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...

Bug

Promise rejections should not be caught by 'try' block
Bug

Collection elements should not be replaced unconditionally
Bug

Errors should not be created without being thrown
Bug

Collection sizes and array length comparisons should make sense
Bug

All branches in a conditional structure should not have exactly the same implementation
Bug

Destructuring patterns should not be empty
Bug

The output of functions that don't return anything should not be used
Bug

Comma and logical OR operators should not be used in switch cases
Bug

Generators should "yield" something
Bug

Attempts should not be made to update "const" variables
Bug

Strict equality operators should not be used with dissimilar types
Bug

"in" should not be used with primitive types

Analyze your code

Bug Critical

The `in` operator tests whether the specified property is in the specified object.

If the right operand is a of primitive type (i.e., not an object) the `in` operator raises a `TypeError`.

Noncompliant Code Example

```
var x = "Foo";
"length" in x; // Noncompliant: TypeError
0 in x;        // Noncompliant: TypeError
```

Compliant Solution

```
var x = new String("Foo");
"length" in x;    // true
0 in x;           // true
"foobar" in x;    // false
```

Available In:

sonarlint sonarcloud sonarqube

© 2008-2022 SonarSource S.A., Switzerland. All content is copyright protected. SONAR, SONARSOURCE, SONARLINT, SONARQUBE and SONARCLOUD are trademarks of SonarSource S.A. All other trademarks and copyrights are the property of their respective owners. All rights are expressly reserved.
Privacy Policy

"new" operators should be used with functions

Bug

Non-existent operators '=+', '=-' and '=!' should not be used

Bug

"NaN" should not be used in comparisons

Bug

Setters should not return values

Bug