TypeScript static code analysis

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

Regular expression quantifiers and character classes should be used concisely

Analyze your code

Code Smell | Minor | regex

With regular expressions syntax, it's possible to express the same thing in many ways. For example, to match a two-digit number, one could write `[0-9]{2,2}` or `\d{2}`. Latter is not only shorter in terms of expression length, but also easier to read and thus to maintain. This rule recommends to replace some bulky quantifiers and character classes with more concise equivalents:

- `\d` for `[0-9]` and `\D` for `[^0-9]`
- `\w` for `[A-Za-z0-9_]` and `\W` for `[^A-Za-z0-9_]`
- `.` for character classes matching everything (e.g. `[\w\W]`, `[\d\D]`, or `[\s\S]` with `s` flag)
- `x?` for `x{0,1}`, `x*` for `x{0,}`, `x+` for `x{1,}`, `x{N}` for `x{N,N}`

Noncompliant Code Example

```
/a{1,}/; // Noncompliant, '{1,}' quantifier is the same as '
/[A-Za-z0-9_]/; // Noncompliant, '\w' is equivalent
```

Compliant Solution

```
/a+/;
/\w/;
```

Available In:

sonarlint | sonarcloud | sonarqube

Files should contain an empty newline at the end

Code Smell

An open curly brace should be located at the end of a line

Code Smell

Tabulation characters should not be used

Code Smell

Function and method names should comply with a naming convention

Code Smell