




 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... 

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

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 '!=' should not be used






Bug

Jump statements should not occur in "finally" blocks

Analyze your code

 Bug

 Critical 

 cwe error-handling

Using `return`, `break`, `throw`, and `continue` from a `finally` block overwrites similar statements from the suspended `try` and `catch` blocks.

This rule raises an issue when a jump statement (`break`, `continue`, `return` and `throw`) would force control flow to leave a `finally` block.

Noncompliant Code Example

```
function foo() {
  try {
    return 1; // We expect 1 to be returned
  } catch(err) {
    return 2; // Or 2 in cases of error
  } finally {
    return 3; // Noncompliant: 3 is returned before 1, o
  }
}
```


Compliant Solution


```
function foo() {
  try {
    return 1; // We expect 1 to be returned
  } catch(err) {
    return 2; // Or 2 in cases of error
  }
}
```


See

- [MITRE, CWE-584](#) - Return Inside Finally Block

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](#)

https://rules.sonarsource.com/typescript/RSPEC-1143

1/2

 Bug
"NaN" should not be used in comparisons  Bug
A "for" loop update clause should move the counter in the right direction  Bug
Return values from functions without side effects should not be ignored  Bug
Special identifiers should not be bound or assigned