




 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... 🔍

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

Formatting SQL queries is security-sensitive

Security Hotspot

Comma operator should not be used

Code Smell

Regular expressions should not contain empty groups

Code Smell

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 explicit quantifier

Code Smell

### Conditionals should start on new lines

Analyze your code

Code Smell

Critical

Quick Fix

suspicious

Code is clearest when each statement has its own line. Nonetheless, it is a common pattern to combine on the same line an *if* and its resulting *then* statement. However, when an *if* is placed on the same line as the closing *}* from a preceding *then*, *else* or *else if* part, it is either an error - *else* is missing - or the invitation to a future error as maintainers fail to understand that the two statements are unconnected.

#### Noncompliant Code Example

```
if (condition1) {
  // ...
} if (condition2) { // Noncompliant
  //...
}
```

#### Compliant Solution

```
if (condition1) {
  // ...
} else if (condition2) {
  //...
}
```

Or

```
if (condition1) {
  // ...
}

if (condition2) {
  //...
}
```

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-3972

1/2

<div>expressions should be followed by an expression that can't match the empty string</div> <div> Code Smell</div>
<div>Tests should check which exception is thrown</div> <div> Code Smell</div>
<div>Character classes in regular expressions should not contain the same character twice</div> <div> Code Smell</div>
<div>Names of regular expressions named groups should be used</div>