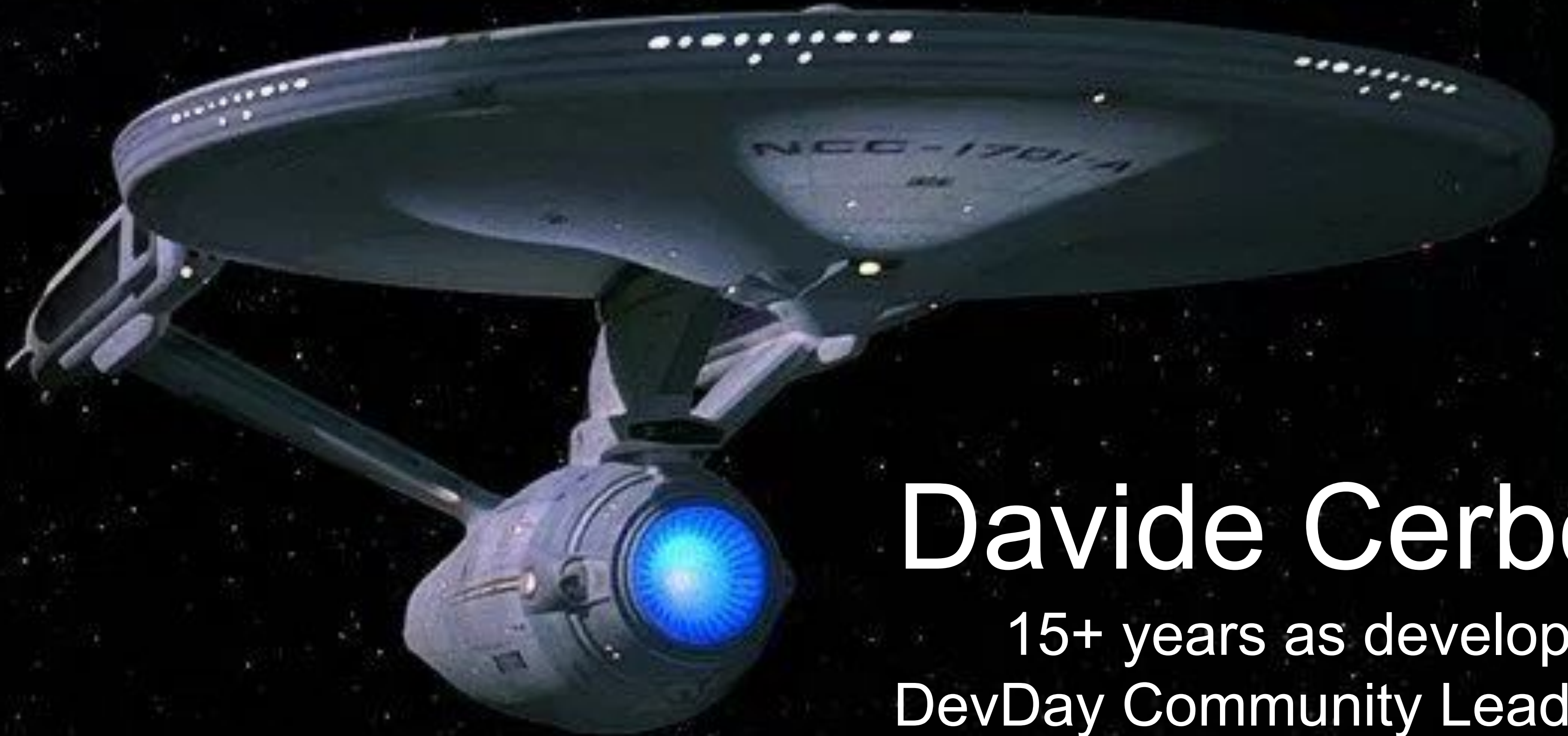
Kotlin loves React

Davide Cerbo
Software Architect @ E.m.m. Informatica

Pescara - 22 February 2019



Davide Cerbo

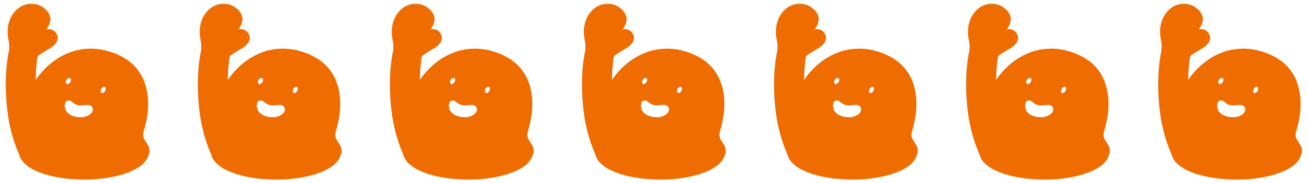
15+ years as developer

DevDay Community Leader

@davide_cerbo

https://medium.com/@davidecerbo

Raise your hands!!!



My first API with Ktor

```
fun main(args: Array<String>) {
   val server = embeddedServer(Netty, 8080) {
       install(ContentNegotiation) {
           jackson { enable(SerializationFeature.INDENT_OUTPUT) // Pretty Prints the JSON }
       }
       routing {
           get("/greetings") {
               call.respond(Greetings().sayGreetings())
           }
       }
   }
   server.start(wait = true)
}
```

but...

KotlinJS can!

- No longer experimental since 1.1.0
- Can run anywhere where JS runs
- Can interoperate with JS
- Can reuse code
- Can use the same Kotlin/JVM IDE

wait, React?

React

A JavaScript library for building user interfaces.

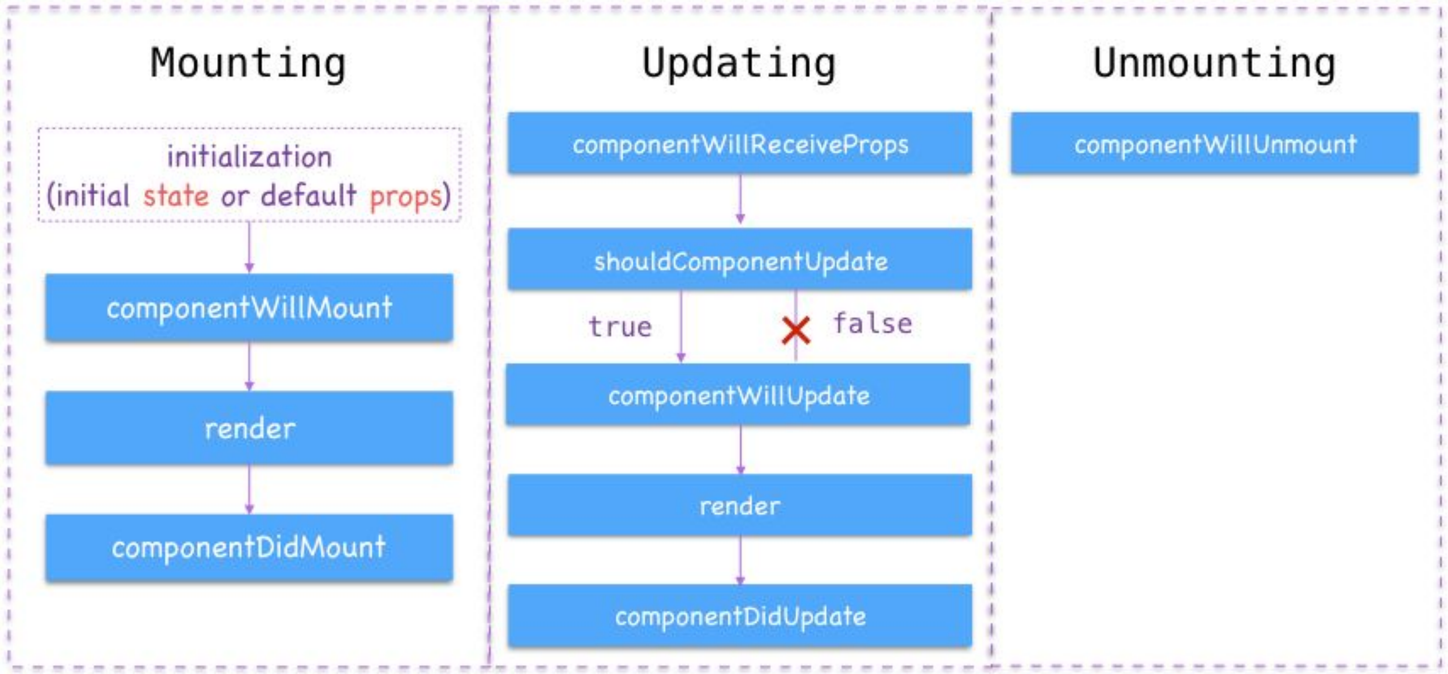
The V in M**V**C

Components, not templates.

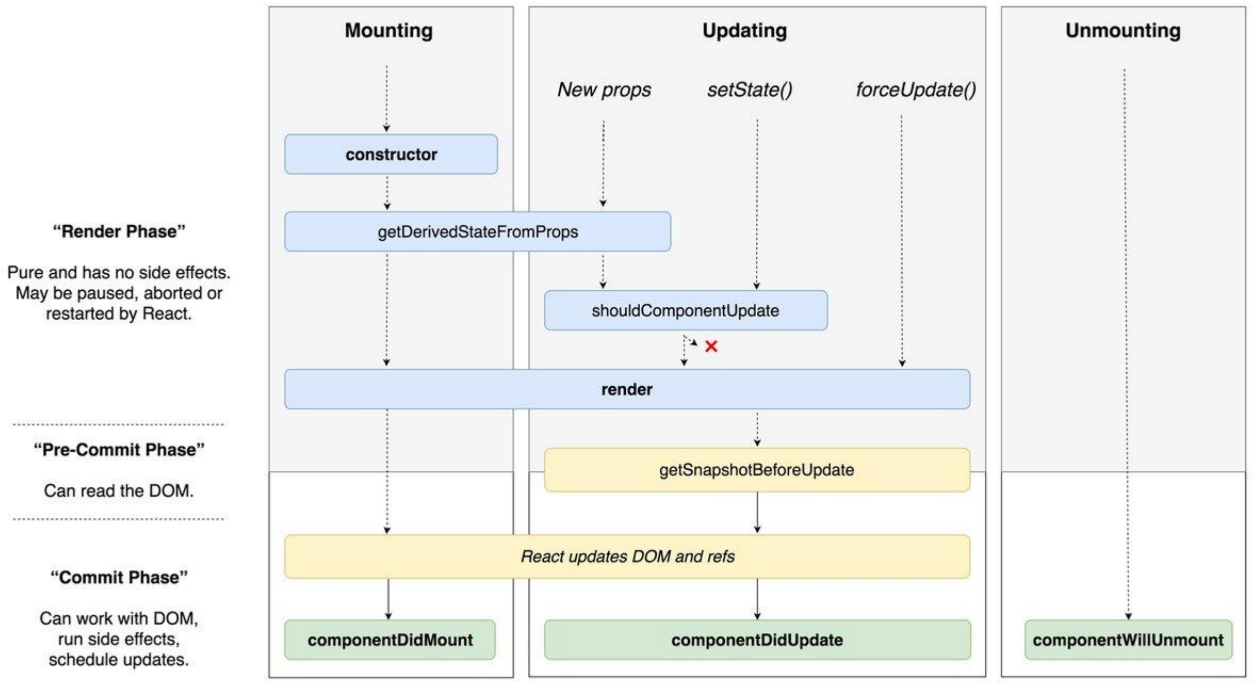
Re-render, don’t mutate.

Virtual DOM is simple and fast

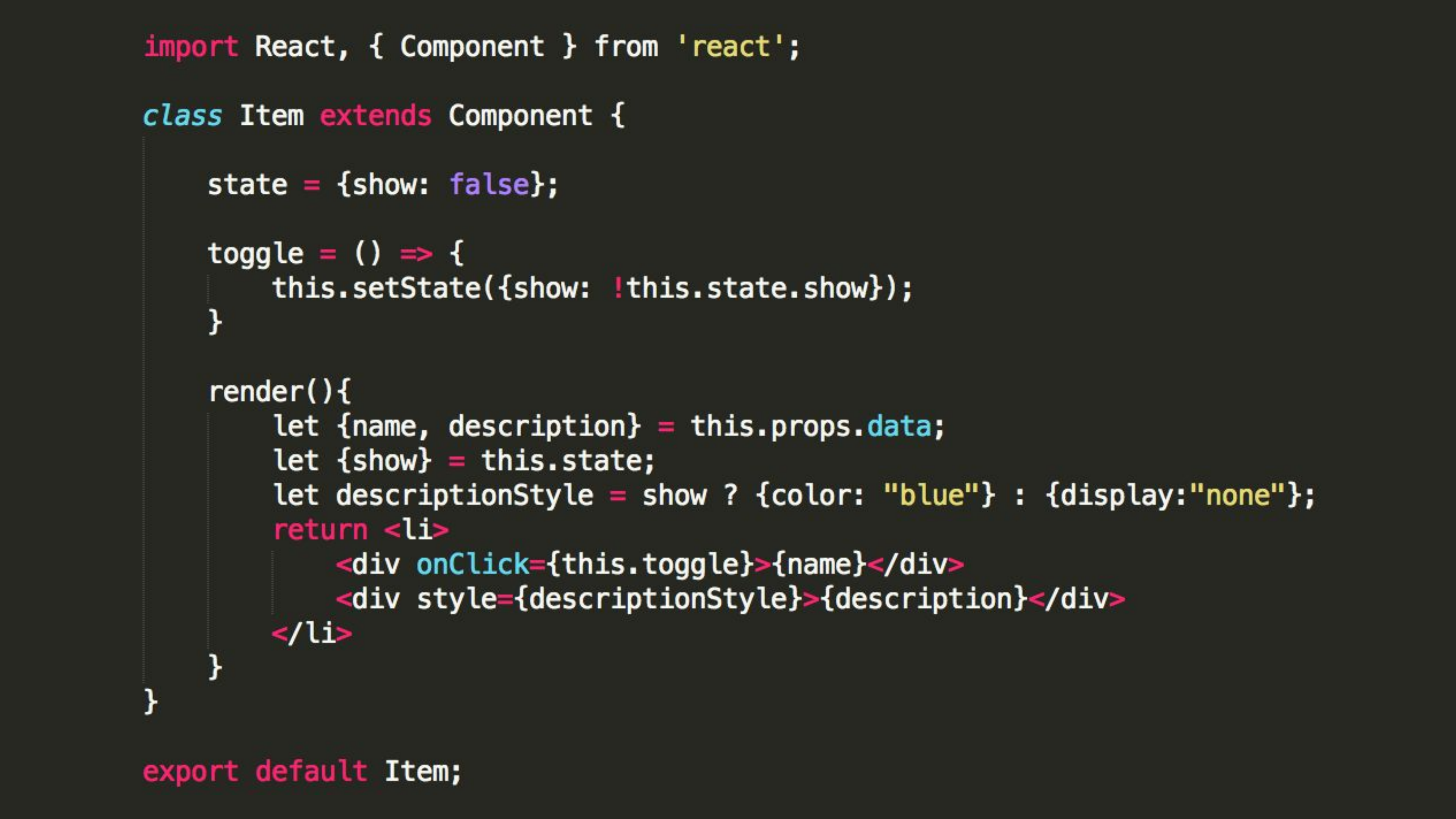
Component lifecycle < 16.3



Component lifecycle > 16.3



```
setState({a:1})
```



```
import React, { Component } from 'react';

class Item extends Component {

    state = {show: false};

    toggle = () => {
        this.setState({show: !this.state.show});
    }

    render(){
        let {name, description} = this.props.data;
        let {show} = this.state;
        let descriptionStyle = show ? {color: "blue"} : {display:"none"};
        return <li>
            <div onClick={this.toggle}>{name}</div>
            <div style={descriptionStyle}>{description}</div>
        </li>
    }
}

export default Item;
```

Kotlin React Wrapper

https://github.com/JetBrains/kotlin-wrappers/blob/master/kotlin-react/src/main/kotlin/react/React.kt

Kotlin (experimental) Multiplatform

https://github.com/gbaldeck/react-js-jvm-kotlin-multiplatform

Demo!!!

https://github.com/jesty/kotlin-loves-react

Greetings

```
actual class Greetings {
    actual fun sayGreetings(): String {
        return "Hello from JVM world!!!"
    }
}
```

Main

```
import app.*
import kotlinext.js.*
import react.dom.*
import kotlin.browser.*

fun main(args: Array<String>) {
   requireAll(require.context("src", true, js("/\\.css$/")))

   render(document.getElementById("root")) {
       app()
   }
}
```

Components

```
class Welcome :
RComponent<RProps,
RState>() {
```

State & Props

```
interface WelcomeProps : RProps {
   var name: String
}

interface WelcomeState : RState {
   var color: String
}
```

Components with state & props

```
class Welcome : RComponent<WelcomeProps, WelcomeState>() {
```

Render

Weird! This pass this

```
override fun RBuilder.render() {
   div {
       +"Hello, ${props.name} - ${state.color}"
       button {
           +"Click me ${props.name}"
           attrs {
               onClickFunction = {
                   setState { color = "red" }
               } } } }
    }
```

https://kotlinlang.org/docs/reference/type-safe-builders.html

Style

```
import kotlinext.js.js
//...
   div {
          attr.title = “Hello”
          //Not typesafe, js body function must be a constant
          attr.style= js { color: “red” }
          //no constant issue
          jsStyle { color = “red” }
      }
//...
```

Routing

```
hashRouter {
   switch {
      route("/", IndexComponent::class, exact = true)
      route("/about", AboutComponent::class, exact = true)
   }
}
//Link
routeLink("/about") { +"About" }
```

setState {a =1}

JS interoperation

```
fun dynamicExample() {
    val a: dynamic = js("{ foo: function () { console.log(Hi!')} }")
    a.foo()
    a.bar() //Uncaught TypeError: a.bar is not a function
}
```

```
fun externalExample() {
   val e = E()
   e.foo()
   //e.bar() //Compile time error!!!
}

external class E {
   fun foo()
}

//helloworld.js
var E = function(){
   this.foo = function () {
      console.log('Hello world! (external)')
   } }
```

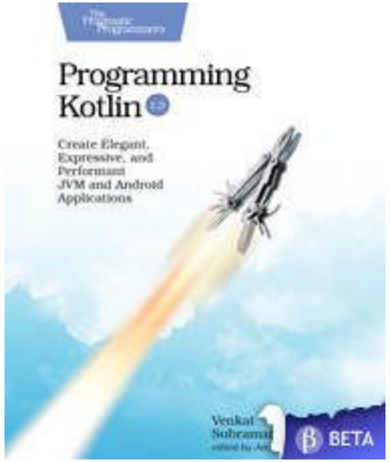
```kotlin
fun actualExpectExample() {
   val ea = EA()
   ea.foo()
}

expect class EA {
   fun foo()
}
```

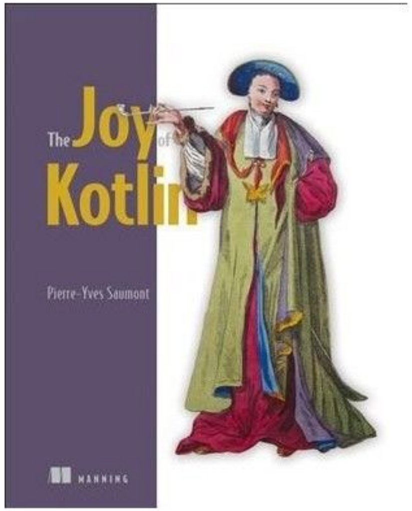
```kotlin
//in a JS project
actual class EA {
   fun foo() { println("Hello from JS!") }
}
```

```kotlin
//in a JVM project
actual class EA {
   fun foo() { println("Hello from JVM!") }
}
```

(Auto)references

- https://github.com/jesty/kotlin-fossavotabona
- https://github.com/jesty/reactiveredis


The Pragmatic Programmers
Programming Kotlin
Create Elegant, Expressive, and Performant JVM and Android Applications
Venkat Subramaniam
BETA

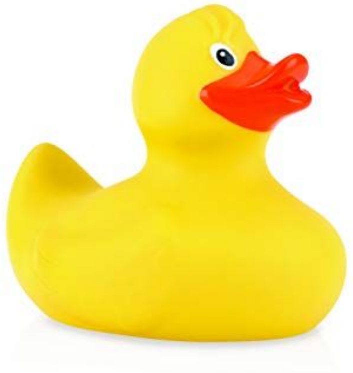


At the end…

Dynamic Vs. Static typed

Static: Types checked before run-time

Dynamic: Types checked on the fly, during execution



Typescript Vs. KotlinJs

TypeScript is a superset of Javascript.

Kotlin aims to be full-stack for creating apps.

https://www.slant.co/versus/378/1543/~typescript_vs_kotlin

https://discuss.kotlinlang.org/t/feedback-on-our-migration-from-typescript-to-kotlin/2578

Javascript vs. KotlinJS

Using KoltinJS doesn’t avoid you to learn Javascript!

Generation size

219K main.9e9685a3.js

+ **261K** vendors.4841eeb9.js

= **480K**

The end!

https://twitter.com/davide_cerbo

let's talk?

let’s talk?

http://devday.it