




 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


lines of code

 Code Smell


Track comments matching a regular expression

 Code Smell


Statements should be on separate lines

 Code Smell


Magic numbers should not be used

 Code Smell


Collapsible "if" statements should be merged

 Code Smell


Standard outputs should not be used directly to log anything

 Code Smell


Files should not have too many lines of code

 Code Smell


Lines should not be too long

 Code Smell


Debugger statements should not be used

 Vulnerability


Regular expressions using Unicode character classes or property escapes should enable the unicode flag

 Bug

The base should be provided to "parseInt"

 Bug


Function declarations should not be made within blocks


 Bug


Tags ▾

Search by name... 🔍

### Variables should be used in the blocks where they are declared

 Code Smell

 Major ?

 pitfall

Variables that are declared inside a block but used outside of it (which is possible with a var-style declaration) should be declared outside the block.

#### Noncompliant Code Example

```
function doSomething(a, b) {
  if (a > b) {
    var x = a - b; // Noncompliant
  }

  if (a > 4) {
    console.log(x);
  }

  for (var i = 0; i < m; i++) { // Noncompliant, both loops
  }

  for (var i = 0; i < n; i++) {
  }

  return a + b;
}
```

#### Compliant Solution

```
function doSomething(a, b) {
  var x;

  if (a > b) {
    x = a - b;
  }

  if (a > 4) {
    console.log(x);
  }

  for (let i = 0; i < m; i++) {
  }

  for (let i = 0; i < n; i++) {
  }


  return a + b;
}
```

Available In:


https://rules.sonarsource.com/typescript/RSPEC-2392

1/2


Writing cookies is security-sensitive

 Security Hotspot


"continue" should not be used

 Code Smell




Primitive return types should be used

 Code Smell

Default type parameters should be omitted

 Code Smell

Type assertions should use "as"

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](#)