TypeScript static code analysis

Unique rules to find Bugs, Vulnerabilities, Security Hotspots, and Code Smells in your TYPESCRIPT code

Multiline string literals should not be used
Code Smell

Local variables should not be declared and then immediately returned or thrown
Code Smell

Function call arguments should not start on new lines
Code Smell

"switch" statements should have at least 3 "case" clauses
Code Smell

A "while" loop should be used instead of a "for" loop
Code Smell

Unnecessary imports should be removed
Code Smell

Boolean literals should not be used in comparisons
Code Smell

Extra semicolons should be removed
Code Smell

Class names should comply with a naming convention
Code Smell

Track uses of "TODO" tags
Code Smell

Web SQL databases should not be used
Vulnerability

Variables declared with "var" should be declared before they are used
Code Smell

Regular expressions should not contain empty groups

Analyze your code

Code Smell Major regex

There are several reasons to use a group in a regular expression:

- to change the precedence (e.g. `do(g|or)` will match 'dog' and 'door')
- to remember parenthesised part of the match in the case of capturing group
- to improve readability

In any case, having an empty group is most probably a mistake. Either it is a leftover after refactoring and should be removed, or the actual parentheses were intended and were not escaped.

Noncompliant Code Example

```
const dateRegex = /^(?:0[1-9]|[12][0-9]|3[01])[- /.](?:0[1-9
const methodCallRegex = /foo()/;  // Noncompliant, will matc
```

Compliant Solution

```
const dateRegex = /^(?:0[1-9]|[12][0-9]|3[01])[- /.](?:0[1-9
const methodCallRegex = /foo\(\)/; // OK, matches 'foo()'
```

Available In:

sonarlint | sonarcloud | sonarqube

Track lack of copyright and license headers

Code Smell

Reading the Standard Input is security-sensitive

Security Hotspot

Using command line arguments is security-sensitive

Security Hotspot

Using Sockets is security-sensitive

Security Hotspot