

































-  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



## TypeScript static code analysis

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

All rules279

Vulnerability27

Bug51


Security Hotspot43

Code Smell158


Quick Fix50

Tags ▾


Search by name... 🔍

 Code Smell


"if ... else if" constructs should end with "else" clauses




Control structures should use curly braces




String literals should not be duplicated




Expressions should not be too complex




Template literal placeholder syntax should not be used in regular strings




Built-in objects should not be overridden




"for...in" loops should filter properties before acting on them




Results of operations on strings should not be ignored




Increment (++) and decrement (--) operators should not be used in a method call or mixed with other operators in an expression



Optional boolean parameters should have default value





Union types should not have too many elements




### Functions should not have identical implementations

Analyze your code

 Code Smell

 Major ?

 confusing duplicate suspicious

When two functions have the same implementation, either it was a mistake - something else was intended - or the duplication was intentional, but may be confusing to maintainers. In the latter case, the code should be refactored.

#### Noncompliant Code Example

```
function calculateCode() {
  doTheThing();
  doOtherThing();
  return code;
}

function getName() { // Noncompliant
  doTheThing();
  doOtherThing();
  return code;
}
```

#### Compliant Solution




```
function calculateCode() {
  doTheThing();
  doOtherThing();
  return code;
}

function getName() {
  return calculateCode();
}
```

#### Exceptions

Functions with fewer than 3 lines are ignored.





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/typescript/RSPEC-4144

1/2

<div>Dependencies should be explicit</div> <div> Code Smell</div>
<div>"this" should not be assigned to variables</div> <div> Code Smell</div>
<div>The "any" type should not be used</div> <div> Code Smell</div>
<div>"for in" should not be used with iterables</div> <div> Code Smell</div>