




 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... 🔍


Comparison operators should not be used with strings




Property getters and setters should come in pairs




JavaScript parser failure




The ternary operator should not be used




"===" and "!==" should be used instead of "==" and "!="




Functions should not have too many lines of code




Track comments matching a regular expression




Statements should be on separate lines




Magic numbers should not be used




Collapsible "if" statements should be merged



Standard outputs should not be used directly to log anything






Files should not have too many lines of code



Ternary operators should not be nested

Analyze your code

 Code Smell  Major ?  confusing

Just because you *can* do something, doesn't mean you should, and that's the case with nested ternary operations. Nesting ternary operators results in the kind of code that may seem clear as day when you write it, but six months later will leave maintainers (or worse - future you) scratching their heads and cursing.

Instead, err on the side of clarity, and use another line to express the nested operation as a separate statement.




Noncompliant Code Example

```
function getReadableStatus(job) {
  return job.isRunning() ? "Running" : job.hasErrors() ? "Failed" : "Succeeded";
}
```

Compliant Solution

```
function getReadableStatus(job) {
  if (job.isRunning()) {
    return "Running";
  }
  return job.hasErrors() ? "Failed" : "Succeeded";
}
```





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-3358

1/2

<div>Lines should not be too long</div> <div> Code Smell</div>
<div>Debugger statements should not be used</div> <div> Vulnerability</div>
<div>"alert(...)" should not be used</div> <div> Vulnerability</div>
<div>Regular expressions using Unicode character classes or property escapes should enable the unicode flag</div> <div> Bug</div>