có git rồi, xóa đi.

// đừng đánh dấu lãnh thổ nữa
 ////////////////////////
```

Alway refactor

Always add extra time to refactor when estimate