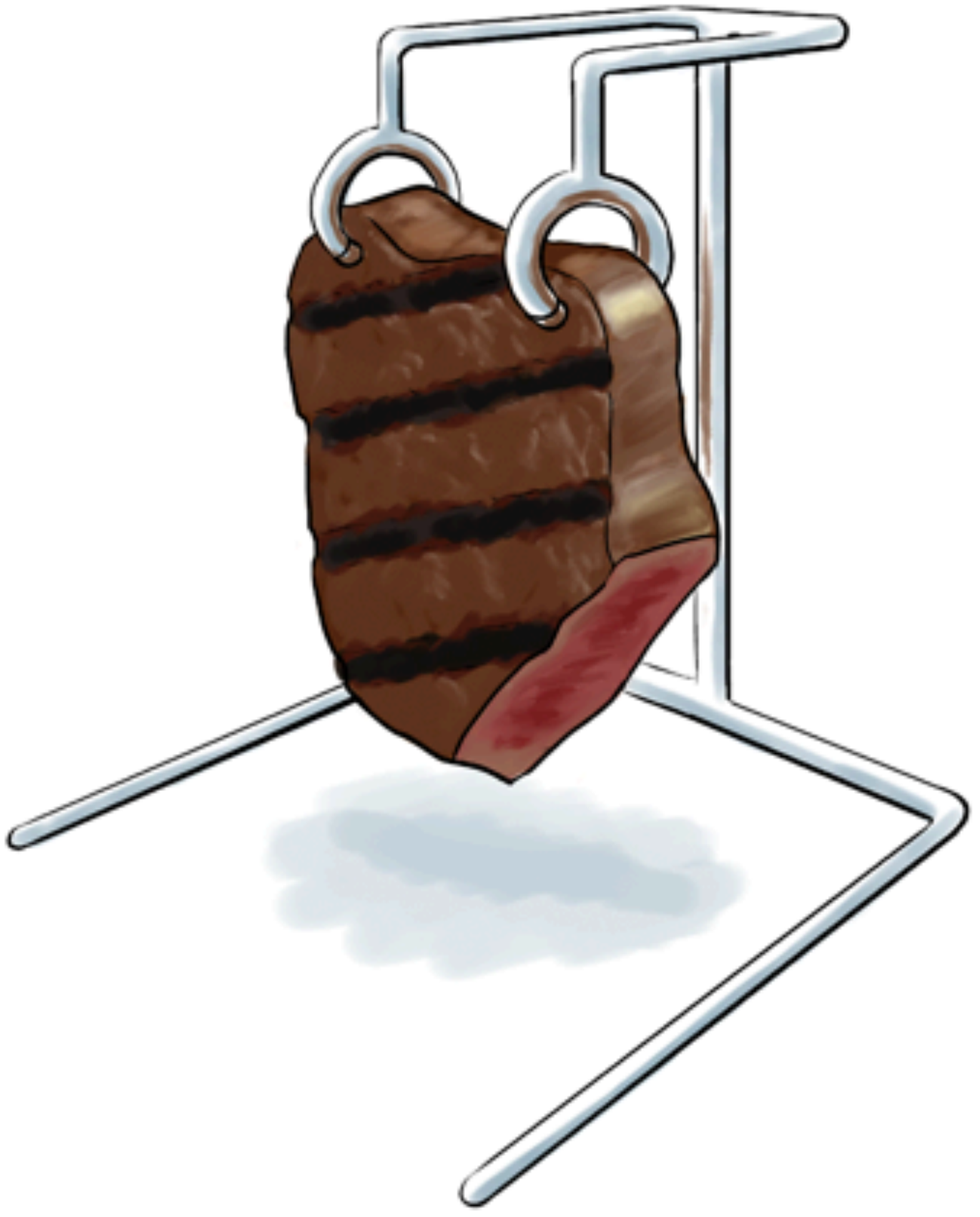
Selling Kotlin to Management

Yes we can!

I did say managers, and you probably came here for that


lied

It's truly about the stakeholders



It's truly about the stakeholders

- You

It's truly about the stakeholders

- You
- Colleagues

It's truly about the stakeholders

- You
- Colleagues
- Management and business

You??? Maybe you aren't truly convinced

Why is Kotlin right for me?

It's official

Null safety

Immutability

Code size reduction

I still was not convinced

Groovy Broke My ❤

Kotlin looked weird.

We started using GraphQL, everything was nullable

I had no desire

for crazy nested if statements.

```
if (article != null
        && article.promotionalProperties() != null
        && article.creativeWork() != null
        && article.creativeWork().bylines() != null
        && article.creativeWork().bylines().get(0) != null
        && article.creativeWork().bylines().get(0).renderedRepresentation() != null
        && article.creativeWork().summary() != null
        && article.creativeWork().headline() != null
        && article.creativeWork().headline().headlineDefault() != null) {

    String cardTypeLiteral = article.promotionalProperties().cardType();
    String byLine = article.creativeWork().bylines().get(0).renderedRepresentation();
    String summary = article.creativeWork().summary();
    String headline = article.creativeWork().headline().headlineDefault();
```

Kotlin saved me

```
val createdCard = safeLet(article,
        article.promotionalProperties()?.cardType(),
        article.creativeWork()?.bylines()?.get(0)?.renderedRepresentation(),
        article.creativeWork()?.summary(),
        article.creativeWork()?.headline()?.headlineDefault())

{ article, cardTypeLiteral, byLine, summary, headline ->

```

There's fun stuff too!

```
val disposable = programStore.get(key)
compositeDisposable += disposable
```

Using extensions!

```
operator fun CompositeDisposable.plusAssign(disposable: Disposable) {
    this.add(disposable)
}
```

That's my story, on to convincing colleagues

Maybe teammates are nervous or don't have the time to learn something new

It's all about discussion and dialogue

what you can do

what you can do

- Try sending out daily resources (articles on operators, tips, and best practices)

what you can do

- Try sending out daily resources (articles on operators, tips, and best practices)
- Examples of Kotlin conversions

what you can do

- Try sending out daily resources (articles on operators, tips, and best practices)
- Examples of Kotlin conversions
- Plan for in office working sessions to demonstrate or dev arch meetings (kotlin koans)

talking points

talking points

- Best strategy by far for me has been discussion of what people think won't
work; they usually bring up

talking points

- Best strategy by far for me has been discussion of what people think won't
  work; they usually bring up
- Don't want to switch entirely to Kotlin (counter with interop)

talking points

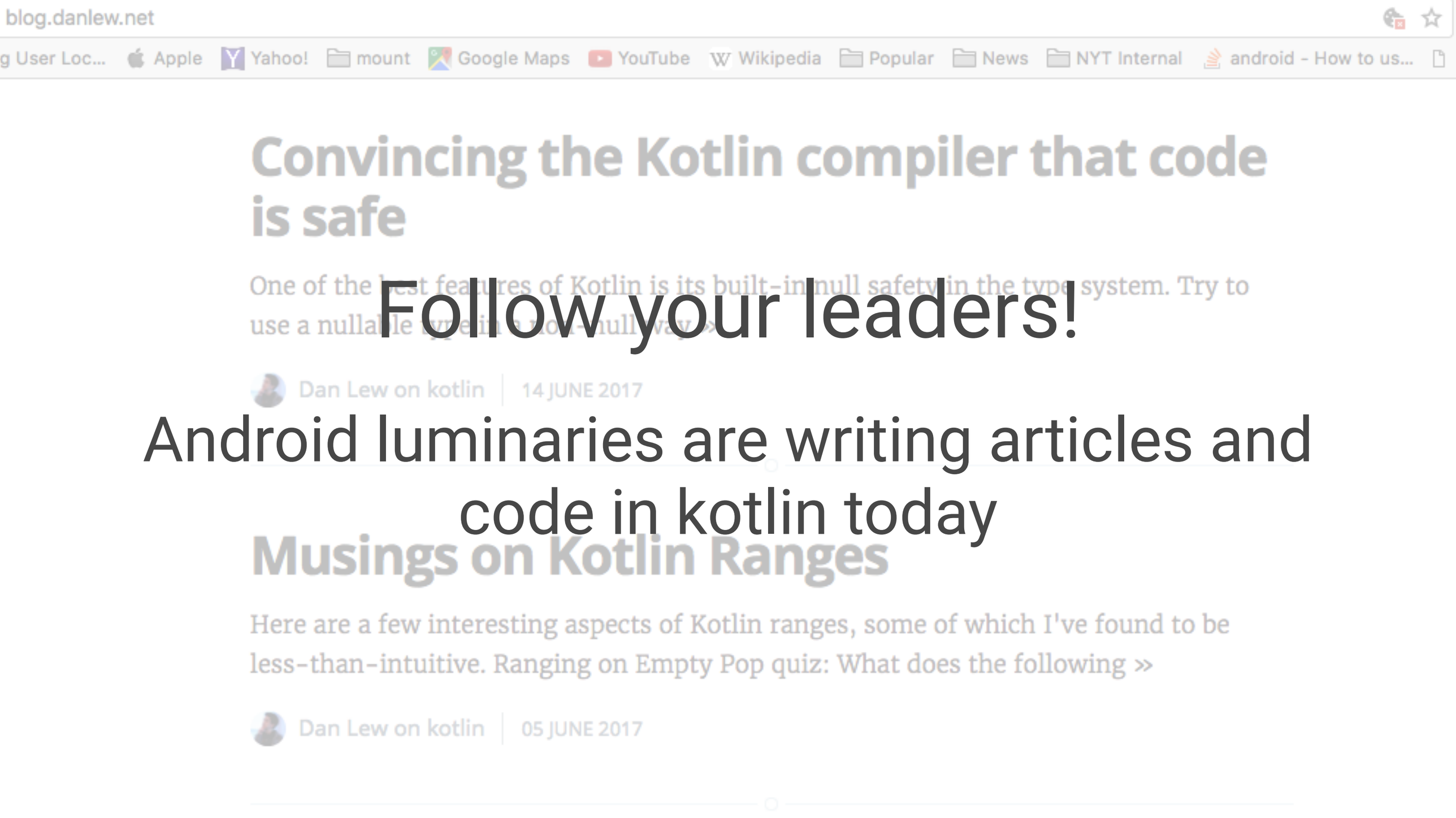
- Best strategy by far for me has been discussion of what people think won't
  work; they usually bring up
- Don't want to switch entirely to Kotlin (counter with interop)
- Is there that much of a point (counter with shorter code/ nullability)

talking points

- Best strategy by far for me has been discussion of what people think won't
  work; they usually bring up
- Don't want to switch entirely to Kotlin (counter with interop)
- Is there that much of a point (counter with shorter code/ nullability)
- support, libraries, tooling (counter with all these work)

Official Google samples are now written in Kotlin

Conference slides in Kotlin


blog.danlew.net
g User Loc... Apple Yahoo! mount Google Maps YouTube Wikipedia Popular News NYT Internal android - How to us...
Convincing the Kotlin compiler that code is safe
One of the best features of Kotlin is its built-in null safety in the type system. Try to use a nullable type in a non-null way »
Dan Lew on kotlin | 14 JUNE 2017
Musings on Kotlin Ranges
Here are a few interesting aspects of Kotlin ranges, some of which I've found to be less-than-intuitive. Ranging on Empty Pop quiz: What does the following »
Dan Lew on kotlin | 05 JUNE 2017
Follow your leaders!
Android luminaries are writing articles and code in kotlin today

On to convincing the suits!

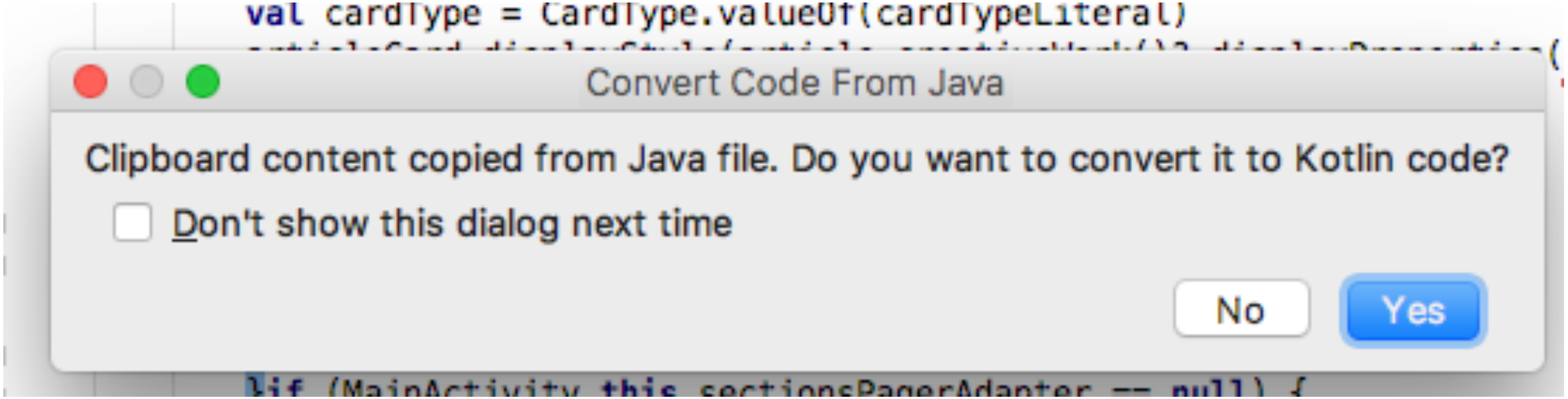
Officially supported by Google

What does this mean?,

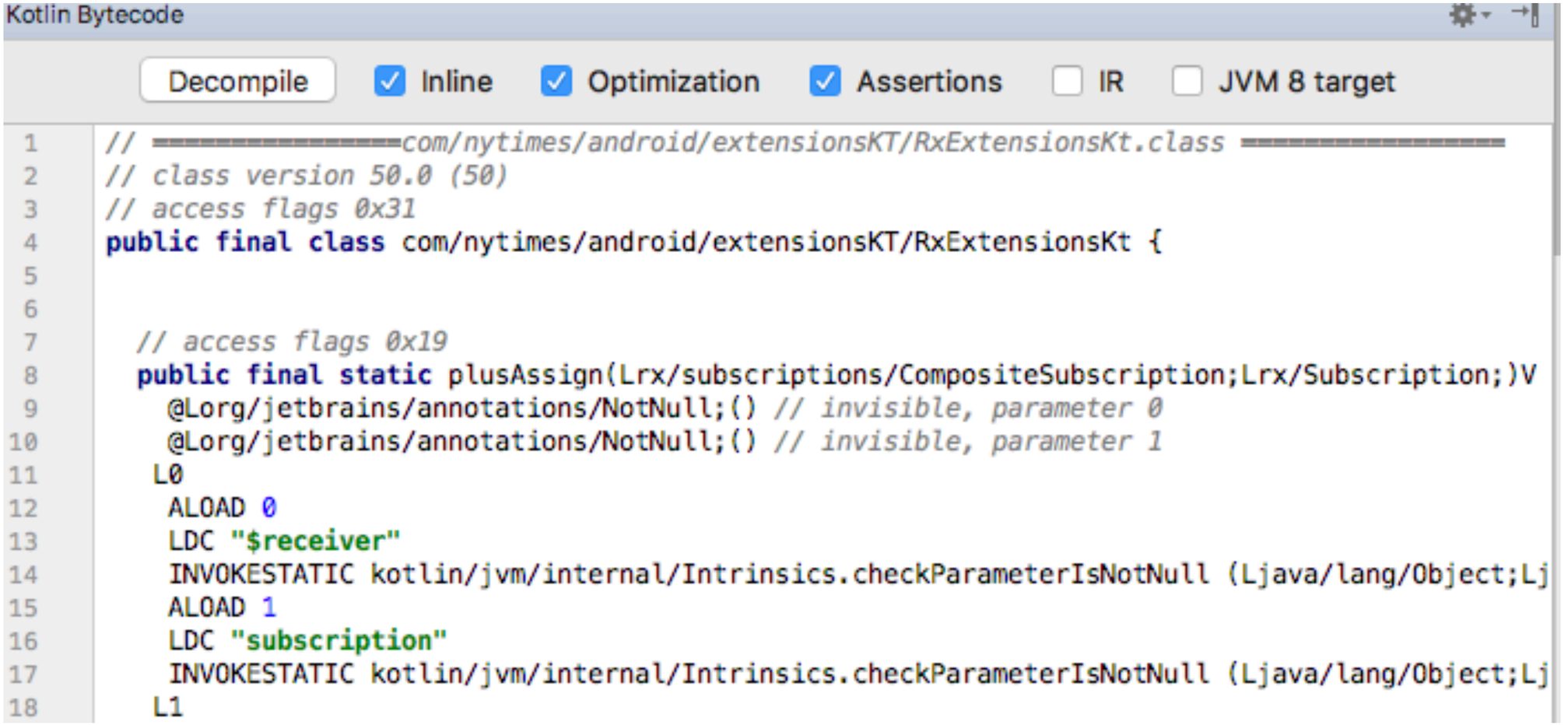
tooling works, bugs directly to Google

Jetbrains support

Copy paste in editor



Disassemble and examine bytecode



Proper test support

```
@Test
fun getConstraintSet() {
    CardConstraint.values().forEach {
        val card = mock(ArticleCard::class.java)
        Mockito.`when`(card.cardType()).thenReturn(it.cardType)
        Mockito.`when`(card.rendition()).thenReturn(it.rendition)

        if (it.columnCount > 0) {
            assertMatching(card, it, it.columnCount)
        } else {
            for (columnCount in 1..4) {
                assertMatching(card, it, columnCount)
            }
        }
    }
}
```

Works with Data Binding!

```
override fun bind(binding: CardArticleBinding, position: Int) {
        ConstraintSet().apply {
            load(binding.root.context, constraintSetLayoutRes)
            applyTo(binding.root as ConstraintLayout)
        }

        binding.root.post { binding.card = card }
    }
```

Latest Android and Gradle tools, incremental and instant compilation

3.3.0 (January 2019)

This version of the Android plugin requires the following:

- Gradle 4.10.1 or higher. To learn more, read the section about updating Gradle.

  > **Note:** When using Gradle 5.0 and higher, the default Gradle daemon memory heap size decreases from 1 GB to 512 MB. This might result in a build performance regression. To override this default setting, specify the Gradle daemon heap size in your project's **`gradle.properties`** file.

- SDK Build Tools 28.0.3 or higher.

Lastly,

It just works with everything; Picasso, RxJava, Guava, Dagger etc....

Managers don't care about any of that they care about

ROI (return on investment)

ROI

ROI

- Reduced issues overall (type safety forces safer code)

ROI

- Reduced issues overall (type safety forces safer code)
- Reduced time spent fixing production issues (less bugs, see above)

ROI

- Reduced issues overall (type safety forces safer code)
- Reduced time spent fixing production issues (less bugs, see above)
- Reduced time on boilerplate code (less code to write)

ROI

- Reduced issues overall (type safety forces safer code)
- Reduced time spent fixing production issues (less bugs, see above)
- Reduced time on boilerplate code (less code to write)
- Reduced code review time (less code to review)

Leads to:

Increased feature development &

Happier Developers

Kotlin != Golden Hammer

Good managers want to know the bad as well

The cons

The cons

- Limited static analyzers

The cons

- Limited static analyzers
- Minor increase in build time

The cons

- Limited static analyzers
- Minor increase in build time
- Mockito needs some tweaking (when, open keyword)

How to get your feet wet low risk

How to get your feet wet low risk

- Unit tests, not in source (we wrote kotlin tests that called java classes)

How to get your feet wet low risk

- Unit tests, not in source (we wrote kotlin tests that called java classes)
- Annotation processor rework or impl (apollo uses kotlin to generate java for graphql)

How to get your feet wet low risk

- Unit tests, not in source (we wrote kotlin tests that called java classes)
- Annotation processor rework or impl (apollo uses kotlin to generate java for graphql)
- Put behind interfaces to hide implementation

How to get your feet wet low risk

- Unit tests, not in source (we wrote kotlin tests that called java classes)
- Annotation processor rework or impl (apollo uses kotlin to generate java for graphql)
- Put behind interfaces to hide implementation
- Know that mistakes will be made as everyone learns it

Links and resources

Kotlin weekly (www.kotlinweekly.net)

Kotlin koans (kotlinlang.org/docs/tutorials/koans.html)

Kotlin by Example (hadihariri.com/2017/06/12/kotlin-by-example)

Intro to Kotlin Google IO 2017 (youtube.com/watch?v=X1RVYt2QKQE)