




 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

Tags ▾

Search by name... 🔍

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

"await" should not be used redundantly

Code Smell

"for of" should be used with Iterables

Code Smell

Imports from the same modules should be merged

Code Smell

Variables should not be self-assigned

Analyze your code

Bug Major ?

There is no reason to re-assign a variable to itself. Either this statement is redundant and should be removed, or the re-assignment is a mistake and some other value or variable was intended for the assignment instead.

Noncompliant Code Example

```
function setName(name) {
    name = name;
}
```

Compliant Solution

```
function setName(name) {
    this.name = name;
}
```





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](#)

https://rules.sonarsource.com/javascript/RSPEC-1656

1/2

<div><div>Jump statements should not be redundant</div><div> Code Smell</div></div>
<div><div>Default export names and file names should match</div><div> Code Smell</div></div>
<div><div>The global "this" object should not be used</div><div> Code Smell</div></div>
<div><div>"catch" clauses should do more than rethrow</div><div> Code Smell</div></div>