




 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... 🔍


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

 Code Smell


Optional boolean parameters should have default value

 Code Smell


Union types should not have too many elements

 Code Smell


Dependencies should be explicit

 Code Smell


"this" should not be assigned to variables

 Code Smell


The "any" type should not be used

 Code Smell


"for in" should not be used with iterables

 Code Smell


Functions should use "return" consistently

 Code Smell


"arguments" should not be accessed directly

 Code Smell

Comparison operators should not be used with strings

 Code Smell


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


 Code Smell


Property getters and setters should

Array indexes should be numeric

Analyze your code

 Code Smell

 Major ?

 bad-practice

Associative arrays allow you to store values in an array with either numeric or named indexes. But creating and populating an object is just as easy as an array, and more reliable if you need named members.




Noncompliant Code Example

```
let arr = [];  
arr[0] = 'a';  
arr['name'] = 'bob'; // Noncompliant  
arr[1] = 'foo';
```

Compliant Solution

```
let obj = {  
  name: 'bob',  
  arr: ['a', 'foo']  
};
```





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-3579

1/2

<b>come in pairs</b>  Code Smell
<b>JavaScript parser failure</b>  Code Smell
<b>The ternary operator should not be used</b>  Code Smell
<b>"===" and "!===" should be used instead of "==" and "!="</b>  Code Smell
<b>Functions should not have too many lines of code</b>