












































-  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

- Object literal shorthand syntax should be used
-  Code Smell
- Strings and non-strings should not be added
-  Code Smell
- Object literal syntax should be used
-  Code Smell
- "undefined" should not be assigned
-  Code Smell
- Trailing commas should not be used
-  Code Smell
- Array constructors should not be used
-  Code Smell
- Quotes for string literals should be used consistently
-  Code Smell
- Statements should end with semicolons
-  Code Smell
- Comments should not be located at the end of lines of code
-  Code Smell
- Loops should not contain more than a single "break" or "continue" statement
-  Code Smell
- Variable, property and parameter names should comply with a naming convention
-  Code Smell
- Lines should not end with trailing whitespaces
-  Code Smell

Tags ▾

Search by name... 

"await" should not be used redundantly

Analyze your code

-  Code Smell
-  Minor 
-  redundant

An async function always wraps the return value in a Promise. Using return await is therefore redundant.

Noncompliant Code Example

```
async function foo() {
  // ...
}

async function bar() {
  // ...
  return await foo(); // Noncompliant
}
```

Compliant Solution





```
async function foo() {
  // ...
}

async function bar() {
  // ...
  return 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](#)

<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>