




 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... 🔍

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

Code Smell

Track lack of copyright and license headers

Code Smell

Reading the Standard Input is security-sensitive

Security Hotspot

Using command line arguments is security-sensitive

Security Hotspot

Using Sockets is security-sensitive

Security Hotspot

Executing XPath expressions is security-sensitive

Security Hotspot

Encrypting data is security-sensitive

Security Hotspot

Using regular expressions is security-sensitive

Security Hotspot

Class methods should be used instead of "prototype" assignments

Code Smell

Variables should be declared with "let" or "const"

Code Smell

Unchanged variables should be marked "const"

Code Smell

Wildcard imports should not be used

Code Smell

Character classes in regular expressions should not contain the same character twice

Analyze your code

Code Smell

Major ?

regex

Character classes in regular expressions are a convenient way to match one of several possible characters by listing the allowed characters or ranges of characters. If the same character is listed twice in the same character class or if the character class contains overlapping ranges, this has no effect.

Thus duplicate characters in a character class are either a simple oversight or a sign that a range in the character class matches more than is intended or that the author misunderstood how character classes work and wanted to match more than one character. A common example of the latter mistake is trying to use a range like [ 0-99 ] to match numbers of up to two digits, when in fact it is equivalent to [ 0-9 ]. Another common cause is forgetting to escape the - character, creating an unintended range that overlaps with other characters in the character class.




Noncompliant Code Example

```
/[0-99]/ // Noncompliant, this won't actually match strings
/[0-9.-_]/ // Noncompliant, .-_ is a range that already cont
```

Compliant Solution

```
/[0-9]{1,2}/
/[0-9.-_\]/
```





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-5869

1/2

<div><div>"switch" statements should not be nested</div><div> Code Smell</div></div>
<div><div>Cyclomatic Complexity of functions should not be too high</div><div> Code Smell</div></div>
<div><div>"strict" mode should be used with caution</div><div> Code Smell</div></div>
<div><div>Control flow statements "if", "for", "while", "switch" and "try" should not be nested too deeply</div><div> Code Smell</div></div>