




 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... 🔍

"in" should not be used on arrays

Code Smell

Assignments should not be redundant

Code Smell

Functions should not have identical implementations

Code Smell

Sparse arrays should not be declared

Code Smell

Array-mutating methods should not be used misleadingly

Code Smell

Collection and array contents should be used

Code Smell

Literals should not be thrown

Code Smell

Array indexes should be numeric

Code Smell

Assertion arguments should be passed in the correct order

Code Smell

Ternary operators should not be nested

Code Smell

"delete" should not be used on arrays

Code Smell

Variables and functions should not be redeclared

Code Smell

"indexOf" checks should not be for positive numbers

Regular expressions should not contain control characters

Analyze your code

Bug Major ? regex

Entries in the ASCII table below code 32 are known as control characters or non-printing characters. As they are not common in JavaScript strings, using these invisible characters in regular expressions is most likely a mistake.

Noncompliant Code Example

```
const pattern1 = /\x1a/;
const pattern2 = new RegExp('\x1a');
```

Compliant Solution

```
const pattern1 = /\x20/;
const pattern2 = new RegExp('\x20');
```





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-6324

1/2

 Code Smell
"arguments.caller" and "arguments.callee" should not be used  Code Smell
Multiline blocks should be enclosed in curly braces  Code Smell
Boolean expressions should not be gratuitous  Code Smell
Variables should be used in the blocks where they are declared