




 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... 🔍

Multiline string literals should not be used

Code Smell

Local variables should not be declared and then immediately returned or thrown

Code Smell

Function call arguments should not start on new lines

Code Smell

"switch" statements should have at least 3 "case" clauses

Code Smell

A "while" loop should be used instead of a "for" loop

Code Smell

Unnecessary imports should be removed

Code Smell

Boolean literals should not be used in comparisons

Code Smell

Extra semicolons should be removed

Code Smell

Class names should comply with a naming convention

Code Smell

Track uses of "TODO" tags

Code Smell

Web SQL databases should not be used

Vulnerability

Variables declared with "var" should be declared before they are used

Code Smell

Regular expressions should not contain empty groups

Analyze your code

Code Smell

Major

regex

There are several reasons to use a group in a regular expression:

- to change the precedence (e.g. `do(g|or)` will match 'dog' and 'door')
- to remember parenthesised part of the match in the case of capturing group
- to improve readability

In any case, having an empty group is most probably a mistake. Either it is a leftover after refactoring and should be removed, or the actual parentheses were intended and were not escaped.

Noncompliant Code Example

```
const dateRegex = /^(?:0[1-9]|[12][0-9]|3[01])[ -./](?:0[1-9]
const methodCallRegex = /foo()/; // Noncompliant, will match
```

Compliant Solution

```
const dateRegex = /^(?:0[1-9]|[12][0-9]|3[01])[ -./](?:0[1-9]
const methodCallRegex = /foo\(\)/; // OK, matches 'foo()'
```





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-6331

1/2

<div>Track lack of copyright and license headers</div> <div> Code Smell</div>
<div>Reading the Standard Input is security-sensitive</div> <div> Security Hotspot</div>
<div>Using command line arguments is security-sensitive</div> <div> Security Hotspot</div>
<div>Using Sockets is security-sensitive</div> <div> Security Hotspot</div>