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...

Vulnerability

Function calls should not pass extra arguments
Bug

Regular expressions should be syntactically valid
Bug

Getters and setters should access the expected fields
Bug

"super()" should be invoked appropriately
Bug

"Symbol" should not be used as a constructor
Bug

Results of "in" and "instanceof" should be negated rather than operands
Bug

"in" should not be used with primitive types
Bug

A compare function should be provided when using "Array.prototype.sort()"
Bug

Jump statements should not occur in "finally" blocks
Bug

Using slow regular expressions is security-sensitive
Security Hotspot

Using publicly writable directories is security-sensitive
Security Hotspot

Variables should be declared explicitly

Analyze your code

Code Smell Blocker pitfall

JavaScript variable scope can be particularly difficult to understand and get right. The situation gets even worse when you consider the *accidental* creation of global variables, which is what happens when you declare a variable inside a function or the `for` clause of a for-loop without using the `let`, `const` or `var` keywords.

`let` and `const` were introduced in ECMAScript 2015, and are now the preferred keywords for variable declaration.

Noncompliant Code Example

```
function f(){
  i = 1;         // Noncompliant; i is global

  for (j = 0; j < array.length; j++) {  // Noncompliant; j i
    // ...
  }
}
```

Compliant Solution

```
function f(){
  var i = 1;

  for (let j = 0; j < array.length; j++) {
    // ...
  }
}
```

Available In:
sonarlint sonarcloud sonarqube

Using clear-text protocols is security-sensitive

Security Hotspot

Expanding archive files without controlling resource consumption is security-sensitive

Security Hotspot

Using weak hashing algorithms is security-sensitive

Security Hotspot

Disabling CSRF protections is security-sensitive