




 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

Code Smell
Primitive types should be omitted from initialized or defaulted declarations
Code Smell
Non-null assertions should not be used
Code Smell
"undefined" should not be assigned
Code Smell
Trailing commas should not be used
Code Smell
Array constructors should not be used
Code Smell
Quotes for string literals should be used consistently
Code Smell
Statements should end with semicolons
Code Smell
Comments should not be located at the end of lines of code
Code Smell
Loops should not contain more than a single "break" or "continue" statement
Code Smell
Variable, property and parameter names should comply with a naming convention
Code Smell
Lines should not end with trailing whitespaces
Code Smell

Regular expression quantifiers and character classes should be used concisely

Analyze your code

Code Smell

Minor

regex

With regular expressions syntax, it's possible to express the same thing in many ways. For example, to match a two-digit number, one could write `[0-9]{2,2}` or `\d{2}`. Latter is not only shorter in terms of expression length, but also easier to read and thus to maintain. This rule recommends to replace some bulky quantifiers and character classes with more concise equivalents:

- `\d` for `[0-9]` and `\D` for `[^0-9]`
- `\w` for `[A-Za-z0-9_]` and `\W` for `[^A-Za-z0-9_]`
- `.` for character classes matching everything (e.g. `[\w\W]`, `[\d\D]`, or `[\s\S]` with `s` flag)
- `x?` for `x{0,1}`, `x*` for `x{0,}`, `x+` for `x{1,}`, `x{N}` for `x{N,N}`

Noncompliant Code Example

```
/a{1,}/; // Noncompliant, '{1,}' quantifier is the same as '[A-Za-z0-9_]'; // Noncompliant, '\w' is equivalent
```

Compliant Solution

```
/a+/;  
/\w/;
```

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](#)

<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>