




 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


Server-side requests should not be vulnerable to forging attacks

 Vulnerability


Non-empty statements should change control flow or have at least one side-effect

 Bug


Regular expressions with the global flag should be used with caution

 Bug


Replacement strings should reference existing regular expression groups

 Bug


Regular expressions should not contain control characters

 Bug


Alternation in regular expressions should not contain empty alternatives

 Bug


Mocha timeout should be disabled by setting it to "0".

 Bug


Unicode Grapheme Clusters should be avoided inside regex character classes

 Bug

Assertions should not be given twice the same argument

 Bug





Alternatives in regular expressions should be grouped when used with anchors

 Bug

Promise rejections should not be caught by 'try' block

Regular expressions should be syntactically valid

Analyze your code

 Bug  Critical   regex

Regular expressions have their own syntax that is understood by regular expression engines. Those engines will throw an exception at runtime if they are given a regular expression that does not conform to that syntax.

To avoid syntax errors, special characters should be escaped with backslashes when they are intended to be matched literally and references to capturing groups should use the correctly spelled name or number of the group.

To match a literal string, rather than a regular expression, either all special characters should be escaped or methods that don't use regular expressions should be used.




Noncompliant Code Example

```
new RegExp(" ( ");
str.match(" ( ");
```

Compliant Solution

```
new RegExp("\\(\\[ ");
str.match("\\(\\[ ");
str.replace(" ( ", "{");
```





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](#)

https://rules.sonarsource.com/typescript/RSPEC-5856

1/2

 Bug
Collection elements should not be replaced unconditionally  Bug
Constructors should not be declared inside interfaces  Bug
Errors should not be created without being thrown  Bug
Collection sizes and array length comparisons should make sense