

Kotlin/JS

Stay typesafe in the browser with Kotlin

Harald Pehl
02/2021

Java User Group
Metropolregion Nürnberg



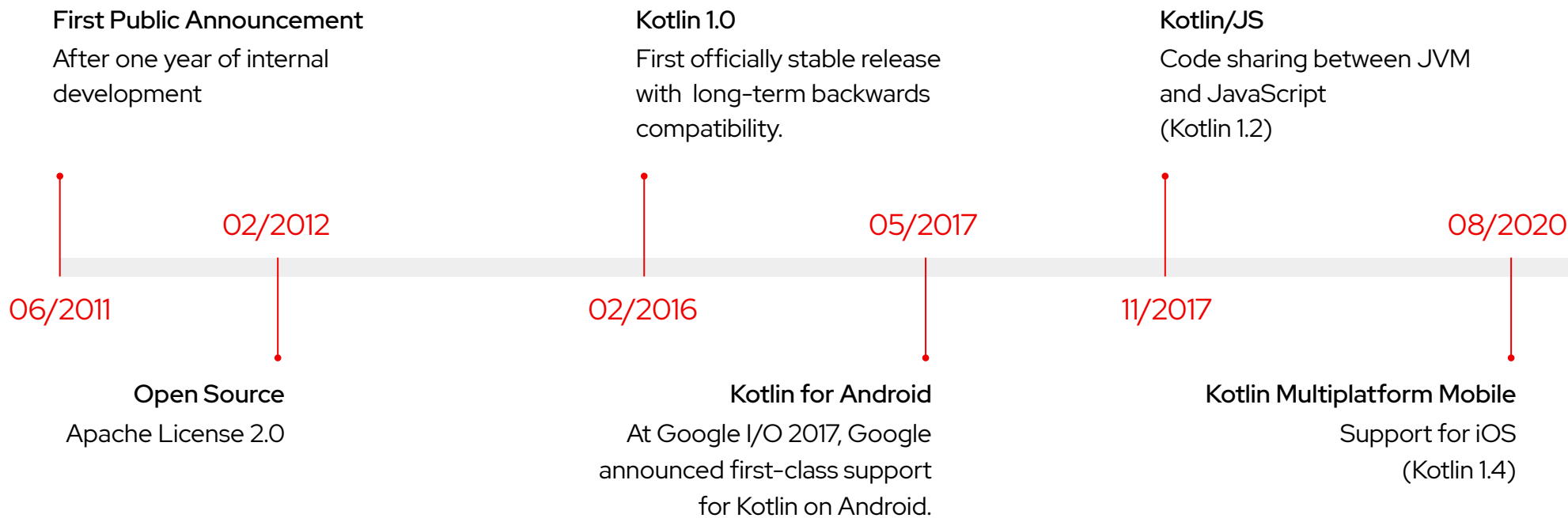
Harald Pehl

- ▶ Senior Software Engineer at Red Hat
- ▶ WildFly Management / HAL / halOS
- ▶ PatternFly Fritz2

What we'll discuss today

- ▶ Kotlin
- ▶ Getting Started
- ▶ Ecosystem
- ▶ JavaScript
- ▶ React
- ▶ Fritz2 / PatternFly

A Brief History



Highlights

Why should I use Kotlin?



Java Interoperability

No setup necessary
Getters / Setters
Static Members



Concise Syntax

Type Inference
Data Classes
Extension Functions



Safety First

Null Safety
Immutable



Concurrency

Suspend
Coroutines
Channels

```

// POJO with getters, setters, `equals()`, `hashCode()`,
// `toString()` and `copy()` in a single line
data class Customer(val name: String, val email: String, val company: String)

// Use `object` to create a singleton
object ThisIsASingleton {
    val companyName: String = "Red Hat"
}

// Extension function for string
fun String.reverseCase(): String = this.map {
    if (it.isUpperCase()) it.toLowerCase() else it.toUpperCase()
}.joinToString("")

println("Hello World".reverseCase()) // "hELLO wORLD"

// Operator extension function for List<Int>
operator fun List<Int>.times(by: Int): List<Int> = this.map { it * by }

// filter a list using a lambda
val numbers = listOf(4, -6, 2, -8, 12).filter { it > 0 }

// same as `numbers.times(2)`
println(numbers * 2) // "[8, 4, 24]"

```

Concise Syntax

Data classes

Objects

Extension functions

Operator overloading

```
var a: String = "abc" // Regular initialization means non-null
a = null // compilation error
```

```
var b: String? = "abc" // can be set null
b = null // ok
```

```
val l = a.length
val l = b.length // error: variable 'b' can be null
val l = if (b != null) b.length else -1
val l = b?.length ?: -1
val l = b!!.length
```

```
println(a?.length) // Unnecessary safe call
println(b?.length)
```

```
fun calculateTotal(obj: Any) {
    if (obj is Invoice)
        obj.calculateTotal()
}
```

Safety

Built in null safety

Smart casts

```

import kotlinx.coroutines.async
import kotlinx.coroutines.delay
import kotlinx.coroutines.runBlocking

suspend fun compute(n: Long): Long {
    delay(100) // simulate computation
    return n
}

val sum = runBlocking {
    (1..1_000_000L).map {
        async { compute(it) }
    }.sumOf { it.await() }
}
println("Sum: $sum")

```

delay	100	200	500	1000	2000
Ø time of 10 runs	2489	2533	2687	2978	3821

Concurrency

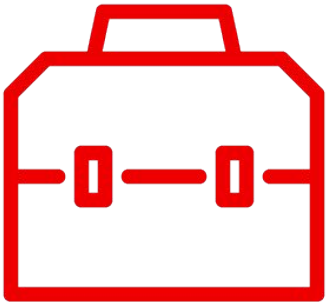
Coroutines

Suspend

Async

Kotlin/JS Setup

How to create a new project



Gradle Build Script

Write from scratch or generate using IDE wizards.

Standard Library

Most of the Kotlin standard library is available for Kotlin/JS.

Webpack

Webpack is used to build, bundle and run the application.

NPM / Yarn

Declare npm dependencies in the Gradle build script.

Run Debug Test

Using the Kotlin/JS
gradle plugin



Run

The run task that lets you run Kotlin/JS projects without additional configuration.



Debug

Source maps are generated automatically for debugging the code using browser development tools.



Test

Run tests through a variety of test runners that can be specified via the Gradle configuration.

For advanced testing see <https://kotest.io>.

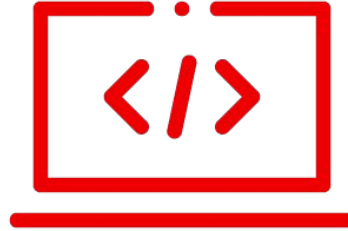
```
plugins {  
    kotlin("js") version "1.4.30"  
}  
  
group = "com.redhat.kotlinjs"  
version = "0.0.1"  
  
dependencies {  
    testImplementation(kotlin("test-js"))  
}  
  
kotlin {  
    js {  
        browser {  
            testTask {  
                useKarma {  
                    useChromeHeadless()  
                }  
            }  
            binaries.executable()  
        }  
    }  
}
```

Gradle Build Script

Project coordinates

Dependencies

Run configuration

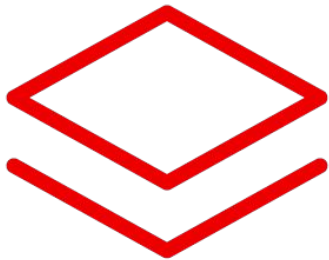


getting-started

<https://github.com/hpehl/kotlinjs-talk/tree/main/getting-started>

Kotlin Ecosystem

Useful libraries for the JS platform



kotlinx-html

DSL for typesafe HTML

kotlinx-serialization-json

(De)serialize from / to JSON

kotlinx-coroutines-core

Concurrency in the browser

ktor-client

Fetch data from the backend

```
ul {  
  listOf("Java", "Kotlin", "Scala").forEach {  
    li {  
      +it  
    }  
  }  
}
```

```
<ul>  
  <li>Java</li>  
  <li>Kotlin</li>  
  <li>Scala</li>  
</ul>
```

Typesafe HTML

Based on a DSL

Fully integrated

```
@Serializable
data class Todo(
    val id: Int,
    val name: String,
    val done: Boolean
)

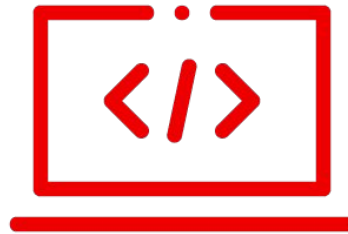
{
    "id": 23,
    "name": "Buy milk",
    "done": false
}
```

Serialization

Reasonable defaults

Multiple formats

Annotation based



ecosystem

<https://github.com/hpehl/kotlinjs-talk/tree/main/ecosystem>

JavaScript Interop

How to interact with the JavaScript ecosystem



Browser & DOM API

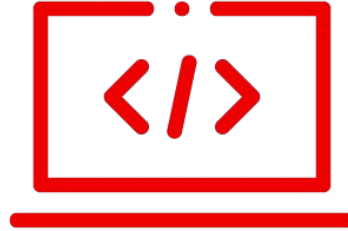
Typesafe wrappers for the DOM API

Use JavaScript code

Dynamic type, external declarations, Dukat

NPM dependencies

Managed by the gradle build script

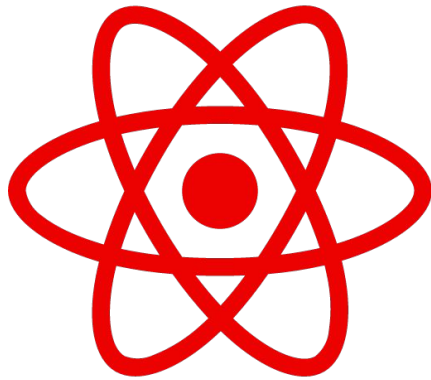


jsinterop

<https://github.com/hpehl/kotlinjs-talk/tree/main/jsinterop>

Kotlin/JS for React

How to create own and use existing React components



IDE Support

Wizards to get started

DSL

Write typesafe HTML and CSS

Components

Create own and integrate existing components

```
external interface CounterProps : RProps {
    var step: Int
}

external interface CounterState : RState {
    var value: Int
}

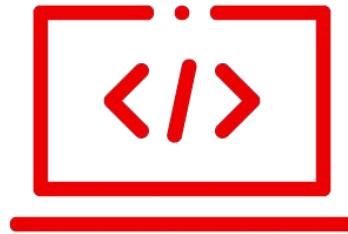
class CounterComponent : RComponent<CounterProps, CounterState>() {

    override fun CounterState.init() { value = 0 }

    override fun RBuilder.render() {
        button {
            +"dec"
            attrs.onClickFunction = { setState { value -= props.step } }
        }
        +state.value.toString()
        button {
            +"inc"
            attrs.onClickFunction = { setState { value += props.step } }
        }
    }
}
```

Typesafe
React Component

Properties
State
Component



react

<https://github.com/hpehl/kotlinjs-talk/tree/main/react>

fritz2

Reactive web-apps in pure Kotlin



Pure Kotlin

No external dependencies, just coroutines and flows

Databinding

No virtual DOM necessary

Stores & Handlers

Built around just a few basic concepts

```
class CounterStore(val step: Int) : RootStore<Int>(0) {
    val decrement: Handler<Unit> = handle { current -> current - step }
    val increment: Handler<Unit> = handle { current -> current + step }
}

fun RenderContext.counter(step: Int) {
    val store = CounterStore(step)
    div {
        button {
            +"dec"
            clicks handledBy store.decrement
        }
        store.data.asText()
        button {
            +"inc"
            clicks handledBy store.increment
        }
    }
}

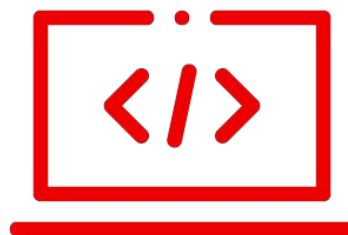
fun main() {
    appendToBody(renderElement {
        counter(10)
    })
}
```

Fritz2

Store

Handler

Tags



fritz2

<https://github.com/hpehl/kotlinjs-talk/tree/main/fritz2>

PatternFly Fritz2

PatternFly

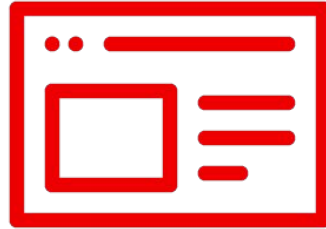
Open source design system using clear standards, components and layouts.

Fritz2

Reactive applications in pure Kotlin using coroutines and flows

PatternFly Fritz2

Implements PatternFly components using fritz2.



PatternFly Fritz2 Showcase

<https://patternfly-kotlin.github.io/patternfly-fritz2-showcase/#home>

Links

- ▶ [Kotlin](#)
- ▶ [Kotlin/JS](#)
- ▶ [Slack Channel](#)
- ▶ [Playground](#)
- ▶ [React Hands-On](#)
- ▶ [PatternFly Fritz2](#)
- ▶ [Code Samples](#)

Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.

 linkedin.com/company/red-hat

 youtube.com/user/RedHatVideos

 facebook.com/redhatinc

 twitter.com/RedHat