




 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



JavaScript static code analysis

Unique rules to find Bugs, Vulnerabilities, Security Hotspots, and Code Smells in your JAVASCRIPT code

- All rules 285
- Vulnerability 29
- Bug 62
- Security Hotspot 43
- Code Smell 151
- Quick Fix 41

Variables should not be self-assigned	Bug
Function argument names should be unique	Bug
Property names should not be duplicated within a class or object literal	Bug
Bitwise operators should not be used in boolean contexts	Bug
Constructing arguments of system commands from user input is security-sensitive	Security Hotspot
Allowing requests with excessive content length is security-sensitive	Security Hotspot
Statically serving hidden files is security-sensitive	Security Hotspot
Using intrusive permissions is security-sensitive	Security Hotspot
Disabling auto-escaping in template engines is security-sensitive	Security Hotspot
Using shell interpreter when executing OS commands is security-sensitive	Security Hotspot
Setting loose POSIX file permissions is security-sensitive	Security Hotspot

Tags ▾

Search by name... 🔍

"await" should only be used with promises

Analyze your code

Code Smell

Critical

confusing

It is possible to use `await` on values which are not Promises, but it's useless and misleading. The point of `await` is to pause execution until the Promise's asynchronous code has run to completion. With anything other than a Promise, there's nothing to wait for.

This rule raises an issue when an awaited value is guaranteed not to be a Promise.

Noncompliant Code Example

```
let x = 42;
await x; // Noncompliant
```

Compliant Solution





```
let x = new Promise(resolve => resolve(42));
await x;

let y = p ? 42 : new Promise(resolve => resolve(42));
await y;
```

Available In:

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](#)

<div>Formatting SQL queries is security-sensitive</div> <div> Security Hotspot</div>
<div>Comma operator should not be used</div> <div> Code Smell</div>
<div>Regular expressions should not contain empty groups</div> <div> Code Smell</div>
<div>Regular expressions should not contain multiple spaces</div> <div> Code Smell</div>
<div>Chai assertions should have only one</div>