JavaScript Code Linting 101

`Keep your code clean`

TABLE OF CONTENTS

1. What is code linting? (beginner)
2. Purpose of code linting (beginner)
3. JavaScript Linting Tools (beginner)
4. Overview of popular linter configurations and plugins (beginner)
5. Integrating linter into the project and Continuous Integration (beginner)
6. Writing own configurations (intermediate)
7. Overview of useful rules (intermediate)
8. Writing plugins for linter (advanced)
9. Deep overview: Abstract Syntax Tree, ESLint structure and more (advanced)
10. Purpose of writing own plugins, perspectives (advanced)
11. Get the most out of linters with plugins (advanced)
12. TypeScript Linting (bonus)

AT THE END:

"Questions section"

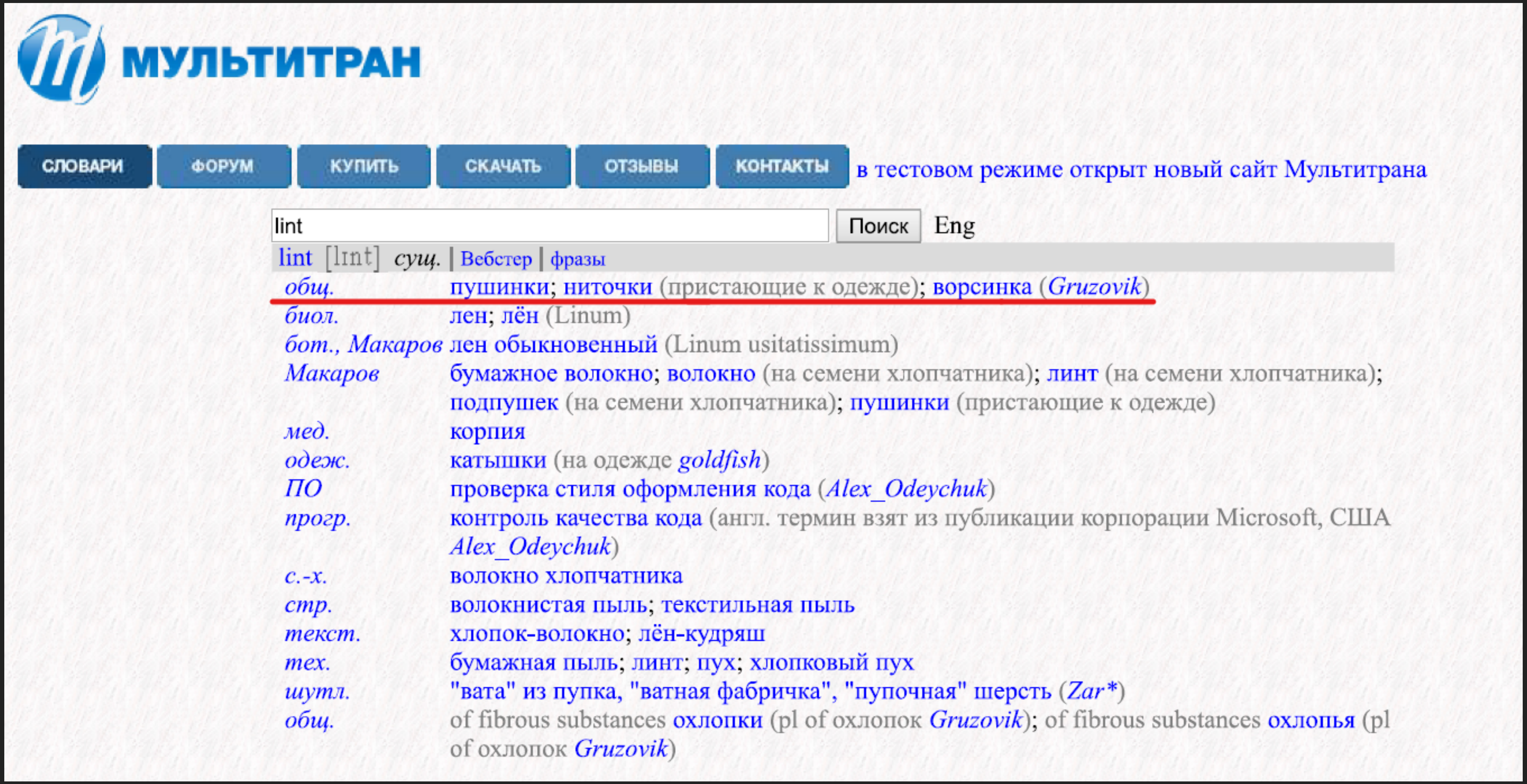
```
/**
 * Take notes, write down slides numbers and we will
 * get back later. We will also have some intermediate
 * questions sections :)
 */
```

WHAT IS CODE LINTING?

Lint, or a linter, is a tool that applies static analysis to source code to flag programming errors, bugs, stylistic errors, and suspicious constructs.

"Wikipedia"

The term originates from a UNIX utility called "Lint" that examined C language source code.

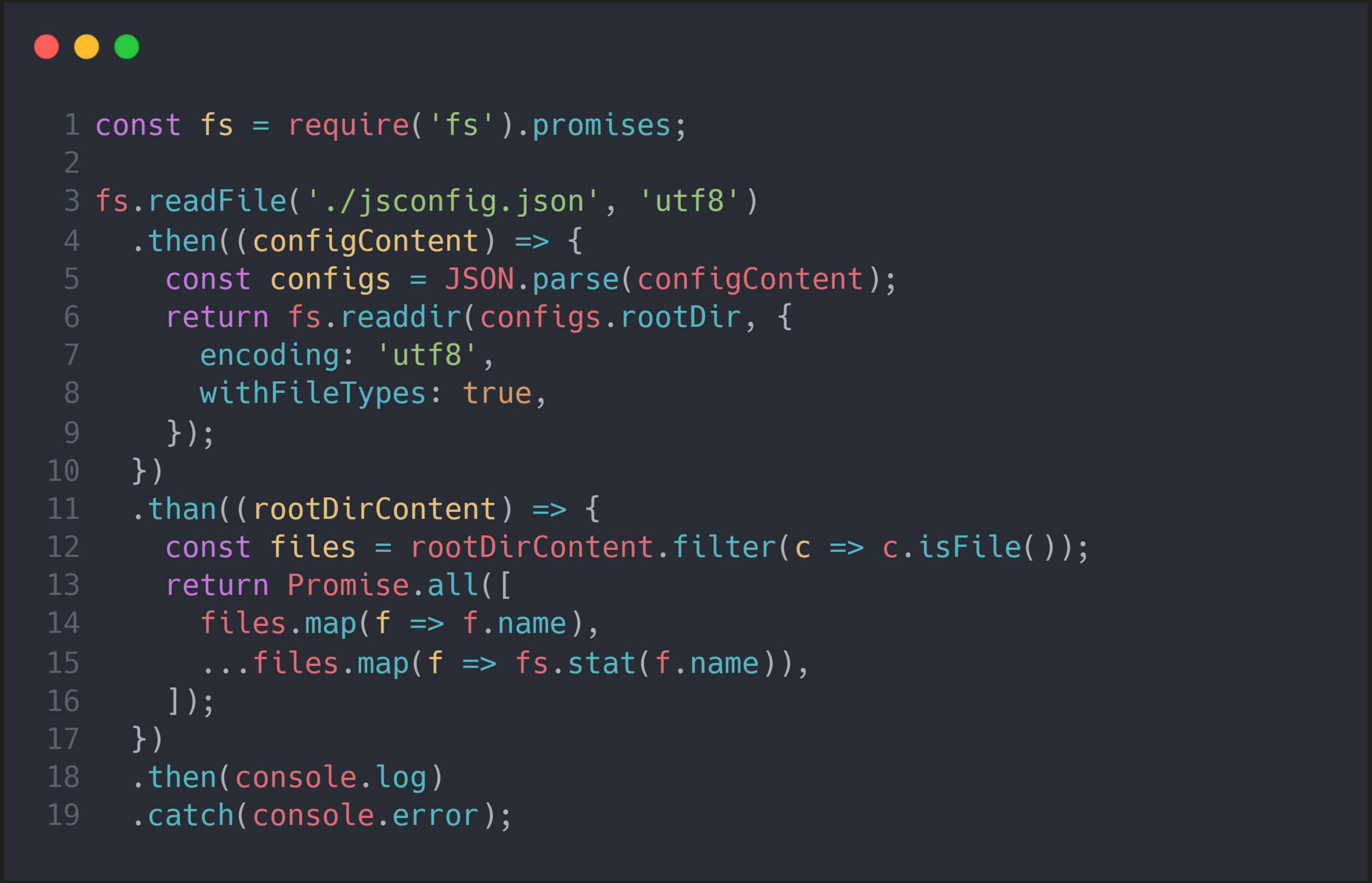


lint [lɪnt] *сущ.* | Вебстер | фразы

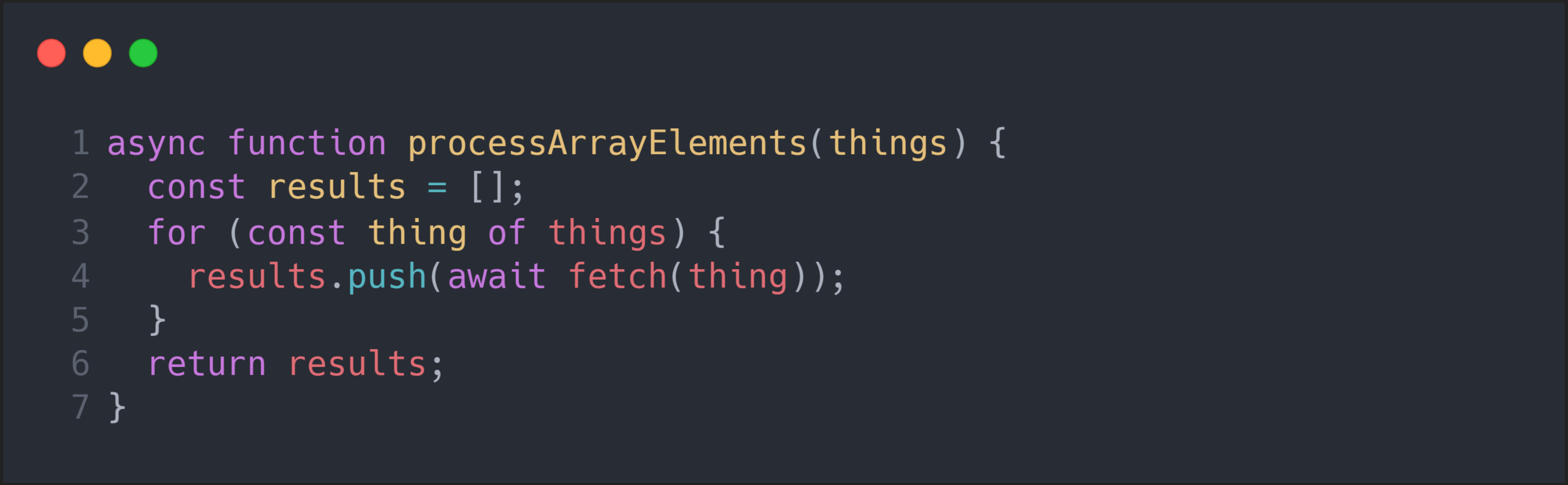
общ.	пушинки; ниточки (пристающие к одежде); ворсинка (*Gruzovik*)
биол.	лен; лён (Linum)
бот., Макаров	лен обыкновенный (Linum usitatissimum)
Макаров	бумажное волокно; волокно (на семени хлопчатника); линт (на семени хлопчатника); подпушек (на семени хлопчатника); пушинки (пристающие к одежде)
мед.	корпия
одеж.	катышки (на одежде *goldfish*)
ПО	проверка стиля оформления кода (*Alex_Odeychuk*)
прогр.	контроль качества кода (англ. термин взят из публикации корпорации Microsoft, США *Alex_Odeychuk*)
с.-х.	волокно хлопчатника
стр.	волокнистая пыль; текстильная пыль
текст.	хлопок-волокно; лён-кудряш
тех.	бумажная пыль; линт; пух; хлопковый пух
шутл.	"вата" из пупка, "ватная фабричка", "пупочная" шерсть (*Zar**)
общ.	of fibrous substances охлопки (pl of охлопок *Gruzovik*); of fibrous substances охлопья (pl of охлопок *Gruzovik*)

PURPOSE OF LINTING CODE?

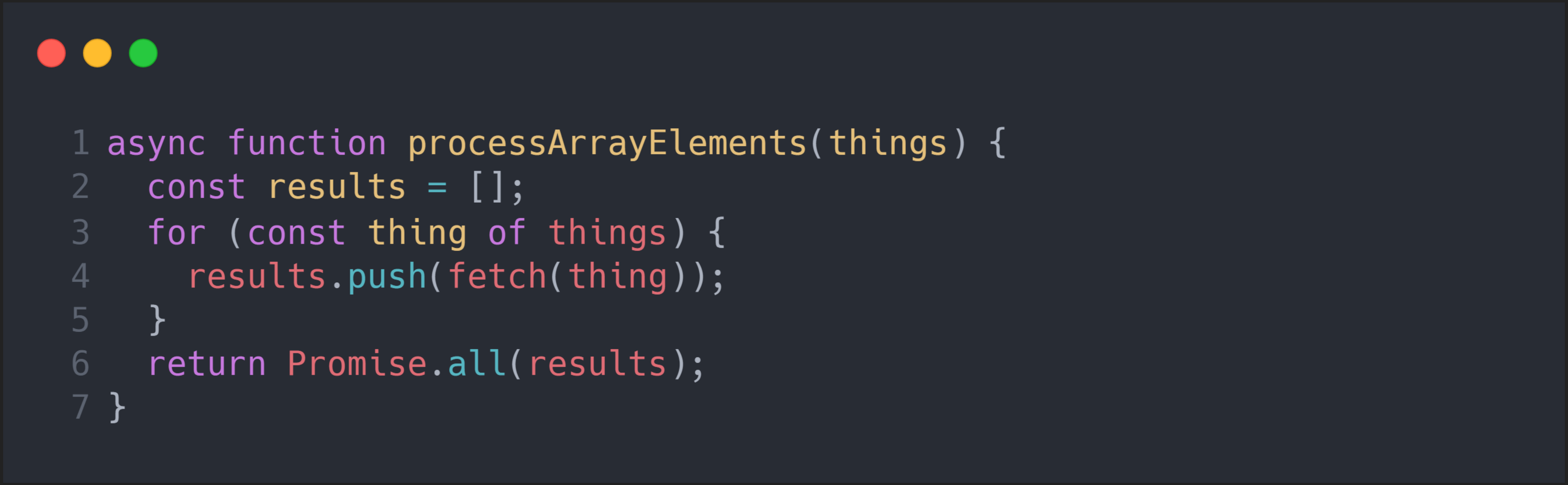
- Detect possible and known bugs, errors and problems



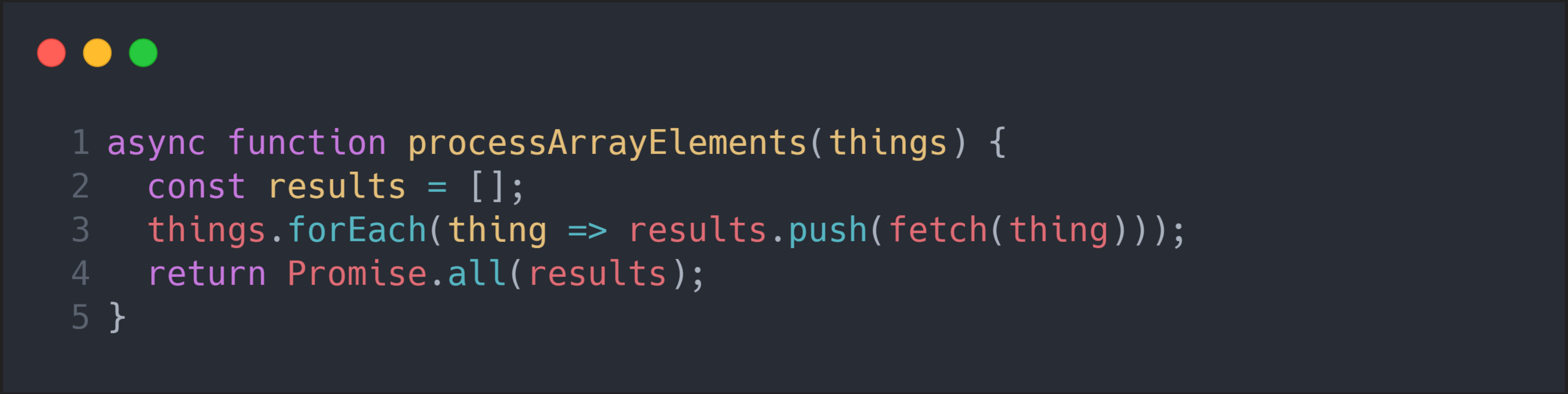
```javascript
const fs = require('fs').promises;

fs.readFile('./jsconfig.json', 'utf8')
  .then((configContent) => {
    const configs = JSON.parse(configContent);
    return fs.readdir(configs.rootDir, {
      encoding: 'utf8',
      withFileTypes: true,
    });
  })
  .than((rootDirContent) => {
    const files = rootDirContent.filter(c => c.isFile());
    return Promise.all([
      files.map(f => f.name),
      ...files.map(f => fs.stat(f.name)),
    ]);
  })
  .then(console.log)
  .catch(console.error);
```



```
async function processArrayElements(things) {
  const results = [];
  for (const thing of things) {
    results.push(await fetch(thing));
  }
  return results;
}
```



```
async function processArrayElements(things) {
  const results = [];
  for (const thing of things) {
    results.push(fetch(thing));
  }
  return Promise.all(results);
}
```



```javascript
async function processArrayElements(things) {
  const results = [];
  things.forEach(thing => results.push(fetch(thing)));
  return Promise.all(results);
}
```

PURPOSE OF LINTING CODE?

- Detect possible and known bugs, errors and problems
- Our code should adhere to a certain syntax conventions
- Our code should match any standard and/or rules
- Can help us keep our code clean
- Can help us do a code reviews
- Detect known heisenbugs

Heisenbug, Bohrbug, Mandelbug, Schroedinbug, Hindenbug, Higgs-bugson

JAVASCRIPT LINTING TOOLS

Tool	Downloads	Issues	PRs	Updates	Stars	License
Eslint	> 7.5M	98	40	4-6/week	> 13.5K	MIT
JSHint	> 540K	356	58	1-2/week	> 8K	MIT
Standard	> 163K	72	8	3-5/year	> 20K	MIT
JSLint	> 25K	15	0	1/month	475	BSD-3-Clause

Deprecated: JSCS (merged with ESLint), Validator Pro

Out of scope: TSLint (for TypeScript), Google Closure Compiler

POPULAR ESLINT CONFIGURATIONS

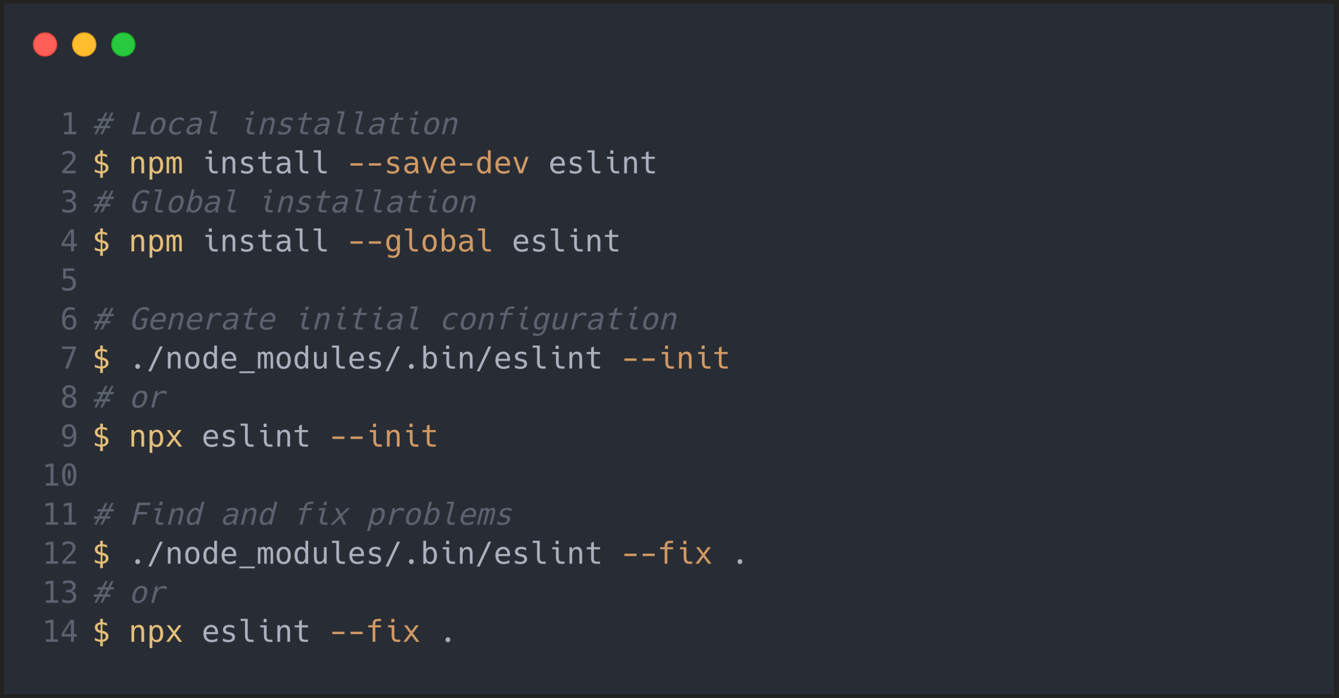
- Airbnb (eslint-config-airbnb)
- Standard (eslint-config-standard)
- Canonical (eslint-config-canonical)
- Problems (eslint-config-problems)
- ESLint recommended (already installed with ESLint)
- Facebook (eslint-config-fbjs)
- Google (eslint-config-google)
- ES (eslint-config-es)

POPULAR ESLINT PLUGINS

- Angular (eslint-plugin-angular)
- React (eslint-plugin-react)
- React Native (eslint-plugin-react-native)
- Vue (eslint-plugin-vue)
- GraphQL (eslint-plugin-graphql)
- JSDoc (eslint-plugin-jsdoc)
- Lodash (eslint-plugin-lodash)
- HTML (eslint-plugin-html)
- Import (eslint-plugin-import)
- Security (eslint-plugin-security)
- MongoDB (eslint-plugin-mongodb)
- JSON (eslint-plugin-json)
- Markdown (eslint-plugin-markdown)
- SQL (eslint-plugin-sql)
- Optimize Regexp (eslint-plugin-optimize-regex)
- Node (eslint-plugin-node)

...and much more plugins for testing environments (Mocha, Jest, Chai, Jasmine), project structures, filenames, code comments, preventing security issues (like XSS detection) and so on.

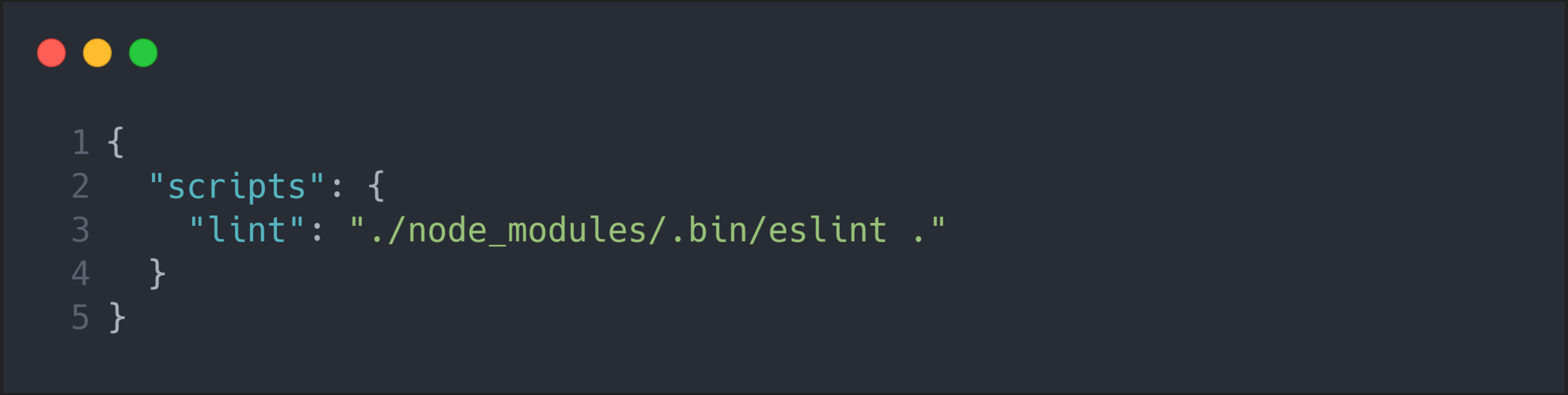
ESLINT INSTALLATION AND CONFIGURATION



```
# Local installation
$ npm install --save-dev eslint
# Global installation
$ npm install --global eslint

# Generate initial configuration
$ ./node_modules/.bin/eslint --init
# or
$ npx eslint --init

# Find and fix problems
$ ./node_modules/.bin/eslint --fix .
# or
$ npx eslint --fix .
```

Configuration file (.eslintrc.*) will appear in project's root

ADD ESLINT TO PACKAGE.JSON



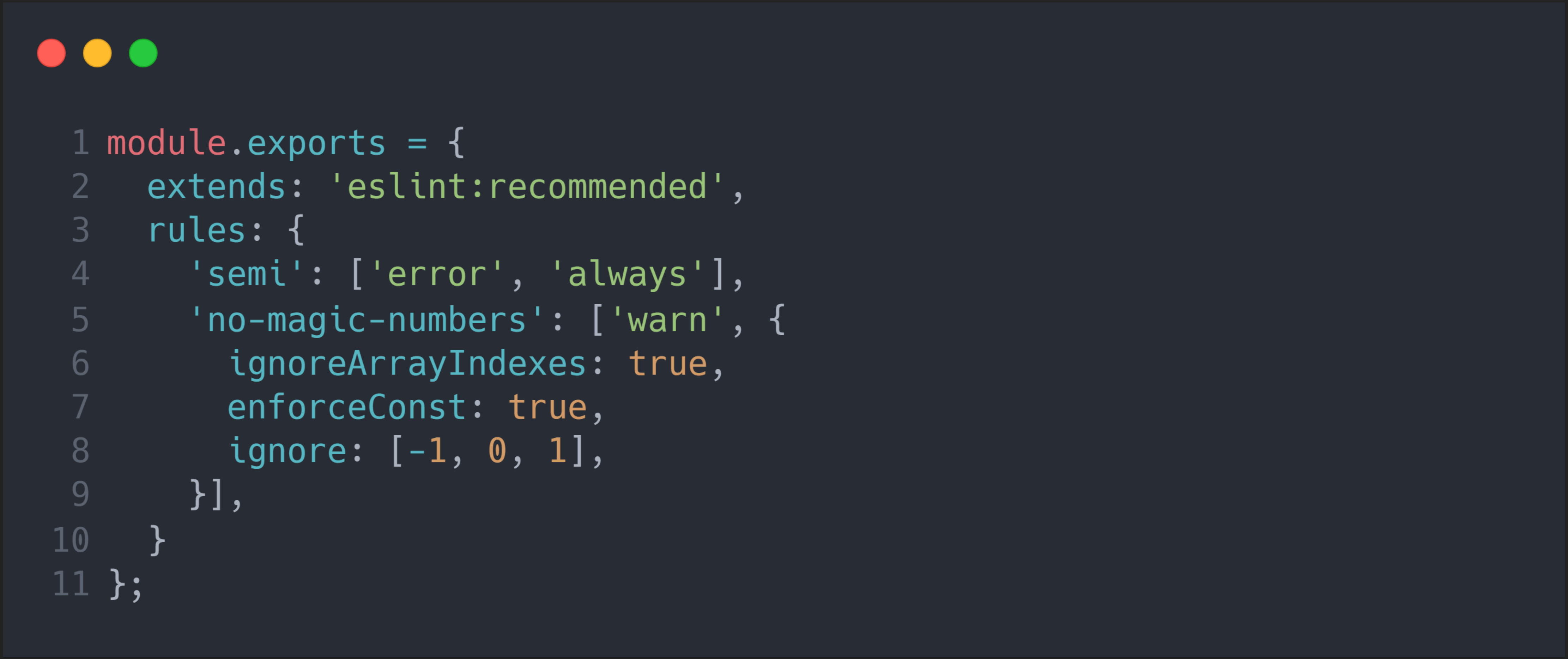
```json
{
  "scripts": {
    "lint": "./node_modules/.bin/eslint ."
  }
}
```

IDE INTEGRATION

ESLint for Visual Studio Code

...but also available for: Atom, JetBrains editors, Brackets, Eclipse, Emacs, Vim and Sublime Text 3

WRITING OWN (SHAREABLE) CONFIGURATIONS



```js
module.exports = {
  extends: 'eslint:recommended',
  rules: {
    'semi': ['error', 'always'],
    'no-magic-numbers': ['warn', {
      ignoreArrayIndexes: true,
      enforceConst: true,
      ignore: [-1, 0, 1],
    }],
  }
};
```

WHAT COULD BE INSIDE OF CONFIGURATION FILE?

- Parser: Espree (default), @typescript-eslint/parser
- parserOptions:
  - ecmaVersion: 2019 (2015, 5 (default), 6, 9)
  - sourceType: 'script' (default) / 'module'
  - ecmaFeatures:
    - globalReturn
    - impliedStrict
    - jsx
- extends: string / array
- globals: object
- plugins: array
- overrides: array of objects
- rules: object
  - off / 0
  - warn / 1
  - error / 2

WHAT COULD BE INSIDE OF CONFIGURATION FILE?

- env:
  - browser
  - node
  - shared-node-browser
  - es6
  - commonjs
  - amd
  - worker
  - serviceworker
  - webextensions
  - mocha / chai / jasmine / jest
  - jquery
  - environments from plugins

DISABLE ESLINT WITH COMMENTS

```
/* eslint quotes: ["error", "double"], curly: 2 */
/* eslint-disable */
/* eslint-enable */
/* eslint-disable no-global-assign, curly */
// eslint-disable-line
// eslint-disable-line yoda
// eslint-disable-next-line
```

ESLINTIGNORE

With ".eslintignore" file we can disable ESLint for specific files or folders. Same syntax to .gitignore, .npmignore etc.

BEFORE PUBLISHING

```json
{
  "name": "eslint-config-our-pretty-name",
  "keywords": [
    "eslint",
    "eslintconfig"
  ],
  "peerDependencies": {
    "eslint": ">=5.14.0"
  }
}
```

WHERE TO FIND RULES LIST?

https://eslint.org/docs/rules

RULES CATEGORIES

Possible Errors

Best Practices

Variables

Node.js and Common.js

Stylistic Issues

ECMAScript 6

WHAT WE CAN HANDLE WITH BUILT-IN RULES?

WRITING ESLINT PLUGIN

The easiest way: with Yeoman generator (generator-eslint)

```
$ npm install --global yo generator-eslint

# create a new ESLint plugin
$ yo eslint:plugin

# create a new ESLint rule
$ yo eslint:rule
```

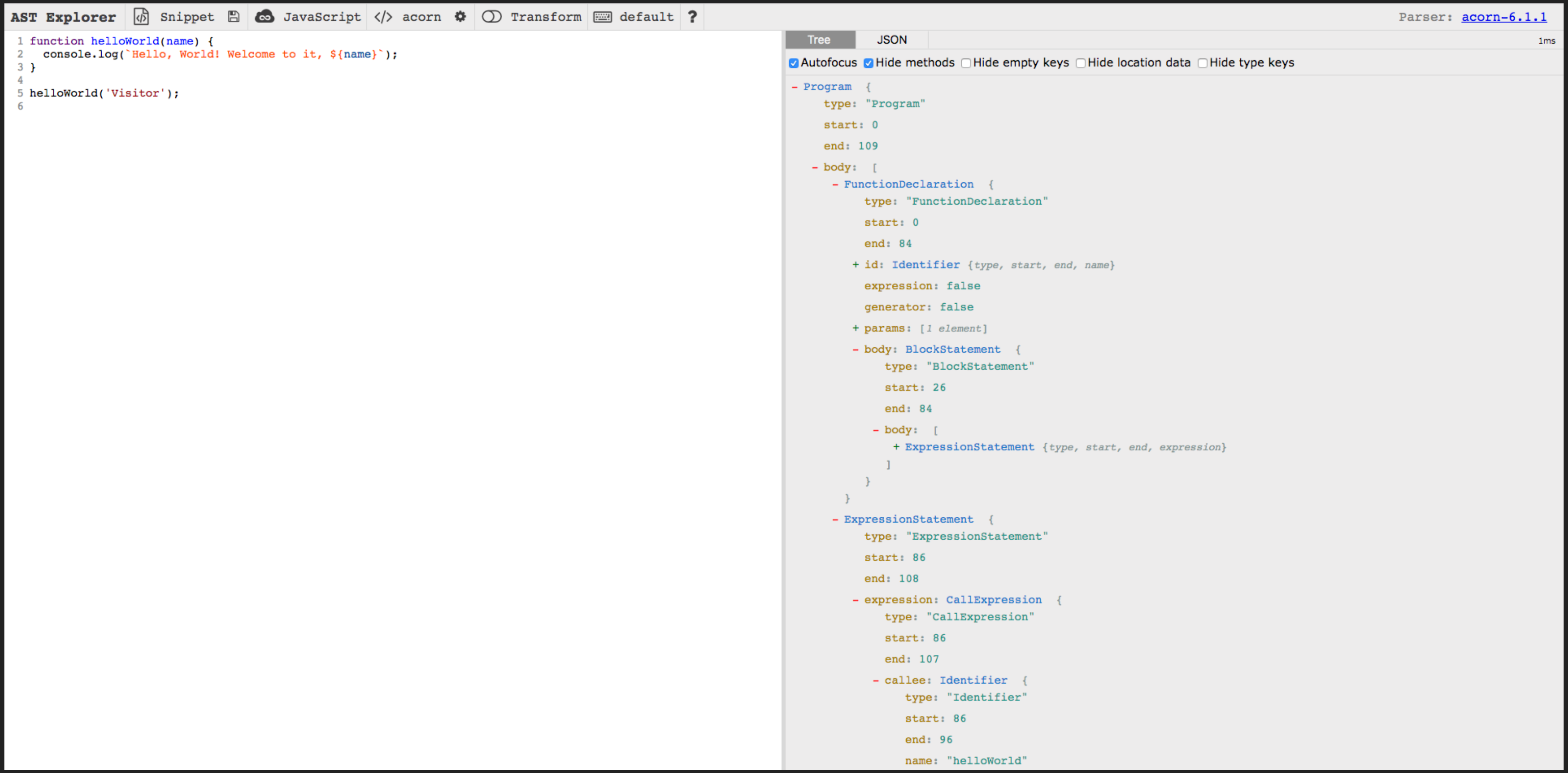
WHAT CAN WE HAVE INSIDE OF A PLUGIN?

1. Custom rules
2. Custom environments (define globals and parser options)
3. Create file processors (even with auto-fix)
4. Custom configs (including our rules)

HOW DOES ESLINT SEE OUR CODE?

AST - Abstract Syntax Tree

HTTPS://ASTEXPLORER.NET

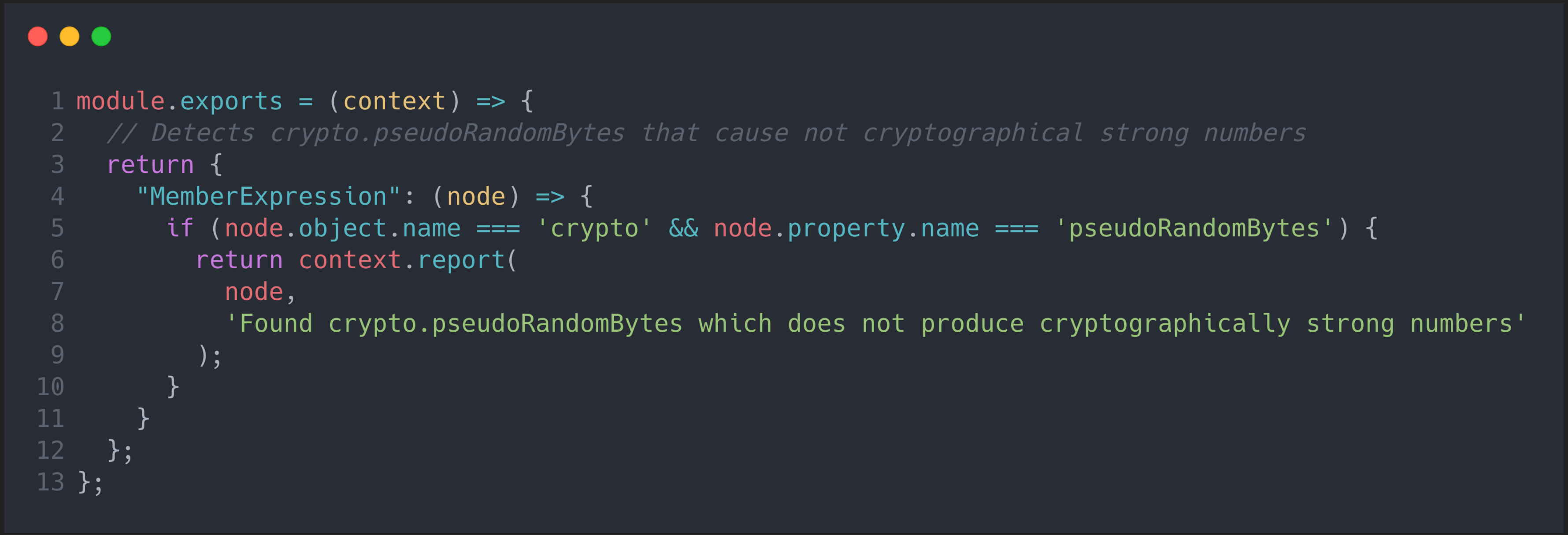


```javascript
function helloWorld(name) {
  console.log(`Hello, World! Welcome to it, ${name}`);
}

helloWorld('Visitor');
```

[✓] Autofocus [✓] Hide methods [ ] Hide empty keys [ ] Hide location data [ ] Hide type keys

```
- Program  {
     type: "Program"
     start: 0
     end: 109
   - body:  [
      - FunctionDeclaration  {
           type: "FunctionDeclaration"
           start: 0
           end: 84
         + id: Identifier {type, start, end, name}
           expression: false
           generator: false
         + params: [1 element]
         - body: BlockStatement  {
              type: "BlockStatement"
              start: 26
              end: 84
            - body:  [
               + ExpressionStatement {type, start, end, expression}
              ]
           }
        }
      - ExpressionStatement  {
           type: "ExpressionStatement"
           start: 86
           end: 108
         - expression: CallExpression  {
              type: "CallExpression"
              start: 86
              end: 107
            - callee: Identifier  {
                 type: "Identifier"
                 start: 86
                 end: 96
                 name: "helloWorld"
```

RULE EXAMPLE

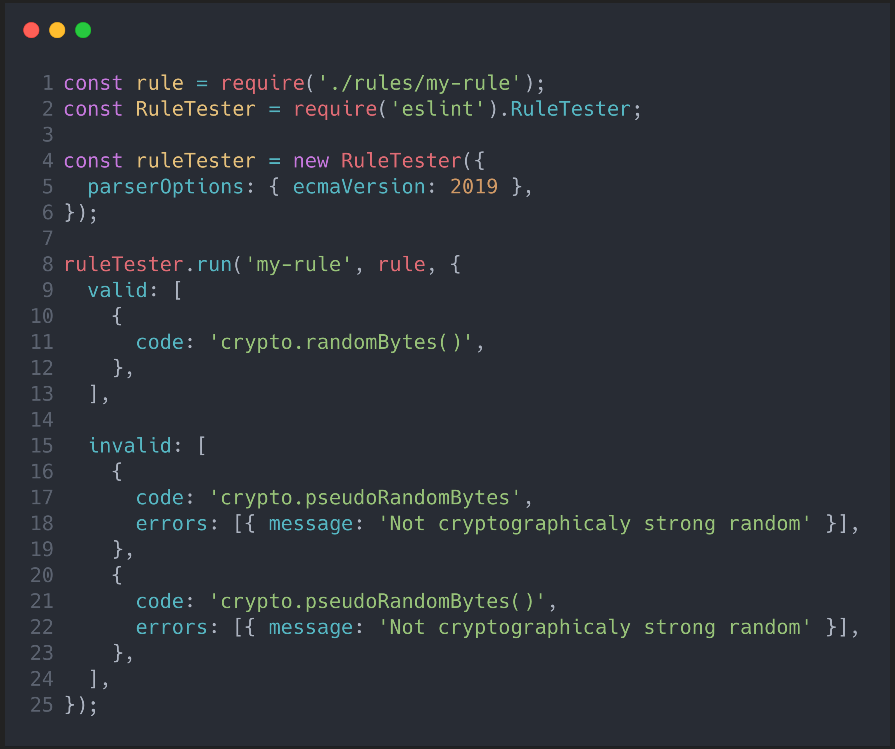


```
module.exports = (context) => {
  // Detects crypto.pseudoRandomBytes that cause not cryptographical strong numbers
  return {
    "MemberExpression": (node) => {
      if (node.object.name === 'crypto' && node.property.name === 'pseudoRandomBytes') {
        return context.report(
          node,
          'Found crypto.pseudoRandomBytes which does not produce cryptographically strong numbers'
        );
      }
    }
  };
};
```

https://eslint.org/docs/developer-guide/working-with-rules

HOW THEN WE TEST OUR PLUGIN RULES?

ESLint gives us a RuleTester



```javascript
const rule = require('./rules/my-rule');
const RuleTester = require('eslint').RuleTester;

const ruleTester = new RuleTester({
  parserOptions: { ecmaVersion: 2019 },
});

ruleTester.run('my-rule', rule, {
  valid: [
    {
      code: 'crypto.randomBytes()',
    },
  ],

  invalid: [
    {
      code: 'crypto.pseudoRandomBytes',
      errors: [{ message: 'Not cryptographicaly strong random' }],
    },
    {
      code: 'crypto.pseudoRandomBytes()',
      errors: [{ message: 'Not cryptographicaly strong random' }],
    },
  ],
});
```

YOU ARE AWESOME!

Thank you for participation =)