




 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 |
|  | All branches in a conditional structure should not have exactly the same implementation |
|  | Destructuring patterns should not be empty |
|  | The output of functions that don't return anything should not be used |
|  | Comma and logical OR operators should not be used in switch cases |
|  | Generators should "yield" something |
|  | Attempts should not be made to update "const" variables |
|  | Strict equality operators should not be used with dissimilar types |
|  | "new" operators should be used with functions |
|  | Non-existent operators '+=', '==', and '!=' should not be used |
| | "NaN" should not be used in comparisons |
| | Setters should not return values |

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:

 |  | 

| |
|--|
| <p>Properties of variables with "null" or "undefined" values should not be accessed</p> <p> Bug</p> |
| <p>A "for" loop update clause should move the counter in the right direction</p> <p> Bug</p> |
| <p>Return values from functions without side effects should not be ignored</p> <p> Bug</p> |
| <p>Special identifiers should not be bound or assigned</p> <p> Bug</p> |