




 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 rules 279

Vulnerability 27

Bug 51


Security Hotspot 43


Code Smell 158


Quick Fix 50


Tags ▾


Search by name... 🔍


Mocha timeout should be disabled by setting it to "0".
 Bug


Unicode Grapheme Clusters should be avoided inside regex character classes
 Bug


Assertions should not be given twice the same argument
 Bug


Alternatives in regular expressions should be grouped when used with anchors
 Bug


Promise rejections should not be caught by 'try' block
 Bug


Collection elements should not be replaced unconditionally
 Bug

Constructors should not be declared inside interfaces
 Bug

Errors should not be created without being thrown
 Bug



Collection sizes and array length comparisons should make sense
 Bug

All branches in a conditional structure should not have exactly the same implementation
 Bug

Destructuring patterns should not be empty
 Bug

"super()" should be invoked appropriately

Analyze your code

 Bug  Critical ?

There are situations where `super ()` must be invoked and situations where `super ()` cannot be invoked.

The basic rule is: a constructor in a non-derived class cannot invoke `super ()`; a constructor in a derived class must invoke `super ()`.

Furthermore:

- `super ()` must be invoked before the `this` and `super` keywords can be used.
- `super ()` must be invoked with the same number of arguments as the base class' constructor.
- `super ()` can only be invoked in a constructor - not in any other method.
- `super ()` cannot be invoked multiple times in the same constructor.

Known Limitations




- False negatives: some issues are not raised if the base class is not defined in the same file as the current class.

Noncompliant Code Example

```
class Dog extends Animal {
  constructor(name) {
    super();
    this.name = name;
    super();           // Noncompliant
    super.doSomething();
  }
}
```

Compliant Solution






```
class Dog extends Animal {
  constructor(name) {
    super();
    this.name = name;
    super.doSomething();
  }
}
```

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-3854

1/2


<p>The output of functions that don't return anything should not be used</p> <p> Bug</p>
<p>Comma and logical OR operators should not be used in switch cases</p> <p> Bug</p>
<p>Generators should "yield" something</p> <p> Bug</p>
<p>"new" operators should be used with functions</p> <p> Bug</p>