

sonar


RULES


Products


▼


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


Primitive types should be omitted from initialized or defaulted declarations




Non-null assertions should not be used




"undefined" should not be assigned




Trailing commas should not be used




Array constructors should not be used




Quotes for string literals should be used consistently




Statements should end with semicolons




Comments should not be located at the end of lines of code




Loops should not contain more than a single "break" or "continue" statement



Variable, property and parameter names should comply with a naming convention





Lines should not end with trailing whitespaces





Private properties that are only assigned in the constructor or at declaration should be "readonly"

Analyze your code

Code Smell

Major

Quick Fix

confusing

readonly properties can only be assigned in a class constructor or at the point of declaration. If a class has a property that's not marked `readonly` but is only set in the constructor, it could cause confusion about the property's intended use. To avoid confusion, such properties should be marked `readonly` to make their intended use explicit, and to prevent future maintainers from inadvertently changing their use.

Noncompliant Code Example

```
class Person {
  private _birthYear: number; // Noncompliant
  constructor(birthYear: number) {
    this._birthYear = birthYear;
  }
}
```

Compliant Solution

```
class Person {
  private readonly _birthYear: number;
  constructor(birthYear: number) {
    this._birthYear = birthYear;
  }
}
```

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-2933

1/2

<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>