

Style Guides



Universidad
de La Laguna

Introduction

- What does this function do?

```
const badFunc = (a,b) => {let r = 1, i = 0; for(;i<b;i++)r*=a;return r};
```

What are we going to see?

- Coding Style
- Google Style Guide
- ESLint
- JSDoc

Coding Style (I) - Functions



```
// function declarations
function createElement() {
  ...
}

function takeElement(elem) {
  ...
}

function dropElement(elem) {
  ...
}

// the code which uses them
let elem = createElement();
takeElement(elem);
dropElement(elem);
```



```
// the code which uses the functions
let elem = createElement();
takeElement(elem);
dropElement(elem);

// --- helper functions ---
function createElement() {
  ...
}

function takeElement(elem) {
  ...
}

function dropElement(elem) {
  ...
}
```



Coding Style (I) - Functions



```
function main() {  
  const myArgs = processArgs();  
  
  isLowercase(myArgs) ? console.log('La letra es minúscula')  
    : console.log('La letra es mayúscula');  
  
  isVowal(myArgs) ? console.log('La letra es una vocal')  
    : console.log('La letra es una consonante');  
}  
  
if (require.main === module) {  
  main();  
}
```

```
const isVowel = function(letter) {  
  switch (letter) {  
    case 'a':  
    case 'e':  
    case 'i':  
    case 'o':  
    case 'u':  
    case 'A':  
    case 'E':  
    case 'I':  
    case 'O':  
    case 'U':  
      return true;  
    default:  
      return false;  
  }  
};
```



Coding Style (II) – Length of a line



```
let github = "GitHub, Inc. is a provider of Internet hosting for software development and version control using Git. It offers the distributed version control and source code management functionality of Git, plus its own features."
```



```
let github = `
```

```
    GitHub, Inc. is a provider of Internet  
    hosting for software development and  
    version control using Git. It offers  
    the distributed version control and  
    source code management functionality  
    of Git, plus its own features.
```

```
`
```



Coding Style (III) – Indentation



```
function pow(x, n) {  
  let result = 1;  
  for (let i = 0; i < n; i++) {  
    result *= x;  
  }  
  return result;  
}
```



```
function pow(x, n) {  
  let result = 1;  
  
  for (let i = 0; i < n; i++) {  
    result *= x;  
  }  
  
  return result;  
}
```



Coding Style (IV) – If else



```
if (n < 0) {console.log(`This is not well done`);}
```

```
if (n < 0)  
  console.log(`Easy to have errors`);
```




```
if (n < 0) console.log(`This is acceptable, if it is short`);
```


```
if (n < 0) {  
  console.log(`This is a work of art`);  
}
```



Coding Style (V) – Variables



```
var a = "";  
const auxFunction = function(param1, param2) {  
  var b = param1 * param2;  
  a = `The product between ${param1} and ${param2} is ${b}`;  
}
```



```
const productString = function(number1, number2) {  
  let result = number1 * number2;  
  return `The product between ${number1} and ${number2} is ${result}`;  
}
```

Google Style Guide

- Created by Google to reinforce their coding standards
- Covers aesthetics as well as conventions and code standard
- The style guide is very extensive, so it is not possible to cover everything

Google Style Guide – Basic Styling

- Non-ASCII characters are permitted

```
const units = '\u03bcs';
```



```
const units = 'μs';
```



- Files should always be encoded in UTF-8
- Filenames are allowed to include `-` and `_` as special character.

Google Style Guide – Naming

- Variables and Functions
 - Written in lowerCamelCase
- Constants
 - Written in CONSTANT_CASE
- Classes and Enums
 - Written in UpperCamelCase
- Always use concise wording e.g., `errorCount` instead of `cErr`

Google Style Guide - Formatting

- 2 more spaces per block
- Horizontal Alignment is discouraged, but permitted
- General line limit: 80

```
{  
  tiny: 42, // this is great  
  longer: 435, // this too  
};  
  
{  
  tiny:    42,  // permitted, but future edits  
  longer: 435, // may leave it unaligned  
};
```

Ln 12, Col 5 Spaces: 2 UTF-8 LF

- In VSCode the Indent can be set at the bottom right corner

Google Style Guide – General Styling

- Arrow Functions are preferred

```
const moduleLocalFunc = (numParam, strParam) => numParam + Number(strParam);
```

- Use single quotes, when possible
- Switch-Case statements need a default case
- If possible, use for-of loops

```
switch (input) {  
  case 1:  
  case 2:  
    prepareOneOrTwo();  
    // fall through  
  case 3:  
    handleOneTwoOrThree();  
    break;  
  default:  
    handleLargeNumber(input);  
}
```

Google Style Guide - Equality

- Use the `===` / `!==` operator, except for one case:
Catching null and undefined values in one case

```
if (someObjectOrPrimitive == null) {  
    // Checking for null catches both null and undefined for objects and  
    // primitives, but does not catch other falsy values like 0 or the empty  
    // string.  
}
```

Google Style Guide – Eval()

- Do NOT use eval or the Function(...String) constructor
- Both are always a security risk

```
const ADMIN_PASSWORD = 'superSecretPassword';  
  
const simpleCalc = (userInput) => {  
  | console.log(eval(userInput));  
};
```


Google Style Guide - Imports

```
import '../directory/file.js';  
  
import * as bigAnimals from './biganimals.js';
```

- Always add the file extension
- Do NOT import a file multiple times
- When doing named imports use lowerCamelCase

ESLint

- Helps to follow style guide rules
- Can automatically format code
- NOT a replacement for knowing style guides



ESLint: Installation

Global install

- ``npm install -g eslint``
- Removes need to install for every project
- It is possible to create default ruleset for every project

Project install

- ``npm install eslint --save-dev``
- Easier to install for new people in the project
- All files in one folder

ESLint: Initialisation

- ``npm init @eslint/config``

```
usuario@Ubunto-18-PAI-1:~/test$ npm init @eslint/config
? How would you like to use ESLint? ...
  To check syntax only
  ▶ To check syntax and find problems
    To check syntax, find problems, and enforce code style
```

- For this class: To check syntax, find problems, and enforce code style

ESLint: Initialisation

```
usuario@Ubunto-18-PAI-1:~/test$ npm init @eslint/config
✓ How would you like to use ESLint? · style
? What type of modules does your project use? ...
  ▶ JavaScript modules (import/export)
    CommonsJS (require/exports)
    None of these
```

- In this class: JavaScript modules

ESLint: Initialisation

```
? Which framework does your project use? ...  
  React  
  Vue.js  
▶ None of these
```

- In this class: Non of these

```
? Does your project use TypeScript? ▶ No / Yes
```

- In this class: No

ESLint: Initialisation

? Where does your code run?
✓ Browser
✓ Node

- In this class: Node

ESLint: Initialisation

```
? How would you like to define a style for your project?  
▸ Use a popular style guide  
Answer questions about your style
```

- In this class: Use a popular style guide

```
? Which style guide do you want to follow? ...  
Airbnb: https://github.com/airbnb/javascript  
Standard: https://github.com/standard/standard  
▸ Google: https://github.com/google/eslint-config-google  
XO: https://github.com/xojs/eslint-config-xo
```

- In this class: Google

ESLint: Initialisation

```
? What format do you want your config file to be in? ...  
▸ JavaScript  
  YAML  
  JSON
```

- In this presentation: JavaScript

```
Checking peerDependencies of eslint-config-google@latest  
Local ESLint installation not found.  
The config that you've selected requires the following dependencies:  
  
eslint-config-google@latest eslint@>=5.16.0  
? Would you like to install them now with npm? ▸ No / Yes
```

- Select yes

.eslintrc

- This file is used to change eslint settings
- It is possible to add, remove or modify rules, set style guides etc.

```
'rules': {  
  'max-len': ['error', {'code': 80, 'ignoreComments': false}],  
},
```

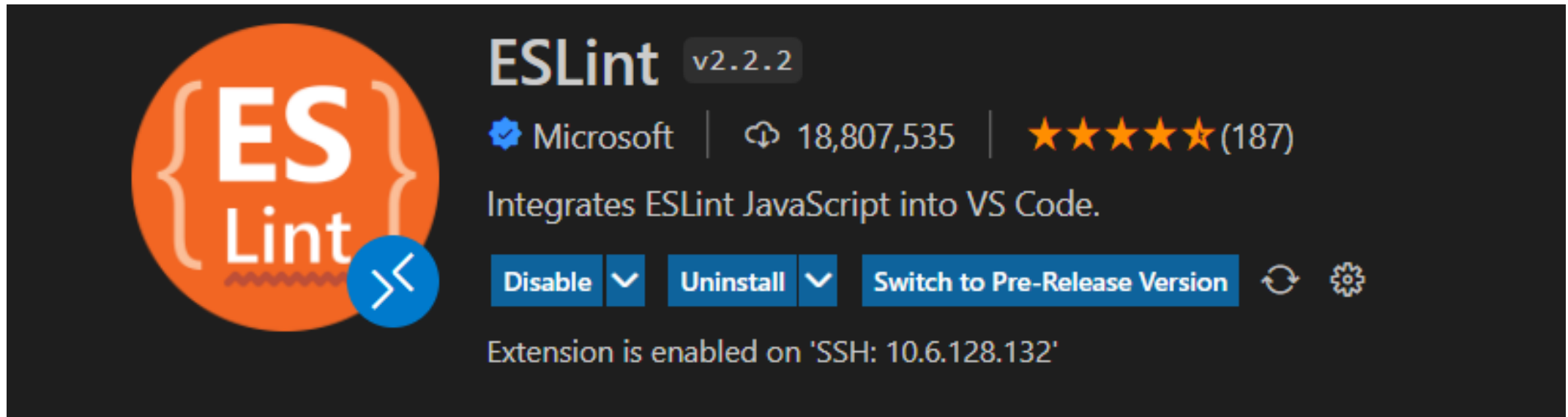
.eslintignore

- This file is used to ignore files from eslint
- It basically works like a .gitignore file

```
1 // For the VSCode ESLint Plugin this file needs to be in root
2 src/eslint/ignored.js
```

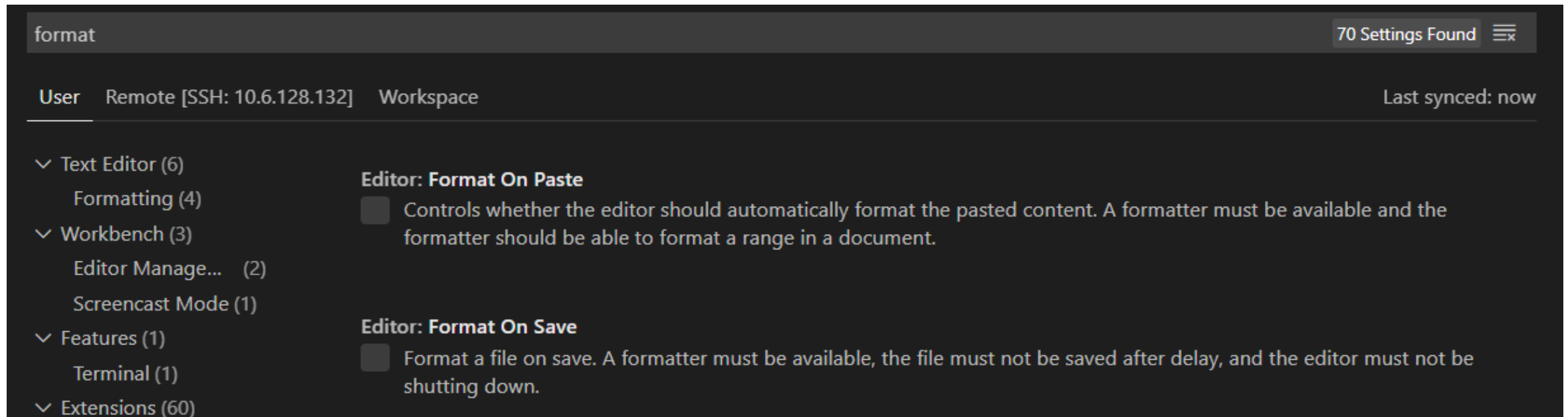
VSCode and ESLint

- Install the ESLint Extension



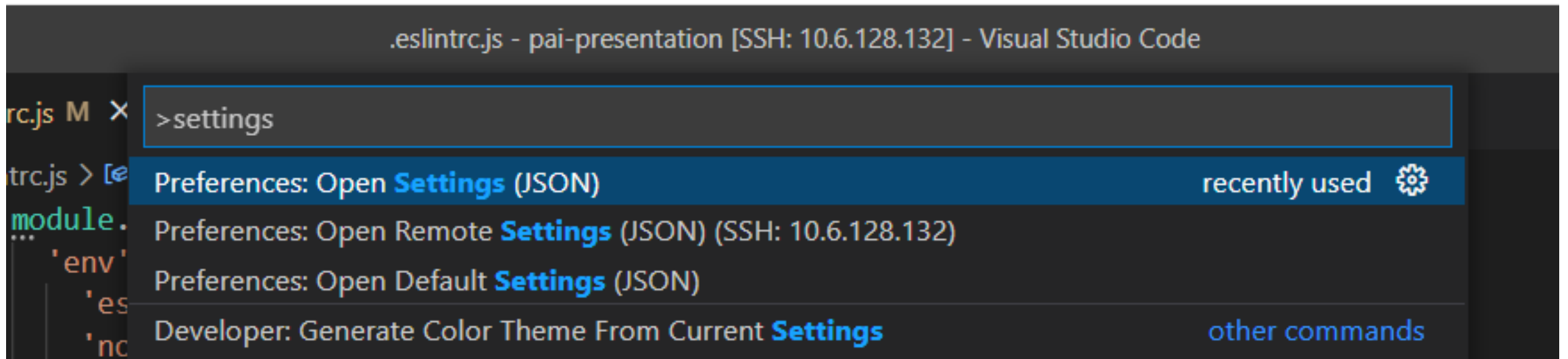
VSCode and ESLint

- Disable Autoformat
 - File -> Preferences -> Settings -> type 'format' and disable Format on Paste and Format on Save



VSCode and ESLint

- Press `F1` and enter settings to open the Settings in JSON format



- Important: Open the local file, not the remote one

VSCode and ESLint

```
"eslint.alwaysShowStatus": true,  
"editor.codeActionsOnSave": {  
  "source.fixAll.eslint": true  
},  
"eslint.run": "onSave",  
"eslint.options": {  
  "resolvePluginsRelativeTo": "/home/usuario/node_modules"  
},
```

- 1: shows indication of the bottom of the IDE
- 2-4: formats automatically on save
- 5: when should eslint run: onSave or onType
- 6-8: set path to style guide, only necessary on global installs

JSDoc – What is it?



- Documentation tool.
- Free software.
- Produce documentation accessible in formats like HTML.
- Quite similar to Javadoc.
- Same purpose as other used tools in the degree as Rdoc or Doxygen.

JSDoc – How to use it

- Installation

- \$ npm install -g jsdoc -> Globally
- \$ npm install --save-dev jsdoc -> Locally (package.json)

- Development

- Use JSDoc

JSDoc – How to use it

- Installation
- Development
 - Document your code
 - Write the code
- Use JSDoc

JSDoc – Headers



```
/**
 * Universidad de La Laguna
 * Escuela Superior de Ingeniería y Tecnología
 * Grado en Ingeniería Informática
 * Programación de Aplicaciones Interactivas*
 *
 * @author Jorge Hdez. Batista
 * @since Feb 28 2022
 * @desc Documentation
 *   This is the way we should comment a header, so all the information is added.
 *   During this file, you will find an example of how we can use the tool JSDoc in the most important structures
 *
 * @see {@link https://jsdoc.app}
 *
 */
```

JSDoc – Variables



```
/** @const
    @type {string}
    @default
 */
const RED = 'FF0000';
```

```
/**
 * A number, or a string containing a number.
 * @typedef {(number|string)} NumberLike
 */

/**
 * Set the magic number.
 * @param {NumberLike} x - The magic number.
 */
function setMagicNumber(x) {
}
```

JSDoc – Functions



```
/**
 * @throws {InvalidArgumentException} If empty string
 * @param {String} sentence
 * @desc Reverse
 *       It returns the sentence but in backwards
 * @return {String} The backwards argument.
 */
function reverse(sentence) {
    if (sentence.length === 0) throw InvalidArgumentException;
    let reverseString = "";
    for (let i = sentence.length; i >= 0; i--) {
        reverseString += sentence[i];
    }
    return reverseString;
}
```

JSDoc – Classes (I)



```
/**
 * @classdesc Class representing a Pokémon
 * @class
 * @property {string} name
 * @property {string} [nickname]
 */
class Pokemon() {
```

```
class Pokemon() {
  constructor (pokedex, number, nickname = "") {
    /**
     * Name is a property that contains the name of the pokémon
     * @type {string}
     * @public
     */
    this.name = pokedex.name(number);
    /**
     * Nickname is an optional property that may contains the name
     * the trainer gived to his pokémon
     * @type {string}
     * @public
     */
    this.nickname = nickname;
    // ...
  }
}
```

JSDoc – Classes (II)

```
class Pokemon() {  
  constructor (pokedex, number, nickname = "") {  
    /**  
     * Name is a property that contains the name of the pokémon  
     * @type {string}  
     * @public  
     */  
    this.name = pokedex.name(number);  
    /**  
     * Nickname is an optional property that may contains the name  
     * the trainer given to his pokémon  
     * @type {string}  
     * @public  
     */  
    this.nickname = nickname;  
    // ...  
  }  
  attack (move, opponent) {  
    //...  
  }  
}
```

```
/**  
 * @classdesc Class representing a Pikachu  
 * @class  
 * @extends Pokémon  
 */  
class Pikachu extends Pokemon() {  
  // ...  
  /**  
   * Pikachu attacks another pokémon with thunderbolt  
   * @override  
   */  
  attack (opponent) {  
    move = "Thunderbolt";  
  }  
}
```

JSDoc – How to use it

- Installation
- Development
- Use JSDoc
 - jsdoc file.js



/out

Sources

- <https://google.github.io/styleguide/jsguide.html> (02.03.2022)
- <https://github.com/yannickcr/eslint-plugin-react/issues/2339> (02.03.2022)
- <https://eslint.org/docs/user-guide/getting-started> (02.03.2022)
- <https://lenguajejs.com/javascript/caracteristicas/eslint/> (02.03.2022)
- <https://www.freecodecamp.org/news/google-publishes-a-javascript-style-guide-here-are-some-key-lessons-1810b8ad050b/> (02.03.2022)
- https://www.w3schools.com/js/js_best_practices.asp (02.03.2022)
- <https://javascript.info/coding-style> (02.03.2022)
- <https://codezen.rishimohan.me> (02.03.2022)
- <https://jsdoc.app> (02.03.2022)
- <https://stackoverflow.com/questions/41715994/how-to-document-ecma6-classes-with-jsdoc> (02.03.2022)
- <https://www.geeksforgeeks.org/documentation-comments-in-jsdoc/> (02.03.2022)