








-  **sonar** RULES
-  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
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


Type assertions should use "as"

Analyze your code

-  Code Smell
-  Minor ?
-  confusing

Type assertion can be done in two ways: with `as MyType` or with `<MyType>`. But since there is an ambiguity in the latter when using JSX and there is no ambiguity in the former, `as` is preferred.

Noncompliant Code Example

```
var foo = <any>"foo"; // Noncompliant
```

Compliant Solution

```
var foo = "foo" as any;
```

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](#)

<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>