




 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... 🔍


 Code Smell


"switch" statements should not be nested





 Cyclomatic Complexity of functions should not be too high





 "strict" mode should be used with caution





 Control flow statements "if", "for", "while", "switch" and "try" should not be nested too deeply



 "switch" statements should have "default" clauses



 "if ... else if" constructs should end with "else" clauses



 Control structures should use curly braces

 String literals should not be duplicated


 Expressions should not be too complex


 Template literal placeholder syntax should not be used in regular strings


 Built-in objects should not be overridden


"undefined" should not be passed as the value of optional parameters

Analyze your code

 Code Smell

 Major

 Quick Fix

 confusing

Unlike in JavaScript, where every parameter can be omitted, in TypeScript you need to explicitly declare this in the function signature. Either you add ? in the parameter declaration and undefined will be automatically applied to this parameter. Or you add an initializer with a default value in the parameter declaration. In the latter case, when passing undefined for such parameter, default value will be applied as well. So it's better to avoid passing undefined value to an optional or default parameter because it creates more confusion than it brings clarity. Note, that this rule is only applied to the last arguments in function call.

Noncompliant Code Example

```
function foo(x: number, y: string = "default", z?: number) {
  // ...
}




foo(42, undefined); // Noncompliant
foo(42, undefined, undefined); // Noncompliant
foo(42, undefined, 5); // OK, there is no other way to force
```

Compliant Solution

```
function foo(x: number, y: string = "default", z?: number) {
  // ...
}

foo(42);
```




Available In:

 |  | 

© 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-4623

1/2

<div><div>"for...in" loops should filter properties before acting on them</div><div> Bug</div></div>
<div><div>Results of operations on strings should not be ignored</div><div> Bug</div></div>
<div><div>Increment (++) and decrement (--) operators should not be used in a method call or mixed with other operators in an expression</div><div> Code Smell</div></div>
<div><div>Optional boolean parameters should have default value</div></div>