sonar RULES

Products

- Secrets
- ABAP
- Apex
- C
- C++
- CloudFormation
- COBOL
- C#
- CSS
- Flex
- Go
- HTML
- Java
- JavaScript
- Kotlin
- Objective C
- PHP
- PL/I
- PL/SQL
- Python
- RPG
- Ruby
- Scala
- Swift
- Terraform
- Text
- TypeScript
- T-SQL
- VB.NET
- VB6
- XML

JS

JavaScript static code analysis

Unique rules to find Bugs, Vulnerabilities, Security Hotspots, and Code Smells in your JAVASCRIPT code

All rules 285 | Vulnerability 29 | Bug 62 | Security Hotspot 43 | Code Smell 151 | Quick Fix 41

Tags

Search by name...

security-sensitive
Security Hotspot

Encrypting data is security-sensitive
Security Hotspot

Using regular expressions is security-sensitive
Security Hotspot

Class methods should be used instead of "prototype" assignments
Code Smell

Function constructors should not be used
Code Smell

Variables should be declared with "let" or "const"
Code Smell

Unchanged variables should be marked "const"
Code Smell

Wildcard imports should not be used
Code Smell

"switch" statements should not be nested
Code Smell

Cyclomatic Complexity of functions should not be too high
Code Smell

"strict" mode should be used with caution
Code Smell

Control flow statements "if", "for", "while", "switch" and "try" should not be nested too deeply
Code Smell

Names of regular expressions named groups should be used

Analyze your code

Code Smell Major regex

Why use named groups only to never use any of them later on in the code?

This rule raises issues every time named groups are:

- defined but never called anywhere in the code through their name;
- defined but called elsewhere in the code by their number instead;
- referenced while not defined.

Noncompliant Code Example

```
const date = "01/02";

const datePattern = /(?<month>[0-9]{2})\/(?<year>[0-9]{2})/;
const dateMatched = date.match(datePattern);

if (dateMatched !== null) {
  checkValidity(dateMatched[1], dateMatched[2]); // Noncompl
  checkValidity(dateMatched.groups.day); // Noncompliant - t
}

// ...

const score = "14:1";

const scorePattern = /(?<player1>[0-9]+):(?<player2>[0-9]+)/
const scoreMatched = score.match(scorePattern);

if (scoreMatched !== null) {
  checkScore(score);
}
```

Compliant Solution

```
const date = "01/02";

const datePattern = /(?<month>[0-9]{2})\/(?<year>[0-9]{2})/;
const dateMatched = date.match(datePattern);

if (dateMatched !== null) {
  checkValidity(dateMatched.groups.month, dateMatched.groups
}

// ...

const score = "14:1";

const scorePattern = /(?<player1>[0-9]+):(?<player2>[0-9]+)/
const scoreMatched = score.match(scorePattern);

if (scoreMatched !== null) {
```

"switch" statements should have "default" clauses

Code Smell

"if ... else if" constructs should end with "else" clauses

Code Smell

Control structures should use curly braces

Code Smell

String literals should not be duplicated

Code Smell

```
  checkScore(scoreMatched.groups.player1);
  checkScore(scoreMatched.groups.player2);
}
```

Available In:

sonarlint sonarcloud sonarqube