




 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... 🔍


 Bug


Results of operations on strings should not be ignored

 Bug


 Code Smell


Increment (++) and decrement (--) operators should not be used in a method call or mixed with other operators in an expression

 Code Smell


 Code Smell


Optional boolean parameters should have default value

 Code Smell


 Code Smell


Union types should not have too many elements

 Code Smell


 Code Smell

Dependencies should be explicit

 Code Smell

 Code Smell

"this" should not be assigned to variables

 Code Smell

 Code Smell

The "any" type should not be used

 Code Smell

 Code Smell

"for in" should not be used with iterables

 Code Smell

 Code Smell

Functions should use "return" consistently

 Code Smell

 Code Smell

"arguments" should not be accessed directly

 Code Smell


 Code Smell


Comparison operators should not be used with strings


 Code Smell


Literals should not be thrown

Analyze your code

 Code Smell

 Major

 Quick Fix

 error-handling api-design

It is a bad practice to throw something that's not derived at some level from `Error`. If you can't find an existing `Error` type that suitably conveys what you need to convey, then you should extend `Error` to create one.

Specifically, part of the point of throwing `Errors` is to communicate about the conditions of the error, but literals have far less ability to communicate meaningfully than `Errors` because they don't include stacktraces.




Noncompliant Code Example

```
throw 404; // Noncompliant
throw "Invalid negative index."; // Noncompliant
```

Compliant Solution

```
throw new Error("Status: " + 404);
throw new Error("Invalid negative index.");{code}
```





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-3696

1/2

<p>Private properties that are only assigned in the constructor or at declaration should be "readonly"</p> <p> Code Smell</p>
<p>Property getters and setters should come in pairs</p> <p> Code Smell</p>
<p>JavaScript parser failure</p> <p> Code Smell</p>
<p>The ternary operator should not be used</p> <p> Code Smell</p>