




 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... 🔍

"import" should be used to include external code

Code Smell

Braces and parentheses should be used consistently with arrow functions

Code Smell

Destructuring syntax should be used for assignments

Code Smell

Template strings should be used instead of concatenation

Code Smell

Shorthand object properties should be grouped at the beginning or end of an object declaration

Code Smell

Object literal shorthand syntax should be used

Code Smell

Strings and non-strings should not be added

Code Smell

Primitive types should be omitted from initialized or defaulted declarations

Code Smell

Non-null assertions should not be used

Code Smell

"undefined" should not be assigned

Code Smell

Trailing commas should not be used

Code Smell

Track uses of "FIXME" tags

Analyze your code

Code SmellMajor?cwe

FIXME tags are commonly used to mark places where a bug is suspected, but which the developer wants to deal with later.

Sometimes the developer will not have the time or will simply forget to get back to that tag.

This rule is meant to track those tags and to ensure that they do not go unnoticed.

Noncompliant Code Example

```
function divide(numerator, denominator) {
  return numerator / denominator;      // FIXME deno
}
```

See

- MITRE, CWE-546 - Suspicious Comment





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-1134

1/2

<div>Array constructors should not be used</div> <div> Code Smell</div>
<div>Quotes for string literals should be used consistently</div> <div> Code Smell</div>
<div>Statements should end with semicolons</div> <div> Code Smell</div>
<div>Comments should not be located at the end of lines of code</div> <div> Code Smell</div>
<div>Loops should not contain more than a</div>