




 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... 🔍


Parameters should be passed in the correct order

 Code Smell


Two branches in a conditional structure should not have exactly the same implementation

 Code Smell


Unused assignments should be 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

Errors should not be created without being thrown

Analyze your code

 Bug Major Quick Fix error-handling

Creating a new **Error** without actually throwing it is useless and is probably due to a mistake.




Noncompliant Code Example

```
if (x < 0) {
  new Error("x must be nonnegative");
}
```

Compliant Solution

```
if (x < 0) {
  throw new Error("x must be nonnegative");
}
```

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-3984

1/2

 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 left empty  Code Smell
Functions should not have too many parameters  Code Smell