




 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





Variables should be declared with "let" or "const"

Analyze your code

 Code Smell

 Critical

 Quick Fix

 es2015 bad-practice

ECMAScript 2015 introduced the `let` and `const` keywords for block-scope variable declaration. Using `const` creates a read-only (constant) variable.

The distinction between the variable types created by `var` and by `let` is significant, and a switch to `let` will help alleviate many of the variable scope issues which have caused confusion in the past.

Because these new keywords create more precise variable types, they are preferred in environments that support ECMAScript 2015. However, some refactoring may be required by the switch from `var` to `let`, and you should be aware that they raise `SyntaxErrors` in pre-ECMAScript 2015 environments.

This rule raises an issue when `var` is used instead of `const` or `let`.

Noncompliant Code Example

```
var color = "blue";
var size = 4;
```

Compliant Solution

```
const color = "blue";
let size = 4;
```

Available In:

sonarlint

sonarcloud

sonarqube

© 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-3504

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>