




 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




### Destructuring syntax should be used for assignments

Analyze your code

 Code Smell

 Minor ?

 es2015 clumsy

ECMAScript 2015 introduced the ability to extract and assign multiple data points from an object or array simultaneously. This is called "destructuring", and it allows you to condense boilerplate code so you can concentrate on logic.

This rule raises an issue when multiple pieces of data are extracted out of the same object or array and assigned to variables.

#### Noncompliant Code Example

```
function foo (obj1, obj2, array) {
  var a = obj1.a; // Noncompliant
  var b = obj1.b;

  var name = obj2.name; // ignored; there's only one extract




  var zero = array[0]; // Noncompliant
  var one = array[1];
}
```

#### Compliant Solution

```
function foo (obj1, obj2, array) {
  var {a, b} = obj1;

  var {name} = obj2; // this syntax works because var name

  var [zero, one] = array;
}
```

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-3514

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>