

































-  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 rules279

Vulnerability27

Bug51


Security Hotspot43

Code Smell158


Quick Fix50

Tags ▾


Search by name... 🔍

 Code Smell


Primitive types should be omitted from initialized or defaulted declarations




Non-null assertions should not be used




"undefined" should not be assigned




Trailing commas should not be used




Array constructors should not be used




Quotes for string literals should be used consistently




Statements should end with semicolons




Comments should not be located at the end of lines of code




Loops should not contain more than a single "break" or "continue" statement



Variable, property and parameter names should comply with a naming convention





Lines should not end with trailing whitespaces




"switch" statements should have "default" clauses

Analyze your code

 Code Smell

 Critical

 cwe

The requirement for a final default clause is defensive programming. The clause should either take appropriate action, or contain a suitable comment as to why no action is taken.

Noncompliant Code Example

```
switch (param) { //missing default clause
  case 0:
    doSomething();
    break;
  case 1:
    doSomethingElse();
    break;
}
```

```
switch (param) {
  default: // default clause should be the last one
    error();
    break;
  case 0:
    doSomething();
    break;
  case 1:
    doSomethingElse();
    break;
}
```

Compliant Solution

```
switch (param) {
  case 0:
    doSomething();
    break;
  case 1:
    doSomethingElse();
    break;
  default:
    error();
    break;
}
```

See

- [MITRE, CWE-478](#) - Missing Default Case in Switch Statement

Available In:





sonarlint

sonarcloud

sonarqube

https://rules.sonarsource.com/typescript/RSPEC-131

1/2

<div>Files should contain an empty newline at the end</div> <div> Code Smell</div>	
<div>An open curly brace should be located at the end of a line</div> <div> Code Smell</div>	
<div>Tabulation characters should not be used</div> <div> Code Smell</div>	
<div>Function and method names should comply with a naming convention</div> <div> Code Smell</div>	

© 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](#)