practical magic

CLOCKWORK

https://clockwork.ing

real world
kotlin
development

who are these people?

“Your hosts for today, brought to you by Clockwork”

David Denton / @daviddenton / david@clockwork.ing

- London-based independent consultant
- Founding Partner, clockwork.ing
- Google Developer Expert
- first deployed Kotlin to production in 2016
- talks at KotlinConf, BCS, Joker, GITEX
- co-author, http4k web toolkit

Ivan Sanchez / @s4nchez / ivan@clockwork.ing

- London-based independent consultant
- Partner, clockwork.ing
- Google Developer Expert
- first deployed Kotlin to production in 2016
- talks at KotlinConf, Joker, GITEX
- co-author, http4k web toolkit

CLOCKWORK
clockwork.ing

why are we here?

“Today in a nutshell”

- Kotlin is a much more succinct language than Java
- It also has a bunch of amazing power features
- But - we aren’t always starting from scratch
- We’re going to:
  - Show you how converting Java to Kotlin allows you to unlock **smaller**, **safer**, **better** code
  - Let you loose on some open source Java projects
  - Prove that your Java code probably isn’t that safe!

prerequisites

“stuff you’ll need to make this do-able!”

- IntelliJ Idea 2024 (Community/Ultimate) with latest Kotlin plugin installed
- Java 21 for simplicity

CLOCKWORK
clockwork.ing

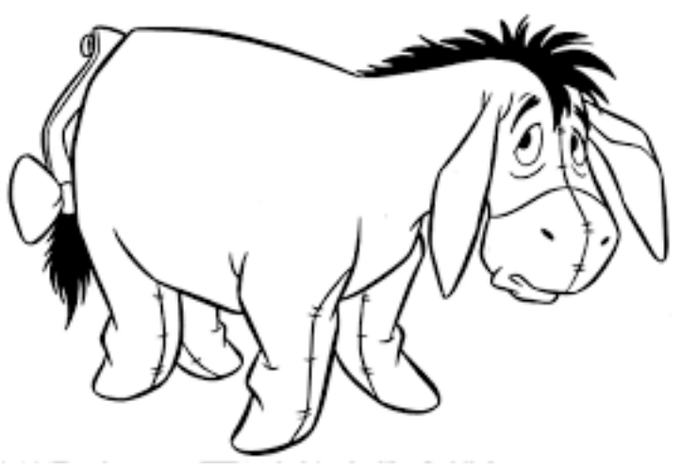
practical magic

“convert a Java codebase to kotlin”

- We’re going to practice the subjects we’ve covered on real Java codebases.
- Your job is to explore those codebases and use IntelliJ to convert bits of them to Kotlin.
- Please clone:

http://www.github.com/realworldkotlin/application

- The handouts contain some suggested starting points and exercises!



conversion gotchas

“Your Java code probably isn’t actually that safe!”

Breaks:

- **Java -> Kotlin** conversion problems:
    - **visibility modifiers** - often too strict
    - var/**property** handling of nullable types
    - **generics** on java methods - can be removed
    - **static method** handling - missing annotations to ensure compatibility

Notes:

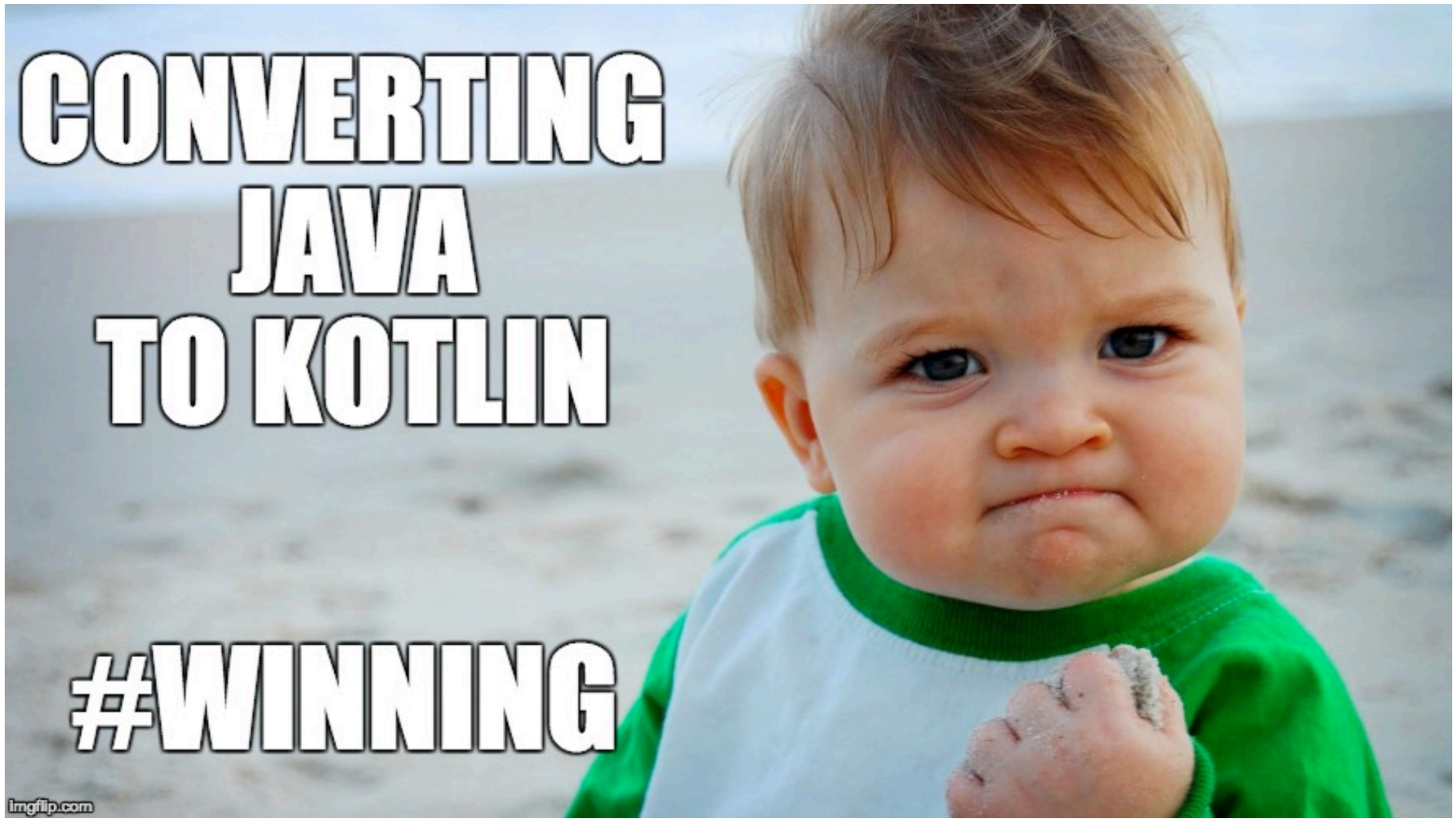
- This process highlights **Java API fragility**
- Good idea to **commit after a successful conversion**
- **Kotlin** and **Java** source code can co-exist
- **Ordering** of conversion is important!

intellij + kotlin

“automatic conversion can get you 95% of the way there”

How:

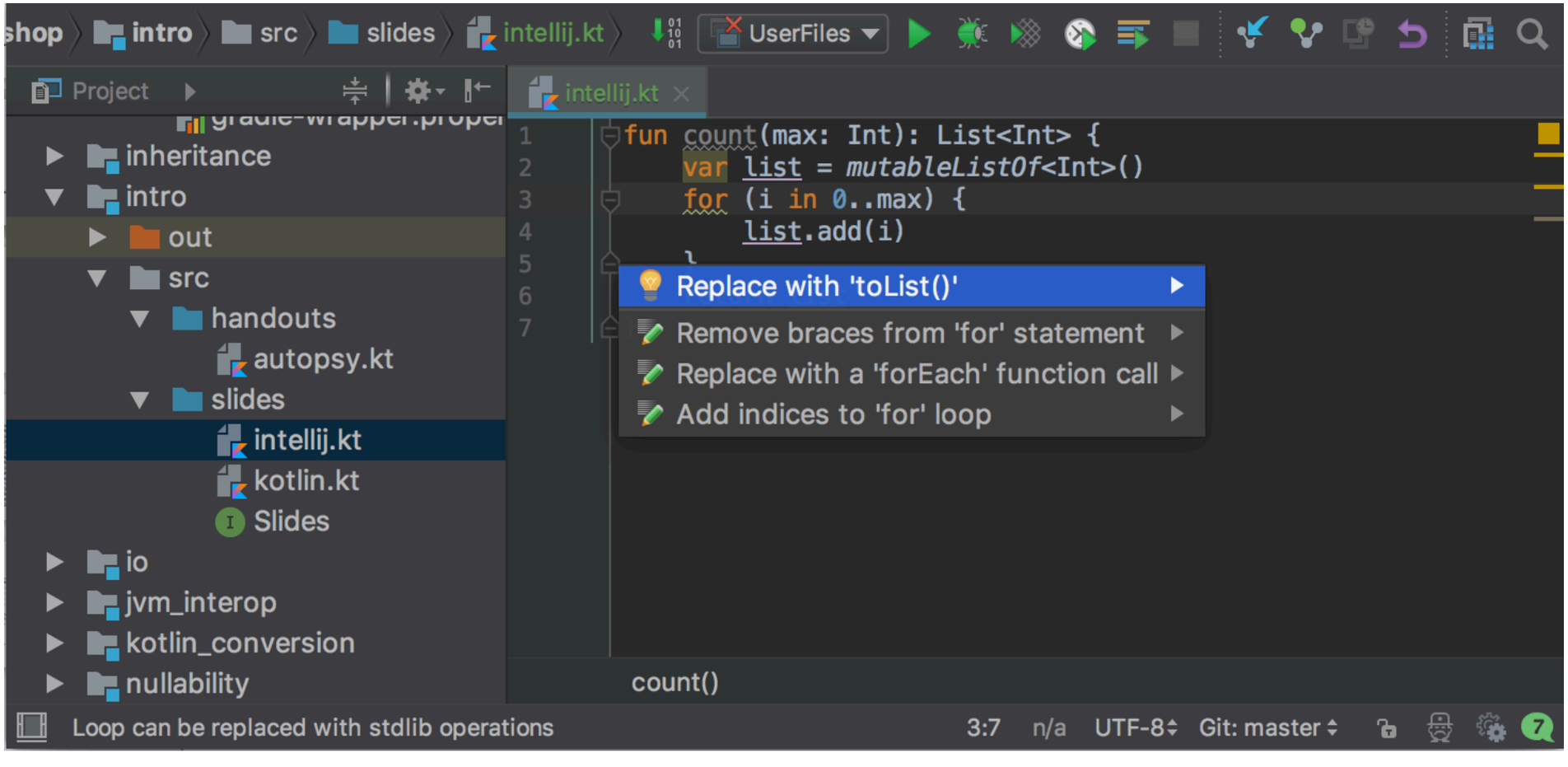
- Open “**Code**” menu -> “**Convert Java File to Kotlin File**”
- On MacOS, the shortcut is **Alt + Cmd + Shift + K**
- Works on **file**, **packages,** entire **projects**!



intelliJ intentions

“the computer is trying to tell you something”

- Any time you see an **odd colour** or **underlining** in your code, IntelliJ is trying to help
- The trough on the right of your IDE shows the current warnings
- Move the carat or hit **F2,** then press **Alt + Enter** to bring up the intentions menu



<<- Look here!

CLOCKWORK

hints for success!

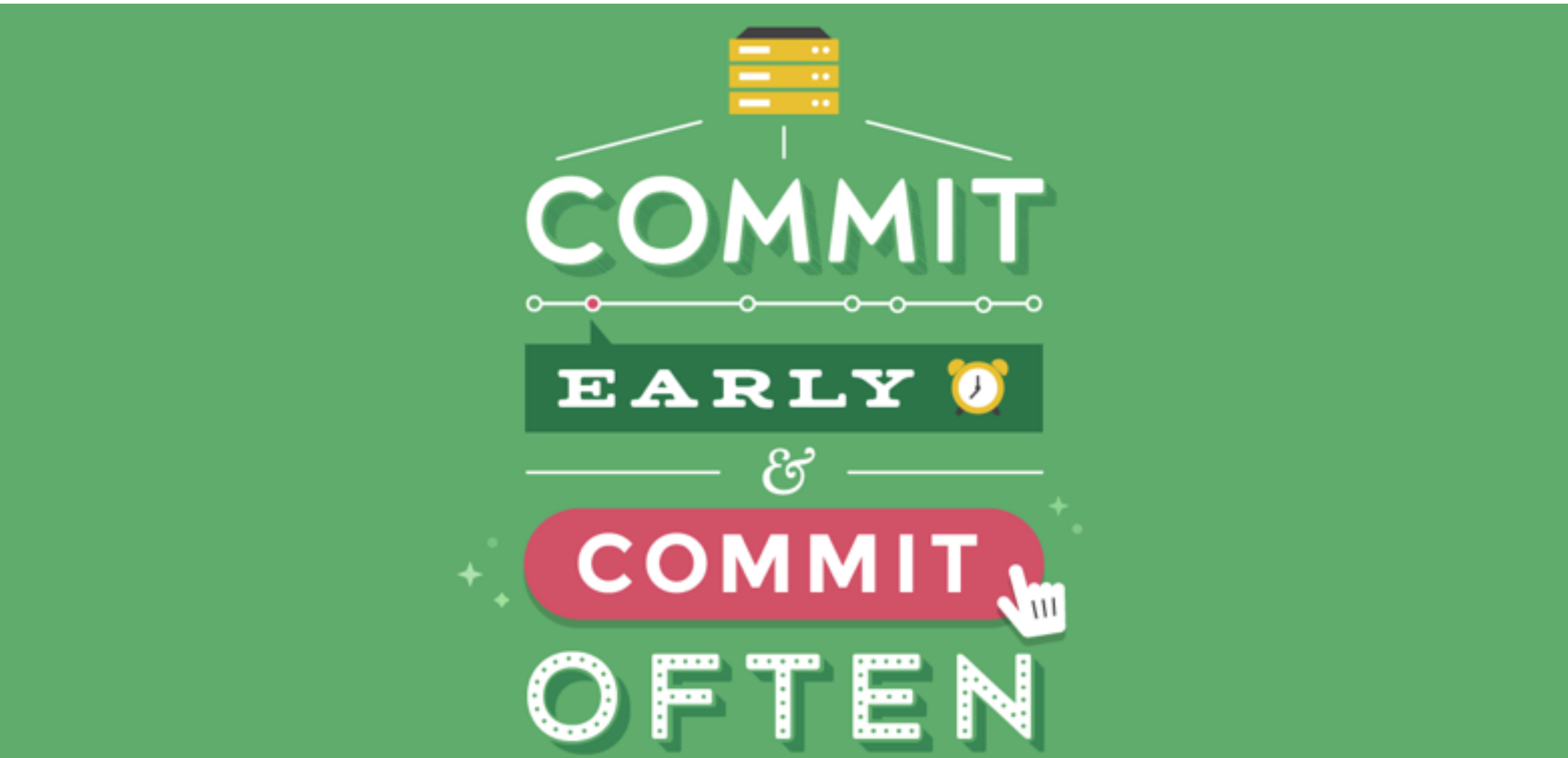
“This will make thing easier”

- Start in a small corner and circle outwards
- Convert single files instead of whole packages
- Follow the process:

1. Convert	2. Fix	3. Tidy

- You will revert a lot, so….

kotlin loves it when you do this one cool trick!



practical magic: starting point

project:	commons-cli	**class:**	DefaultParser	**level:**	*

instructions:

1. Convert to Kotlin (only that file)
2. Fix compilation errors: (missing lateinit, MutableList)
3. Fix warnings (Tidy imports, shadowing, intentions)
4. Reduce number of constructors to 1 (use default parameters, replace primary constructor)
5. Turn methods into single expressions (isArgument/ isNegativeNumber, isOption etc…)
6. Replace **if** with **when** in **handleToken(token: String)**

practical magic: examples

project:	Examples	**class:**	*	**level:**	*

instructions:

1. There are 10 examples in the repository
2. Each one contains some Java code…
3. … which will look much nicer and safer when it’s Kotlin!

practical magic: starting point

project:	joda-money	**class:**	MoneyUtils	**level:**	*

instructions:

1. Convert to Kotlin (only that file)
2. Fix checkNotNull to take nullable arg
3. Fix intention warnings
4. Convert isZero method to extension function
5. Convert to expression and remove return type
6. Convert isZero null check to `this.let?` scope function

practical magic: starting point

project:	commons-cli	**class:**	Util	**level:**	*

instructions:

1. Convert to Kotlin (only that file)
2. Throw error in stripLeadingHyphens when str is null
3. Replace if with when in stripLeadingHyphens, convert to single expression
4. Convert methods to extension functions
5. Tidy stringLeavingAndTrailingQuotes to expression, remove “this.

practical magic: starting point

project:	joda-money	**class:**	MoneyUtils	**level:**	*

instructions:

1. Convert to Kotlin
2. For max()/min()/add()/subtract()
   1. Replace compareTo with operators
   2. Convert to a single if/else block
   3. Lift return
   4. Replace if with when
   5. Remove Braces
   6. Convert to expression

practical magic: starting point

project:	okey-doke	**class:**	InvocationFormatter	**level:**	**

instructions:

1. Convert to Kotlin (only that file)
2. Remove nullability on Invocation to fix compile
3. Remove protected as class is not extended
4. Convert format() method to use apply(), convert to single expression
5. Convert formatArguments() method to use fold() on arguments

practical magic: starting point

project:	commons-cli	**class:**	TypeHandler	**level:**	**

instructions:

1. Convert to Kotlin (only that file)
2. Add missing @JvmStatic
3. Convert createValue()
   1. Replace if with when
   2. Remove braces
   3. Import constants
   4. Convert to single expression
   5. Suppress compiler warning
4. Tidy createClass(), createObject(), createNumber() into single expressions

practical magic: starting point

project:	snake-yaml	**class:**	MappingNode	**level:**	**

instructions:

1. Convert to Kotlin (only that file)
2. Convert enum constructors to take nullable (compile break)
3. Fix DumperOptionsTest to use JVM field instead of setTags
4. Fix up defaultFlowStyle and lineBreak
5. In Version, tidy up the toString()
6. Replace if -> when in ScalarStyle.createStyle
7. In platformLineBreak
   1. Replace and inline lineSeparator
   2. Replace loop with firstOrNull
   3. Make get an expression