Your JS code style matters!

by Nicu Micleușanu

Disclaimer

What follows is my personal opinion and I do not claim that this is the only right way to do things.

I will try to recommend what I consider to be correct based on my experience and will argument it.

If you do not agree with my position feel free to ignore it or start a constructive discussion on the topic without flaming and starting a “Holy War”.

Have a nice time at our JS meetup ;)

A bit about me

I am a code style nerd!

I get angry when I see it …

… when I see ugly code

dO

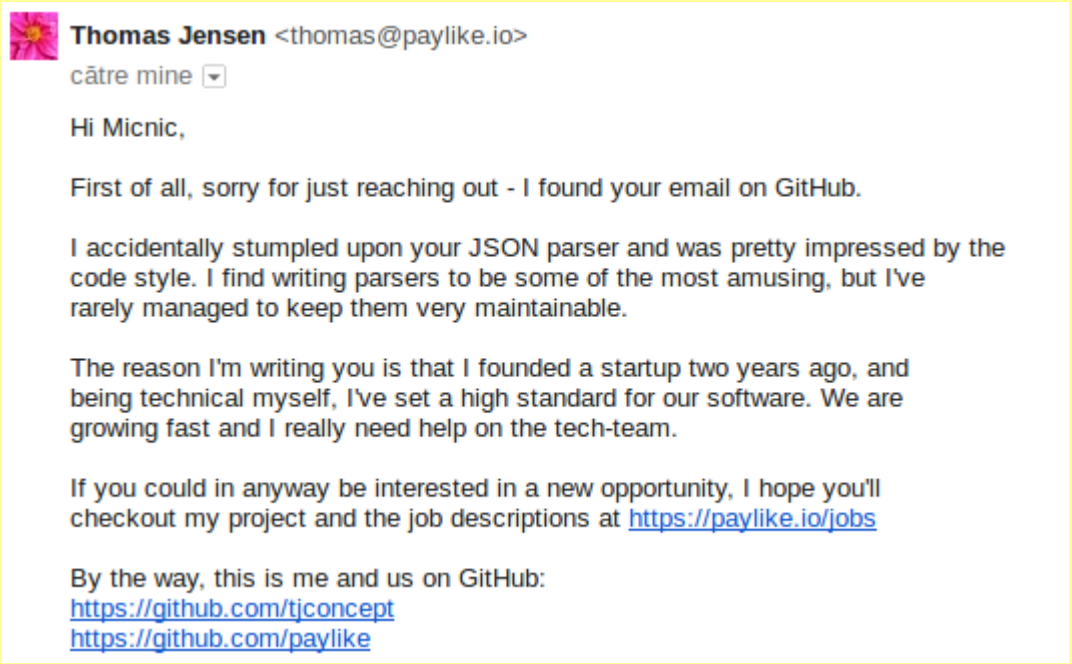
yOu

fEel iT?

My code in 2010

```
tags=/(\[((\*)|(\/?(anchor|b|center|code|i|pre|right|s|u|yt))|(((color=(#[0-9a-f]{6}|[a-z]+))|(

function mytmdinserttextoutside(text){
    params.setStringValue('state_data',text);
    window.opener.document.commandDispatcher.getControllerForCommand('cmd_insertText').QueryInt

function mytmdinserttext(text){
    if(e('editortextbox'))f('editortextbox');
    params.setStringValue('state_data',text);
    document.commandDispatcher.getControllerForCommand('cmd_insertText').QueryInterface(ci.nsIC

function mytmdinsertsmile(smile){
    smiles=mytmdpref('lastsmiles','c').split('.');
    found=false;
    for(i=0;i<smiles.length;i++)if(smiles[i]==smile)found=true;
    if(!found){
        for(i=smiles.length-2;i>=0;i--)smiles[i+1]=smiles[i];
        if(smile==':\'-(')smiles[0]=':\\\'-(';
        else smiles[0]=smile;
        mytmdpref('lastsmiles','c',smiles.join('.'));}
    mytmdinserttextoutside(smile);
    window.close();}

function mytmdfromclipboard(){
    if(!clip)return false;
    trans=cc['@mozilla.org/widget/transferable;1'].createInstance(ci.nsITransferable);
    if(!trans)return false;
    trans.addDataFlavor('text/unicode');
    clip.getData(trans,clip.kGlobalClipboard);
    str=new Object();
    strLength=new Object();
    trans.getTransferData('text/unicode',str,strLength);
    if(str)str=str.value.QueryInterface(ci.nsISupportsString);
    if(str)pastetext=str.data.substring(0,strLength.value/2);
    return pastetext;}
```

Interesting case



Thomas Jensen <thomas@paylike.io>
către mine

Hi Micnic,

First of all, sorry for just reaching out - I found your email on GitHub.

I accidentally stumpled upon your JSON parser and was pretty impressed by the code style. I find writing parsers to be some of the most amusing, but I've rarely managed to keep them very maintainable.

The reason I'm writing you is that I founded a startup two years ago, and being technical myself, I've set a high standard for our software. We are growing fast and I really need help on the tech-team.

If you could in anyway be interested in a new opportunity, I hope you'll checkout my project and the job descriptions at https://paylike.io/jobs

By the way, this is me and us on GitHub:
https://github.com/tjconcept
https://github.com/paylike

Tool time

Hey guys!

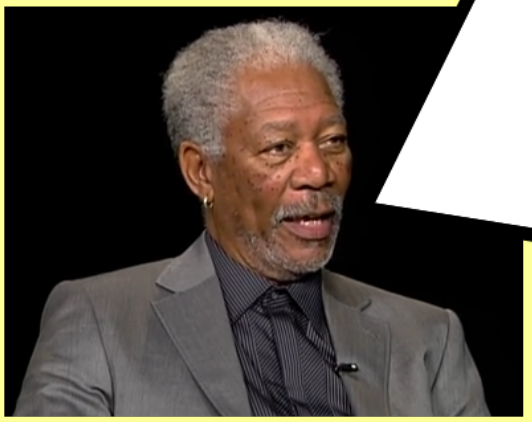
I came here to help with the JS code style :)

Let’s start with indentation and whitespace

Spaces vs Tabs

2 space vs 4 space indentation

And yes, don’t forget about consistency!



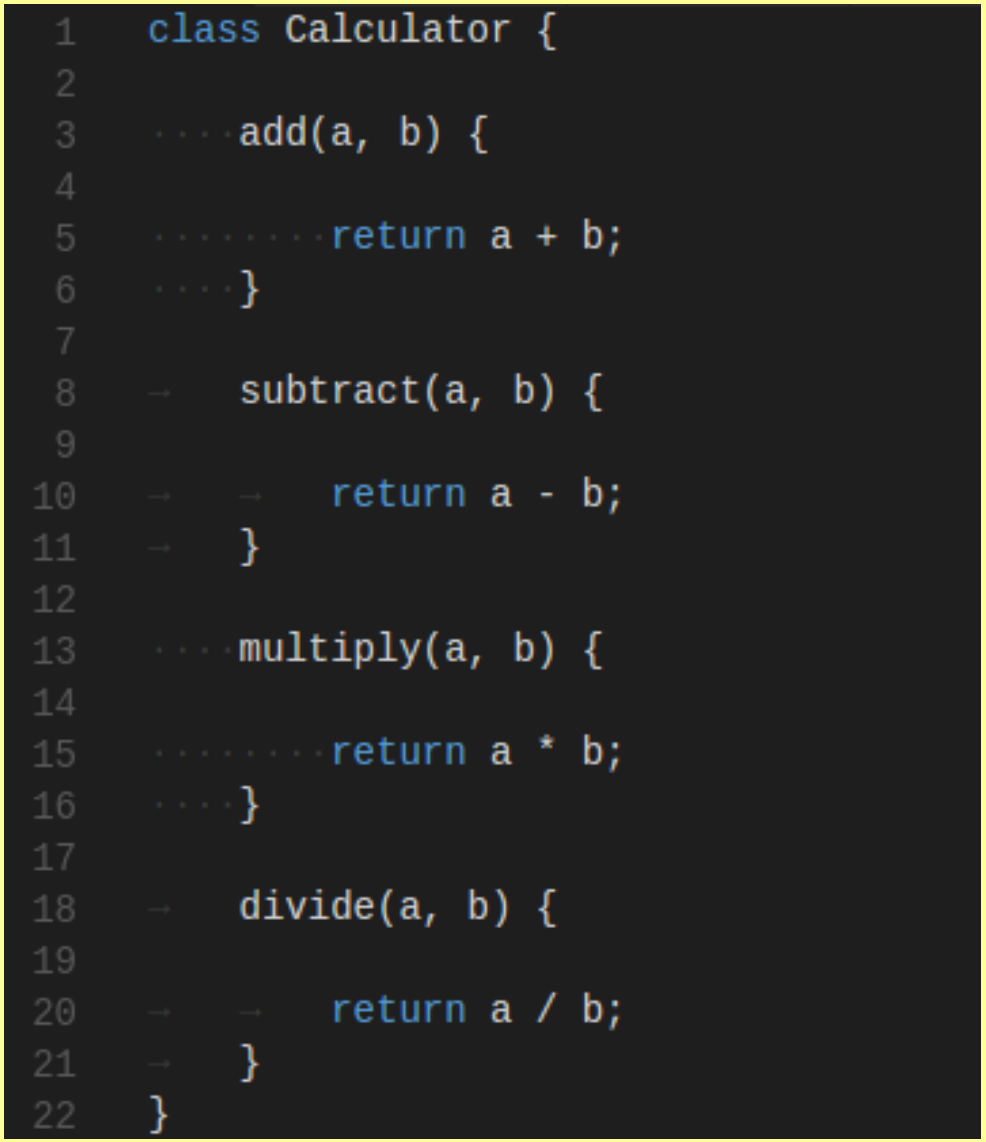
Morgan Freeman

Spaces	Tabs
Look the same everywhere	Compact for source file size
Do not need configuration in the editor	Flexible width
	Meant for indentation
	Faster to navigate using arrow keys
	Impossible to make a partial indentation

What is wrong with this code?

```
1   class Calculator {
2
3       add(a, b) {
4
5           return a + b;
6       }
7
8       subtract(a, b) {
9
10          return a - b;
11      }
12
13      multiply(a, b) {
14
15          return a * b;
16      }
17
18      divide(a, b) {
19
20          return a / b;
21      }
22  }
```

You can not see it until ...
… you do not render whitespace in your editor



ESLint Rules:

- indent (spaces / tabs)
- no-mixed-spaces-and-tabs (recommended)

- max-len (enforce tab size)

- no-irregular-whitespace (recommended)
- no-multi-spaces (fixable)
- no-trailing-spaces (fixable)
- no-whitespace-before-property (fixable)

- space-before-blocks (fixable)
- space-before-function-paren (fixable)
- space-in-parens (fixable)

...

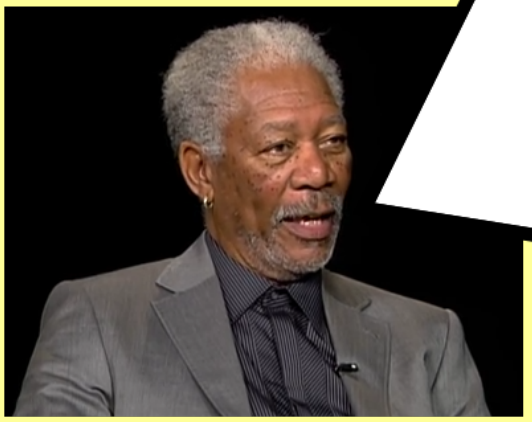
“Semicolon” vs “No semicolon”

BANG!

BANG!

Let’s talk about them

Let’s compare them side by side



Semicolon	No semicolon
Non-ambiguous	ASI (Automatic Semicolon Insertion)
Less linter configuration	Less to write
Friendly for C / C++ / C# / Java / … developers	Beginner friendly
No side effects	
More readable	

ESLint Rules:

- no-extra-semi (recommended)
- semi (fixable)

- semi-spacing (fixable)

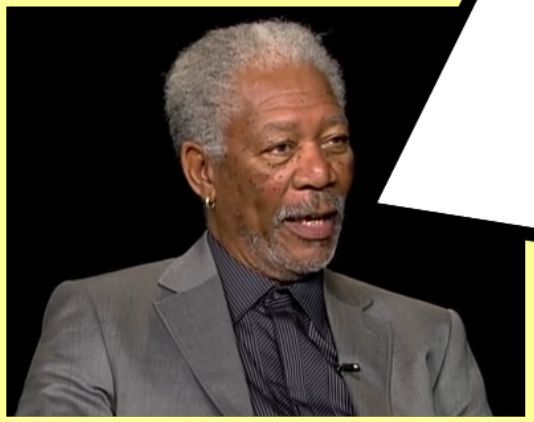
Naming,
this is a real problem

Use meaningful and pronounceable names for everything.
(ddmmyyyy vs currentDate)

Use same vocabulary everywhere.
(not user then client then customer)

Be verbose.
(l vs location)

Numeric values should be defined in constants.
(const secondInHour = 3600;)



Hi Guys!

Use ES6, it’s cool!

R.I.P. var, function

Say hello to your new friends let, const and arrow functions

Jason Statham

ESLint Rules:

- no-var (fixable)
- prefer-const (fixable)
- prefer-arrow-callback (fixable)

- arrow-spacing (fixable)
- arrow-parens (fixable)
- arrow-body-style (fixable)

Avoid bad parts!

==

A ? B : C

block-less statements

break / continue

with() statement

eval() and friends

switch() statement

for() statement

ESLint Rules:

- eqeqeq
- no-with
- no-eval
- no-continue
- curly (fixable)

Write a lot of comments!

Be verbose in your comments but not too much.

Do not keep code in the comments, only words.

I use one line comments.

ESLint Rules:

- capitalized-comments (fixable)
- spaced-comment (fixable)
- line-comment-position
- lines-around-comment (fixable)

- no-inline-comments
- require-jsdoc

Code recommendations

```
class ClassName extends ParentClassName {

    constructor(arg1, arg2) {

        super(arg1, arg2);
    }

    // This method does some magic with the provided argument
    method(arg1) {

        const const1 = arg1;
        const const2 = this.property;

        let var1 = const1 * const2;

        // Set max value to 100
        if (var1 > 100) {
            var1 = 100;
        }

        return var1;
    }

    // This method multiplies by 2 the provided argument
    static staticMethod(arg1) {

        return arg1 * 2;
    }
}
```

Hello my friends,
here I am ;)

Let’s talk about functions

Function arguments:
fewer are better (max 4)

Functions should do one
single thing

Chuck Norris

I guess you know you should not overwrite global objects prototypes

Externalize duplicate code to common functions!

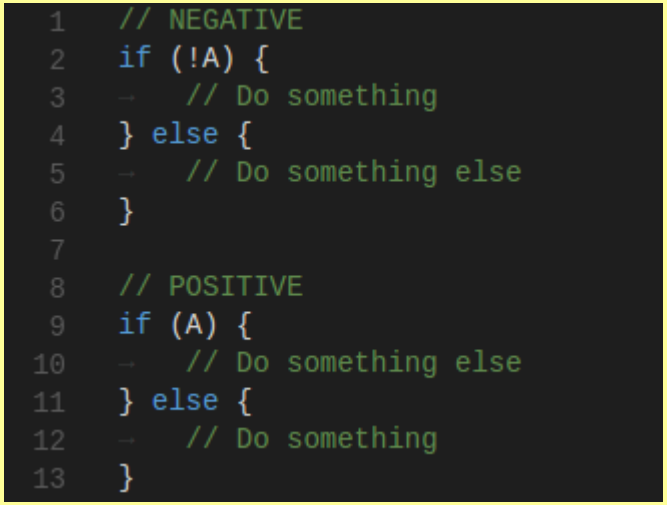
Favor functional programming

Encapsulate (complex) conditions in function

Be positive

and remember
code is for humans
not for machines!

How to be positive



```
// NEGATIVE
if (!A) {
    // Do something
} else {
    // Do something else
}

// POSITIVE
if (A) {
    // Do something else
} else {
    // Do something
}
```

```
this.isTheEnd();
```