Kotlin for the Pragmatic Functionalist

Paco Estevez
@pacomorks
Special thanks to Facebook for allowing me to work on this on my spare time. Kategory is not an internal Facebook project, nor it is an incubator. It is an external project of community contributors.

In any case facebers are bunch of devs excited about Kotlin tool
Kotlin is ready for Functional Programming TODAY

We’re growing
We’re improving
We want all of you
Our community goal

- Use of Kotlin in JS, Native, JVM
- One codebase for mobile, backend, frontend
- Same idioms across languages and ecosystems

The objective of this talk is to give you a peek at bleeding edge technologies for Kotlin, and what future language features will look like.
Where is Kotlin today
What are we bringing into the ecosystem
Where would we like to be
Kotlin today
Key cornerstones

- Data classes, sealed classes
- Inliner
- Extension functions
- Type aliases
- Enhanced resolution of generic parameters
## Data & sealed classes

<table>
<thead>
<tr>
<th>Data</th>
<th>Sealed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plain information</td>
<td>Pure behavior</td>
</tr>
<tr>
<td>Stores specifics</td>
<td>Indicates branching in execution</td>
</tr>
<tr>
<td>Closed and immutable by default</td>
<td>Checked at compile time</td>
</tr>
</tbody>
</table>

@pacoworks
Inliner

- Copies a function in its call site
- Inline functions’ body is copied at compile time
- All call sites with static dispatch are known at compile time
- Enables runtime generics
Extension functions

- Converts static functions into object methods
- All call sites with static dispatch are known at compile time
- Can extend just for specific generic parameters
Kotlin today with Kategory
Key features

- Extension Interfaces
- Ad-hoc polymorphism for generic type constructors
- Imperative async code
- Smarter boilerplate removal
Extension Interfaces

- Implement any interface for already existing types
- Use the extension interface as a parameter
- WIP: automatic extension interface lookup

Extension protocols
interface Jsonable<T> {
    fun toJson(T element): String

    fun fromJson(String json): T
}

fun <T> jsonable(): Jsonable<T> = // lookup code
object EmployeeJsonable: Jsonable<Employee> {
    val gson = Gson()

    fun toJson(Employee element): String =
        gson.toJsonString(element)

    fun fromJson(String json): Employee =
        gson.fromJsonString(json)
}

fun parser(
    company: Company,
    parser: Jsonable<Company> = jsonable<Company>()
): String =
    parser.toJson(company)
Introduce Extension Interfaces into the language under the name “Type Classes”

Compile time lookup & injection

Aim for parity with Swift’s Extension Protocols
Ad-hoc polymorphism for generic type constructors

- Functions that support any generic type constructor
- Express agnosticity to implementation
- Safe downcast from generic to implementation
Generic type constructor

interface Hk<F, A>

- F is the type of the container
- A is the type of the content
Generic constructor implementations

sealed class Option<A>: Hk<OptionHK, A>
data class ListWrap<A>(...) : Hk<ListWrapHK, A>
sealed class Try<A> : Hk<TryHK, A>

- F becomes a “tag” type
- A remains as the type of the content
Returning generic type constructors

```kotlin
fun <F> getUserById(
    id: String
): Hk<F, User> =
???
```
interface Factory<F> {
  fun <A> create(element: A): Hk<F, A>
}

fun <T> factory(): Factory<T> =
// lookup code
Polymorphism for generic type constructors

fun <F> getUserById(
    id: String,
    factory: Factory<F> = factory<F>()
): Hk<F, User> =
    factory.create(getUser(id))

How do I convert that F back to a real value?
Safely downcasting generic containers

- Extension functions can be defined to require just some of the generic parameters
- Extension functions are resolved at compile time
Safely downcasting generic containers

fun <A> Hk<OptionHK, A>.ev() =
    this as Option<A>

fun <A> Hk<TryHK, A>.ev() =
    this as Try<A>
Ad-hoc polymorphism for generic constructors

val a: Option<User> = getUserById("123").ev()
val b: Try<User> = getUserById("123").ev()
Ad-hoc polymorphism for generic constructors

object OptionFactory: Factory<OptionHK> {
    fun <A> create(element: A): Hk<OptionHK, A> = Option.Some(a)
}

object TryFactory: Factory<TryHK> {
    fun <A> create(element: A): Hk<TryHK, A> = Try { a }
}
KindedJ

- Initiative to share generic type constructors under the name “Higher Kinded Types”
- We’re speaking with other FP libraries for Java & eta-lang
Imperative async code

- Express asynchronous sequential and parallel execution as if they were synchronous
- For any existing framework and abstraction
fun <F> getUserFriends(
    id: String
): Observable<List<User>> =
    getUserById(id).flatMap { user ->
        Observable.merge(
            user.friends.map { friend ->
                getUserById(friend.id)
            }
        ).toList()
    }
fun <F> getUserFriends(
    id: String,
    coroutinable: Coroutinable<F> = coroutinable<F>()
): Hk<F, List<User>> =
    coroutinable.bindingE {
        val user = getUserById(id).bind()
        val friendProfiles = user.friends.map { getUserById(it.id).bind() }
        yields(friendProfiles)
    }
Ad-hoc polymorphism for imperative async code

```scala
val a: Either<Exception, List<User>> = getUserFriends("123").ev()
val b: Try<List<User>> = getUserFriends("123").ev()
val c: ObservableKW<List<User>> = getUserFriends("123").ev()
val d: IO<List<User>> = getUserFriends("123").ev()
```
Coroutinable?

typealias Coroutinable<F> = MonadError<F>
Effects with MonadError

- Thin Extension Interface that can be implemented by any existing framework & abstraction
- With error handling, stack safety, threading & cancellation out of the box
- Current integrations: RxJava, kotlin.coroutines, IO
- Potential integrations: CompletableFuture, kotlinx.coroutines, AsyncTask, ArchComponents, Kovenant...
Implementations in Kategory

- Error handling: Option, Try, Validated, Either, Ior
- Collections: List, Sequence, Map, Set
- RWS: Reader, Writer, State
- Transformers: ReaderT, WriterT, OptionT, StateT, EitherT
- Evaluation: Eval, Trampoline, Free, Function0
- Others: Coproduct, Coreader, Const...
Smarter boilerplate removal

- Eugenio Marletti’s collaboration & mentorship
- Generate boilerplate for all the features show using annotations
Bye bye boring code

- **Generic constructors**
  - `@higherkind`

- **Extension Interfaces**
  - `@derives`
  - `@instance`

- **Manipulation of immutable data & sealed classes**
  - kategory-optics by Simon Vergauwen
Challenges & Pitfalls

- IDE resolution of deeply nested generics
- Scopes with ambiguity for `it` and `this`
- Reified generics are “contagious”
- Unexpected behavior caused by inlining
- Generics and interoperating with Java
- Error recovery in coroutines

And many more, come talk to us!
Where would we like to be
Bring your tools in

- Threading
- Databases
- Network
- Error Handling
- Reactive Programming
Bring your tools in

- Threading
- Databases
- Network
- Error Handling
- Reactive Programming
- Parsers
- Compilers
- Static analyzers
- UI frameworks
- Math & statistics
Bring your tools in

- Threading
- Databases
- Network
- Error Handling
- Reactive Programming
- Parsers
- Compilers
- Static analyzers
- UI frameworks
- Math & statistics
- Data science
- Machine learning
- Videogames
- Computer vision
- Serialization
Bring your tools in

- Category Theory
- Threading
- Web frameworks
- Databases
- New Architectures
- Network
- Error Handling
- Reactive Programming
- Parsers
- Compilers
- Static analyzers
- UI frameworks
- Math & statistics
- Data science
- Machine learning management
- Videogames
- Collection libraries
- Computer vision
- Stream pipelines
- Serialization
We’re growing our tools
We’re improving our ecosystem
We want all of you to be part of it
Kategory: [category.io](http://category.io)

Introduction to Kategory: [tinyurl.com/KatIntro](http://tinyurl.com/KatIntro)

KEEP–87: [github.com/Kotlin/KEEP/pull/87](https://github.com/Kotlin/KEEP/pull/87)

Gitter: [gitter.im/kategory](https://gitter.im/kategory)

#kategory in KotlinLang
#kategory in ASG

Special thanks to all Kategory contributors & supporters!

[ pacoworks.com](http://pacoworks.com)
[ @pacoworks](http://twitter.com/pacoworks)
[ github.com/pakoito](http://github.com/pakoito)
[ Slides: tinyurl.com/FunKotlinSF17](http://tinyurl.com/FunKotlinSF17)