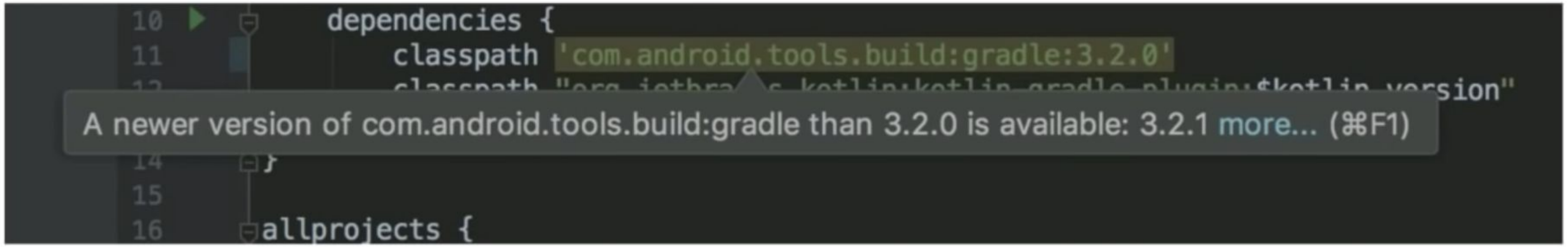
Writing Custom Lint Rules

Hitanshu Dhawan
Android Developer @ Urban Company

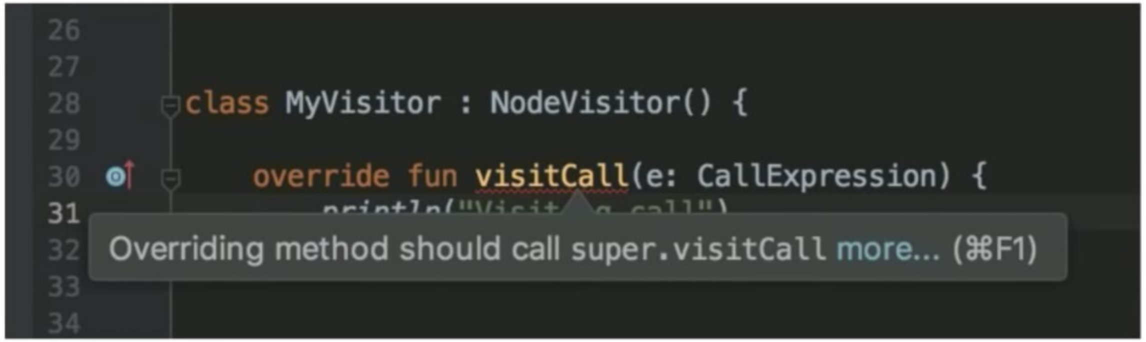
What is Lint ?



```
    dependencies {
        classpath 'com.android.tools.build:gradle:3.2.0'
        classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version"
    }

allprojects {
```

A newer version of com.android.tools.build:gradle than 3.2.0 is available: 3.2.1 more... (⌘F1)



```
class MyVisitor : NodeVisitor() {

    override fun visitCall(e: CallExpression) {
        println("Visiting call")
```

Overriding method should call super.visitCall more... (⌘F1)

Ways to use Lint

Studio

- On-the-fly
- Inspect Code...
- Run Inspection by Name...

Gradle

- `./gradlew lintDebug`
- HTML and XML reports

Let’s start...

MyTextView

Initial setup

Create a java library module for your lint rules.

```
apply plugin: 'java-library'
apply plugin: 'kotlin'


dependencies {
    // Lint
    compileOnly "com.android.tools.lint:lint-api:$lintVersion"
    compileOnly "com.android.tools.lint:lint-checks:$lintVersion"
    // Lint Testing
    testImplementation "com.android.tools.lint:lint:$lintVersion"
    testImplementation "com.android.tools.lint:lint-tests:$lintVersion"
    testImplementation "junit:junit:4.12"
}
```

```
apply plugin: 'com.android.application'

dependencies {
    lintChecks project(path: ':lint-rules')
}
```

IssueRegistry

Registry which provides a list of checks to be performed.

```kotlin
class IssueRegistry : IssueRegistry() {

    override val issues: List<Issue>
        get() = listOf(
            MyTextViewDetector.ISSUE
        )

}
```

```
// Add a file in the following location in the :lint-rules module
// resources/META-INF/services/com.android.tools.lint.client.api.IssueRegistry

com.example.IssueRegistry
```

Issue

An issue is a potential bug in an Android application.

An issue is discovered by a `Detector`, and has an associated `Severity`.

```kotlin
val ISSUE = Issue.create(
    id = "TextViewUsageWarning",
    briefDescription = "The TextView should not be used",
    explanation = "Don't use TextView, use MyTextView instead",
    category = Category.CORRECTNESS,
    priority = 3,
    severity = Severity.WARNING,
    implementation = Implementation(
        MyTextViewDetector::class.java,
        Scope.RESOURCE_FILE_SCOPE
    )
)
```

Detector

A detector is able to find a particular problem.

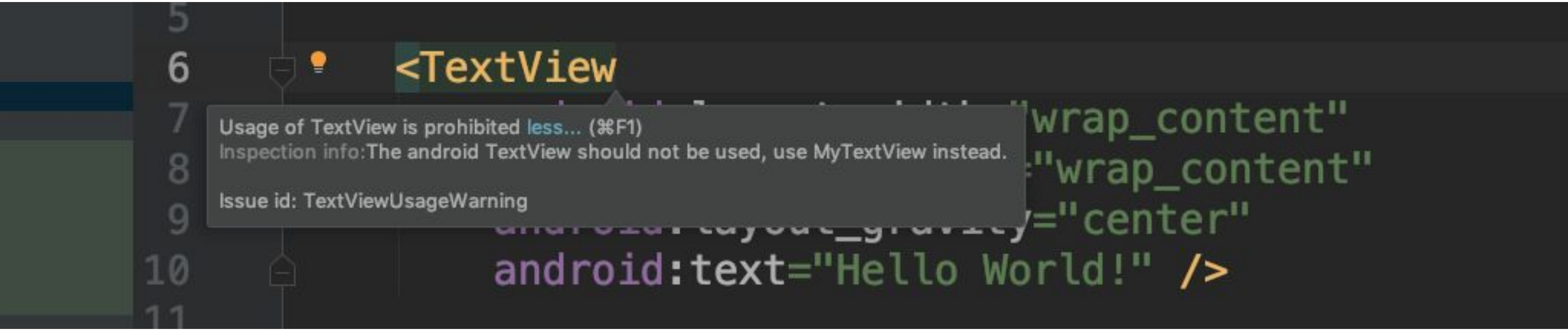
Each problem type is uniquely identified as an `Issue`.

```kotlin
class MyTextViewDetector : Detector(), XmlScanner {

    override fun getApplicableElements(): Collection<String> {
        return listOf("TextView")
    }

    override fun visitElement(context: XmlContext, element: Element) {
        context.report(
            issue = ISSUE,
            location = context.getNameLocation(element),
            message = "Usage of TextView is prohibited"
        )
    }

}
```



```
<TextView
                                  ="wrap_content"
                                  ="wrap_content"
                                 y="center"
    android:text="Hello World!" />
```

Usage of TextView is prohibited less... (⌘F1)
Inspection info:The android TextView should not be used, use MyTextView instead.

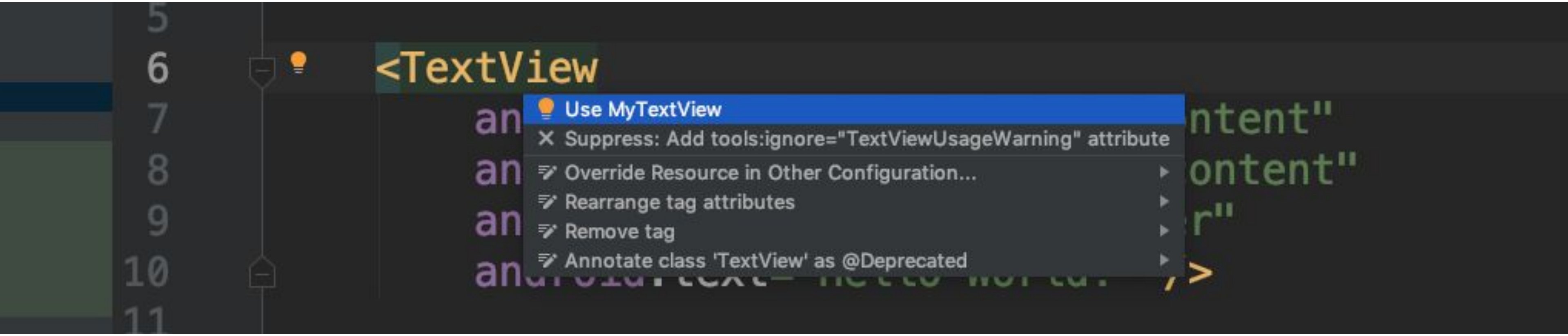
Issue id: TextViewUsageWarning

LintFix

A description of a quickfix for a lint warning, which provides structured data for use by the IDE to create an actual fix implementation.

```kotlin
val quickfixData = LintFix.create()
    .name("Use MyTextView")
    .replace()
    .text("TextView")
    .with("com.example.MyTextView")
    .autoFix()
    .build()

context.report(
    issue = ISSUE,
    location = context.getNameLocation(element),
    message = "Usage of TextView is prohibited",
    quickfixData = quickfixData
)
```



```
<TextView
    an                                    ntent"
    an                                    ontent"
    an                                    r"
    an                                    />
```

Use MyTextView
Suppress: Add tools:ignore="TextViewUsageWarning" attribute
Override Resource in Other Configuration...
Rearrange tag attributes
Remove tag
Annotate class 'TextView' as @Deprecated

Testing

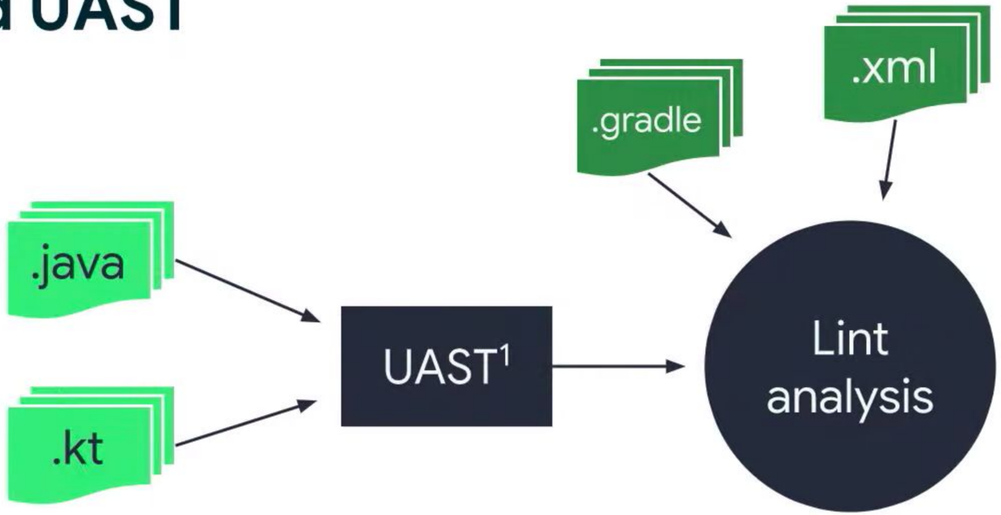
```
lint()
    .files(
        xml(
            "res/layout/layout.xml",
            """
                <merge>
                    <TextView
                        android:text="Hello World!" />
                </merge>
            """
        ).indented()
    )
    .issues(MyTextViewDetector.ISSUE)
    .run()
    .expectWarningCount(1)
```

```
lint()
    .files(...)
    .issues(MyTextViewDetector.ISSUE)
    .run()
    .expectWarningCount(1)
    .verifyFixes()
    .checkFix(
        null,
        xml(
            "res/layout/layout.xml",
            """
                <merge>
                    <com.example.MyTextView
                        android:text="Hello World!" />
                </merge>
            """
        ).indented()
    )
```

MyLog

Lint internals

PSI and UAST



[1] Universal Abstract Syntax Tree

```
class IssueRegistry : IssueRegistry() {

    override val issues: List<Issue>
        get() = listOf(
            MyTextViewDetector.ISSUE,
            MyLogDetector.ISSUE
        )

}
```

```kotlin
val ISSUE = Issue.create(
    id = "LogUsageWarning",
    briefDescription = "The Log should not be used",
    explanation = "Don't use Log, use MyLog instead",
    category = Category.CORRECTNESS,
    priority = 3,
    severity = Severity.WARNING,
    implementation = Implementation(
        MyLogDetector::class.java,
        Scope.JAVA_FILE_SCOPE
    )
)
```

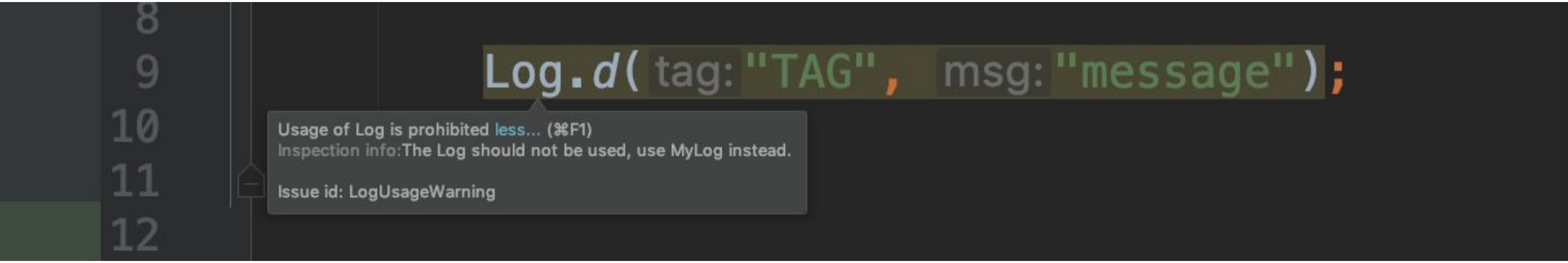
```kotlin
class MyLogDetector : Detector(), SourceCodeScanner {

    override fun getApplicableMethodNames(): List<String> {
        return listOf("v", "d", "i", "w", "e")
    }

    override fun visitMethodCall(context: ..., node: ..., method: ...) {
        val evaluator = context.evaluator
        if (evaluator.isMemberInClass(method, "android.util.Log")) {
            reportUsage(context, node, method)
        }
    }

}
```

```kotlin
private fun reportUsage(context: ..., node: ..., method: ...) {
    context.report(
        issue = MyLogDetector.ISSUE,
        scope = node,
        location = context.getCallLocation(
            call = node,
            includeReceiver = true,
            includeArguments = true
        ),
        message = "Usage of Log is prohibited"
    )
}
```

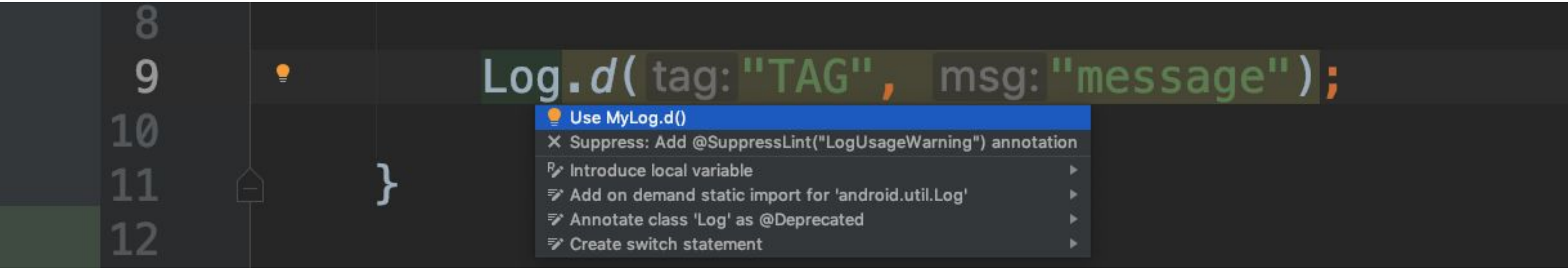


```
Log.d( tag: "TAG",  msg: "message");
```

Usage of Log is prohibited less... (⌘F1)
Inspection info:The Log should not be used, use MyLog instead.

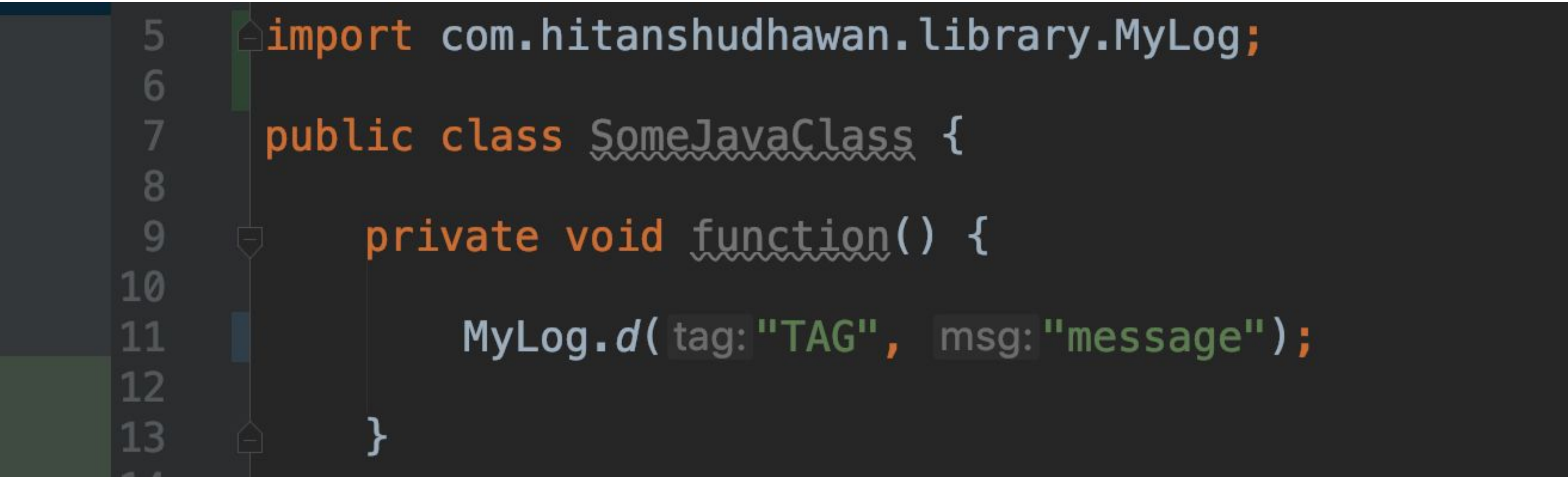
Issue id: LogUsageWarning

```
private fun reportUsage(context: ..., node: ..., method: ...) {
    context.report(
        ...
        quickfixData = fix()
            .name("Use MyLog.${method.name}()")
            .replace()
            .text("Log")
            .with("com.example.MyLog")
            .shortenNames()
            .reformat(true)
            .autoFix()
            .build()
    )
}
```



```
Log.d( tag: "TAG",  msg: "message");
}
```

Use MyLog.d()
Suppress: Add @SuppressLint("LogUsageWarning") annotation
Introduce local variable
Add on demand static import for 'android.util.Log'
Annotate class 'Log' as @Deprecated
Create switch statement



```
import com.hitanshudhawan.library.MyLog;

public class SomeJavaClass {

    private void function() {

        MyLog.d( tag: "TAG",  msg: "message");

    }
```

Let’s recap…

Resources

- ➔ GitHub Repository

  https://github.com/hitanshu-dhawan/CustomLintRules
- ➔ KotlinConf 2017 - Kotlin Static Analysis with Android Lint by Tor Norbye

  https://youtu.be/p8yX5-lPS6o
- ➔ Android source-code for Lints/Detectors

  https://android.googlesource.com/platform/tools/base/+/refs/heads/studio-master-dev/lint/libs/lint-checks/src/main/java/com/android/tools/lint/checks

Thank You...

/in/hitanshu-dhawan

/hitanshu-dhawan

/@hitanshudhawan