




 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

Array indexes should be numeric	Code Smell
Assertion arguments should be passed in the correct order	Code Smell
Ternary operators should not be nested	Code Smell
"delete" should not be used on arrays	Code Smell
Variables and functions should not be redeclared	Code Smell
"indexOf" checks should not be for positive numbers	Code Smell
"arguments.caller" and "arguments.callee" should not be used	Code Smell
Multiline blocks should be enclosed in curly braces	Code Smell
Boolean expressions should not be gratuitous	Code Smell
Variables should be used in the blocks where they are declared	Code Smell
Parameters should be passed in the correct order	Code Smell
Two branches in a conditional structure should not have exactly the	

Alternatives in regular expressions should be grouped when used with anchors

Analyze your code

Bug

Major

regex

In regular expressions, anchors `^` and `$` have higher precedence than the `|` operator. So in a regular expression like `^a1t1|a1t2|a1t3$`, `a1t1` would be anchored to the beginning, `a1t3` to the end and `a1t2` wouldn't be anchored at all. Usually the intended behavior is that all alternatives are anchored at both ends. To achieve this, a non-capturing group should be used around the alternatives.

In cases where it is intended that the anchors only apply to one alternative each, adding (non-capturing) groups around the anchors and the parts that they apply to will make it explicit which parts are anchored and avoid readers misunderstanding the precedence or changing it because they mistakenly assume the precedence was not intended.

Noncompliant Code Example

```
^a|b|c$
```

Compliant Solution

```
^(?:a|b|c)$
```

or





```
^a$|^b$|^c$
```

or, if you do want the anchors to only apply to a and c respectively:

```
(?:^a)|b|(?:c$)
```

Available In:

sonarlint | sonarcloud | sonarqube

<div>structure should not have exactly the same implementation</div> <div> Code Smell</div>
<div>Unused assignments should be removed</div> <div> Code Smell</div>
<div>Function parameters with default values should be last</div> <div> Code Smell</div>
<div>Functions should not be defined inside loops</div> <div> Code Smell</div>