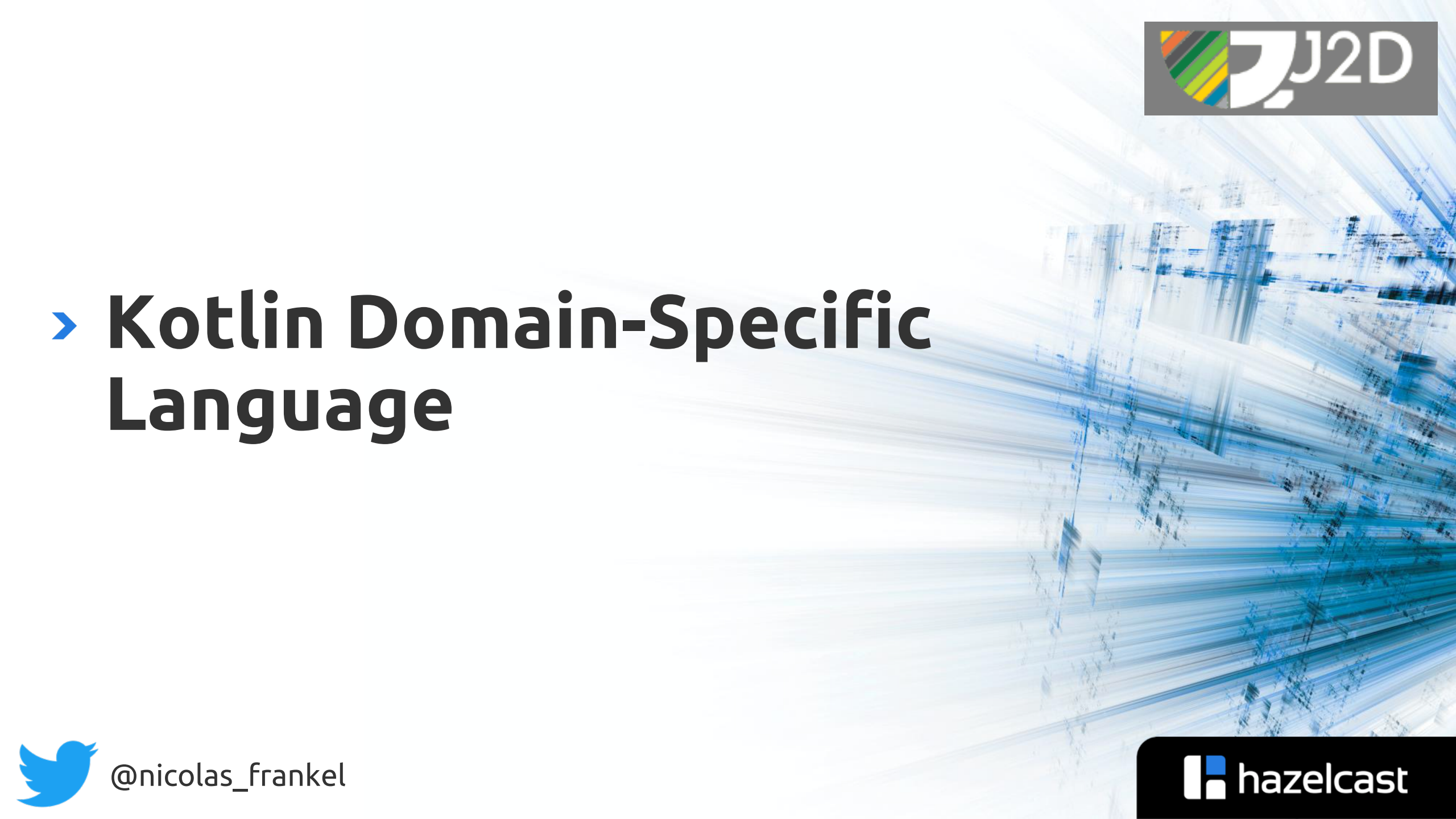


Kotlin Domain-Specific Language

@nicolas_frankel

hazelcast

Me, myself and I

- Developer, team lead, architect, **whatever it takes**
- Recent Developer Advocate
- In love with Kotlin!

Hazelcast

HAZELCAST IMDG is an **operational, in-memory**, distributed computing platform that manages data using in-memory storage, and performs parallel execution for breakthrough application speed and scale.

HAZELCAST JET is the ultra fast, application embeddable, 3rd generation stream processing engine for low latency batch and stream processing.

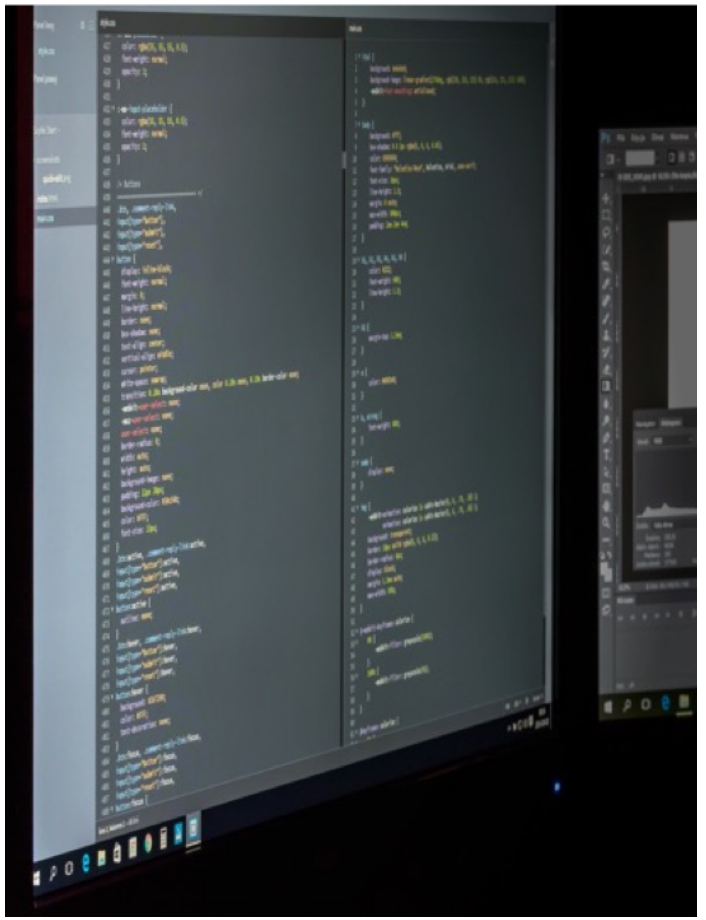
@nicolas_frankel

hazelcast

Domain-Specific Language?

"A **Domain-Specific Language** is a computer language **specialized to a particular application domain**. This is in contrast to a General-Purpose Language, which is broadly applicable across domains."

--Wikipedia

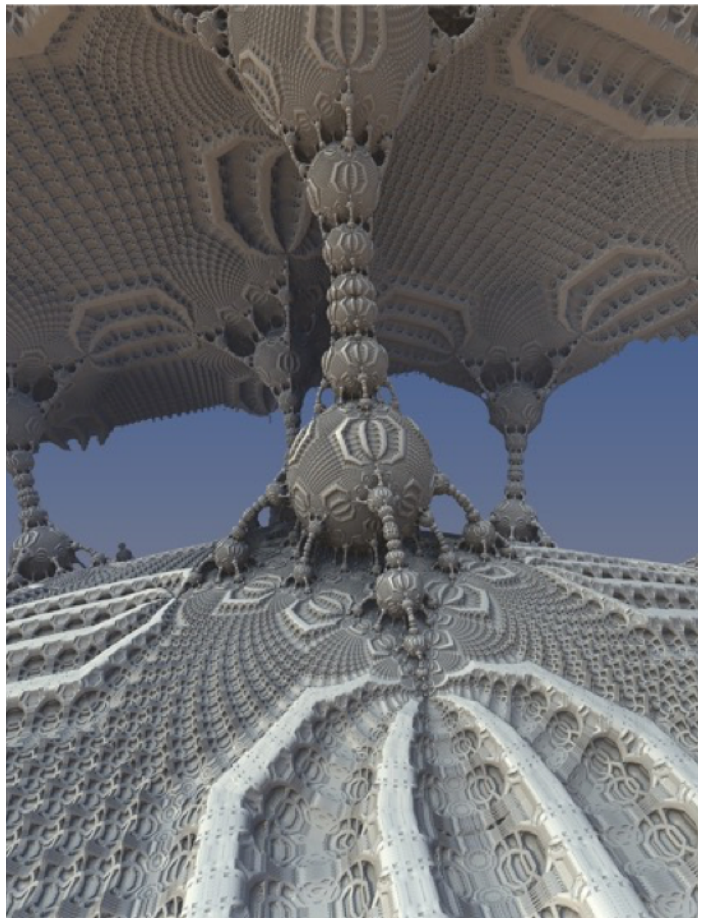


GPL/DSL examples

GPL	DSL
XML	• SVG • MathML
Java	• Hamcrest • AssertJ • Fest Assert
Kotlin	• Anko • Kaadin

Designing a DSL on Java

- Method Sequencing
- Method Chaining
- Nested Method Calls
- Lambdas



@nicolas_frankel

hazelcast

AssertJ example

```
assertThat(frodo.getName())
    .startsWith("Fro")
    .endsWith("do")
    .isEqualToIgnoringCase("frodo");


assertThat(fellowshipOfTheRing)
    .filteredOn(c ->
        c.getName().contains("o"))
    .containsOnly(aragorn, frodo, legolas)
    .extracting(c -> c.getRace().getName())
    .contains("Hobbit", "Elf", "Man");
```

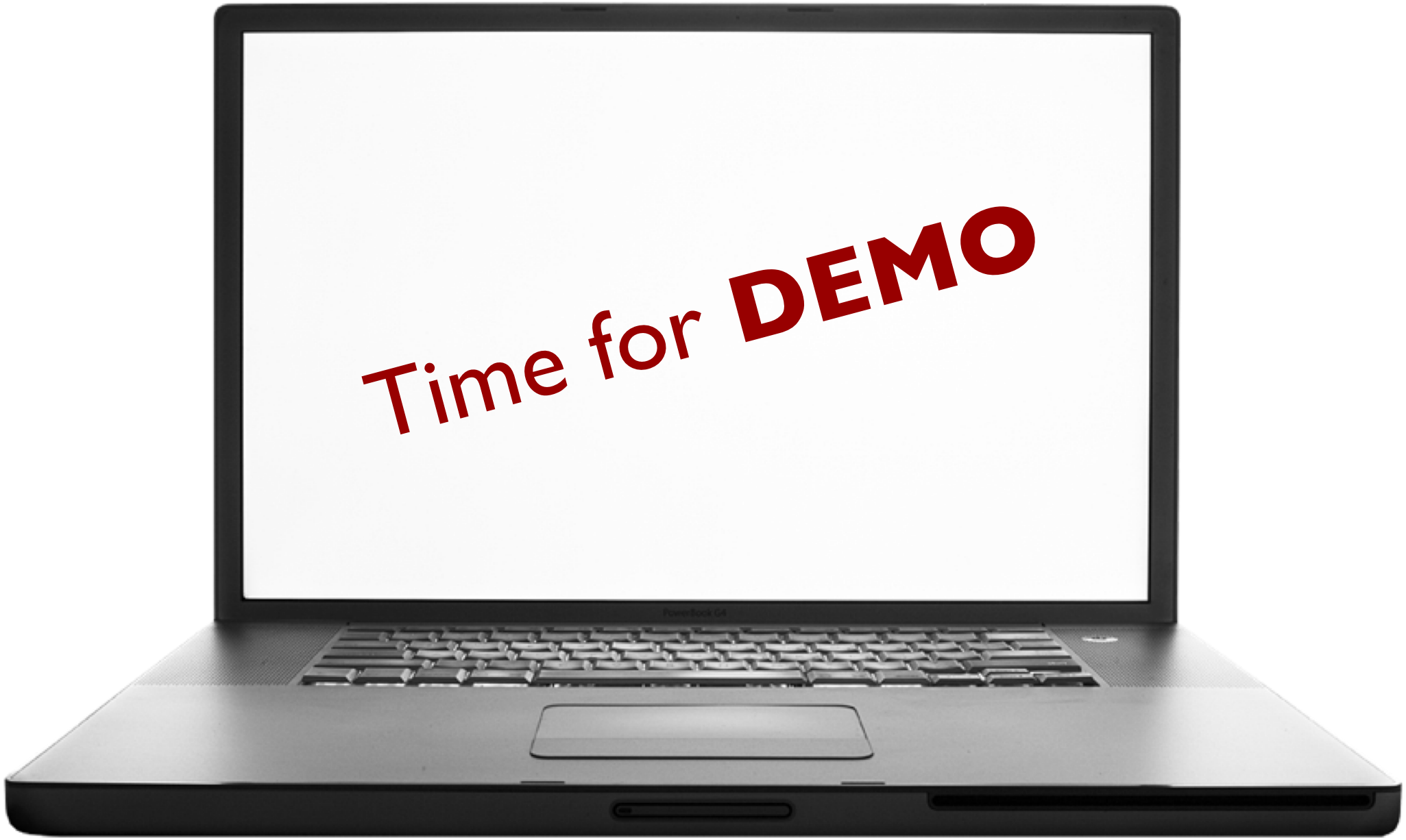
Kotlin

“Statically typed programming language
for the JVM, Android and the browser”

-- http://kotlinlang.org/

Our goal for DSL

- “Looks” declarative…
- But **is code**


Time for DEMO

Why not Groovy?

```
email {
   from 'dsl-guru@mycompany.com'
   to 'john.doe@waitaminute.com'
   subject 'The president has resigned!'
   body {
      p 'Really, the president did resign!'
   }
}
```

Why not Scala?

```
object SquareRoot extends Baysick {
  def main(args:Array[String]) = {
    10 PRINT "Enter a number"
    20 INPUT 'n
    30 PRINT "Square root of " % "'n is " % SQRT('n)
    40 END RUN
  }
}
```

@nicolas_frankel

hazelcast

Takeaways

1. 100% Java interoperability
2. Properties
3. Extension Functions
4. Named parameters
5. Default values
6. Lambdas with receiver
7. (Operator overloading)
8. (Reified generics)
9. (Infix)

Thanks a lot!

- https://blog.frankel.ch/
- @nicolas_frankel
- https://git.io/vd8d9