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

TypeScript static code analysis

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

All rules 279 | Vulnerability 27 | Bug 51 | Security Hotspot 43 | Code Smell 158 | Quick Fix 50

Tags | Search by name...

- Mocha timeout should be disabled by setting it to "0". — Bug
- Unicode Grapheme Clusters should be avoided inside regex character classes — Bug
- Assertions should not be given twice the same argument — Bug
- Alternatives in regular expressions should be grouped when used with anchors — Bug
- Promise rejections should not be caught by 'try' block — Bug
- Collection elements should not be replaced unconditionally — Bug
- Constructors should not be declared inside interfaces — 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

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

Analyze your code

Bug | Critical | Quick Fix

Mixing up the order of operations will almost always yield unexpected results.

Similarly, mis-applied negation will also yield bad results. For instance consider the difference between `!key in dict` and `!(key in dict)`. The first looks for a boolean value (`!key`) in `dict`, and the other looks for a string and inverts the result. `!obj instanceof SomeClass` has the same problem.

This rule raises an issue when the left operand of an `in` or `instanceof` operator is negated.

Noncompliant Code Example

```
if (!"prop" in myObj) {  // Noncompliant;  "in" operator is
  doTheThing();  // this block will be never executed
}

if (!foo instanceof MyClass) {  // Noncompliant; "!foo" retu
  doTheOtherThing();  // this block is never executed
}
```

Compliant Solution

```
if (!("prop" in myObj)) {
  doTheThing();
}

if (!(foo instanceof MyClass)) {
  doTheOtherThing();
}
```

Available In:

sonarlint sonarcloud sonarqube

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

"new" operators should be used with functions

Bug

Non-existent operators '=+', '=-' and '=!'