




 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...


Allowing mixed-content is security-sensitive

 Security Hotspot


Disabling content security policy fetch directives is security-sensitive

 Security Hotspot


Disabling resource integrity features is security-sensitive

 Security Hotspot


Disclosing fingerprints from web application technologies is security-sensitive

 Security Hotspot


Having a permissive Cross-Origin Resource Sharing policy is security-sensitive

 Security Hotspot


Delivering code in production with debug features activated is security-sensitive

 Security Hotspot


Creating cookies without the "HttpOnly" flag is security-sensitive

 Security Hotspot


Creating cookies without the "secure" flag is security-sensitive

 Security Hotspot


Using hardcoded IP addresses is security-sensitive

 Security Hotspot

Regular expression quantifiers and character classes should be used concisely



 Code Smell

Regular expression literals should be used when possible

 Code Smell

Identical expressions should not be used on both sides of a binary operator

Analyze your code

 Bug Major

Using the same value on either side of a binary operator is almost always a mistake. In the case of logical operators, it is either a copy/paste error and therefore a bug, or it is simply wasted code, and should be simplified. In the case of bitwise operators and most binary mathematical operators, having the same value on both sides of an operator yields predictable results, and should be simplified.

Noncompliant Code Example

```
if (a == b && a == b) { // if the first one is true, the second one is also true
  doX();
}
if (a > a) { // always false
  doW();
}

var j = 5 / 5; //always 1
var k = 5 - 5; //always 0
```

Exceptions

The specific case of testing one variable against itself is a valid test for NaN and is therefore ignored.

Similarly, left-shifting 1 onto 1 is common in the construction of bit masks, and is ignored.

Moreover comma operator , and instanceof operator are ignored as there are use-cases when there usage is valid.




```
if (f !== f) { // test for NaN value
  console.log("f is NaN");
}

var i = 1 << 1; // Compliant
var j = a << a; // Noncompliant
```

See

- {rule:javascript:S1656} - Implements a check on =.





Available In:

 |  | 

https://rules.sonarsource.com/typescript/RSPEC-1764

© 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.

1/2

| |
|--|
| |
| "await" should not be used redundantly  Code Smell |
| Redundant casts and non-null assertions should be avoided  Code Smell |
| Type aliases should be used  Code Smell |
| Type guards should be used  Code Smell |
| |