



# Static Code Analysis



Dario Campagna  
Head of Research

# Static Code Analysis

Static code (or program) analysis is the analysis of computer software that is performed without actually executing programs.

- Identification of coding errors
- Individuation of duplicated code
- Computation of software metrics
- Formal methods (e.g., Hoare logic, Model Checking)

```
59 //is the element inside the visible window?
60 var a = w.scrollLeft();
61 var b = w.scrollTop();
62 var o = t.offset();
63 var x = o.left;
64 var y = o.top;
65
66 var ax = settings.accX;
67 var ay = settings.accY;
68 var th = t.height();
69 var wh = w.height();
70 var tw = t.width();
71 var ww = w.width();
72
73 if (y + th + ay >= b &&
74     y <= b + wh + ay &&
75     x + tw + ax >= a &&
76     x <= a + ww + ax) {
77     //trigger the custom event
78     if (!t.appeared) t.trigger('appear', settings.data);
79
80 } else {
81     //it scrolled out of view
82     t.appeared = false;
83 }
84 };
85
86 //create a modified fn with some additional logic
87 var modifiedFn = function() {
88     //mark the element as visible
89     t.appeared = true;
90
91     //is this supposed to happen only once?
92     if (settings.one) {
93         //remove the check
94         $(t).unbind('scroll', check);
95         $(t).bind('scroll', check, $.fn.appear.checks);
96     }
97 }
```



# Automated Tools

Automated tools that perform static code analysis can identify issues that reduce code readability, maintainability, quality.

- IDE code analysis functionalities (e.g., [IntelliJ inspections](#))
- [SonarQube](#)

The image shows a code editor snippet with a warning: "A 'NullPointerException' could be thrown, 'providedClass' is nullable here." The code is as follows:

```
246 if (Provider.class == roleTypeClass) {
247     Type providedType = ReflectionUtils.getLastTypeGenericArgument(dependencyD
248     class providedClass = ReflectionUtils.getTypeClass(providedType);
249
250     if (this.componentManager.hasComponent(providedType, dependencyDescriptor.)
251         || providedClass.isAssignableFrom(List.class) || providedClass.isA
252
253     }
254 }
```

Next to the code is a SonarQube quality gate dashboard with the following metrics:

- RELIABILITY:** 0 Bugs, Quality Gate Passed (All conditions passed)
- SECURITY:** 0 Vulnerabilities, 1 Hotspots
- MAINTAINABILITY:** 4 Code Smells, 5 Debt (min)

# Using automated tools for static-code analysis

- Check/select/configure the inspection rules in default profiles/configuration
  - Too restrictive/permissive rules
  - Rules that you prefer to ignore
  - Rules you want to check and are not included
- Review the rule documentation to confirm it checks for the expected conditions
- Use inspection results as a way to uncover information about your code
- Do not fix violations just because they are reported by the tool, take informed decisions
- Maintain the inspection rules

