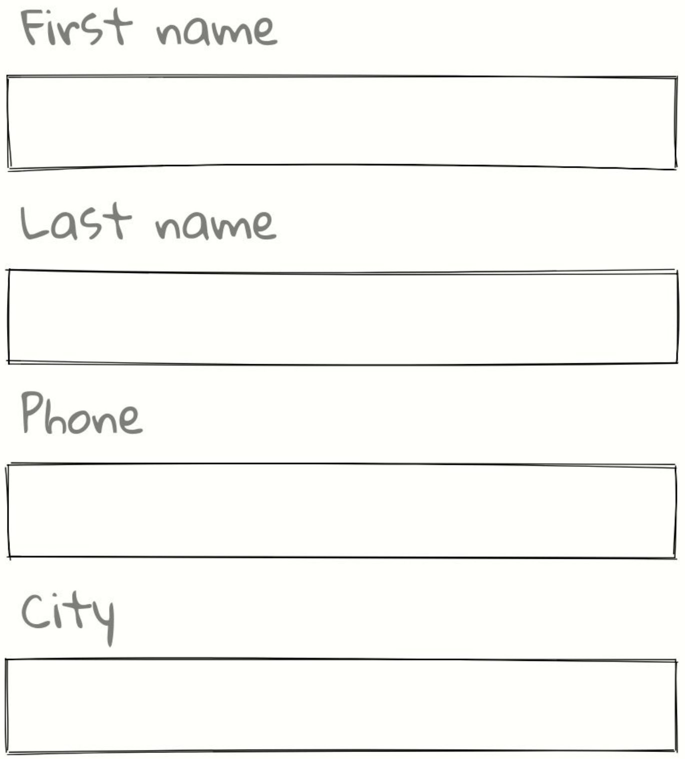
Going beyond JVM with Kotlin

Victor Kropp
@kropp
victor.kropp.name

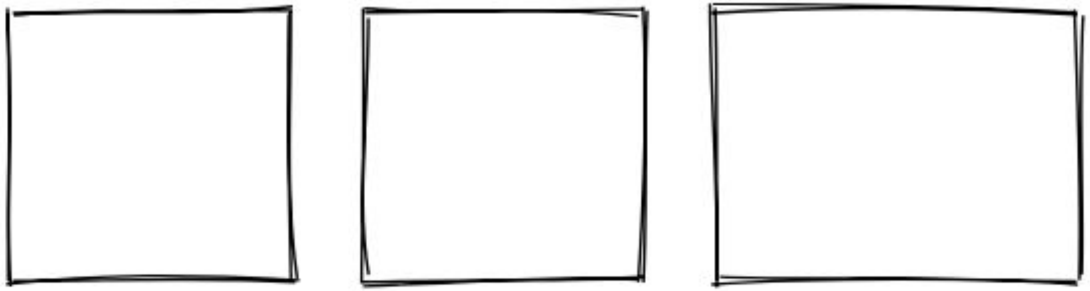
2XU

1261
2XU

2XU
ZONE3
IRONMAN
1261
Victor
M30-34 M

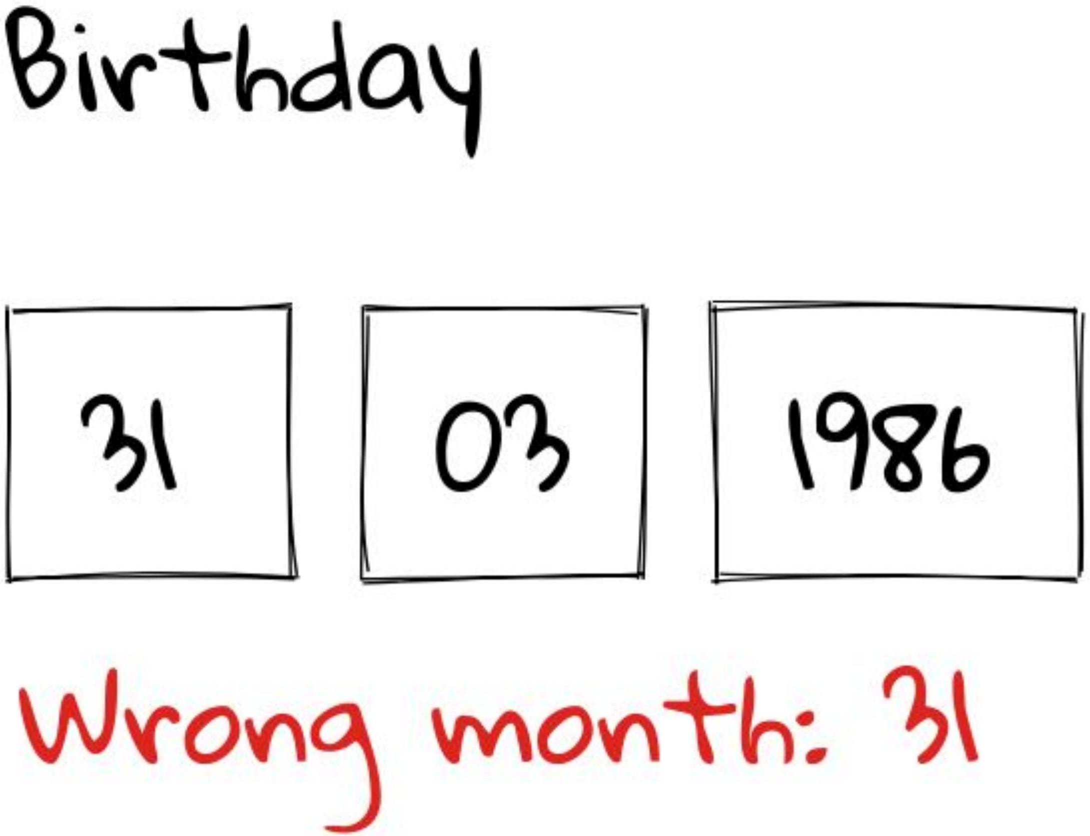

First name
Last name
Phone
City

Birthday



Birthday

31	03	1986

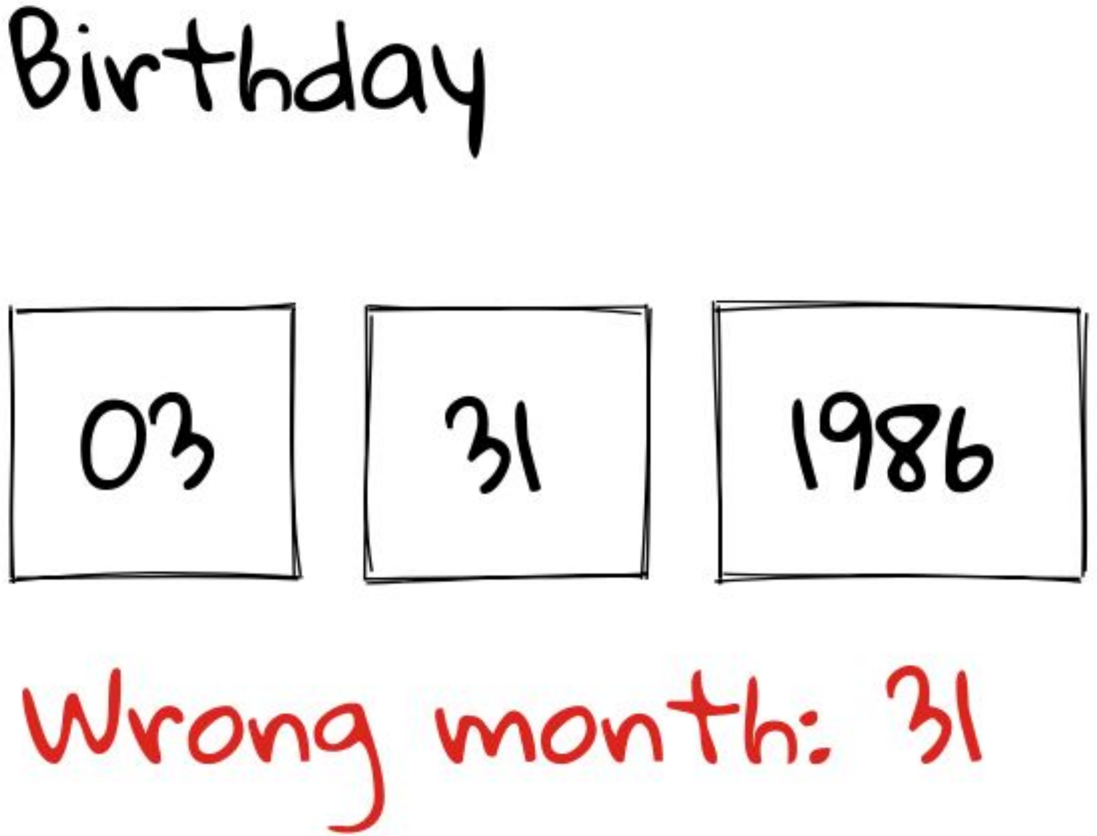

Birthday

31	03	1986

Wrong month: 31

Birthday

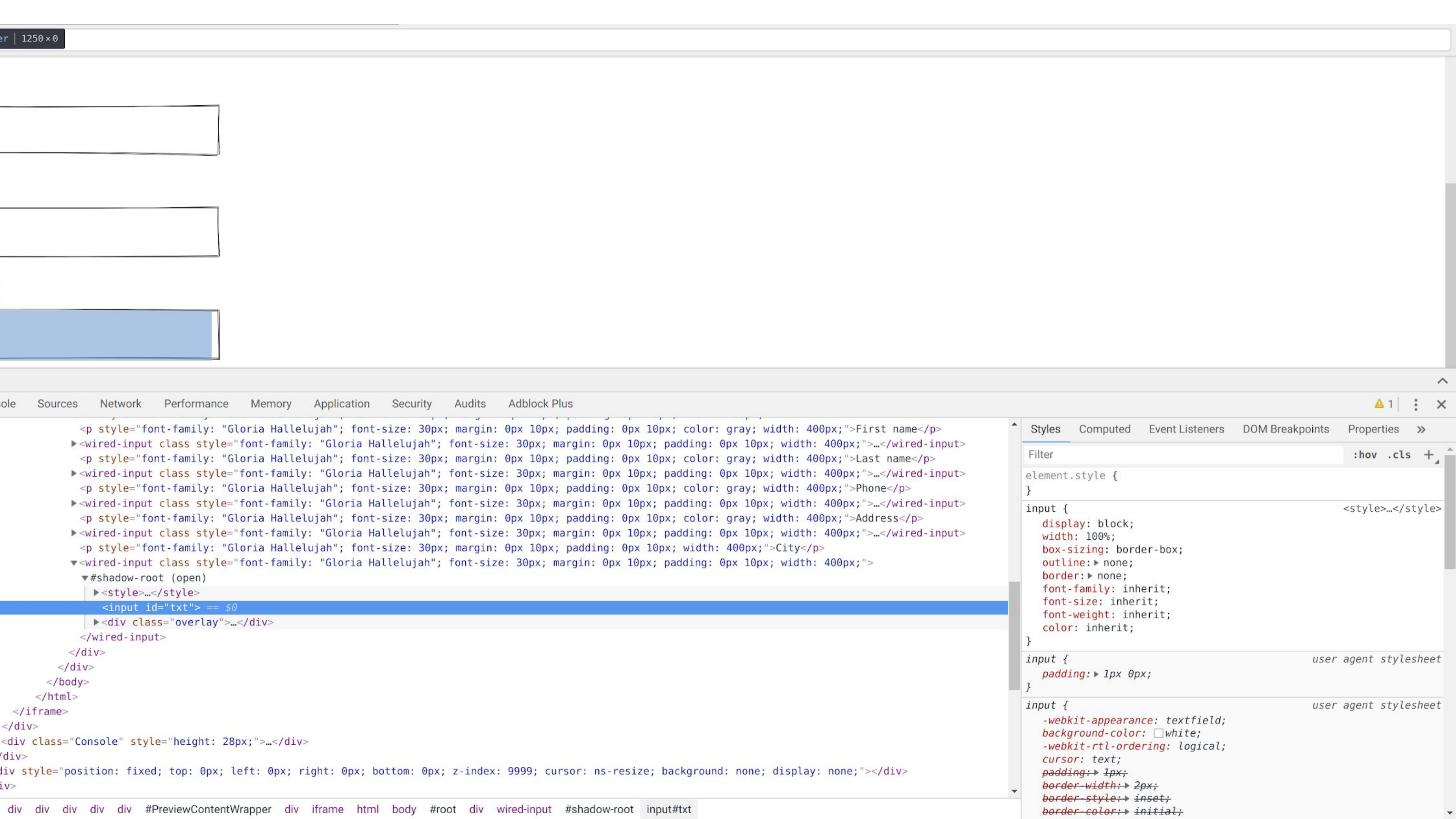
03	31	1986



Birthday

03	31	1986

Wrong month: 31



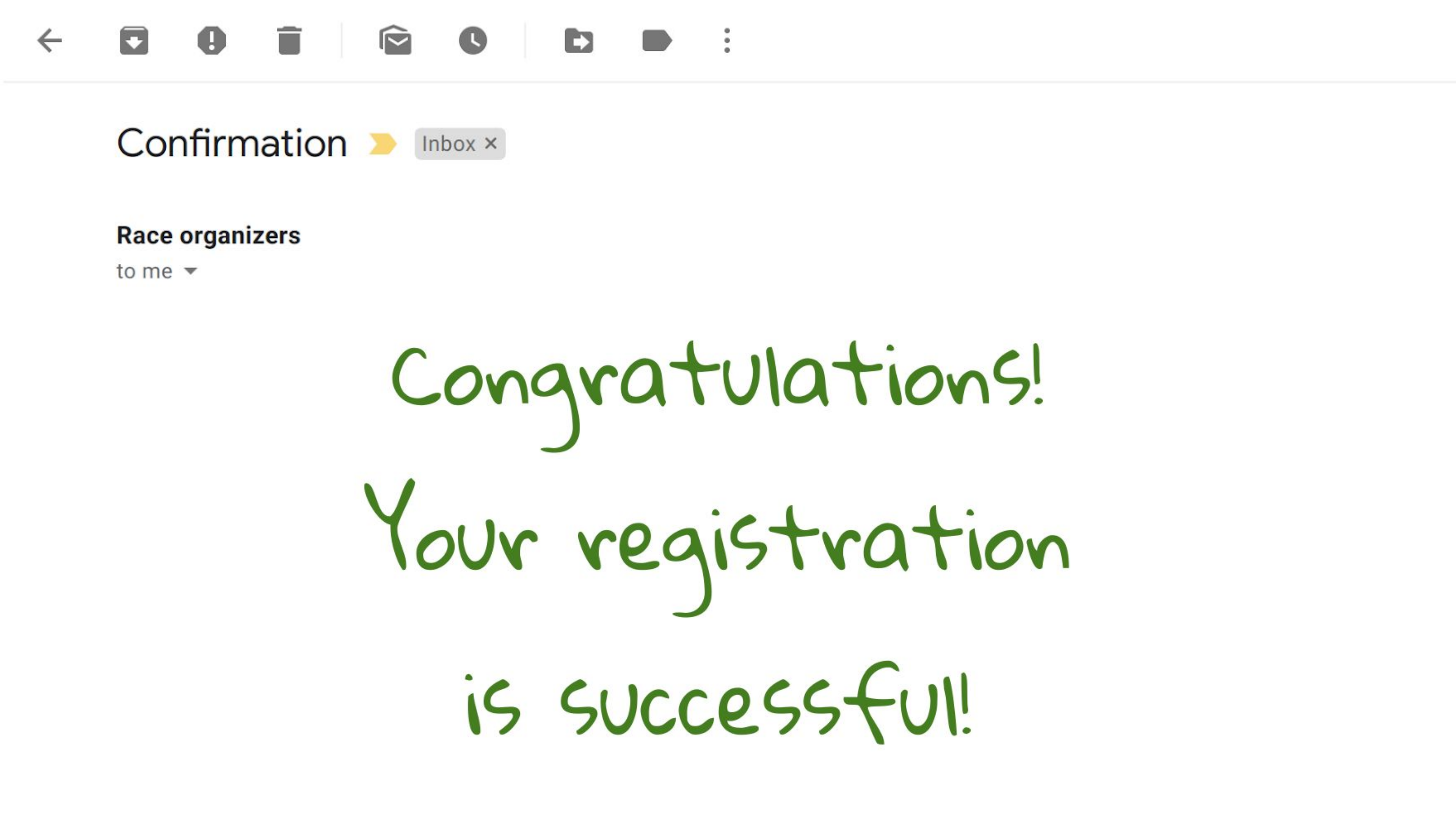
```
        <p style="font-family: "Gloria Hallelujah"; font-size: 30px; margin: 0px 10px; padding: 0px 10px; color: gray; width: 400px;">First name</p>
       ▶<wired-input class style="font-family: "Gloria Hallelujah"; font-size: 30px; margin: 0px 10px; padding: 0px 10px; width: 400px;">…</wired-input>
        <p style="font-family: "Gloria Hallelujah"; font-size: 30px; margin: 0px 10px; padding: 0px 10px; color: gray; width: 400px;">Last name</p>
       ▶<wired-input class style="font-family: "Gloria Hallelujah"; font-size: 30px; margin: 0px 10px; padding: 0px 10px; width: 400px;">…</wired-input>
        <p style="font-family: "Gloria Hallelujah"; font-size: 30px; margin: 0px 10px; padding: 0px 10px; color: gray; width: 400px;">Phone</p>
       ▶<wired-input class style="font-family: "Gloria Hallelujah"; font-size: 30px; margin: 0px 10px; padding: 0px 10px; width: 400px;">…</wired-input>
        <p style="font-family: "Gloria Hallelujah"; font-size: 30px; margin: 0px 10px; padding: 0px 10px; color: gray; width: 400px;">Address</p>
       ▶<wired-input class style="font-family: "Gloria Hallelujah"; font-size: 30px; margin: 0px 10px; padding: 0px 10px; width: 400px;">…</wired-input>
        <p style="font-family: "Gloria Hallelujah"; font-size: 30px; margin: 0px 10px; padding: 0px 10px; width: 400px;">City</p>
       ▼<wired-input class style="font-family: "Gloria Hallelujah"; font-size: 30px; margin: 0px 10px; padding: 0px 10px; width: 400px;">
         ▼#shadow-root (open)
           ▶<style>…</style>
            <input id="txt"> == $0
           ▶<div class="overlay">…</div>
        </wired-input>
      </div>
    </div>
  </body>
</html>
</iframe>
</div>
<div class="Console" style="height: 28px;">…</div>
/div>
div style="position: fixed; top: 0px; left: 0px; right: 0px; bottom: 0px; z-index: 9999; cursor: ns-resize; background: none; display: none;"></div>
iv>
```

```
div  div  div  div  div  #PreviewContentWrapper  div  iframe  html  body  #root  div  wired-input  #shadow-root  input#txt
```

```
element.style {
}
input {                                      <style>…</style>
    display: block;
    width: 100%;
    box-sizing: border-box;
    outline: ▶ none;
    border: ▶ none;
    font-family: inherit;
    font-size: inherit;
    font-weight: inherit;
    color: inherit;
}
input {                                      user agent stylesheet
    padding: ▶ 1px 0px;
}
input {                                      user agent stylesheet
    -webkit-appearance: textfield;
    background-color: ☐white;
    -webkit-rtl-ordering: logical;
    cursor: text;
    ~~padding: ▶ 1px;~~
    ~~border-width: ▶ 2px;~~
    ~~border-style: ▶ inset;~~
    ~~border-color: ▶ initial;~~
```

Birthday

31	03	1986



Confirmation

Race organizers

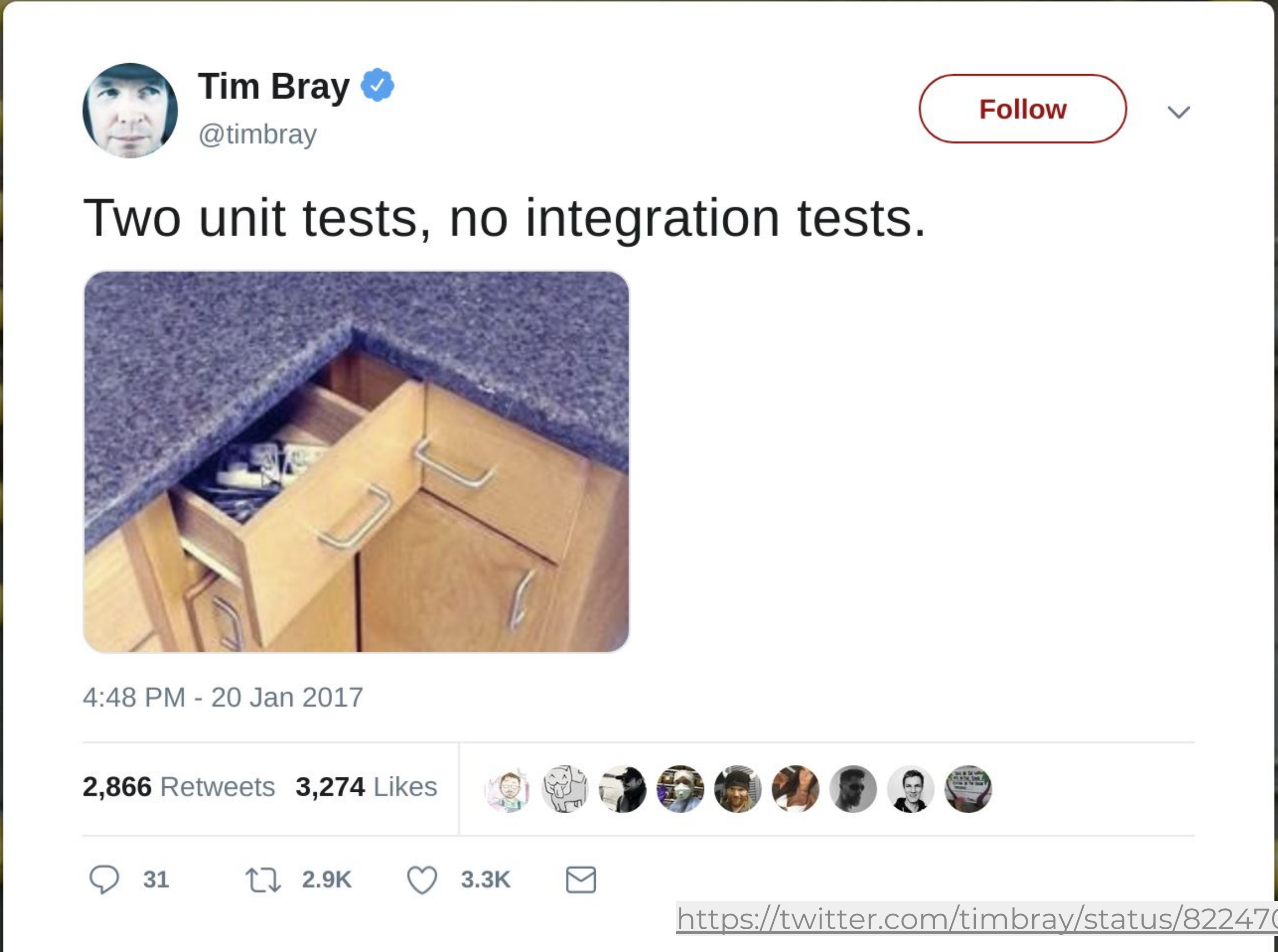
to me

Congratulations!

Your registration

is successful!

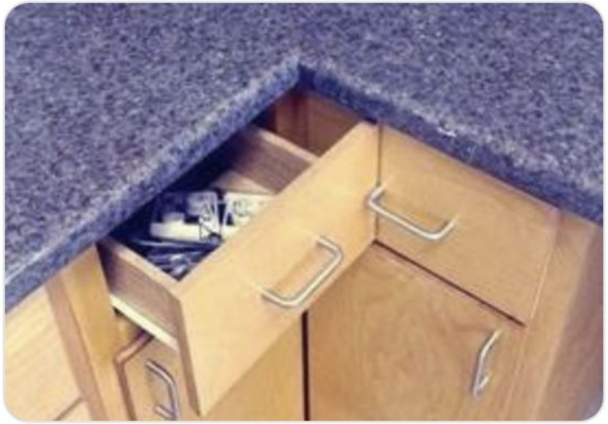
https://imgflip.com/i/2kdpb9



Tim Bray
@timbray

Follow

Two unit tests, no integration tests.



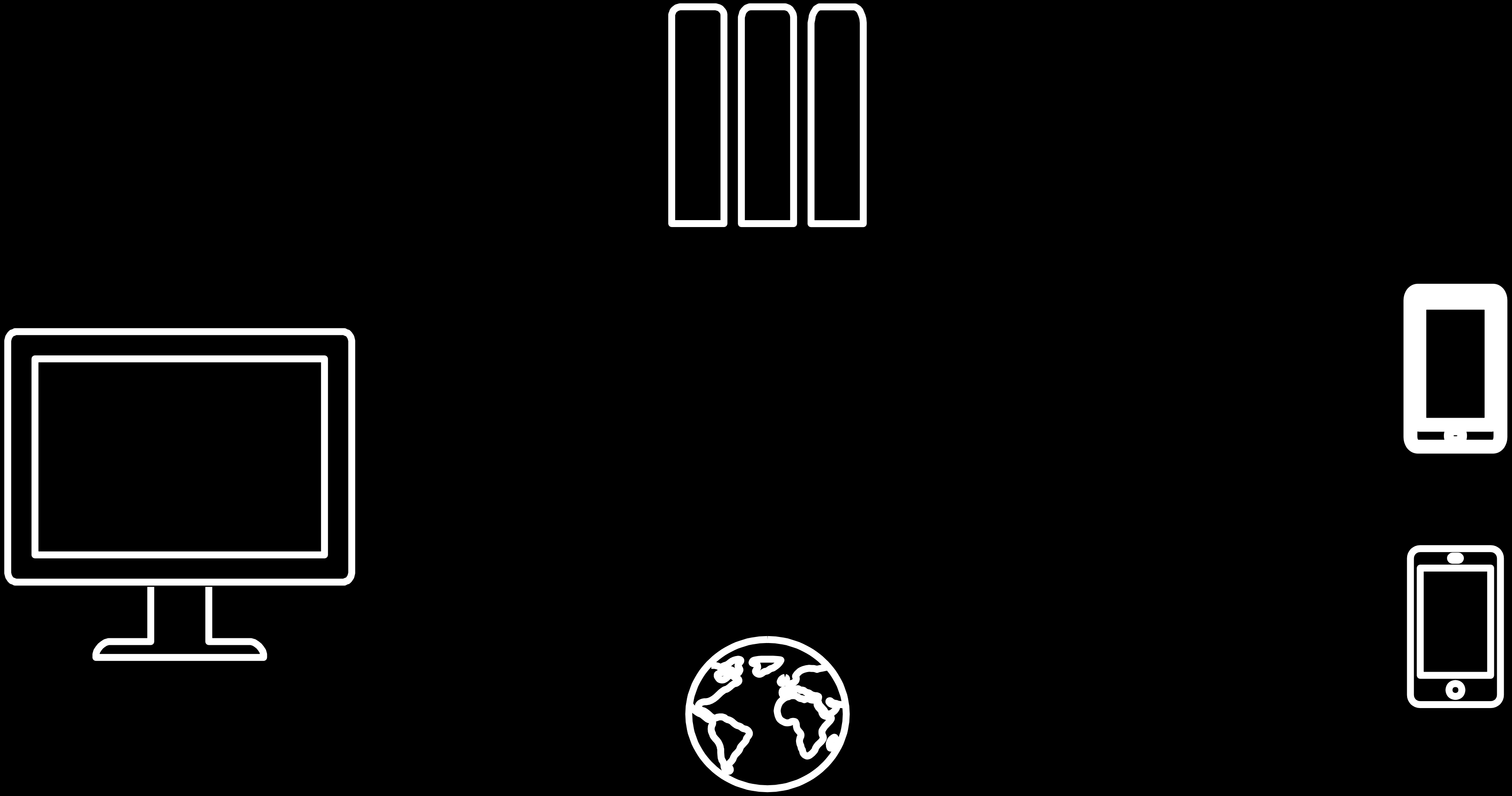
4:48 PM - 20 Jan 2017

2,866 Retweets 3,274 Likes

31 2.9K 3.3K

https://twitter.com/timbray/status/822470746773409794

Modern applications



C
C++
C#
Objective-C
Swift

Java
Python
Ruby

JavaScript
TypeScript

Java

Objective-C
Swift

compiled vs. interpreted

compiled vs. interpreted

static vs. dynamic typing

compiled vs. interpreted

static vs. dynamic typing

manual vs. automatic memory management

C
C++
C#
Objective-C
Swift

Java
Python
Ruby

JavaScript
TypeScript

Java

Objective-C
Swift

~~C~~
~~C++~~
~~C#~~
~~Objective-C~~
~~Swift~~

~~Java~~
~~Python~~
~~Ruby~~

~~JavaScript~~
~~TypeScript~~

~~Java~~

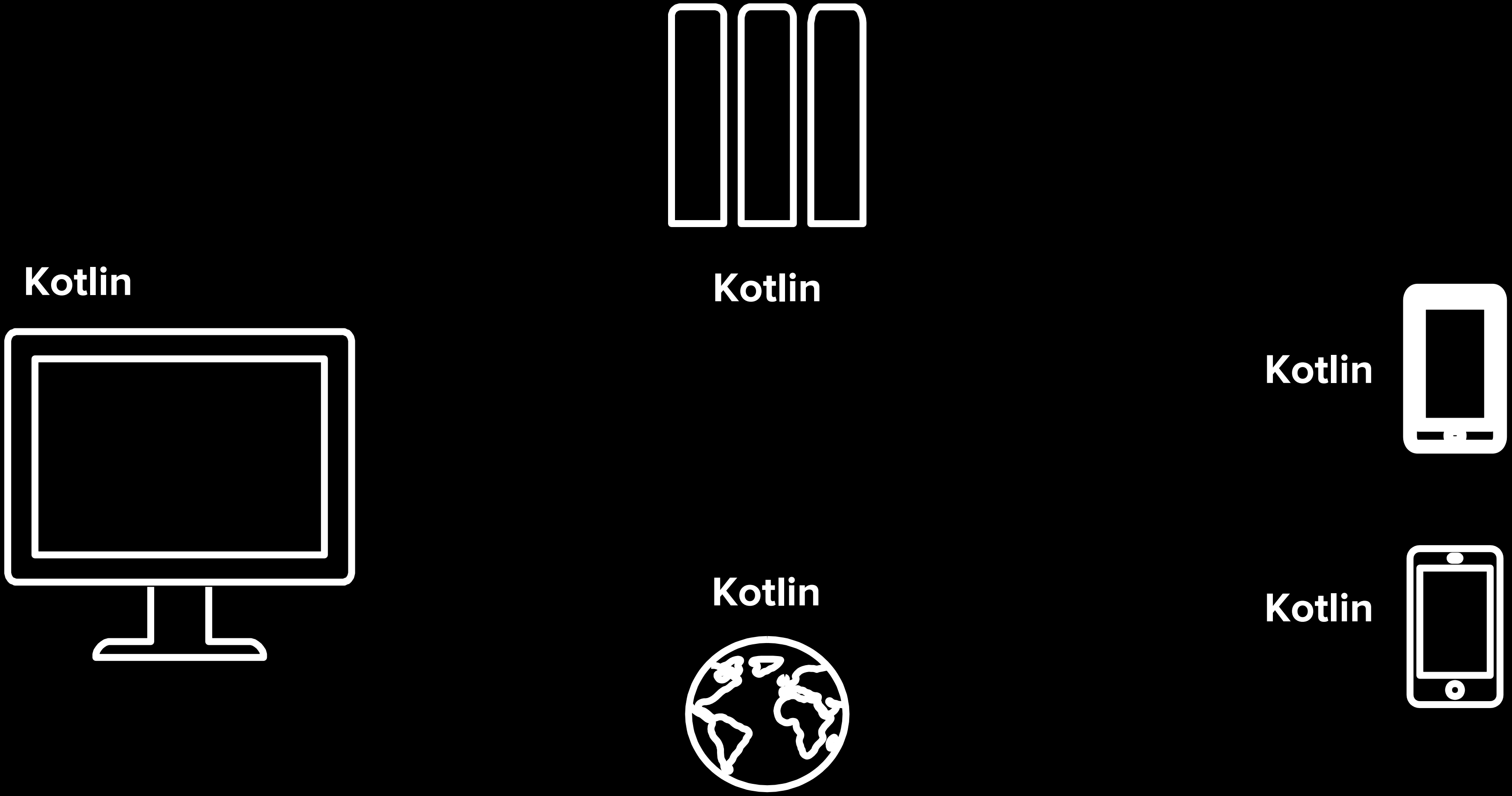
~~Objective-C~~
~~Swift~~

Kotlin

Kotlin

JS

10110
01010
10101


Kotlin
Kotlin
Kotlin
Kotlin
Kotlin

Kotlin/Native

Kotlin/JVM

Kotlin/JVM

Kotlin/JS

Kotlin/Native

Using the same language across all platforms

- Everyone on the team speaks the same language
- Single team working on all apps
- Simplify full-stack development

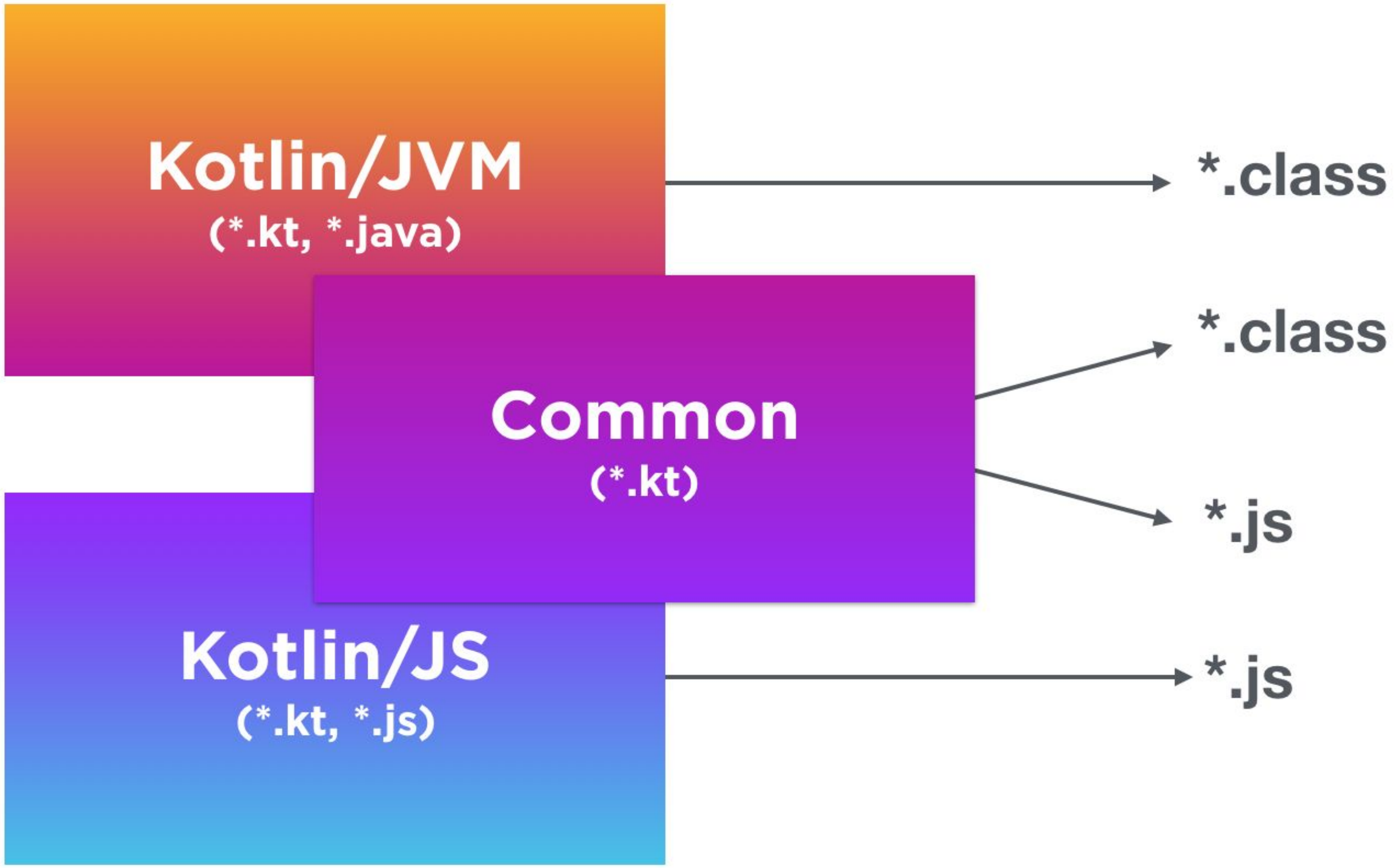
Code sharing

Code sharing

- Share data structures
- Share business logic
- Share tests!

Multiplatform projects in Kotlin

Still experimental


Kotlin/JVM
(*.kt, *.java)
Common
(*.kt)
Kotlin/JS
(*.kt, *.js)
*.class
*.class
*.js
*.js

Common code

```
fun validate(day: Int, month: Int, year: Int): Boolean {
}
```

Common code

```kotlin
fun validate(day: Int, month: Int, year: Int): Boolean {
  return month in 1..12
}
```

Kotlin

Common code

```kotlin
fun validate(day: Int, month: Int, year: Int): Boolean {
  return Date(year, month, day).isValid()
}
```

Common code

```
fun validate(day: Int, month: Int, year: Int): Boolean {
  return Date(year, month, day).isValid()
}

class Date(day: Int, month: Int, year: Int) {
  fun isValid(): Boolean = ...
}
```

Kotlin

Common code

```
class Date(day: Int, month: Int, year: Int) {
  fun isValid(): Boolean = ...
}
```

Interfaces

```
interface Date {
  fun isValid(): Boolean
}
```

Interfaces

```
interface Date {
  fun isValid(): Boolean
}

class DateJvm    : Date { override fun isValid() = true }
class DateJs     : Date { override fun isValid() = true }
class DateNative : Date { override fun isValid() = true }
```

Kotlin

Interfaces

```kotlin
interface Date {
  fun isValid(): Boolean
}

object DateFactory {
  fun createDate(day: Int, month: Int, year: Int) = ...
}
```

expect

```
expect class Date(day: Int, month: Int, year: Int) {
  fun isValid(): Boolean
}
```

expect limitations

```
expect class Date(val day: Int, month: Int, year: Int) {
  fun isValid(): Boolean
}
```

expect limitations

```
expect class Date(~~val~~ day: Int, month: Int, year: Int) {
  fun isValid(): Boolean
}
```

expect limitations

```
expect class Date(day: Int, month: Int, year: Int) {
    val day: Int
    val month: Int
    val year: Int
    fun isValid(): Boolean
}
```

expect limitations

```
expect class Date(day: Int, month: Int, year: Int) {
    private val day: Int
    private val month: Int
    private val year: Int
    fun isValid(): Boolean
}
```

expect limitations

```
expect class Date(day: Int, month: Int, year: Int) {
    fun isValid(): Boolean {
        return true
    }
}
```

expect limitations

```
expect class Date(day: Int, month: Int, year: Int) {
    fun isValid(): Boolean ~~{~~
        ~~return true~~
    ~~}~~
}
```

expect limitations

```
expect class Date(day: Int, month: Int, year: Int) {
}

fun Date.isValid(): Boolean {
  return true
}
```

actual

```
actual class Date
    actual constructor(private val day: Int,
                private val month: Int, private val year: Int) {

    actual fun isValid() = ...
}
```

actual

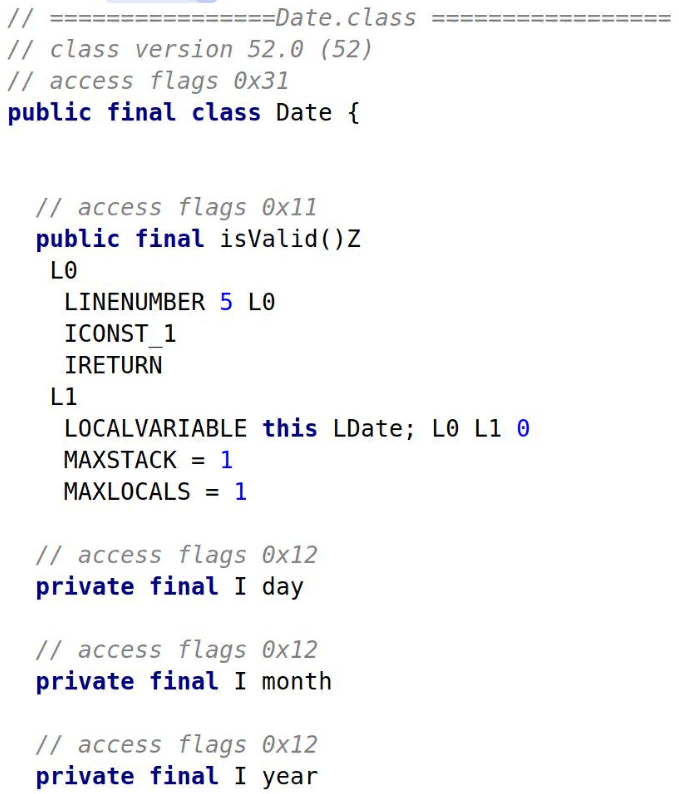
```
actual class Date
  actual constructor(private val day: Int,
          private val month: Int, private val year: Int) {

  actual fun isValid() = true
}
```

All targets

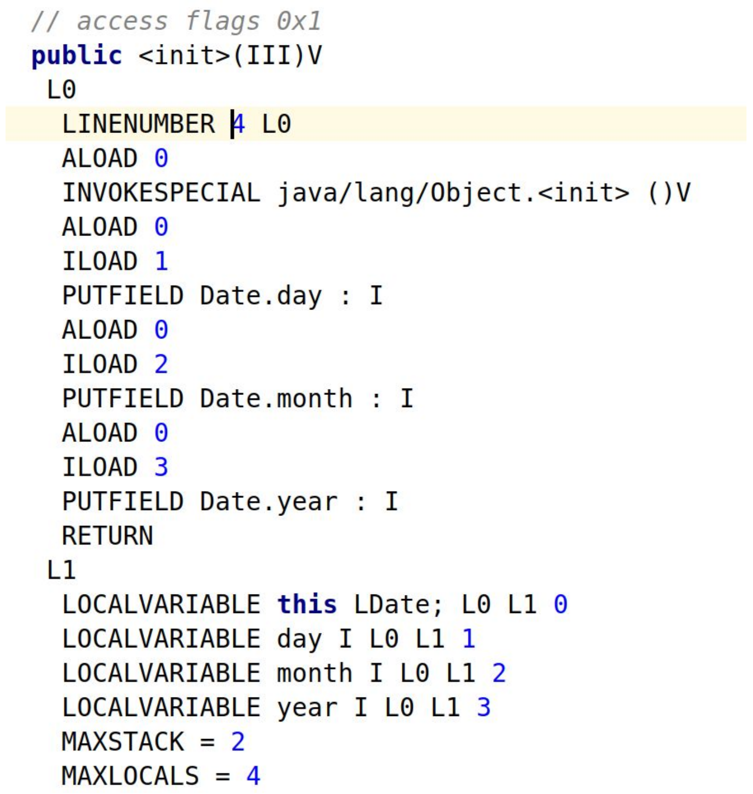
```
actual class Date // JVM
actual class Date // JS
actual class Date // Native
```

Kotlin

No overhead



```
// ================Date.class =================
// class version 52.0 (52)
// access flags 0x31
public final class Date {


  // access flags 0x11
  public final isValid()Z
   L0
    LINENUMBER 5 L0
    ICONST_1
    IRETURN
   L1
    LOCALVARIABLE this LDate; L0 L1 0
    MAXSTACK = 1
    MAXLOCALS = 1

  // access flags 0x12
  private final I day

  // access flags 0x12
  private final I month

  // access flags 0x12
  private final I year
```



```
// access flags 0x1
public <init>(III)V
 L0
  LINENUMBER 4 L0
  ALOAD 0
  INVOKESPECIAL java/lang/Object.<init> ()V
  ALOAD 0
  ILOAD 1
  PUTFIELD Date.day : I
  ALOAD 0
  ILOAD 2
  PUTFIELD Date.month : I
  ALOAD 0
  ILOAD 3
  PUTFIELD Date.year : I
  RETURN
 L1
  LOCALVARIABLE this LDate; L0 L1 0
  LOCALVARIABLE day I L0 L1 1
  LOCALVARIABLE month I L0 L1 2
  LOCALVARIABLE year I L0 L1 3
  MAXSTACK = 2
  MAXLOCALS = 4
```

Kotlin

Existing actual class

```kotlin
expect class LocalDate {
  fun lengthOfYear(): Int
}
```

typealias

```kotlin
expect class LocalDate {
  fun lengthOfYear(): Int
}

// JVM
actual typealias LocalDate = java.time.LocalDate
```

Test typealias

```
expect annotation class Test

actual typealias Test = org.junit.Test
```

Code structure

build.gradle

```
apply plugin: 'kotlin-multiplatform'
```

build.gradle

```
apply plugin: 'kotlin-multiplatform'
kotlin {
    targets {




    }
}
```

build.gradle

```
apply plugin: 'kotlin-multiplatform'
kotlin {
   targets {
       fromPreset(presets.jvm, 'jvm')



   }
}
```

build.gradle

```
apply plugin: 'kotlin-multiplatform'
kotlin {
   targets {
       fromPreset(presets.jvm, 'jvm')
       fromPreset(presets.js, 'js')


   }
}
```

build.gradle

```
apply plugin: 'kotlin-multiplatform'
kotlin {
   targets {
       fromPreset(presets.jvm, 'jvm')
       fromPreset(presets.js, 'js')
       fromPreset(presets.linuxX64, 'linux')
   }
}
```

build.gradle

```
apply plugin: 'kotlin-multiplatform'
kotlin {
    targets {
        fromPreset(presets.jvm)
        fromPreset(presets.js)
        fromPreset(presets.linuxX64, 'linux')
    }
}
```

Kotlin

build.gradle

```
apply plugin: 'kotlin-multiplatform'
kotlin {
    targets {...}
    sourceSets {
    }
}
```

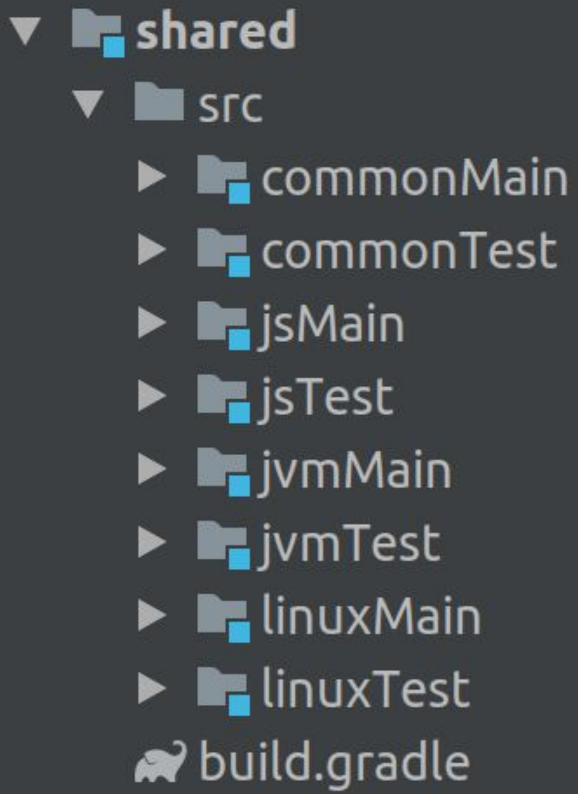
build.gradle

```
apply plugin: 'kotlin-multiplatform'
kotlin {
    targets {...}
    sourceSets {
        commonMain { }
        commonTest { }
    }
}
```

build.gradle

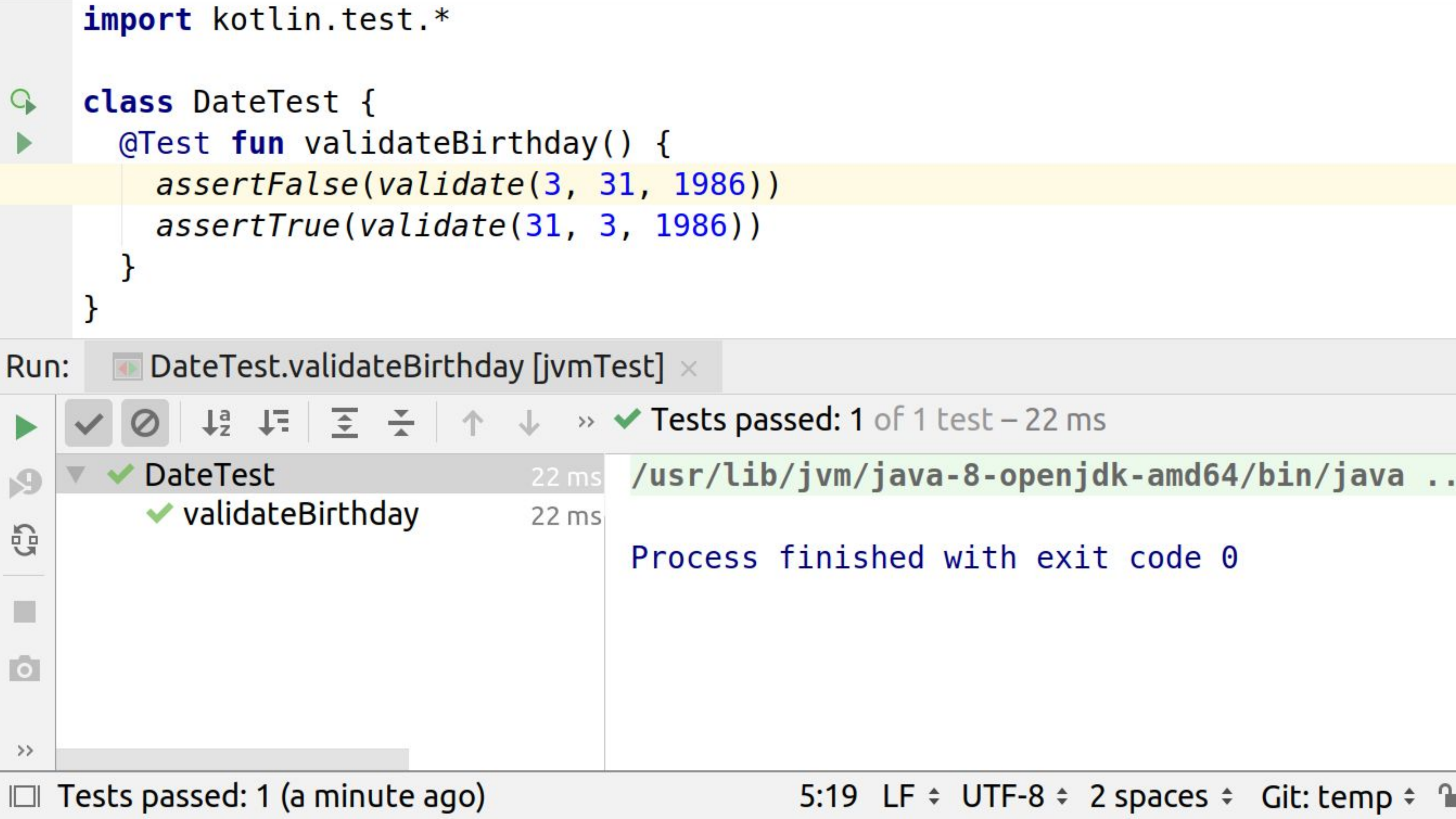
```
sourceSets {
    commonMain { }
    commonTest { }
    jvmMain { }
    jvmTest { }
    jsMain { }
    jsTest { }
    linuxMain { }
    linuxTest { }
}
```

Project structure



```
▼ shared
  ▼ src
    ▶ commonMain
    ▶ commonTest
    ▶ jsMain
    ▶ jsTest
    ▶ jvmMain
    ▶ jvmTest
    ▶ linuxMain
    ▶ linuxTest
  build.gradle
```

Testing



```
import kotlin.test.*

class DateTest {
  @Test fun validateBirthday() {
    assertFalse(validate(3, 31, 1986))
    assertTrue(validate(31, 3, 1986))
  }
}
```

Run:
DateTest.validateBirthday [jvmTest]
Tests passed: 1 of 1 test – 22 ms
DateTest 22 ms
validateBirthday 22 ms

```
/usr/lib/jvm/java-8-openjdk-amd64/bin/java ..

Process finished with exit code 0
```

Tests passed: 1 (a minute ago)
5:19 LF UTF-8 2 spaces Git: temp

```
$ ./gradlew :shared:linuxTest
> Configure project :frontend
> Configure project :shared
> Task :shared:linuxTest
[==========] Running 1 tests from 1 test cases.
[----------] Global test environment set-up.
[----------] 1 tests from DateTest
[ RUN      ] DateTest.validateBirthday
[       OK ] DateTest.validateBirthday (0 ms)
[----------] 1 tests from DateTest (0 ms total)

[----------] Global test environment tear-down
[==========] 1 tests from 1 test cases ran. (0 ms total)
[  PASSED  ] 1 tests.

BUILD SUCCESSFUL in 14s
3 actionable tasks: 3 executed
```

build.gradle

```
commonMain {
  dependencies {
    implementation 'org.jetbrains.kotlin:kotlin-stdlib-common'
  }
}
commonTest {
  dependencies {
    implementation 'org.jetbrains.kotlin:kotlin-test-common'
    implementation 'org.jetbrains.kotlin:kotlin-test-annotations-common'
  }
}
```

build.gradle

```
jvmMain {
  dependencies {
    implementation 'org.jetbrains.kotlin:kotlin-stdlib-jdk8'
   }
}
jvmTest {
  dependencies {
    implementation 'org.jetbrains.kotlin:kotlin-test'
    implementation 'org.jetbrains.kotlin:kotlin-test-junit'
  }
}
```

build.gradle

```
jsMain {
  dependencies {
    implementation 'org.jetbrains.kotlin:kotlin-stdlib-js'
  }
}
jsTest {
  dependencies {
    implementation 'org.jetbrains.kotlin:kotlin-test'
    implementation 'org.jetbrains.kotlin:kotlin-test-js'
  }
}
```

Project dependencies

```
dependencies {
  implementation project(':shared')
}
```

Multiplatform libraries

kotlin.test

kotlin.test

kotlinx.coroutines

kotlin.test

kotlinx.coroutines

kotlinx.serialization

Write your own library!

Publishing

build.gradle

```
apply plugin: 'maven-publish'
```

Console

```
$ ./gradlew publishToMavenLocal
```

Use it

Kotlin

Date/JVM

```
actual class Date {
  actual fun isValid(): Boolean {
  }
}
```

Date/JVM

```
actual class Date actual constructor(day: Int, month: Int,year:Int){
  actual fun isValid(): Boolean {
  }
}
```

Date/JVM

```kotlin
actual class Date actual constructor(private val day: Int, private
 val month: Int, private val year: Int) {
   actual fun isValid(): Boolean {
   }
}
```

Date/JVM

```kotlin
actual class Date actual constructor(private val day: Int, private
 val month: Int, private val year: Int) {
  actual fun isValid() = try {
        val calendar = Calendar.getInstance()
        calendar.isLenient = false
        calendar.set(year, month - 1, day)
        calendar.time
        true
      } catch (e: Exception) {
        false
      }
}
```

Kotlin

Date/JS

```kotlin
actual class Date actual constructor(private val day: Int,
 private val month: Int, private val year: Int) {
 actual fun isValid() = moment(year, month, day).isValid()
}
```

Date/JS

```
actual class Date actual constructor(private val day: Int,
 private val month: Int, private val year: Int) {
 actual fun isValid() = moment(year, month, day).isValid()
}
external interface Moment {
 fun isValid(): Boolean
}
fun moment(day: Int, month: Int, year: Int): Moment =
 definedExternally
```

There is a bug still… 😟

Bug

```kotlin
fun validate(day: Int, month: Int, year: Int): Boolean {
 return Date(year, month, day).isValid()
}

expect class Date(year: Int, month: Int, day: Int) {
 fun isValid(): Boolean
}
```

Fixing a bug

```kotlin
fun validate(day: Int, month: Int, year: Int): Boolean {
 return Date(year, month, day).isValid()
}

expect class Date(year: Int, month: Int, day: Int) {
 fun isValid(): Boolean
}
```

Fixing a bug

```
fun validate(day: Int, month: Int, year: Int): Boolean {
 return Date(year, month, day).isValid()
}

expect class Date(day: Int, month: Int, year: Int) {
 fun isValid(): Boolean
}
```

inline classes

```kotlin
inline class Day(val value: Int)
inline class Month(val value: Int)
inline class Year(val value: Int)
```

Kotlin

inline classes

```kotlin
fun validate(day: Day, month: Month, year: Year): Boolean {
  return Date(day, month, year).isValid()
}
expect class Date(day: Day, month: Month, year: Year) {
  fun isValid(): Boolean
}
inline class Day(val value: Int)
inline class Month(val value: Int)
inline class Year(val value: Int)
```

Kotlin

Compile-time type checks

```
@Test fun validateBirthday() {
  assertFalse(validate(Day(3), Month(31), Year(1986)))
  assertTrue(validate(Month(31), Day(3), Year(1986)))
}
```

```
Type mismatch
Required: Day
Found:    Month
```

Recap

Write once, reuse everywhere

Recap

Write once, reuse everywhere

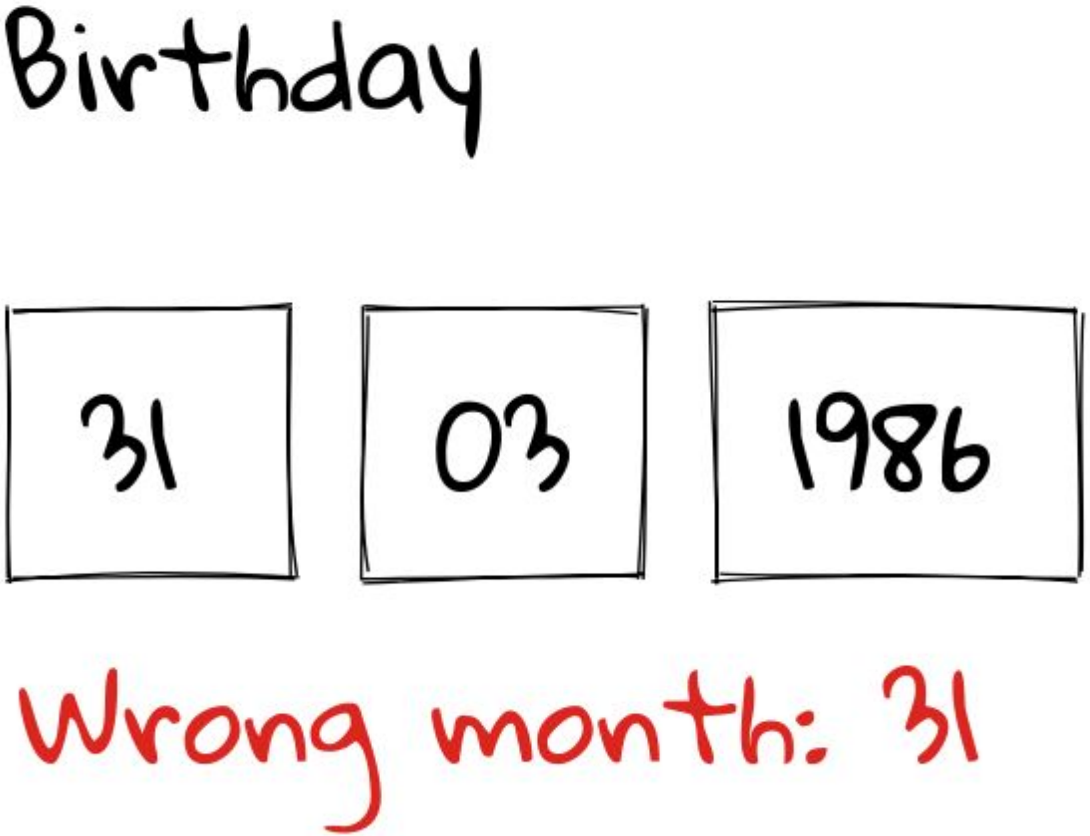
Integrate with platform-specific libraries

Recap

Write once, reuse everywhere

Integrate with platform-specific libraries

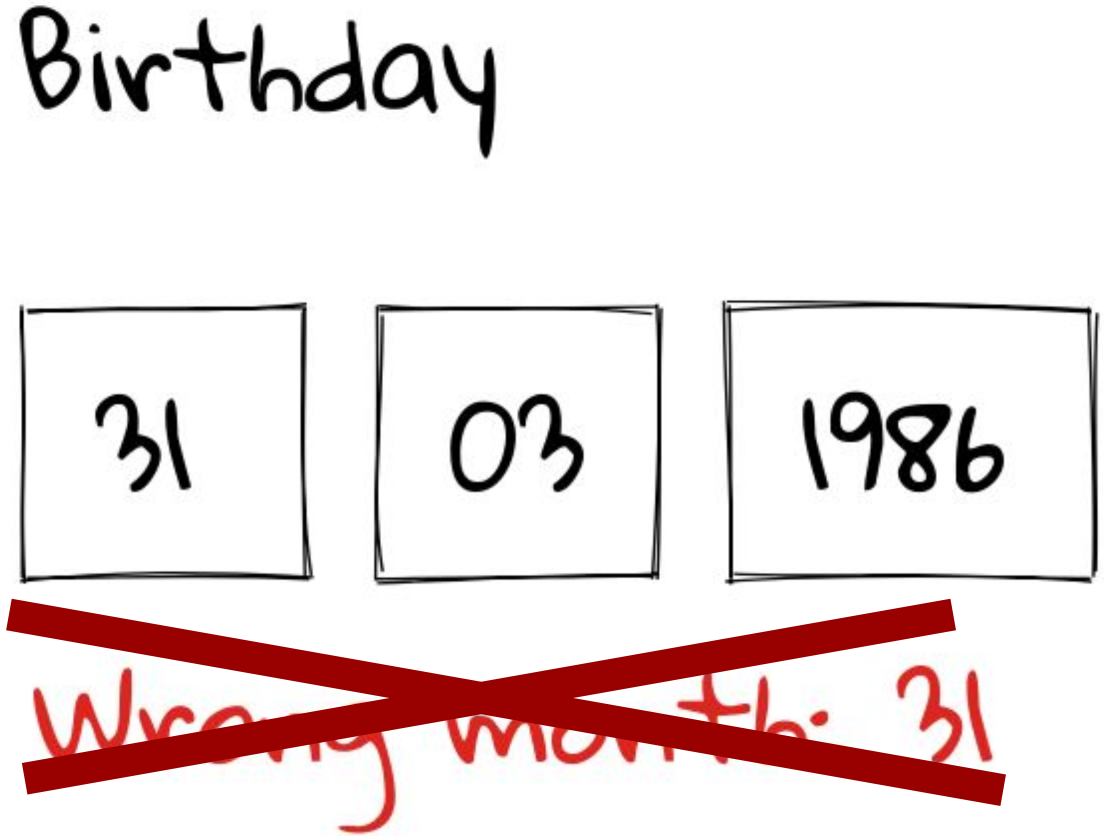
Use all language features on all platforms



Birthday

31	03	1986

Wrong month: 31



Birthday

31	03	1986

~~Wrong month: 31~~

Sample code

https://github.com/kropp/kotlin-multiplatform-sample

Kotlin

Links

Multiplatform projects documentation

https://kotl.in/multiplatform

Configuration samples

https://github.com/h0tk3y/k-new-mpp-samples

Code from this presentation

https://github.com/kropp/kotlin-multiplatform-sample

Thank you!

Victor Kropp
@kropp
victor.kropp.name