




 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

removed
 Code Smell
Function parameters with default values should be last
 Code Smell
Functions should not be defined inside loops
 Code Smell
"switch" statements should not have too many "case" clauses
 Code Smell
Only "while", "do", "for" and "switch" statements should be labelled
 Code Smell
Sections of code should not be commented out
 Code Smell
Unused function parameters should be removed
 Code Smell
Track uses of "FIXME" tags
 Code Smell
Assignments should not be made from within sub-expressions
 Code Smell
Labels should not be used
 Code Smell
Variables should not be shadowed
 Code Smell
Redundant pairs of parentheses should be removed
 Code Smell
Nested blocks of code should not be

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

Analyze your code

 Bug  Major 

The comma operator (,) evaluates its operands, from left to right, and returns the second one. That's useful in some situations, but just wrong in a switch case. You may think you're compactly handling multiple values in the case, but only the last one in the comma-list will ever be handled. The rest will fall through to the default.

Similarly the logical OR operator (||) will not work in a switch case, only the first argument will be considered at execution time.

Noncompliant Code Example





```
switch a {
  case 1,2: // Noncompliant; only 2 is ever handled by this
    doTheThing(a);
  case 3 || 4: // Noncompliant; only '3' is handled
    doThatThing(a);
  case 5:
    doTheOtherThing(a);
  default:
    console.log("Neener, neener!"); // this happens when a=
```

Compliant Solution

```
switch a {
  case 1:
  case 2:
    doTheThing(a);
  case 3:
  case 4:
    doThatThing(a);
  case 5:
    doTheOtherThing(a);
  default:
    console.log("Neener, neener!");
}
```

Available In:

sonarlint  | sonarcloud  | sonarqube 

<div>left empty</div> <div> Code Smell</div>
<div>Functions should not have too many parameters</div> <div> Code Smell</div>
<div>OS commands should not be vulnerable to argument injection attacks</div> <div> Vulnerability</div>
<div>Repeated patterns in regular expressions should not match the empty string</div> <div> Bug</div>