



# How to program in Javascript

The best style

Alejandro Peraza González – Nerea Rodríguez Hernández

# JSDOC

Is an API documentation generator for JavaScript.

The JSDoc tool will scan your source code and generate an HTML documentation website for you.



## How to use it?



### To install:

```
sudo apt install jsdoc-toolkit
```

### To run it:

```
jsdoc <filename>.js
```

### If you have your code in other directory:

```
jsdoc -d=<directoryname> <filename>.js
```

# Featured options.

## **-c <value>, --configure <value>**

The path to a JSDoc configuration file. Defaults to `conf.json` or `conf.json.EXAMPLE` in the directory where JSDoc is installed.

## **-d <value>, --destination <value>**

The path to the output folder for the generated documentation. For JSDoc's built-in Haruki template, use `console` to dump data to the console. Defaults to `./out`.

## **--match <value>**

Only run tests whose names contain `value`.

## **u <value>, --tutorials <value>**

Directory in which JSDoc should search for tutorials. If omitted, no tutorial pages will be generated. See the tutorial instructions for more information.

## **--verbose**

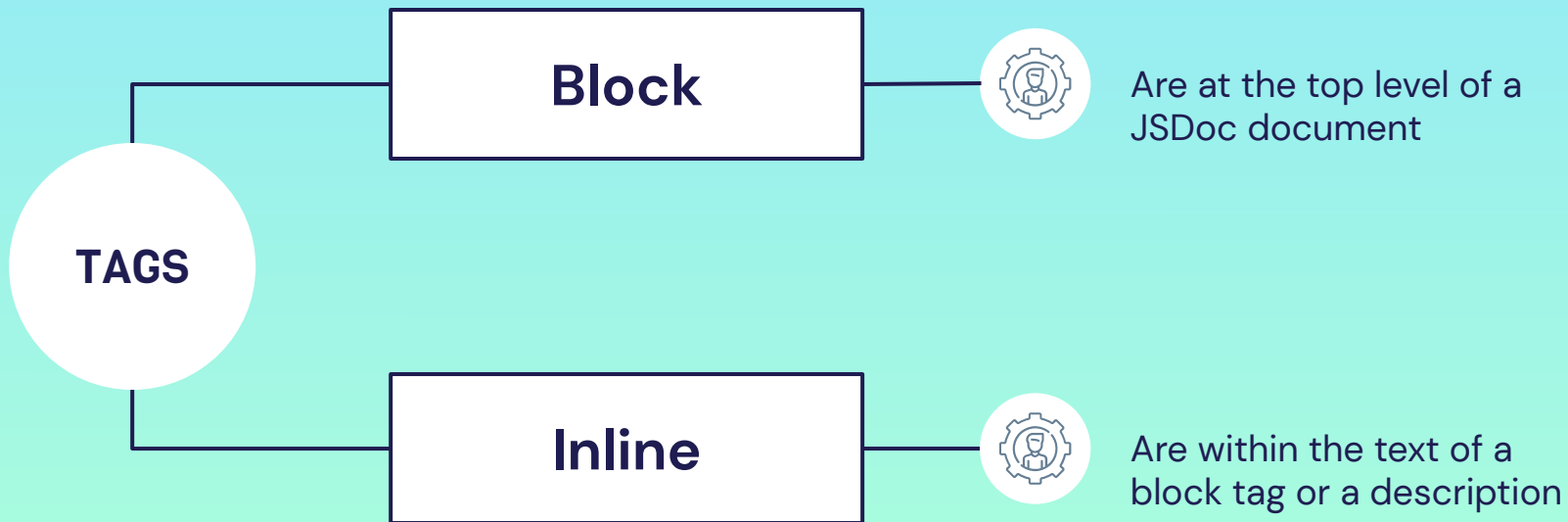
Log detailed information to the console as JSDoc runs. Defaults to `false`.

## **-t <value>, --template <value>**

The path to the template to use for generating output. Defaults to `templates/default`, JSDoc's built-in default template.

# • Tags Block vs Inline

Both types are invoked using @tag



## Tags that we will use the most in practices

- @author
- @class
- @classdesc
- @constant
- @constructs
- @default
- @example
- @exports
- @name
- @param
- @private
- @protected
- @public
- @requires
- @returns
- @see
- @summary
- @tutorial



# Coding Style

Our code must be as clean and easy to read as possible.



# Good practices

## Curly Braces

If (number < 0) alert ('Power  $\{number\}$  is not supported');

**Don't just use one letter to declare variables.**

let n = 'Alejandro';

**Use a meaningful name for the variable.**

let namePerson = 'Nerea';





# Good practices

## Line Length

To have a good practice we will avoid having a long line of code.

```
1 | if (id === 123 && moonPhase === 'Waning Gibbous' && zodiacSign === 'Libra') {  
2 |   letTheSorceseyBegin();  
3 | }
```



Do not do this!

Do this instead!



```
1 | if (  
2 |   id === 123 &&  
3 |   moonPhase === 'Waning Gibbous' &&  
4 |   zodiacSign === 'Libra'  
5 | ) {  
6 |   letTheSorceseyBegin();  
7 | }
```

# Good practices

## Indents

There are two types of indents:

- Horizontal indents: 2 ~~or 4~~ spaces.

```
1  function pow(number, power) {  
2      let result = 1;  
3  
4      for(let i = 0; i < power; i++) {  
5          result *= number;  
6      }  
7      return result;  
8  }
```

- Vertical indents: empty lines splitting code into logical blocks.

# Good practices

## Semicolons

A semicolon should be present after statement, even if it could possibly be skipped.

## Nesting Levels

Try to avoid nesting code too many levels deep.

## Function Placement

If you are writing several “helper” functions and the code that uses them, there are three ways to organize the function:

1. Declare the functions above the code that uses them.
2. Code first, then functions.
3. Mixed: a function is declared where it's first used.

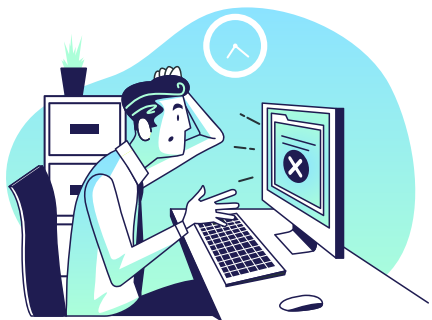


# ESLint

Is a linter that examines JavaScript, following standards customizable by the programmer. It is fully configurable and you can indicate which rules you want to use and which you prefer to ignore.



# Install and run



## Install

```
npm install --save-dev eslint
```



## Run

```
npx eslint
```

We must have the latest version of NPM

# Setting

For the initial configuration of the ESLint we will run:

**`npm eslint --int`**

This will start a wizard to configure ESLint in our project.

A few questions for configurations will be displayed.



### How would you like to use ESLint?

1. To check syntax only
2. To check syntax and find problems
3. To check syntax, find problems and enforce code style

01

### Where does your code run?

1. Browser
2. Node
3. Both

04

### What type of modules does your project use?

1. JavaScript modules (import / export)
2. CommonJS (require/exports)

02

### What format do you want your config file to be in?

JavaScript, YAML and JSON

05

### Which framework does your project use?

1. React
2. Vue.js
3. None of these

03

### Would you like to instal them now with npm?

Automatically install the necessary NPM packages

06

- Use a popular style guide

Promoters	GitHub	Package name
<u><a href="#">Airbnb</a></u>	<u><a href="#">@airbnb/javascript</a></u>	eslint-config-airbnb
<u><a href="#">StardJS</a></u>	<u><a href="#">@standard/standard</a></u>	eslint-config-standard
<u><a href="#">Google</a></u>	<u><a href="#">@google/eslint-config-google</a></u>	eslint-config-google



# Answer questions about style

**What style of indentation do you use?**

Tabulators or spaces.  
rules.indent

**What quotes do you use for strings?**

Double quotes or single quotes for strings  
rules.quotes.

**What line endings do you use?**

Windows or Linux  
rule.linebreak-style

**Do you require semicolons?**

Use semicolon or not use it.  
rule.semi



## Inspect your JavaScript file(s)



This option inspects all our JavaScript files in our project to determine which style guide should be used.

# Configuration format

What format do you want your config file to be in?

## 2. YAML

Will generate a configuration file  
.eslintrc.taml or eslintrc.yml



## 1. JavaScript

Will generate a configuration  
file .eslintrc.js



## 3. JSON

It will generate a configuration file .eslintrc.json.  
This format does not support comments but with  
the ESLint tool it will allow us to support comments  
of the type / \* \* /

**It's free to code  
with style**



# THANKS!



Alejandro Peraza González

[alu0101211770@ull.edu.es](mailto:alu0101211770@ull.edu.es)

[GitHub](#)



Nerea Rodríguez Hernández

[alu0101215693@ull.edu.es](mailto:alu0101215693@ull.edu.es)

[GitHub](#)

