Kotlin 1.4 Online Event

Kotlin/JS in 1.4 and beyond

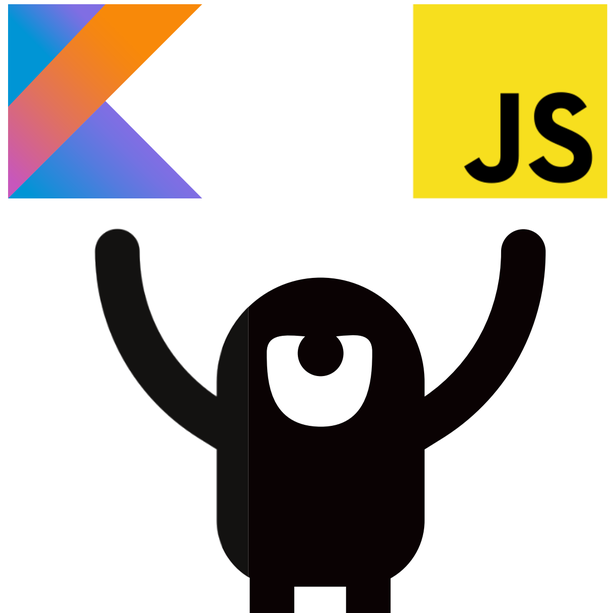
Sebastian Aigner

@sebi_io

October 14, 2020

An overview of Kotlin/JS in 1.4

Kotlin/JS in 1.4 makes the development experience more unified and cohesive, and adds more control for configurations and integrations directly from Gradle.



A clear start

```
plugins {
    kotlin("js") version ”1.4.0"
}
plugins {
    kotlin("multiplatform") version ”1.4.0"
}
```

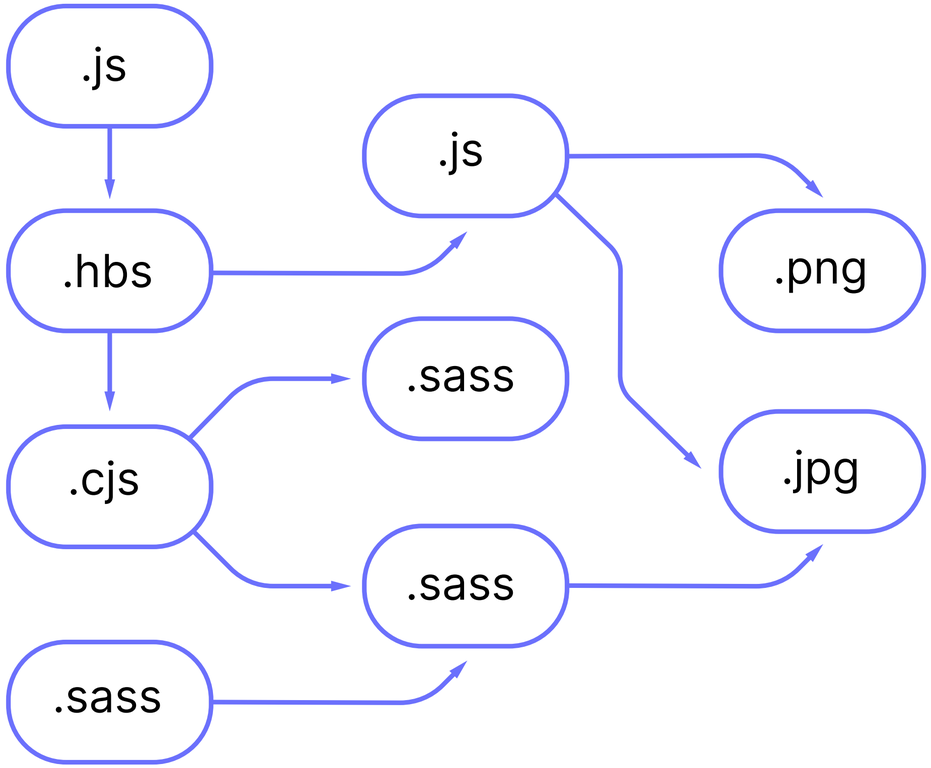
`kotlin2js` and `kotlin-dce-js` are deprecated with Kotlin 1.4.0.

Compatibility between JS and Multiplatform plugins

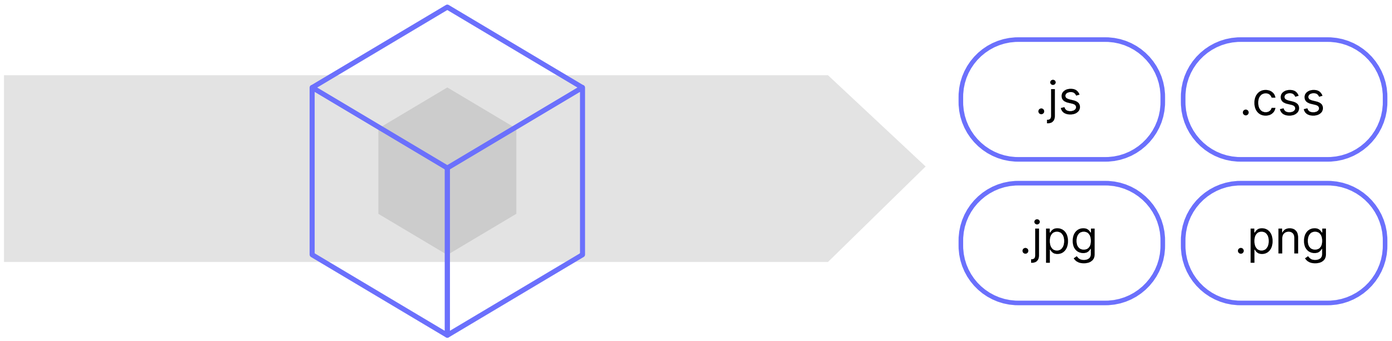
Aligned naming conventions between Gradle plugins

```
kotlin {
    js {
        browser()
        binaries.executable()
    }
}
```

Style support out of the box



Modules with dependencies



Static assets

source: webpack.js.org

Style support out of the box

CSS & style-loader support through Gradle with fine grained configuration.

```
webpackTask {
    cssSupport.enabled = true
}

runTask {
    cssSupport.enabled = true
}

testTask {
    useKarma {
        // . . .
        webpackConfig.cssSupport.enabled = true
    }
}
```

Managing npm dependencies

```
implementation(npm("camelcase", "6.0.0"))
```

```
devNpm
optionalNpm
peerNpm
```

```
{
  "main": "kotlin/myProject.js",
  "devDependencies": {

    . . .
  },
  "dependencies": {
    "camelcase": "6.0.0",
    "kotlin": ”...",
    "kotlin-test": ”..."
  },
  "peerDependencies": {

    . . .
  },
  "optionalDependencies": {

    . . .
  },

  . . .
  "name": ”myProject",
  "version": "1.0.0-SNAPSHOT"
}
```

Generating external declarations

Dukat (experimental) auto-generates Kotlin external declarations, reduces work for using JS dependencies from Kotlin with type-safety.

1) Auto-generate at build time

```
implementation(npm(
        "decamelize", "4.0.0",
        generateExternals = true
))
```

Enable external declaration for individual npm dependendency

```
kotlin.js.generate.externals=true
```

Set default behavior for all npm dependencies (overwritten by individual settings)

Generating external declarations

Dukat (experimental) auto-generates Kotlin external declarations, reduces work for using JS dependencies from Kotlin with type-safety.

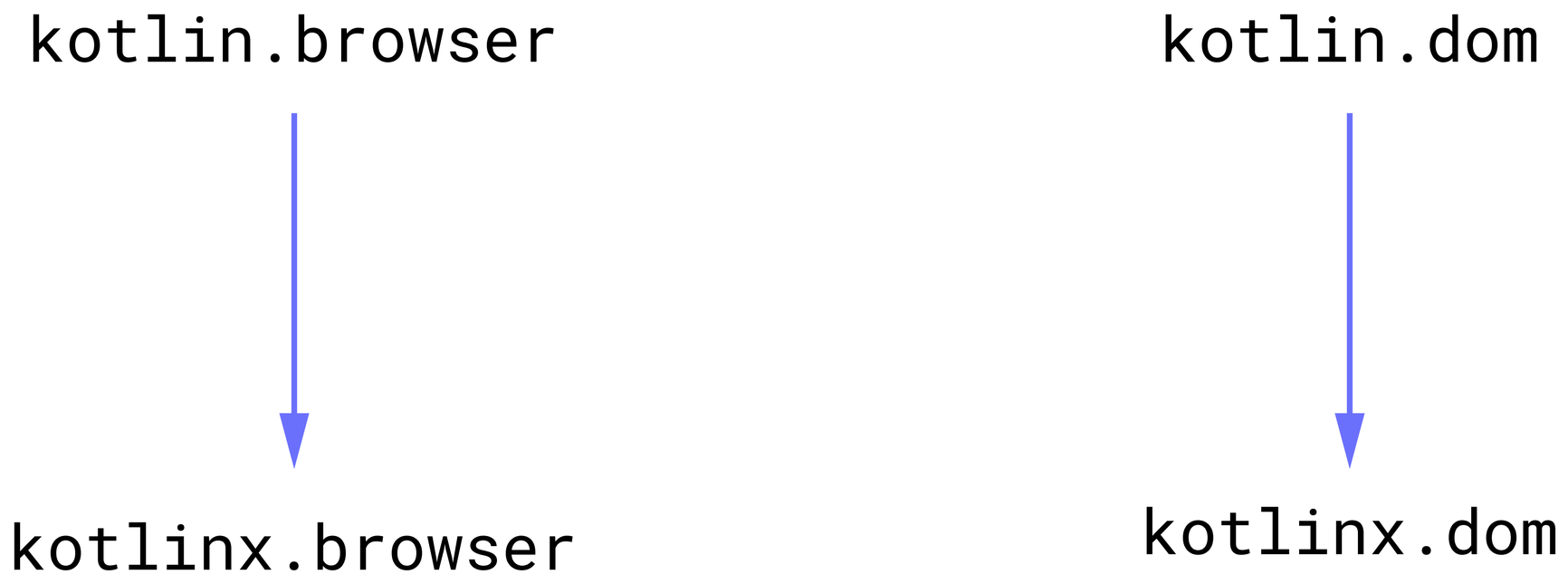
2) Generate & adjust declarations manually

`generateExternals` Gradle task

Creates external declarations (in `./externals/`)

Decoupling the browser APIs

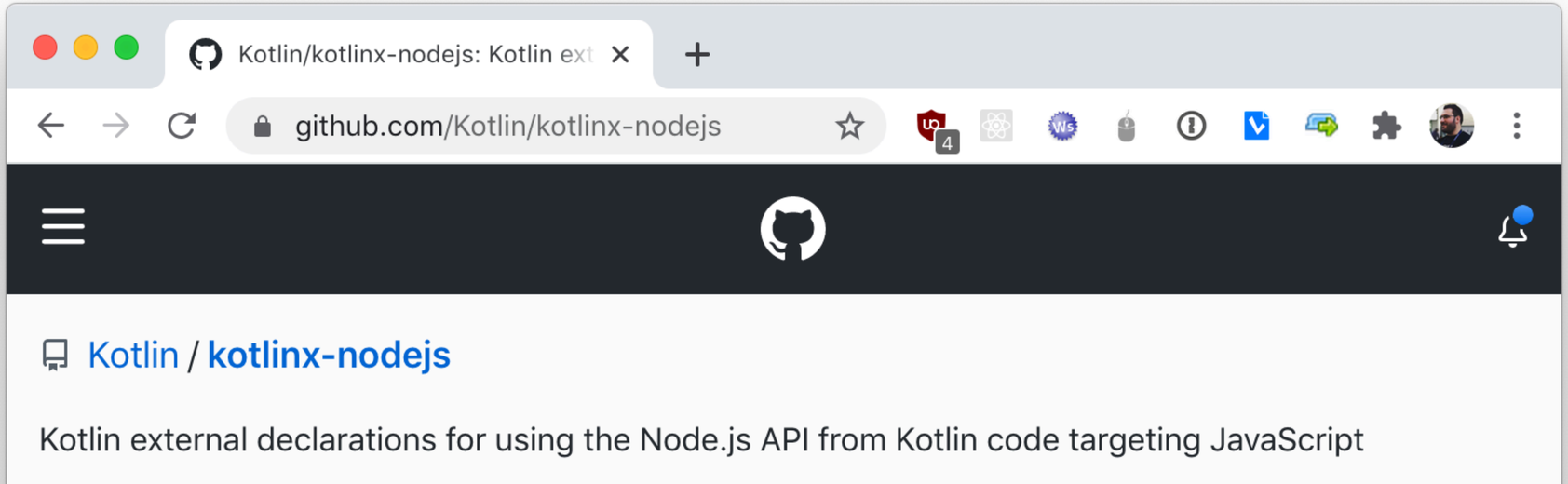
Gradual shift for browser and DOM related packages towards separate artifacts.



Using the Node.js API

`kotlinx-nodejs` (experimental) provides type-safe access to Node.js APIs from Kotlin/JS code

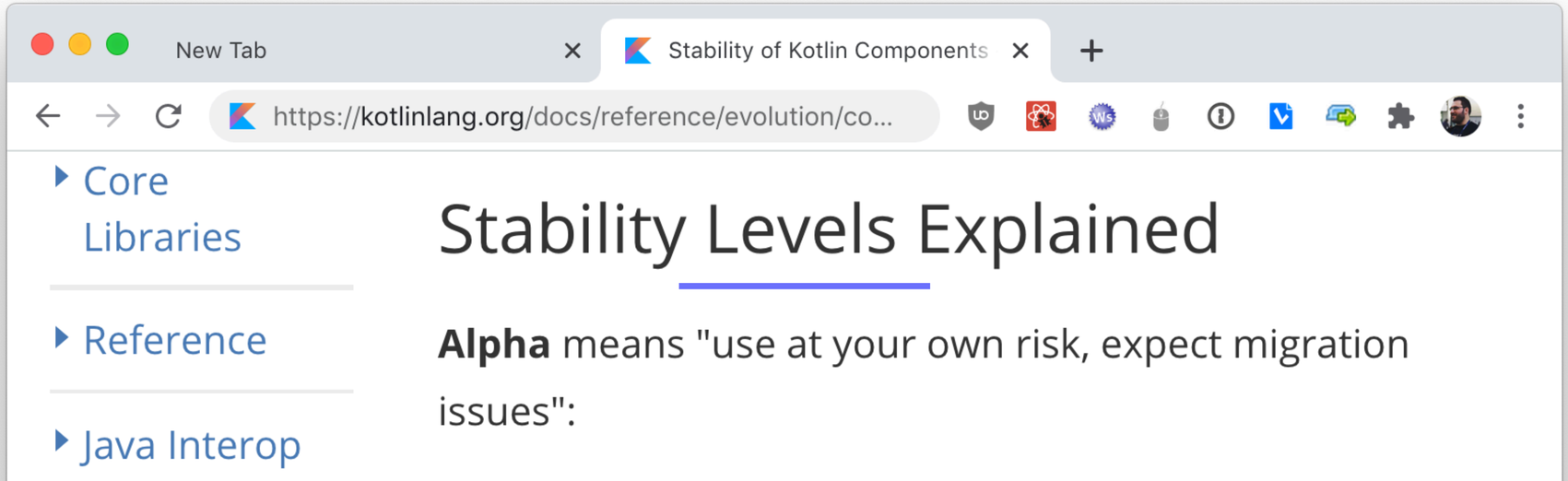
```
fun main() {
    dns.lookup("example.org") { err, address, family ->
        console.log("address: $address, family IPv$family")
    }
}
```



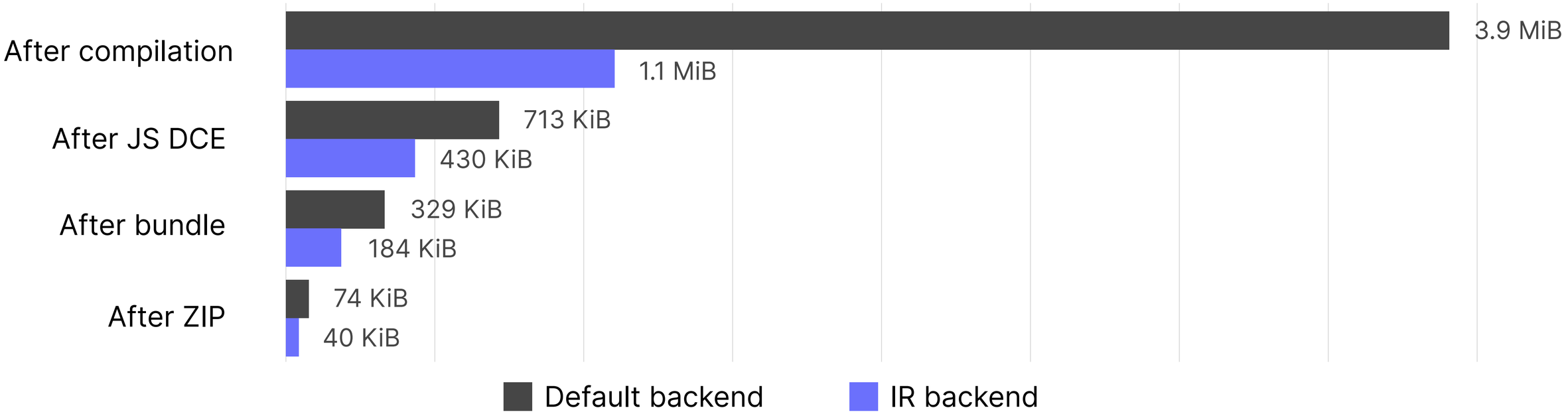
New Alpha IR Compiler for Kotlin/JS

The next evolutionary step,
available to try right now.

The Alpha IR Compiler is the main focus of innovation for Kotlin/JS. Its goals are to improve speed, bundle size, and interoperability with the JavaScript and TypeScript ecosystems.



Addressing bundle size

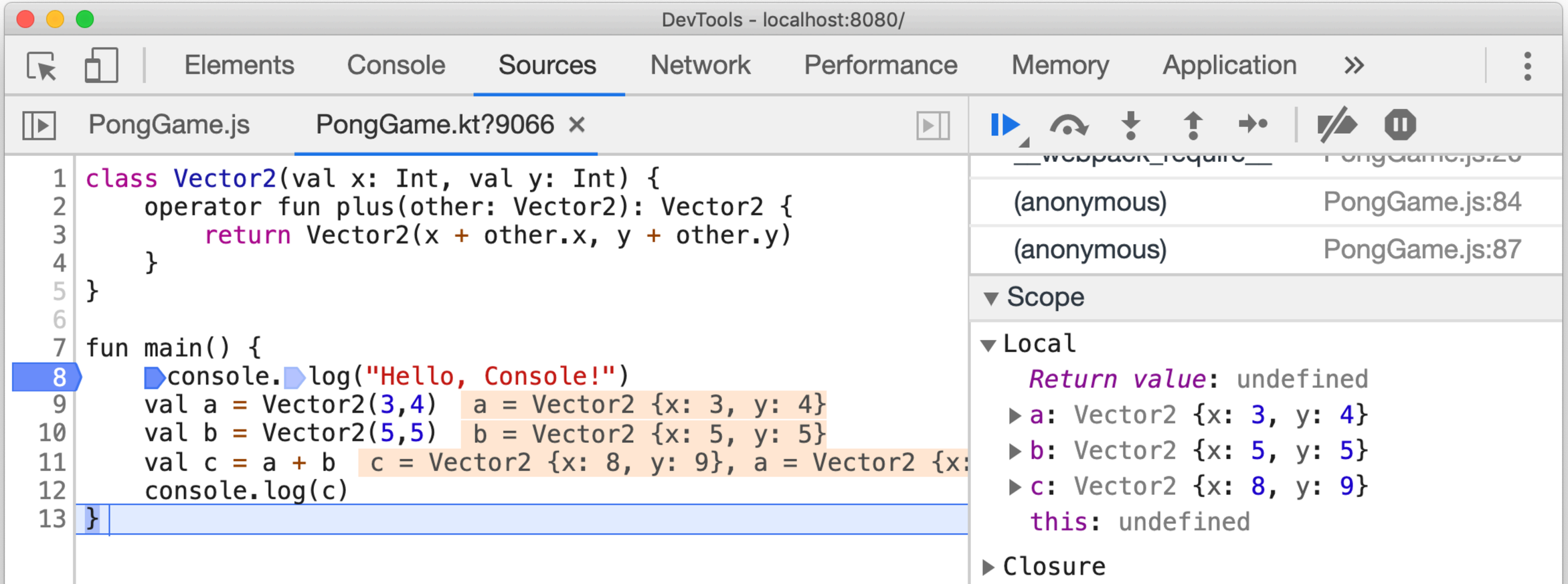


- Strengthening Dead Code Elimination
- Per-Gradle module code splitting
  (configurable from the DSL, coming soon™)

Planned support for ES6 (modules & co.)

- Export your Kotlin code as ES6 modules
- More optimizations regarding bundle size unlocked (webpack tree shaking)
- Gradual addition of more ES6 features (e.g. classes) to the compilation output

Planned feature parity with source maps



Strong interop

- Ability to use Kotlin/JS alongside other web technologies
- No need to “Rewrite It In Kotlin”
- Less constraints for choice of framework, tooling

Introducing @JsExport

```
@JsExport
class KotlinGreeter(val greeting: String) {
    fun greet(name: String): String {
        return "$greeting, $name!"
    }
}

@JsExport
fun wave(person: String): String {
    return "$person waves"
}
```

Preview: TypeScript definitions

```
type Nullable<T> = T | null | undefined
export class KotlinGreeter {
    constructor(greeting: string);
    readonly greeting: string;
    greet(name: string): string;
}
export function wave(person: string): string;
export as namespace library;
```

Using Kotlin code from TypeScript

```
1  import { KotlinGreeter, wave } from "kotlin-library"
2
3  let greeter : KotlinGreeter  = new KotlinGreeter( greeting: "Servus")
4  document.body.innerText = greeter.greet( name: "Sebastian")
5  console.log(wa)
```

```
wave(person: string) (library.d.t…  string
webkitCancelAnimationFrame(handle:  void
```

Smooth transition to the new compiler

build.gradle(.kts)

```
kotlin {
    js(IR) {
        browser {
        }
        binaries.executable()
    }
}
```

gradle.properties

```
kotlin.js.compiler=ir
```

Backported `@JsExport` annotation and `BOTH` mode to ease transition

Smooth transition to the new compiler

build.gradle(.kts)

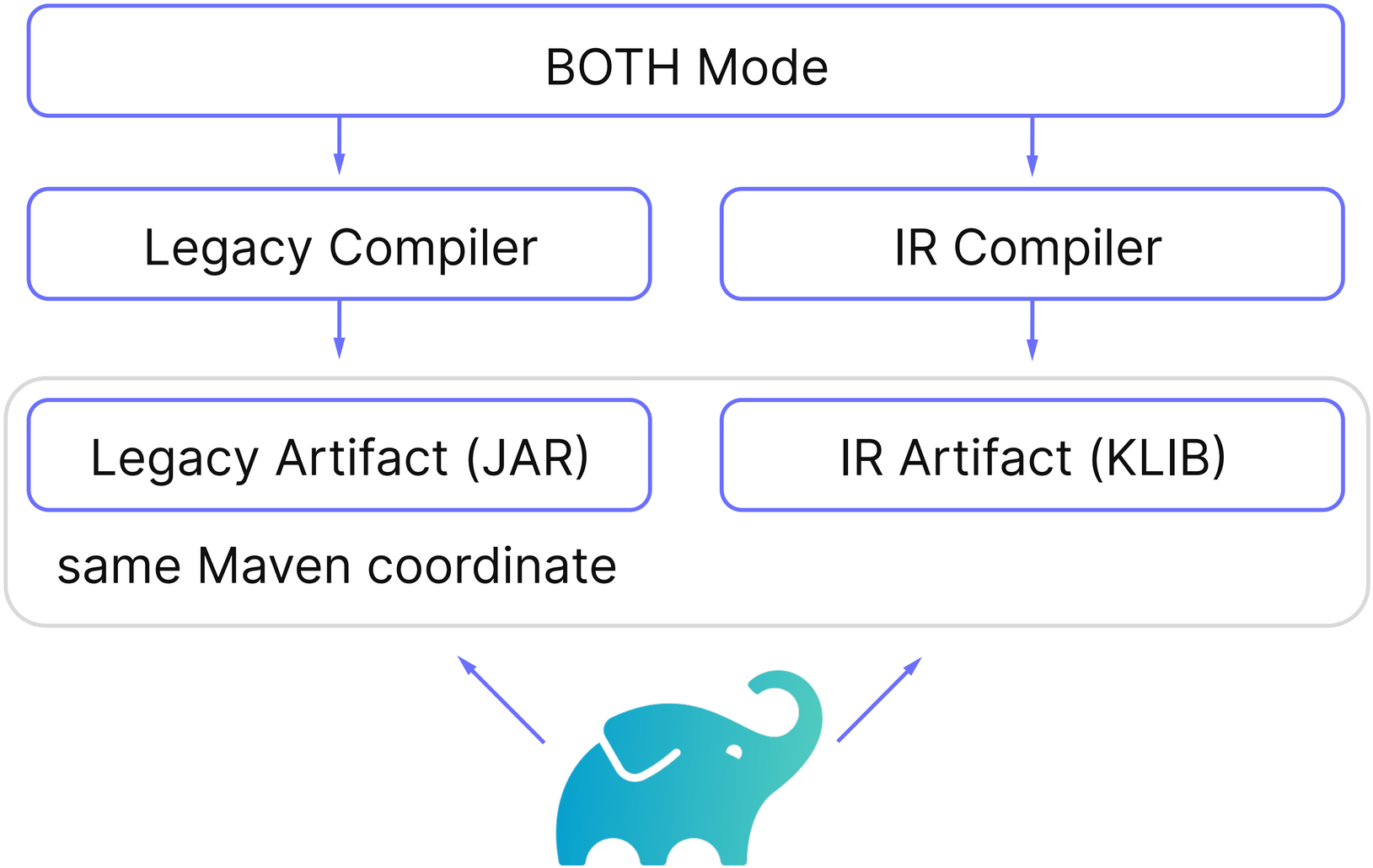
```
kotlin {
    js(IR /* or LEGACY, BOTH */) {
        browser {
        }
        binaries.executable()
    }
}
```

gradle.properties

```
kotlin.js.compiler=ir
/* legacy, both */
```

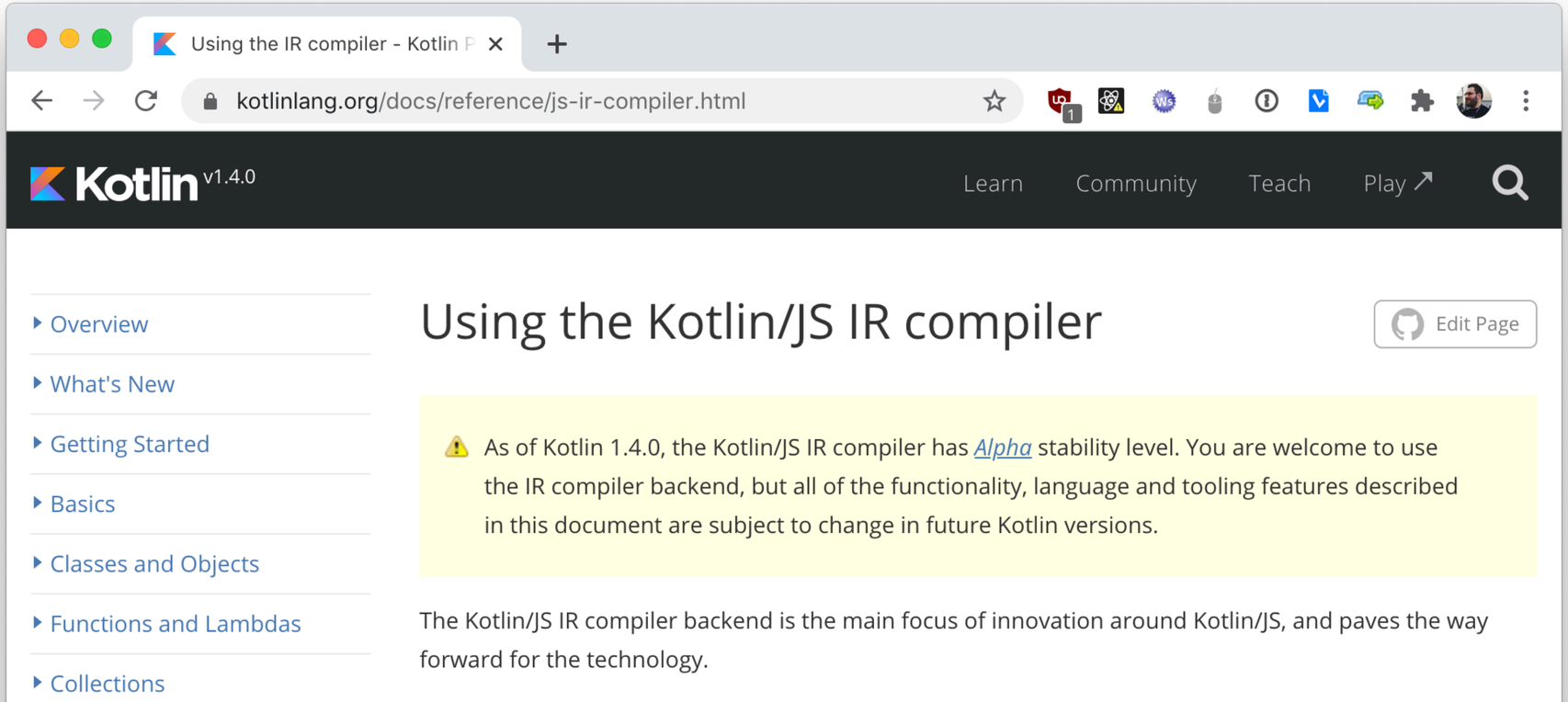
Backported `@JsExport` annotation and `BOTH` mode to ease transition

Both mode and Gradle metadata



Learn more & try the new Kotlin/JS compiler

Kotlin/JS IR compiler documentation available on kotlinlang.org



Kotlin and WebAssembly

A brief update on another Kotlin target

WebAssembly: Making progress!

- Representation in the WebAssembly community working group
- Close interaction with the WebAssembly VM teams
- Work on a first prototype to play around with
- We have high hopes!

Wrapping up

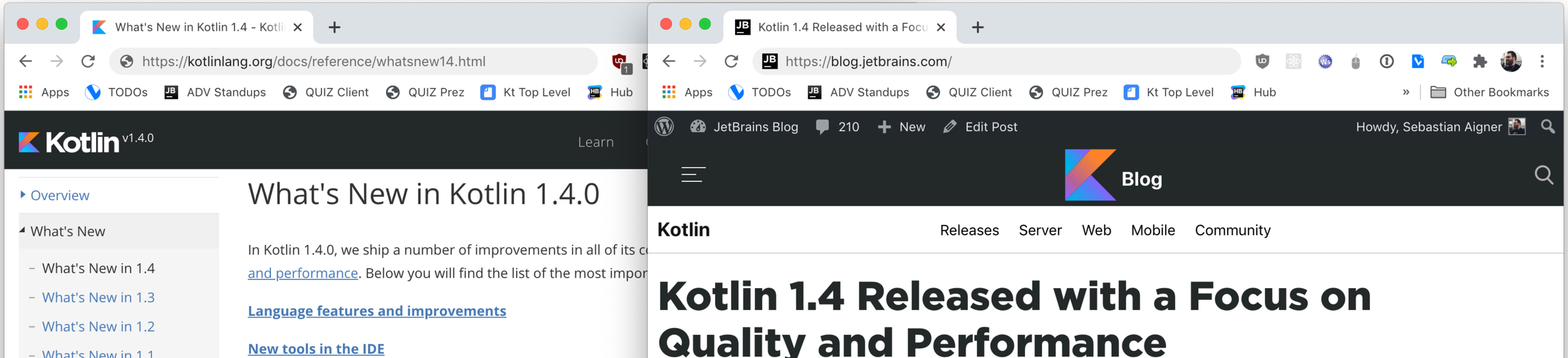
What we’ve seen, and where to go to learn more.

Kotlin/JS in 1.4:

- More unified Gradle DSL
- More control and integrations
- Browser & DOM API evolutions
- Alpha IR Compiler with fancy new features

More about Kotlin/JS

- Check the ”What’s new” section on kotlinlang.org
- Read the release blog post on blog.jetbrains.com/kotlin
- Join the dialogue on kotlinlang.slack.com #javascript



Thanks!
Have a nice Kotlin

@sebi_io