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

# 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...

Track lack of copyright and license headers
Code Smell

Reading the Standard Input is security-sensitive
Security Hotspot

Using command line arguments is security-sensitive
Security Hotspot

Using Sockets is security-sensitive
Security Hotspot

Executing XPath expressions is 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

## Reluctant quantifiers in regular expressions should be followed by an expression that can't match the empty string

Analyze your code

Code Smell Major regex

When a reluctant (or lazy) quantifier is followed by a pattern that can match the empty string or directly by the end of the regex, it will always match zero times for `*?` or one time for `+?`. If a reluctant quantifier is followed directly by the end anchor (`$`), it behaves indistinguishably from a greedy quantifier while being less efficient.

This is likely a sign that the regex does not work as intended.

### Noncompliant Code Example

```
str.split(/.*?x?/); // Noncompliant, this will behave just l
/^.*?$/.test(str); // Noncompliant, replace with ".*"
```

### Compliant Solution

```
str.split(/.*?x/);
/^.*$/.test(str);
```

Available In:

sonarlint sonarcloud sonarqube

"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