# Unleashing Kotlin’s Hidden Arsenal

# `whoami`

- Pasha Finkelshteyn
- Dev 🥑 at JET BRAINS
- ≈10 years in JVM. Mostly ☕ and
- Started in pre-1.0 releases
- 🐦 asm0di0
- @asm0dey@fosstodon.org

# What you already know

## Hopefully

- Type-level null-safety
- Operator overloading
- Building DSLs
- Awesome ☕ interop

# Compatible with

- Spring
- Ktor
- Vert.x
- Jackson
- JPA
- younameit

# But how do we start using it?

# Tests

- Learn the syntax
- Learn the best practices
- Understand the power

# Do we just use JUnit as usual?

You can!

But why limit yourself?

Let’s start with libraries for testing!

# Mocking with `mockk`

https://mockk.io

```kotlin
val car = mockk<Car>()
every { car.drive(Direction.NORTH) } returns Outcome.OK
car.drive(Direction.NORTH) // returns OK
verify { car.drive(Direction.NORTH) }
confirmVerified(car)
```

# Mocking with `mockk`

https://mockk.io

```kotlin
val repository = mockk<Repository>() {
  every { getUsers() } returns listOf(User("Pasha"), User("J. P. Morgan"))
  every { roles() } returns listOf(
    mockk {
      every { name } returns "admin"
      every { rights } returns listOf("delete", "read", "write")
    }
  )
}
val service = Service(repository)
service.findAllUsers()
// checks
confirmVerified(repository) // throws exception because roles were not accessed
```

# What do we do after mocking?

We assert!

# AssertiK

willowtreeapps/assertk

AssertJ-inspired assertions

```kotlin
val person = Person(name = "Bob", age = 18)
assertThat(person.name).isEqualTo("Alice")
// --> expected:<["Alice"]> but was:<["Bob"]>
assertThat(person.age, "age").isGreaterThan(20)
// --> expected [age] to be greater than:<20> but was:<18>
assertThat(person::name).isEqualTo("Alice")
// --> expected [name]:<["Alice"]> but was:<["Bob"]>
```

# kotlin-compile-testing

https://github.com/tschuchortdev/kotlin-compile-testing