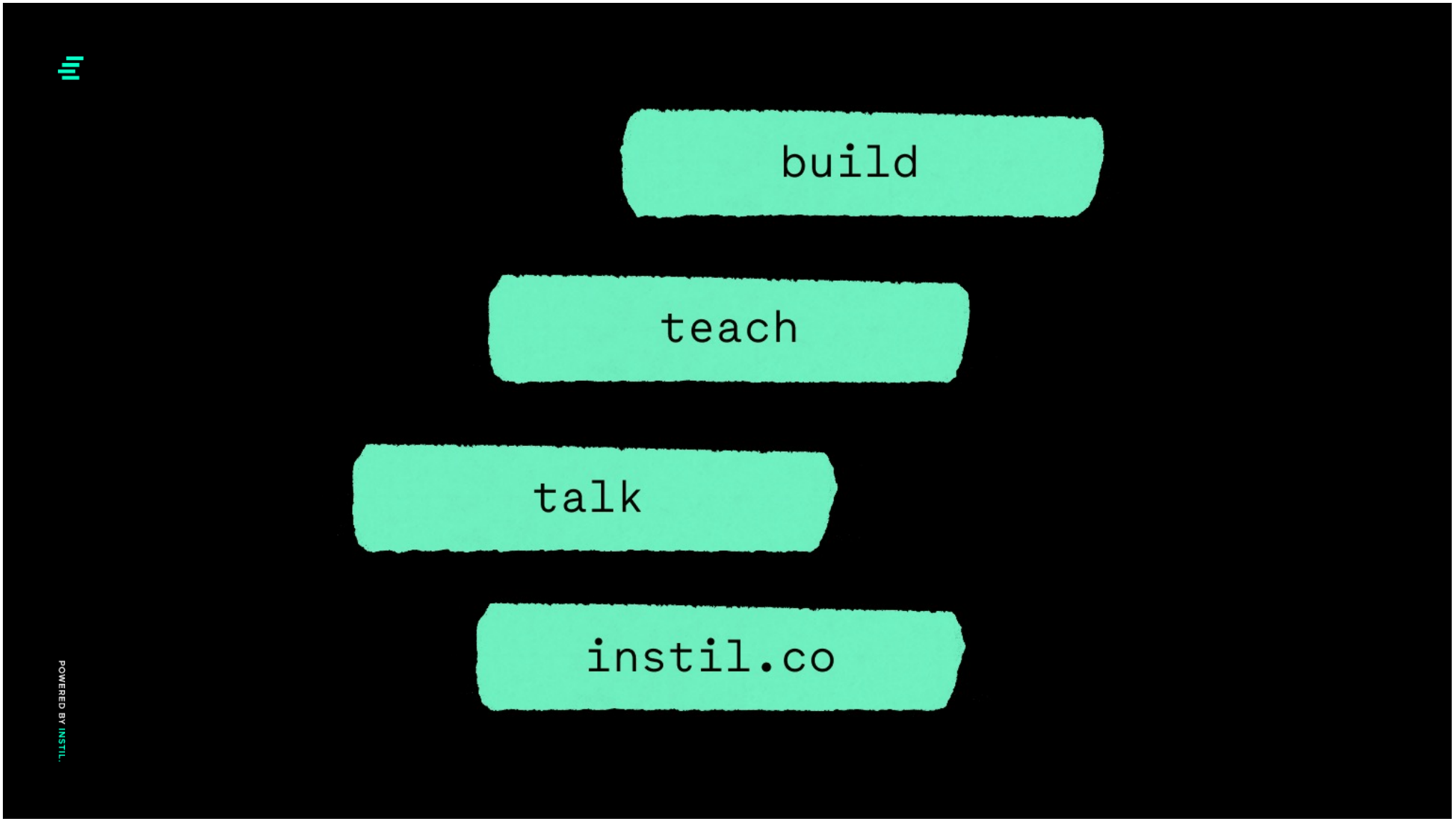
Kotlin Workshop

QUB Google Dev Student Club
November 2022

POWERED BY INSTIL.

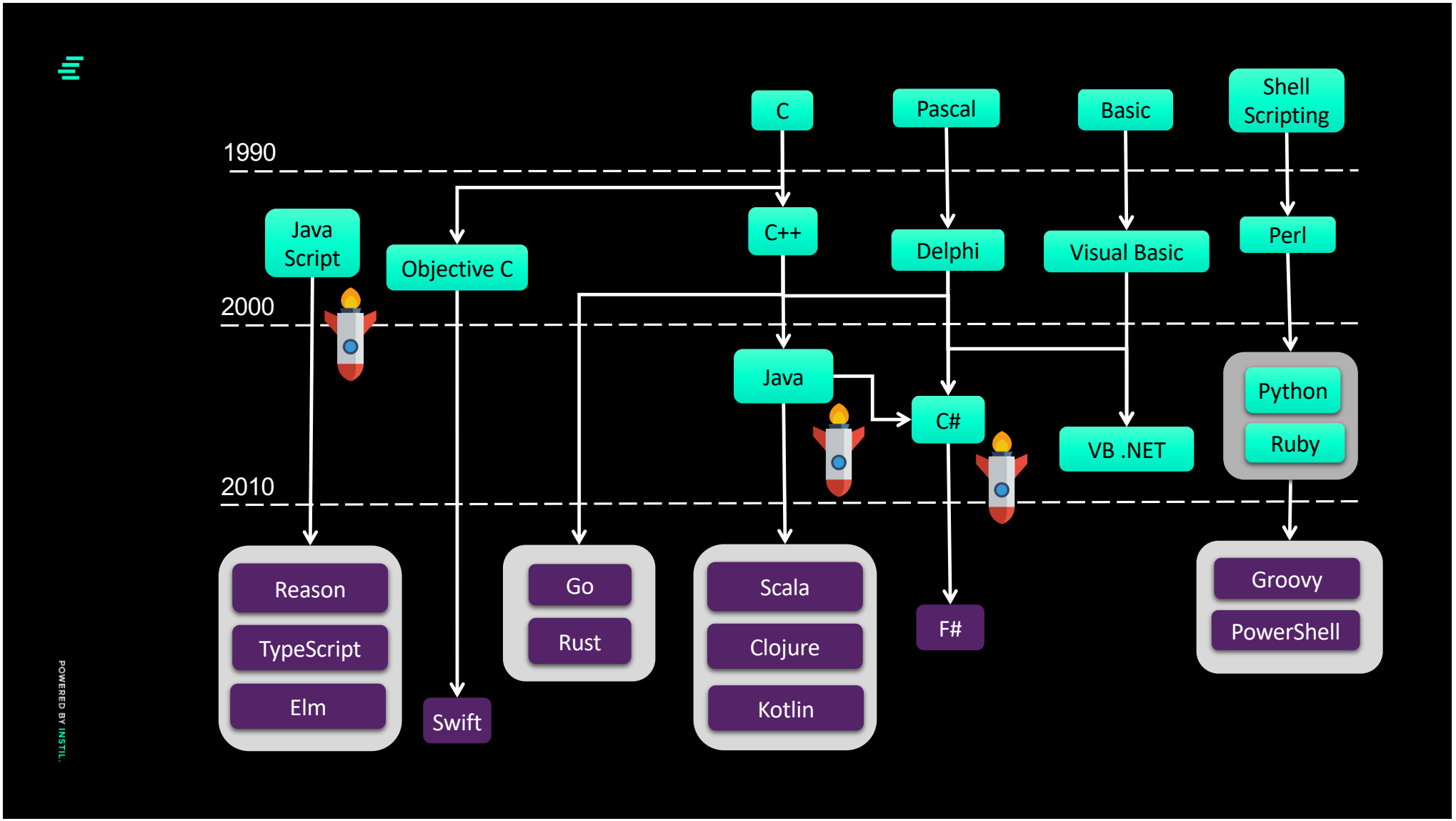

build
teach
talk
instil.co
POWERED BY INSTIL.


AMERICAN EXPRESS
SHOPKEEP
IBM
Liberty Mutual
citi
Bell
ATLASSIAN
intel
Johnson Controls
BMW
Deloitte.
POWERED BY INSTIL

Garth (@GarthGilmour)

Introduction

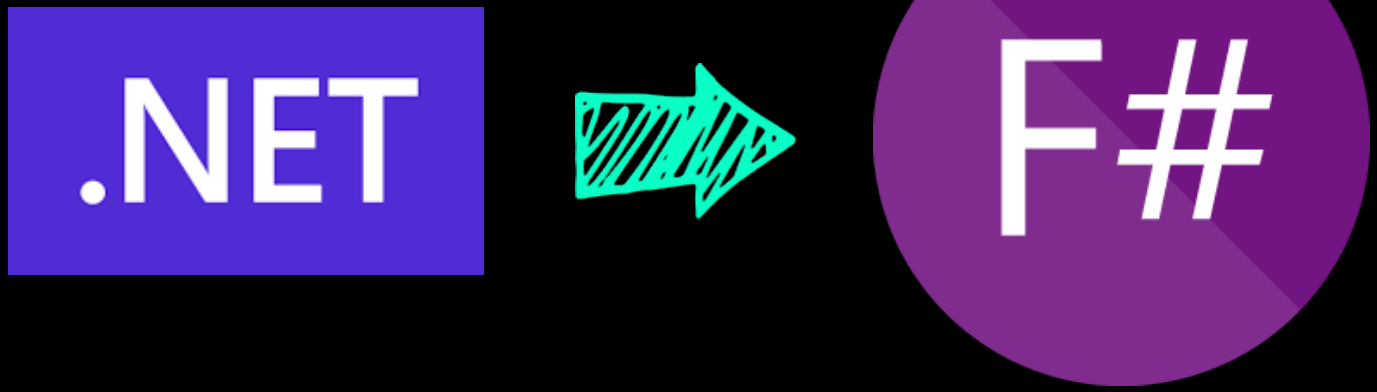
1


1990
2000
2010
C
Pascal
Basic
Shell Scripting
Java Script
Objective C
C++
Delphi
Visual Basic
Perl
Java
C#
VB .NET
Python
Ruby
Reason
TypeScript
Elm
Swift
Go
Rust
Scala
Clojure
Kotlin
F#
Groovy
PowerShell
POWERED BY INSTIL

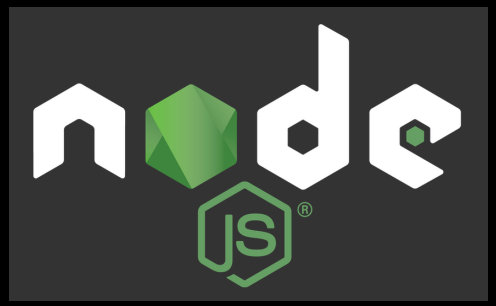
Java and the JVM

New Kids on the (JVM) Block

New Kids on the (CLR) Block

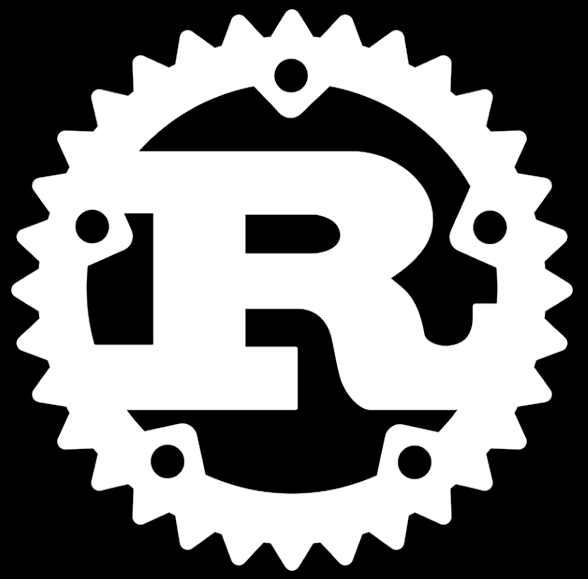


New Kids on the (JS) Block



New Kids on the (Mac) Block

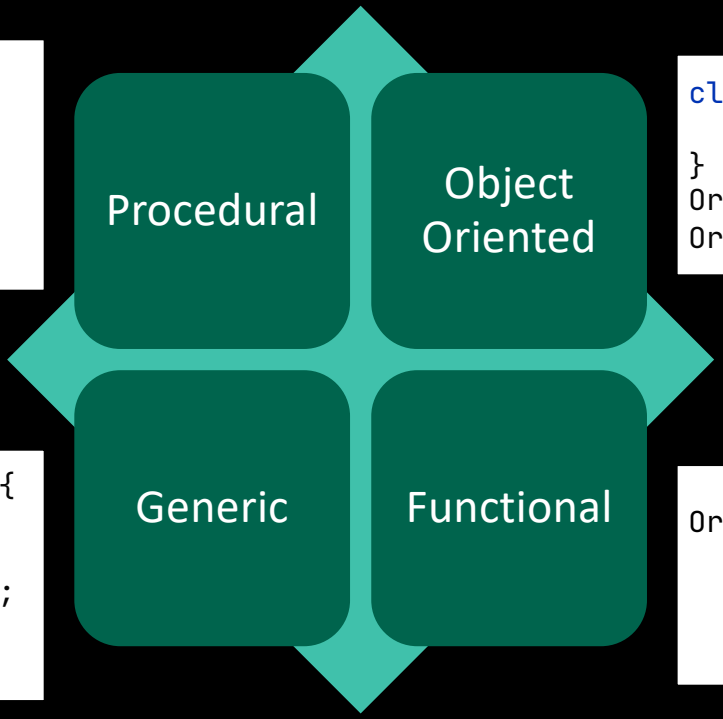
New Kids on the CPU




DON'T
PANIC

There Are Only So Many Paradigms

```
while(orders.hasNext()) {
   Order o = orders.next();
   if(o.priority == URGENT) {
      o.ship();
   }
}
```




```
class Order {
    // . . .
}
Order o1 = new Order();
Order o2 = new Order();
```

```
f1(List<? extends Order> data) {
   Optional<Client> opt = f2();
   for (Order o : data) {
      o.send(opt.getOrElse(me));
   }
}
```

```
Orders
   .filter(o -> o.value > 500)
   .map(o -> o.customer)
   .each(c -> print(c.name));
```

It is my conjecture that there has been nothing new in computing since the 1970's, with the one exception of AI

@IDOMDAVIS

DOM DAVIS

It is my conjecture that there has been nothing new in computing since the 1970's, with the one exception of AI which was 1984. Everything else is just relearning what we knew already

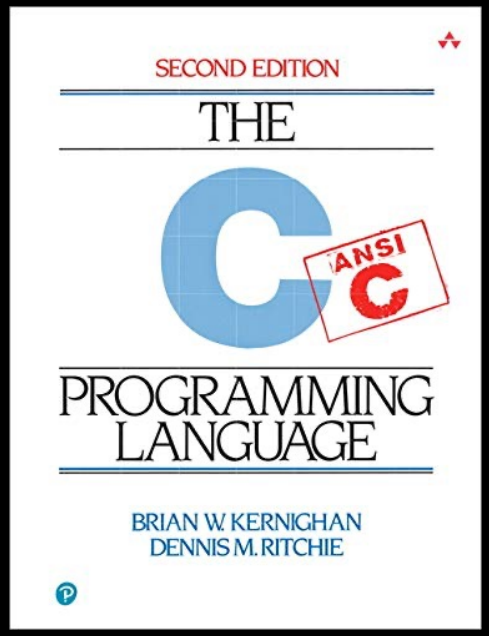
@IDOMDAVIS

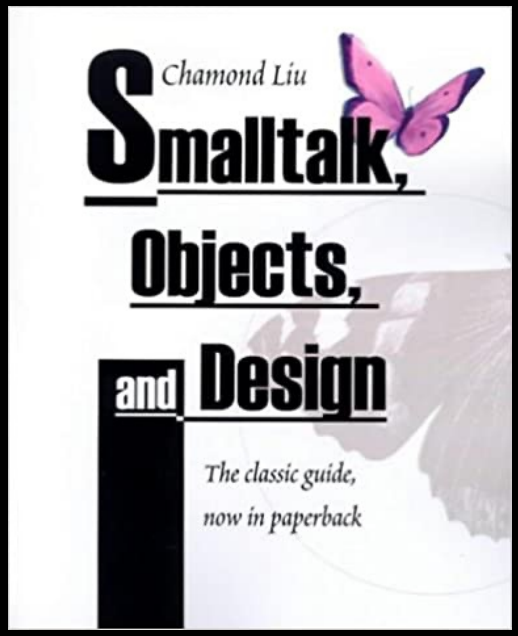
DOM DAVIS

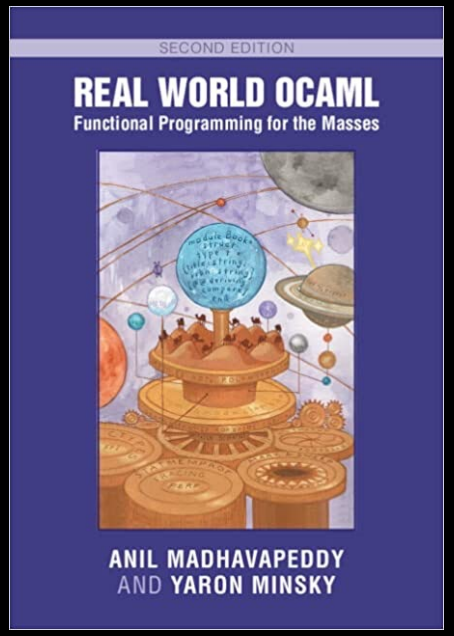
There Are Only So Many Paradigms
take the 'Bruce Wayne approach'

- **Focus on 'Root Styles' not 'Shiny Things'**
  - Procedural (Pull Based)
  - Declarative (Push Based)
  - Functional (Lambda Calculus)
  - Object Oriented (Behavior)
  - Generic (Type Based)

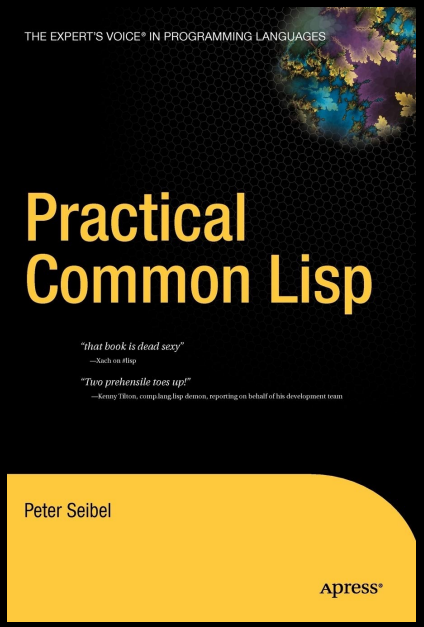
The Root Languages

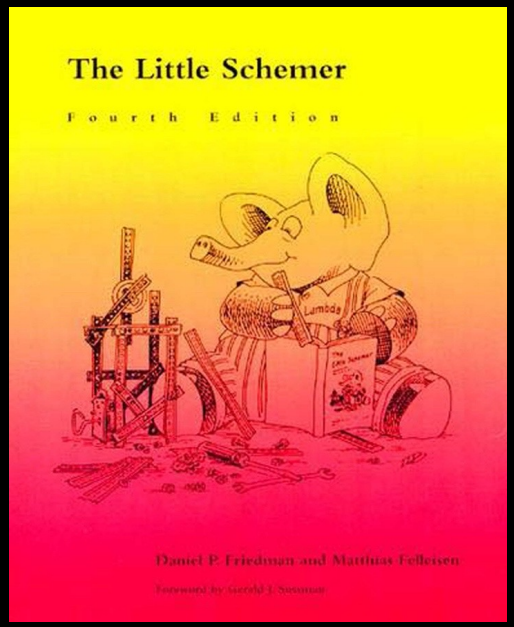






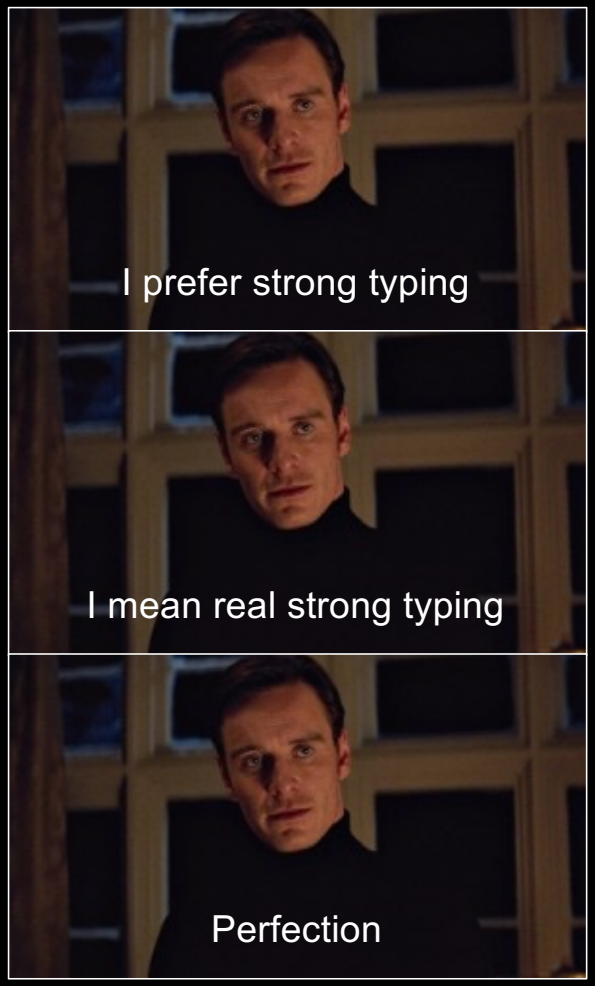
The Root Languages





Convergence

2



let m = new Mutant(“Raven”);

Mutant m = new Mutant(“Raven”);

val m = Mutant(“Raven”)

```scala
class Person(val name: String)

def display(p: Person) = println(s"Person called ${p.name}")

@main def demo() = {
  val things = List("Dave", "Jane", "Pete", "Lucy")

  things
    .map(Person(_))
    .foreach(display)
}
```

```
Person called Dave
Person called Jane
Person called Pete
Person called Lucy
```

```kotlin
class Person(val name: String)

fun display(p: Person) = println("Person called ${p.name}")

fun main() {
  val things = listOf("Dave", "Jane", "Pete", "Lucy")

  things
    .map { Person(it) }
    .forEach(::display)
}
```

```
Person called Dave
Person called Jane
Person called Pete
Person called Lucy
```

```
1   class Person{
2       constructor(public name: String) {}
3   }
4
5   const display = (p: Person) => console.log(`Person called ${p.name}`)
6
7   const things = ["Dave", "Jane", "Pete", "Lucy"]
8
9   things
10      .map(str => new Person(str))
11      .forEach(display)
```

```
Person called Dave
Person called Jane
Person called Pete
Person called Lucy
```

```java
public class Program {
    record Person(String name) {
    }

    static void display(Person p) {
        final var msg : String = "Person called %s".formatted(p.name);
        System.out.println(msg);
    }

    public static void main(String[] args) {
        var things : List<String> = List.of("Dave", "Jane", "Pete", "Lucy");

        things.stream() Stream<String>
                .map(Person::new) Stream<Program.Person>
                .forEach(Program::display);
    }
}
```

Kotlin

3

POWERED BY INSTIL.

kotlin's selling points

- 2011 Null Safety
- 2016 Shrinking Codebase
- 2017 Elegant DSL's
- 2018 Coroutines
- 2019 Multiplatform Libraries

POWERED BY INSTIL

Null Safety

String Templates

Default parameters

Extensions

Free Functions

Coroutines

Single Expression Functions

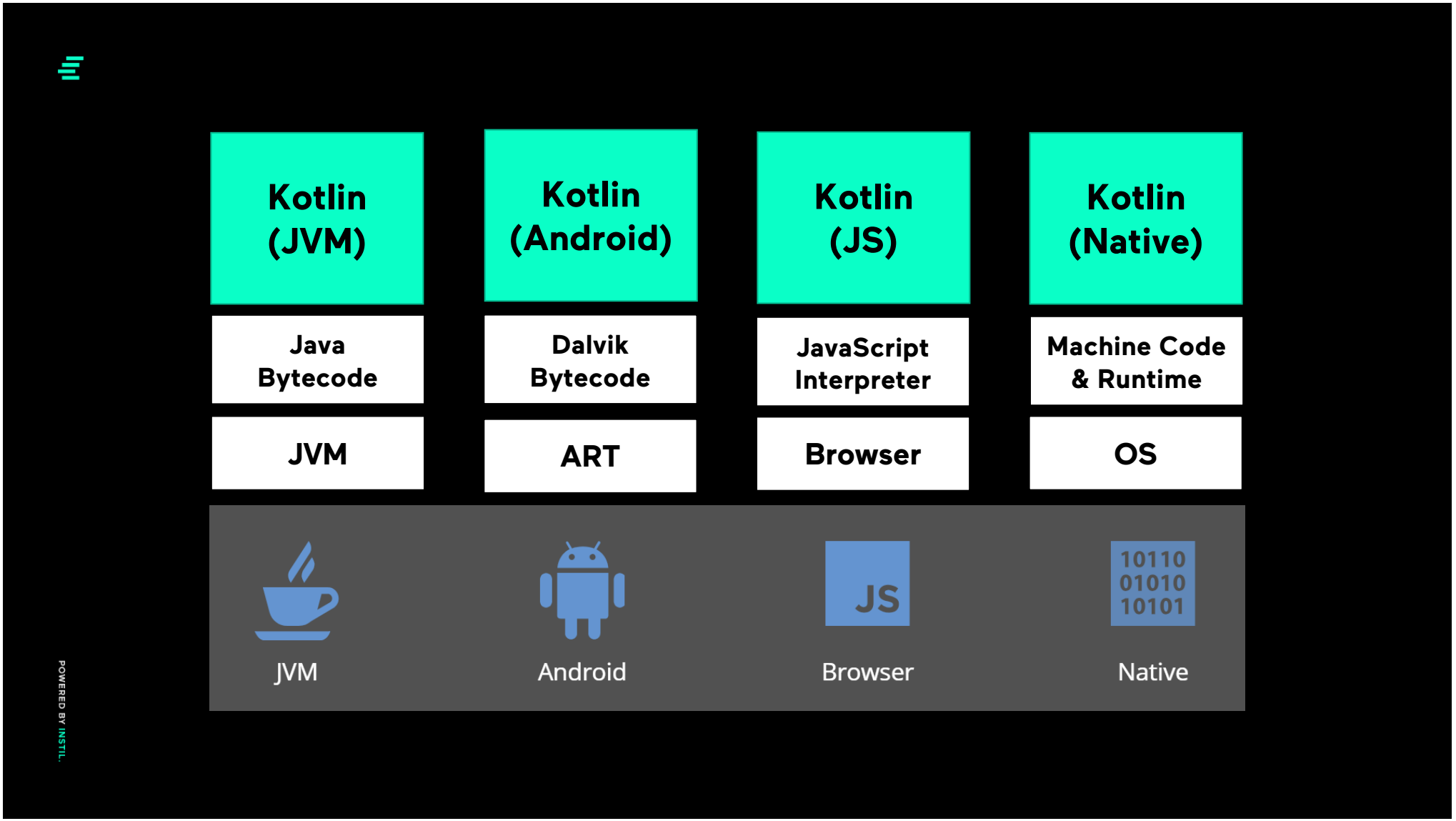
Reified generics

Data classes and Properties

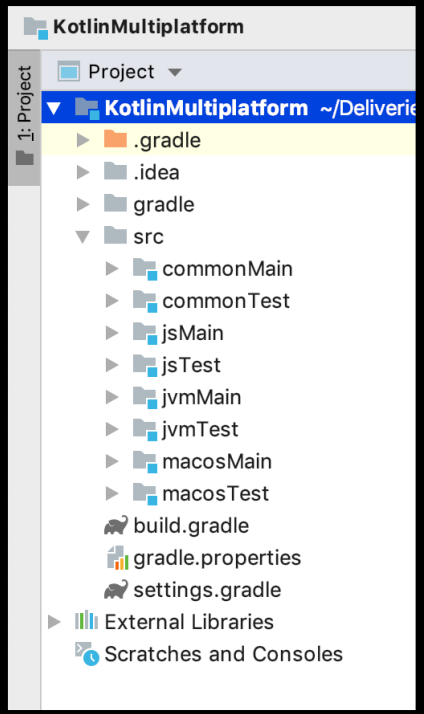
Type Inference

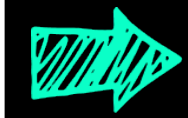
Smart Casts

Operator overloading


Kotlin (JVM)
Kotlin (Android)
Kotlin (JS)
Kotlin (Native)
Java Bytecode
Dalvik Bytecode
JavaScript Interpreter
Machine Code & Runtime
JVM
ART
Browser
OS
JS
10110
01010
10101
JVM
Android
Browser
Native
POWERED BY INSTIL

multiplatform libraries

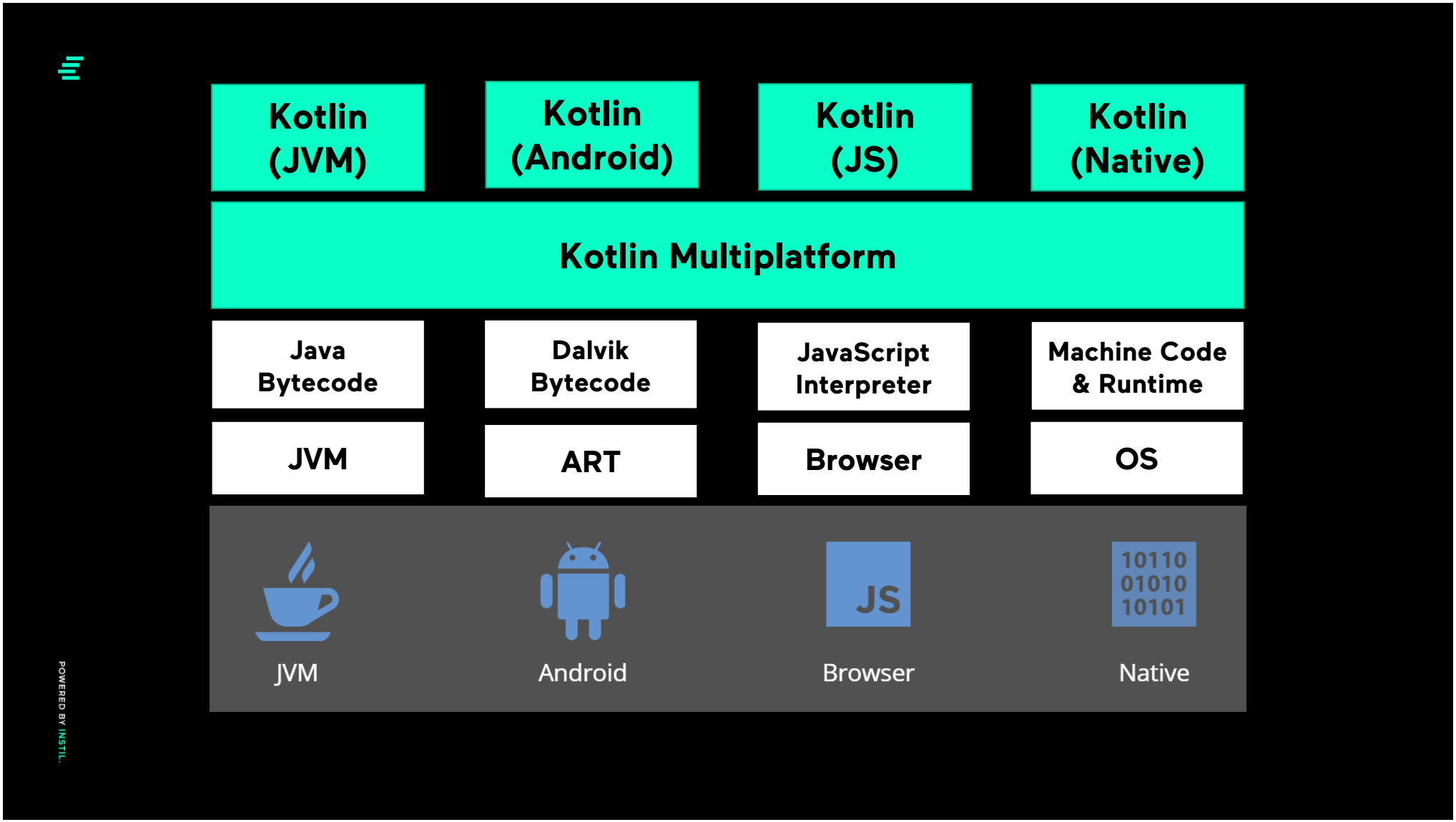




NATIVE ARTEFACT

JS BUNDLE

JAR


Kotlin (JVM)
Kotlin (Android)
Kotlin (JS)
Kotlin (Native)
Kotlin Multiplatform
Java Bytecode
Dalvik Bytecode
JavaScript Interpreter
Machine Code & Runtime
JVM
ART
Browser
OS
JS
10110
01010
10101
JVM
Android
Browser
Native
POWERED BY INSTIL

why use kotlin on android?
developer happiness

- **Codebase reduction by 40%**
  - (JetBrains figure)
- **Simplify the remaining code**
- **Plaster over Androids cracks**
- **Access FP features / patterns (Java 8+)**
- **Future proof your skill set**


Today we're announcing another big step: Android development will become increasingly Kotlin-first. Many new Jetpack APIs and features will be offered first in Kotlin. If you're starting a new project, you should write it in Kotlin
Google I/O 2019
POWERED BY INSTIL

why use kotlin on android?

it’s better with kotlin

Android Jetpack

Jetpack is a suite of libraries, tools, and guidance to help developers write high-quality apps easier. These components help you follow best practices, free you from writing boilerplate code, and simplify complex tasks, so you can focus on the code you care about.

Jetpack comprises the androidx.* package libraries, unbundled from the platform APIs. This means that it offers backward compatibility and is updated more frequently than the Android platform, making sure you always have access to the latest and greatest versions of the Jetpack components.

GET STARTED WATCH THE INTRO VIDEO

Accelerate development

Components are individually adoptable but built to work together while taking advantage of Kotlin language features that make you more productive.

Eliminate boilerplate code

Android Jetpack manages tedious activities like background tasks, navigation, and lifecycle management, so you can focus on what makes your app great.

Build high quality, robust apps

Built around modern design practices, Android Jetpack components enable fewer crashes and less memory leaked with backwards-compatibility baked in.

why use kotlin and spring boot?
it’s better with kotlin

5.2.1. Client Requester

To obtain an `RSocketRequester` on the client side requires connecting to a server along with preparing and sending the initial RSocket `SETUP` frame. `RSocketRequester` provides a builder for that. Internally uses RSocket Java’s `RSocketFactory`.

This is the most basic way to connect with default settings:

Java | Kotlin

KOTLIN

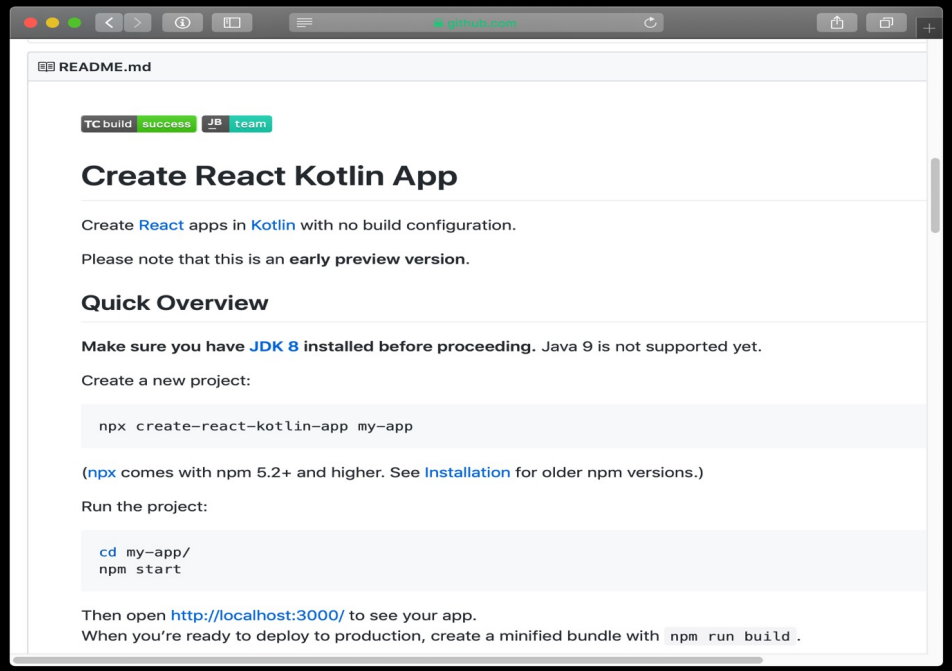
```kotlin
import org.springframework.messaging.rsocket.connectTcpAndAwait
import org.springframework.messaging.rsocket.connectWebSocketAndAwait

val requester = RSocketRequester.builder()
        .connectTcpAndAwait("localhost", 7000)

val requester = RSocketRequester.builder()
        .connectWebSocketAndAwait(URI.create("https://example.org:8080/rsocket"))
```

The above is deferred. To actually connect and use the requester:

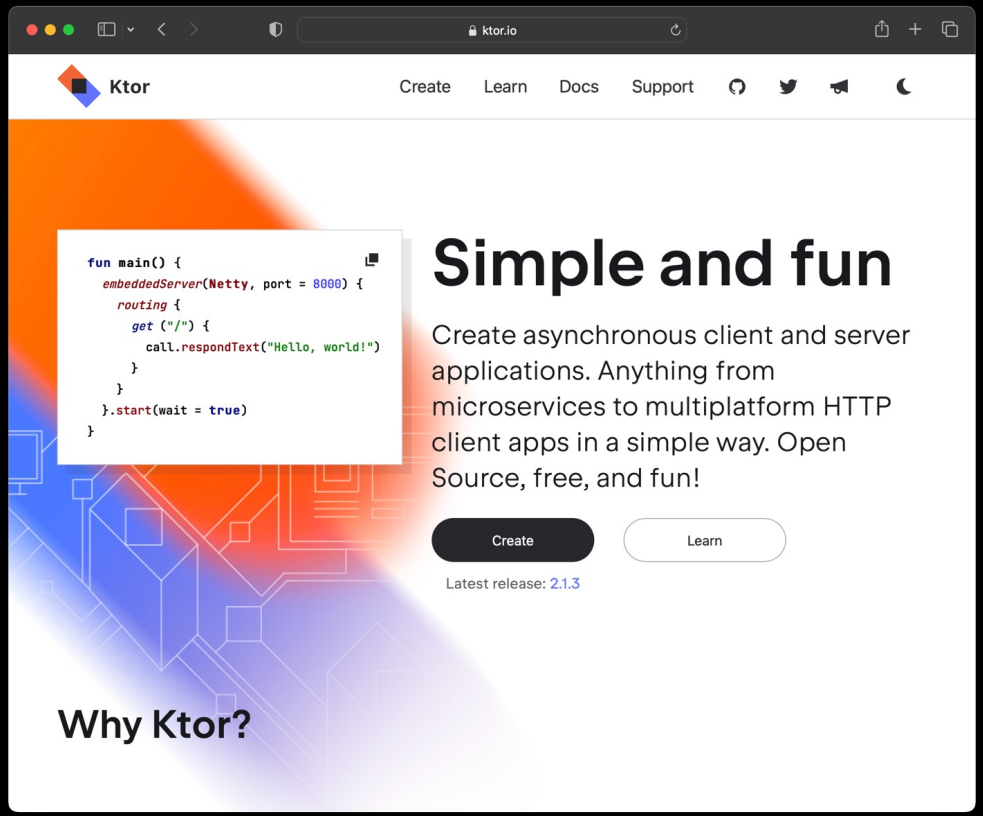
POWERED BY INSTIL.

why use kotlin and spring boot?
it’s better with kotlin

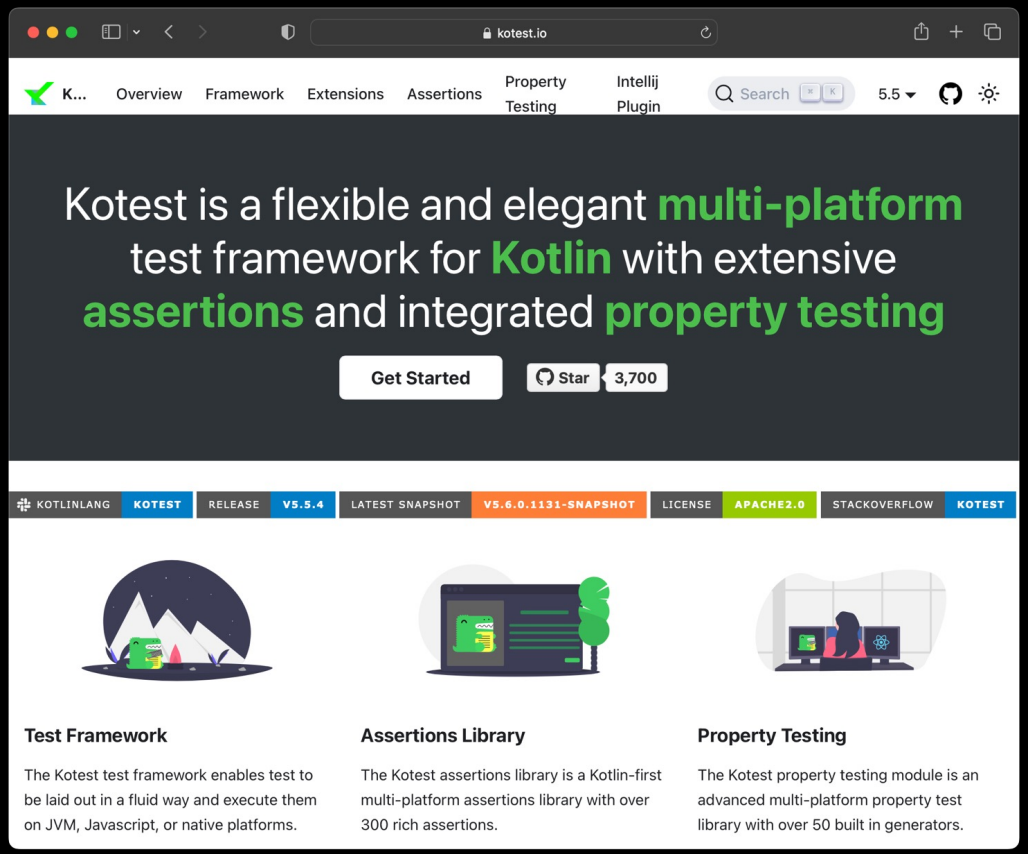
why use kotlin and javascript?
it’s better with kotlin (maybe)



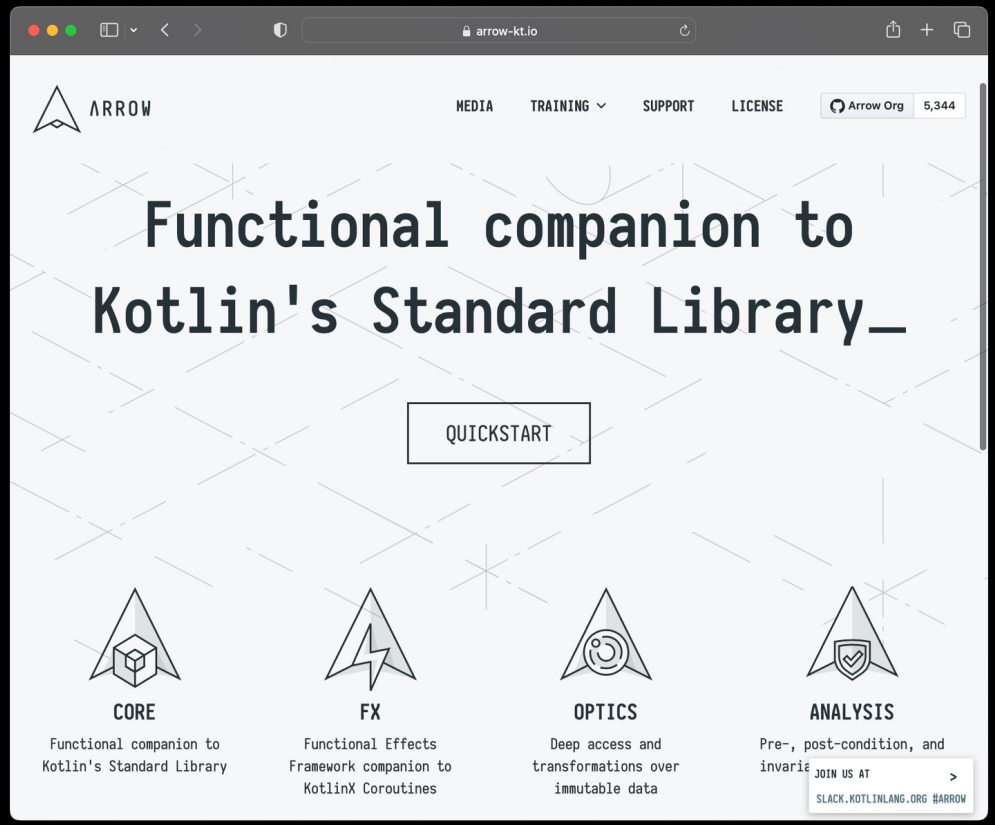
Kotlin Specific Libraries



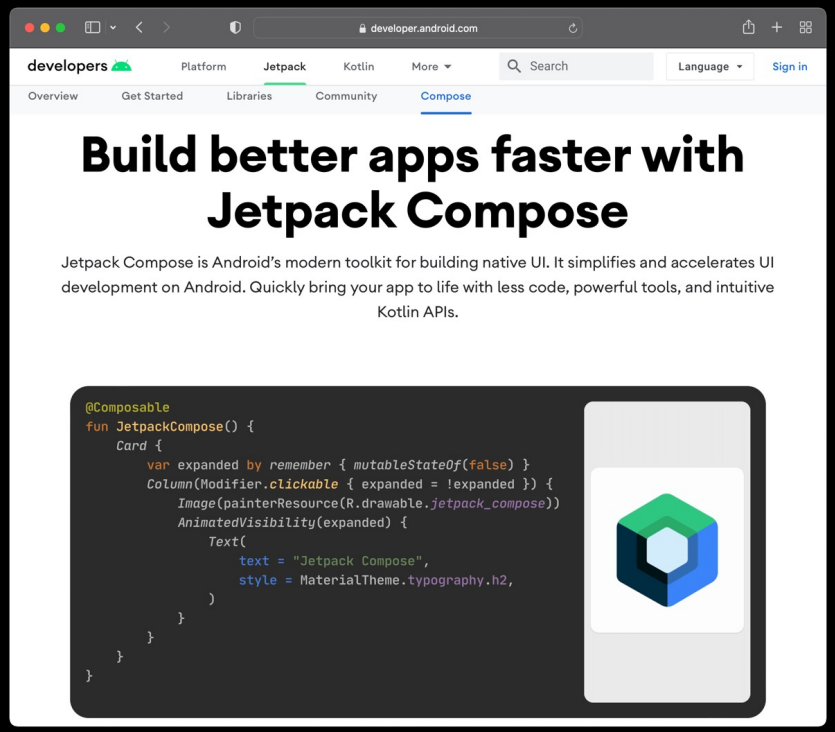
Kotlin Specific Libraries

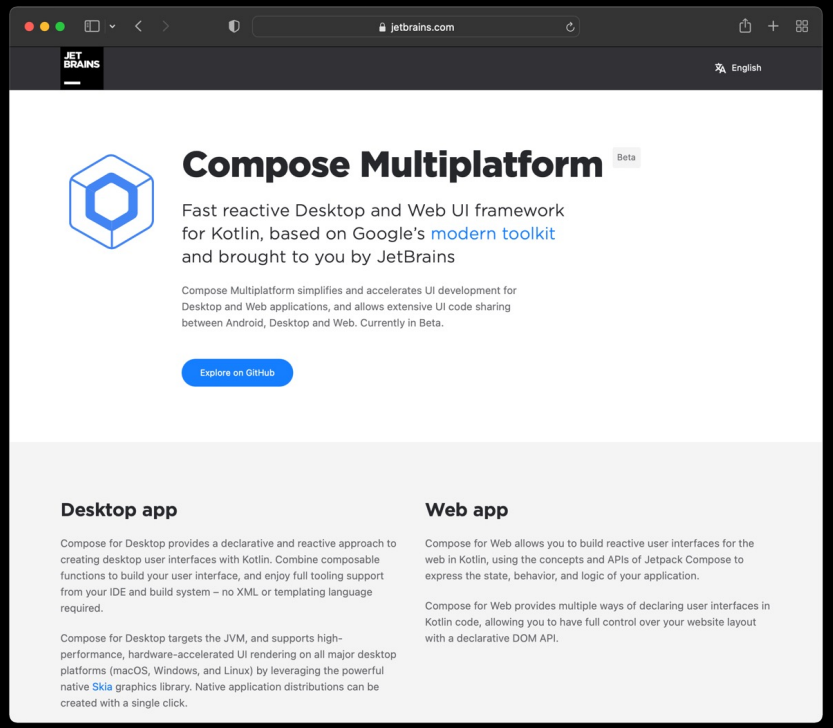


Kotlin Specific Libraries



Kotlin Specific Libraries





Let's Code!

4