




 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 rules285

 Vulnerability29

 Bug62

 Security Hotspot43


 Code Smell151

 Quick Fix41


Tags ▾

Search by name... 🔍


Regular expressions should not contain multiple spaces

 Code Smell


Chai assertions should have only one reason to succeed

 Code Smell


Single-character alternations in regular expressions should be replaced with character classes

 Code Smell


Reluctant quantifiers in regular expressions should be followed by an expression that can't match the empty string

 Code Smell


Tests should check which exception is thrown

 Code Smell


Character classes in regular expressions should not contain the same character twice

 Code Smell


Names of regular expressions named groups should be used

 Code Smell


Regular expressions should not be too complicated

 Code Smell


Shorthand promises should be used

 Code Smell


Template literals should not be nested

 Code Smell

"in" should not be used on arrays


 Code Smell


Assignments should not be redundant


 Code Smell


Functions should not be empty

Analyze your code

 Code Smell

 Critical

 Quick Fix

 suspicious

There are several reasons for a function not to have a function body:

- It is an unintentional omission, and should be fixed to prevent an unexpected behavior in production.
- It is not yet, or never will be, supported. In this case an exception should be thrown in languages where that mechanism is available.
- The method is an intentionally-blank override. In this case a nested comment should explain the reason for the blank override.




Noncompliant Code Example

```
function foo() {  
}  
  
var foo = () => {};
```

Compliant Solution

```
function foo() {  
    // This is intentional  
}  
  
var foo = () => {  
    do_something();  
};
```

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](#)

https://rules.sonarsource.com/javascript/RSPEC-1186

1/2

Assignments should not be redundant
 Code Smell
Functions should not have identical implementations
 Code Smell
Sparse arrays should not be declared
 Code Smell
Array-mutating methods should not be used misleadingly
 Code Smell
Collection and array contents should be used