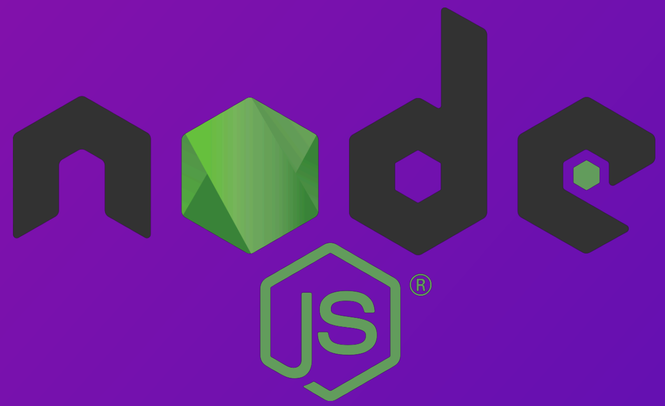
Code Convention & Unit Test In Nodejs

nguyentvk@uiza.io

- Introduction
- Line Length
- Indentation
- Semicolons
- White space
- Unit Test
- Mocha
- Chai
- Demo
- Q&A
- Reference



Line length

No one likes to read a long horizontal line of code.

It’s best practice to split them up and limit the length of your lines.

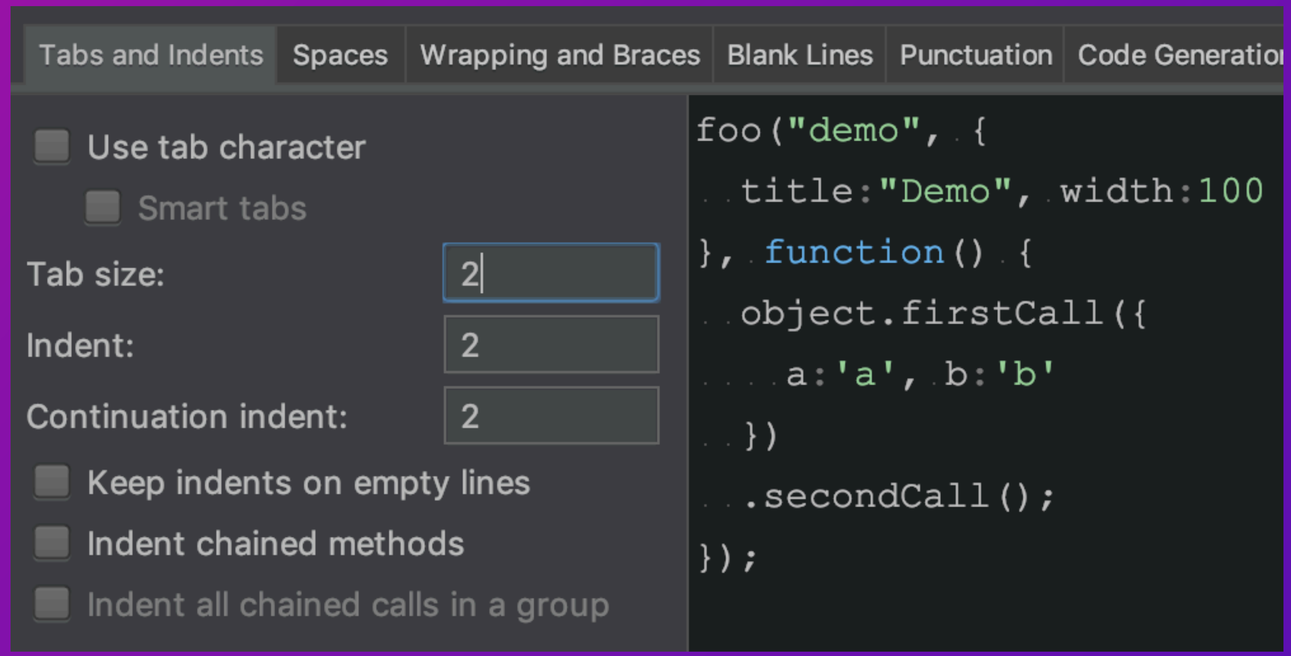
The maximum line length should be agreed upon at the team-level. It’s usually 80 or 120 characters.

Using string template for long statement

Indentation

Use 2 spaces = 1 tab for indenting your code

never mix tabs and spaces



Semicolons

Bad	Good
for(;;) // ✗ avoid	for(var i=0; i< n; i++)
for (let i = 0 ;i < items.length ;i++) {...} // ✗ avoid	for (let i = 0; i < items.length; i++) {...} // ✓ ok
case 'foo': doSomething(); break;	case 'foo': doSomething(); break;
	Put a ; semicolon at the end of each statement let keys = []; let values = [];
if(something){ … };	if(something){ … } //✓ ok

White space

BAD	GOOD
Must clean up any trailing white space in your JS file	Must clean up any trailing white space in your JS file
Whitespace not allowed at end of line.	Whitespace not allowed at end of line.
Add white space after code block	Let a = ‘value a’; Let b = { b : ‘value b’}; if(a === b)

Single quotes

BAD	GOOD
var foo = “bar”; var notOk = “String ‘double’ quotes”	var foo = ‘bar’; var ok = 'String contains "double" quotes’
Using string template	`string text` `string text ${expression} string text`
`let a = 'string text line 1\n' +` `'string text line 2';`	`let a = `string text line 1` `string text line 2`;`
`console.log('Fifteen is ' + (a + b) + ' and\nnot '` `+ (2 * a + b) + '.');`	`console.log(`Fifteen is ${a + b} and` `not ${2 * a + b}.`);`

Variable Declarations

BAD	GOOD
const a1 = new Array(x1, x2, x3);	*const a1 = [x1, x2, x3];*
let a = new Object();	*let a = {};*
let a, b, c;	*Let a;* *Let b;* *Let c;*
Not use const for variable if you reassign	*Const a = ‘aaaa’; //then* *a = ‘bbb’*

Naming

BAD	GOOD	Comment
`function SendMessage()` `function sendmessage()` `function send_message()`	`function sendMessage(){` `}`	Method names are written in `lowerCamelCase`
`var u = new user({` `  name: 'Bob Parr'` `});`	`Class UpperCamelCase` `var user = new User({` `  name: 'Bob Parr'` `});`	Follow the camel case convention
`// Constants` `const numberState = 5;` `const ENUMCONSTANT: ‘value'`	`// Constants` `const NUMBER_STATE = 5;` `const ENUM_CONSTANT: ‘value'`	Constant names use `CONSTANT_CASE`: all uppercase letters, with words separated by underscores

Functional

Write small functions (15 - 20 line)
Keep your functions short. A good function fits on a slide that the people in the last row of a big room can comfortably read.

Return early from functions
To avoid deep nesting of if-statements, always return a function's value as early as possible.

BAD	GOOD	Comment
`function isPercentage(val) {` `  if (val >= 0) {` `  if (val < 100) {` `  return true;` `  } else {` `  return false;` `  }` `  } else {` `  return false;` `  }` `}`	`function isPercentage(val) {` `  if (val < 0)` `  return false;` `  if (val > 100)` `  return false;` `  return true;` `}`	`Return early; To avoid deep nesting of if-statements, always return a function's value as early as possible.`

Operation

BAD	GOOD	
if(a != b) if(a == b)	If(a !== b) if(a === b)	Always use === instead of ==
if(a === null ll a === undefined ll a === {} ll a===[])	import(‘lodash’) if(_.isEmpty(a))	Use lodash
if(a === ‘a’ ll a === ‘b’ ll a === ‘c’ ll a === ‘d’)	if(_.includes([‘a’,’b’,’c’,‘d’], a)	Use lodash
if(a)	if(!!a)	
While (a = checkSomething())	while((a = checkSomething()))	
Var tmp; if(a) tmp = ‘valueA’ Else tmp = ‘valueB’	var tmp = a ? 'valueA' : ‘valueB'	

Error Handler

BAD	GOOD	Comment
Throw `string message error`	Throw new Error(`message error`)	Always create a new Error object with your message. Don't just return a string message to the callback
try{ something(); }catch(error){ throw error; }	try{ something(); }catch(error){ console.error(error); callApiLogging(); alertSlackError(); throw error; }	Always logging error
throw 'error' // ✗ avoid	throw new Error('error') // ✓ ok	

Comment

BAD	GOOD	Comment
	/** * Description: get profile user * Input * Output */	Comment multi line
	// check user is active	Comment in line
	/* comment in code block */	Commnent in code block

Nice async code

‘Await’ only appear in async function

Always await a Promise

Turn callback into a Promise

```
async handler(ctx) {
  try {
    await new Promise((resolve, reject) => {
    });
    await functionReturnPromise();
  } catch(error) {
    console.log(error)
    throw error;
  }
  return {message:'done'}
}
```

```
let mixCbPromise = (url) => {
  return new Promise( (resolve, reject) => {
    request.get(url,(error, response, data)=>{
      if (error){
        console.error(error);
        return reject(error);
      }
      resolve(data);
    });
  });
};
```

Testing

Unit test: function, produce, class, method, property

Integration test: communication between units include
Structure
Functional
Performance
Stress

System testing
…

Unit test

Mocha is a feature-rich JavaScript test framework running on Node.js and in the browser, making asynchronous testing simple and fun

Chai is a assertion library for node and the browser that can be delightfully paired with any javascript testing framework.

Mocha

```
npm install --save-dev mocha
```

```
var assert = require('assert');
describe('Array', function() {
  describe('#indexOf()', function() {
    it('should return -1 when the value is not present', function() {
      assert.equal([1,2,3].indexOf(4), -1);
    });
  });
});
```

ASSERTIONS

should.js - BDD style shown throughout these docs

expect.js - expect() style assertions

chai - expect(), assert() and should-style assertions

better-assert - C-style self-documenting assert()

unexpected - “the extensible BDD assertion toolkit”

Mocha

- ASYNCHRONOUS CODE
- WORKING WITH PROMISES
- USING ASYNC / AWAIT
- SYNCHRONOUS CODE
- ARROW FUNCTIONS
- HOOKS
- DYNAMICALLY GENERATING TESTS
- TIMEOUTS

https://mochajs.org/

Chai

Chai has several interfaces that allow the developer to choose the most comfortable.

Download Chai

4.2.0 / 2018-09-25

for Node

Another platform? Browser Rails

The chai package is available on npm.

```
$ npm install chai
```

View Node Guide

Issues | Fork on GitHub | Releases | Google Group | Build Status

Should

```
chai.should();

foo.should.be.a('string');
foo.should.equal('bar');
foo.should.have.lengthOf(3);
tea.should.have.property('flavors')
  .with.lengthOf(3);
```

Visit Should Guide

Expect

```
var expect = chai.expect;

expect(foo).to.be.a('string');
expect(foo).to.equal('bar');
expect(foo).to.have.lengthOf(3);
expect(tea).to.have.property('flavors')
  .with.lengthOf(3);
```

Visit Expect Guide

Assert

```
var assert = chai.assert;

assert.typeOf(foo, 'string');
assert.equal(foo, 'bar');
assert.lengthOf(foo, 3)
assert.property(tea, 'flavors');
assert.lengthOf(tea.flavors, 3);
```

Visit Assert Guide

Demo Unit Test

Demo Unit Test

```
  Test heal_check
    ✓ returns status 200
    ✓ returns message "ok"

  Test url
    ✓ url = http://localhost:3000/ returns status 200 (2227ms)
    ✓ url = http://localhost:3000/features/video-on-demand returns status 200 (1242ms)
    ✓ url = http://localhost:3000/features/live-streaming returns status 200 (1070ms)
    ✓ url = http://localhost:3000/features/auto-scale-platform returns status 200 (1012ms)
    ✓ url = http://localhost:3000/features/data-analytics returns status 200 (1035ms)
    ✓ url = http://localhost:3000/solutions/social-communication returns status 200 (886ms)
    ✓ url = http://localhost:3000/solutions/ondemand returns status 200 (961ms)
    ✓ url = http://localhost:3000/solutions/digital-media returns status 200 (943ms)
    ✓ url = http://localhost:3000/solutions/education returns status 200 (863ms)
    ✓ url = http://localhost:3000/pricing returns status 200 (971ms)
    ✓ url = http://localhost:3000/careers returns status 200 (947ms)
    ✓ url = http://localhost:3000/about returns status 200 (848ms)
    ✓ url = http://localhost:3000/contact/sales returns status 200 (913ms)
    ✓ url = http://localhost:3000/terms returns status 200 (955ms)
    ✓ url = http://localhost:3000/support-policy returns status 200 (832ms)
Stop server after test done

  End test: all done
    ✓ End test: all done


  21 passing (16s)
```

Q&A

Reference

npm-coding-style
https://docs.npmjs.com/misc/coding-style.html

Google JavaScript Style Guide
https://google.github.io/styleguide/jsguide.html

JavaScript Standard Style
https://standardjs.com/rules.html

NodeJS Coding Style Guidelines
https://blog.zipboard.co/nodejs-coding-style-guidelines-5386b7d57032

Reference

https://en.wikipedia.org/wiki/Software_testing

https://en.wikipedia.org/wiki/Software_testing#Operational_acceptance_testing

https://mochajs.org

https://www.chaijs.com/

Thank **You**