

◀ Kotlin 1.4 Online Event

# Kotlin/JS in 1.4 and beyond

Sebastian Aigner



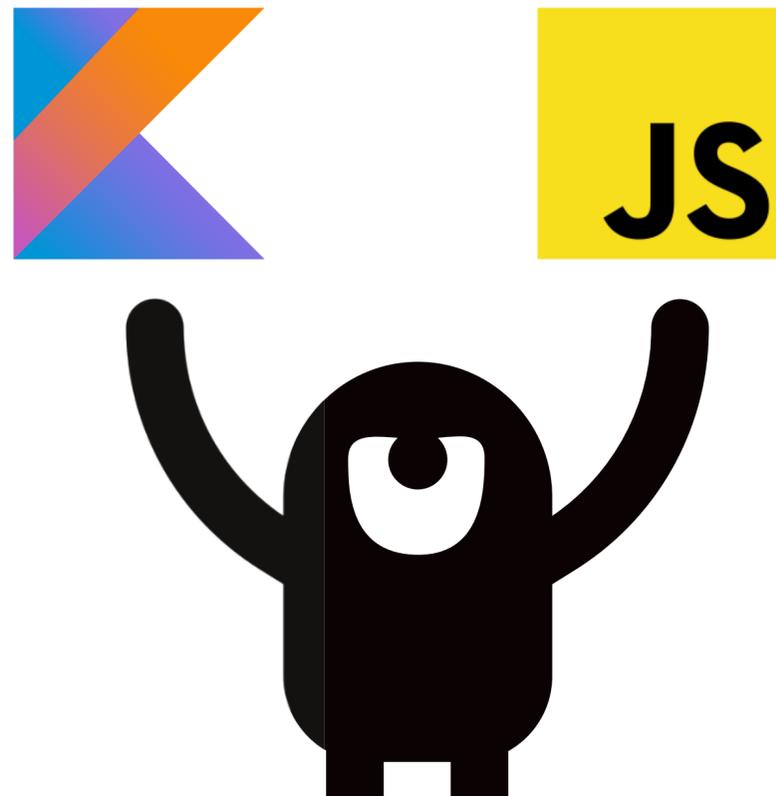
@sebi\_io

October 14, 2020

# An overview of Kotlin/JS in 1.4

The background features a dark grey gradient. On the left side, there are several overlapping, semi-transparent geometric shapes in shades of blue and purple. These shapes include a large triangle pointing downwards and several smaller, overlapping polygons that create a layered, crystalline effect. The overall aesthetic is modern and technical.

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"  
}
```



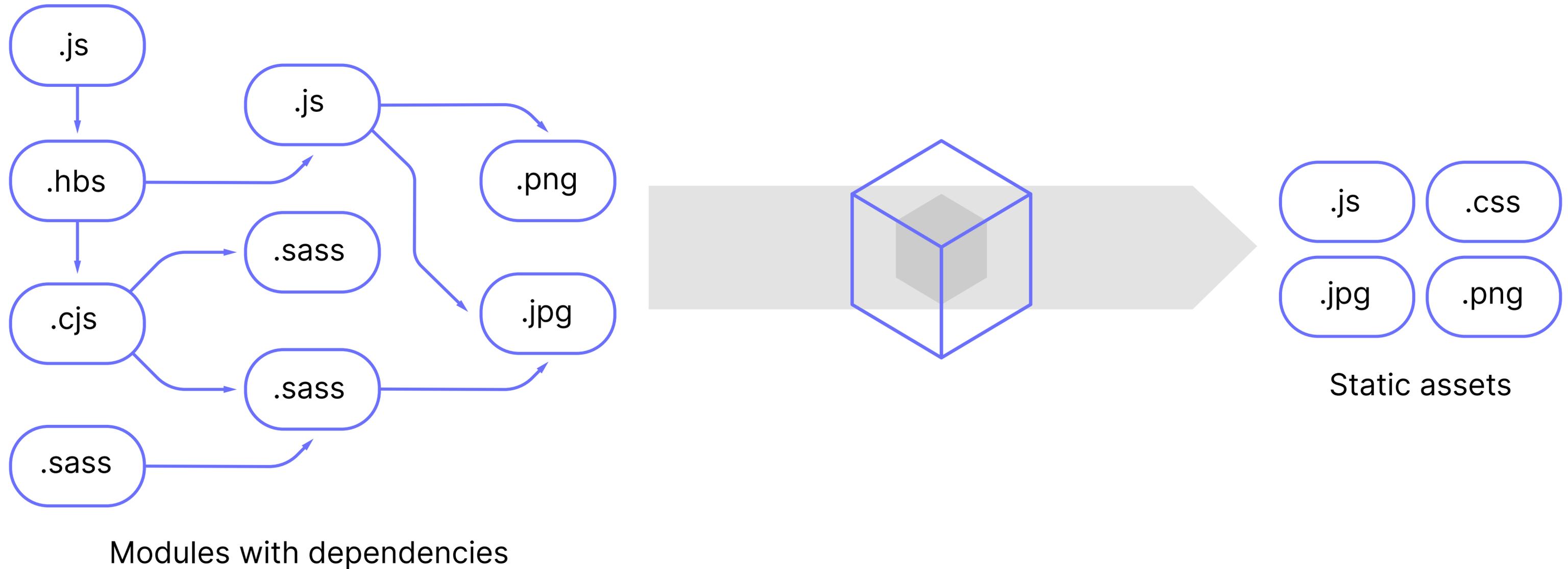
kotlin2js and kotlin-dce-js are deprecated with Kotlin 1.4.0.

# Compatibility between JS and Multiplatform plugins

Aligned naming conventions between Gradle plugins

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

# Style support out of the box



source: [webpack.js.org](http://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  
))
```

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

Enable external declaration for individual npm dependency

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.

`kotlin.browser`



`kotlinx.browser`

`kotlin.dom`

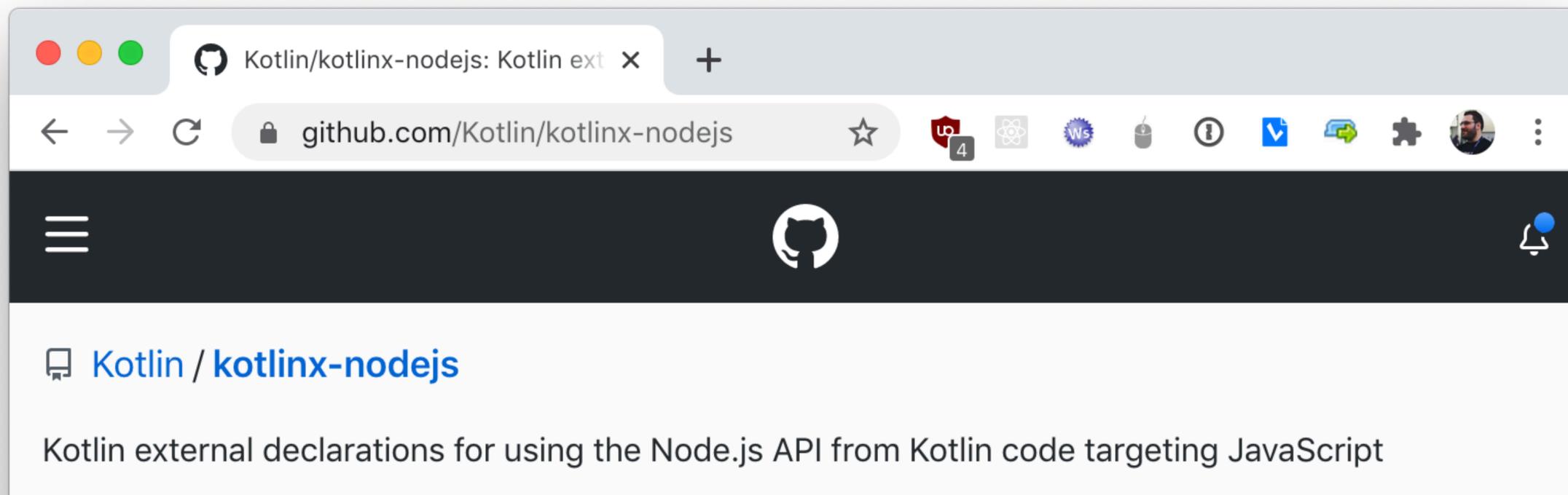


`kotlinx.dom`

# Using the Node.js API

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

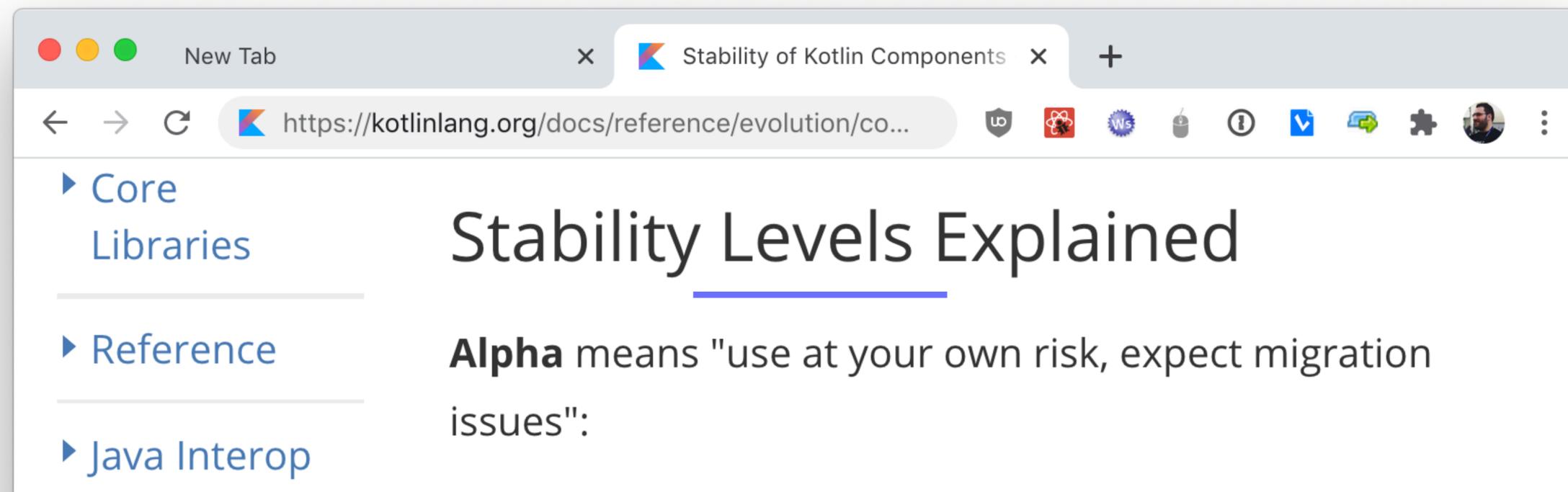
```
fun main() {  
    dns.lookup("example.org") { err, address, family ->  
        console.log("address: $address, family IPv$family")  
    }  
}
```



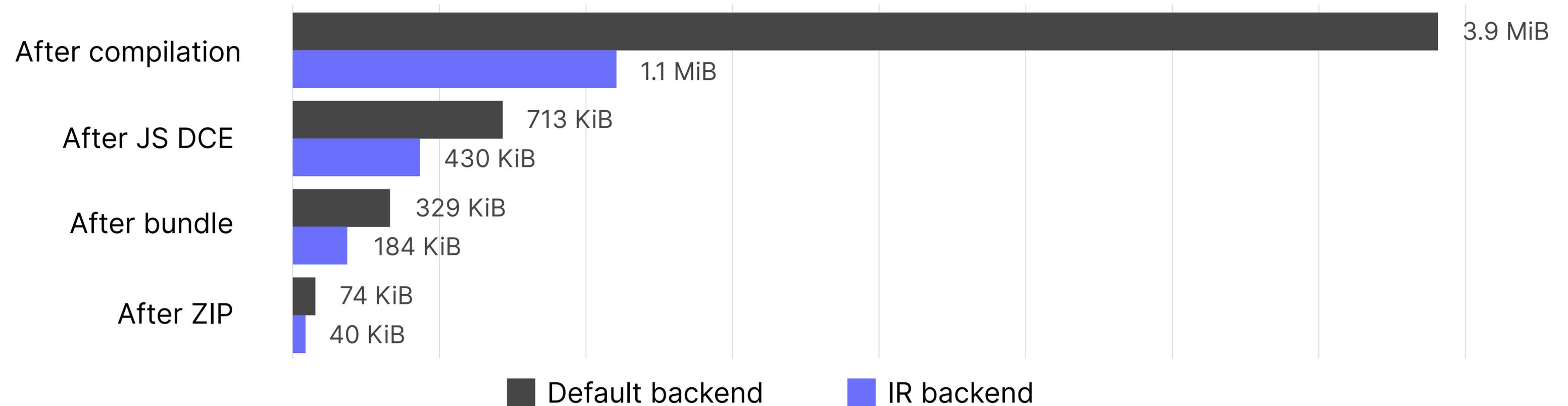
# New Alpha IR Compiler for Kotlin/JS

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

DevTools - localhost:8080/

Elements Console Sources Network Performance Memory Application >>

PongGame.js PongGame.kt?9066 x

```
1 class Vector2(val x: Int, val y: Int) {
2     operator fun plus(other: Vector2): Vector2 {
3         return Vector2(x + other.x, y + other.y)
4     }
5 }
6
7 fun main() {
8     console.log("Hello, Console!")
9     val a = Vector2(3,4)    a = Vector2 {x: 3, y: 4}
10    val b = Vector2(5,5)   b = Vector2 {x: 5, y: 5}
11    val c = a + b         c = Vector2 {x: 8, y: 9}, a = Vector2 {x:
12    console.log(c)
13 }
```

Call stack:

- \_\_webpack\_require\_\_ PongGame.js:20
- (anonymous) PongGame.js:84
- (anonymous) PongGame.js:87

▼ Scope

▼ Local

- Return value*: undefined
- ▶ a: Vector2 {x: 3, y: 4}
- ▶ b: Vector2 {x: 5, y: 5}
- ▶ c: Vector2 {x: 8, y: 9}
- this: undefined

▶ Closure

# 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(wave)
```

• 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)

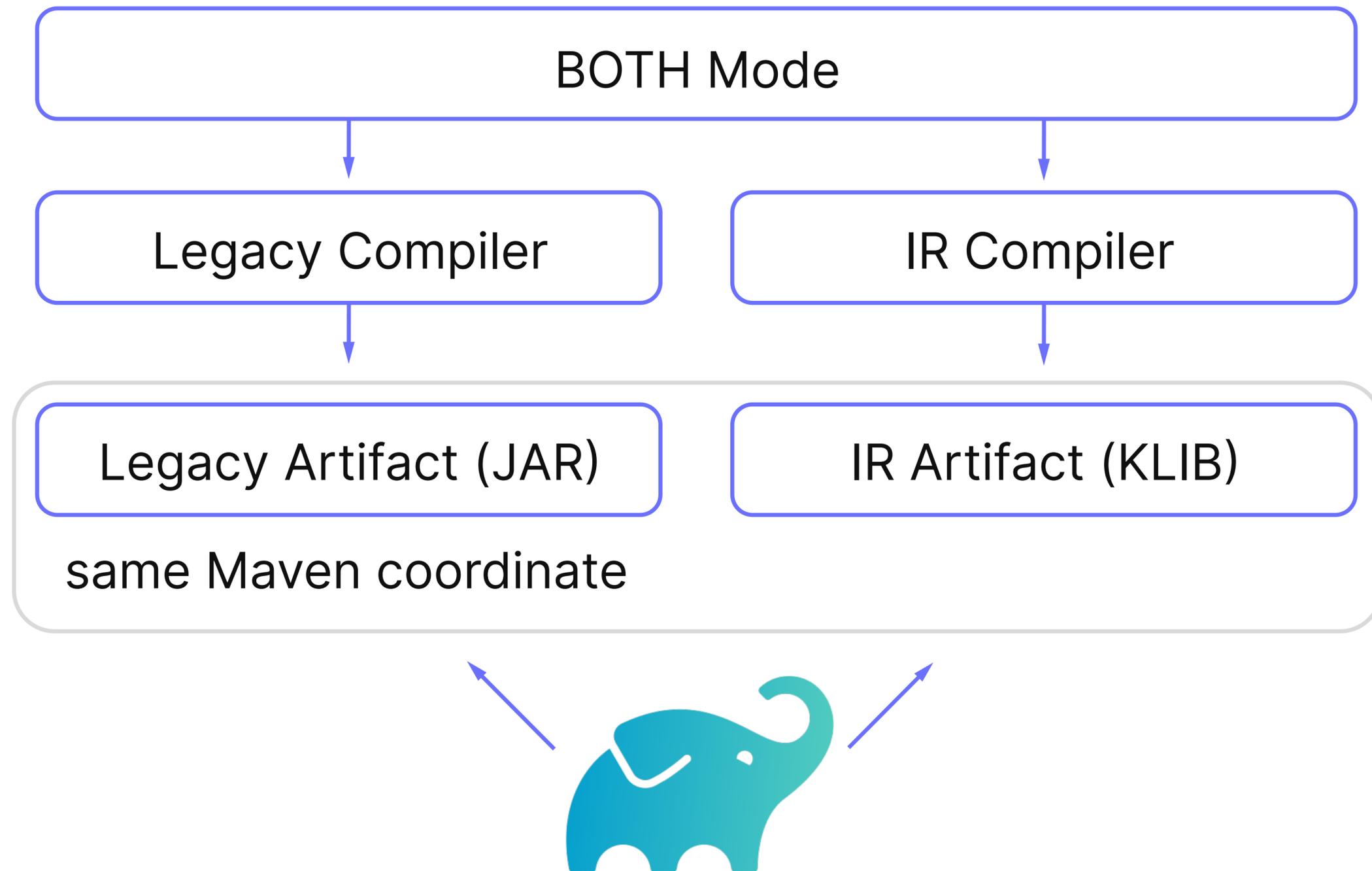
```
kotlin {  
    js(IR /* or LEGACY, BOTH */) {  
        browser {  
        }  
        binaries.executable()  
    }  
}
```

gradle.properties

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

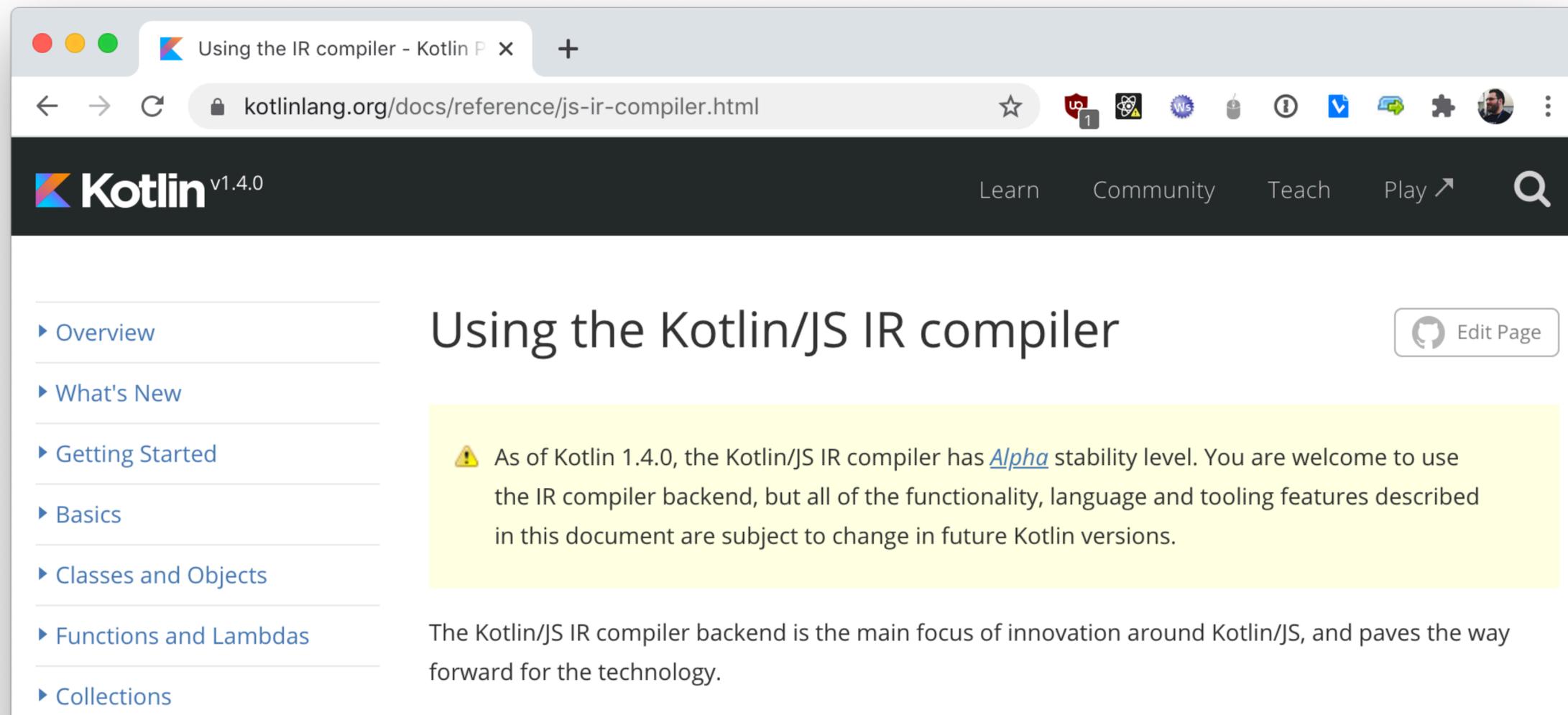
Backported @JsExport annotation and BOTH mode to ease transition

# Both mode and Gradle metadata



# Learn more & try the new Kotlin/JS compiler

Kotlin/JS IR compiler documentation available on [kotlinlang.org](https://kotlinlang.org)



The screenshot shows a web browser window with the URL `kotlinlang.org/docs/reference/js-ir-compiler.html`. The page title is "Using the IR compiler - Kotlin P x". The browser's address bar shows the URL and a star icon. The page header features the Kotlin logo (v1.4.0) and navigation links for "Learn", "Community", "Teach", "Play", and a search icon. The main content area is titled "Using the Kotlin/JS IR compiler" and includes a yellow warning box with a warning icon. The warning text states: "As of Kotlin 1.4.0, the Kotlin/JS IR compiler has *Alpha* stability level. You are welcome to use the IR compiler backend, but all of the functionality, language and tooling features described in this document are subject to change in future Kotlin versions." Below the warning, the text reads: "The Kotlin/JS IR compiler backend is the main focus of innovation around Kotlin/JS, and paves the way forward for the technology." A sidebar on the left contains a list of navigation links: "Overview", "What's New", "Getting Started", "Basics", "Classes and Objects", "Functions and Lambdas", and "Collections". An "Edit Page" button is located in the top right corner of the main content area.

Using the IR compiler - Kotlin P x

kotlinlang.org/docs/reference/js-ir-compiler.html

Kotlin v1.4.0

Learn Community Teach Play

## Using the Kotlin/JS IR compiler

Edit Page

⚠ As of Kotlin 1.4.0, the Kotlin/JS IR compiler has *Alpha* stability level. You are welcome to use the IR compiler backend, but all of the functionality, language and tooling features described in this document are subject to change in future Kotlin versions.

The Kotlin/JS IR compiler backend is the main focus of innovation around Kotlin/JS, and paves the way forward for the technology.

- ▶ Overview
- ▶ What's New
- ▶ Getting Started
- ▶ Basics
- ▶ Classes and Objects
- ▶ Functions and Lambdas
- ▶ Collections

# 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

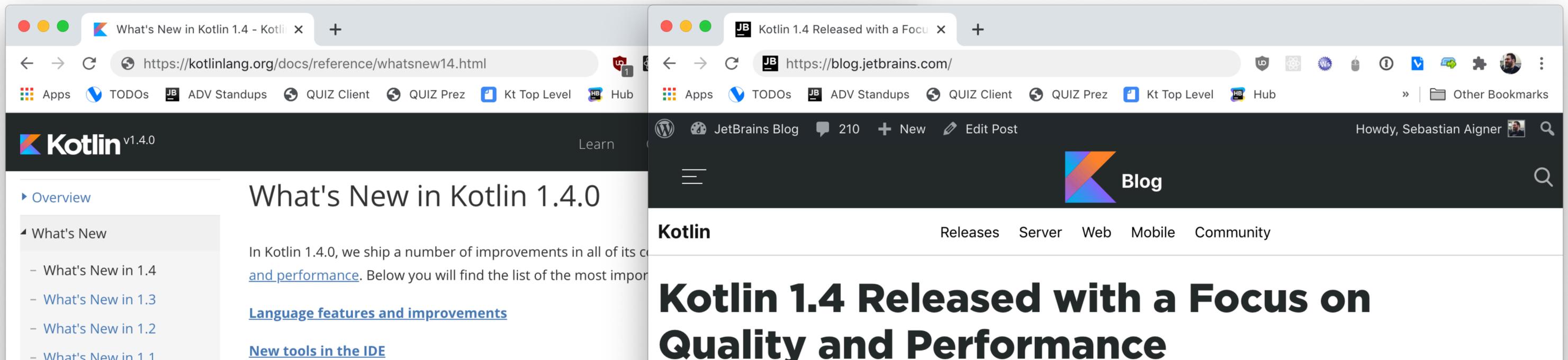
What we've seen, and where to go to learn more.

# 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](https://kotlinlang.org)
- Read the release blog post on [blog.jetbrains.com/kotlin](https://blog.jetbrains.com/kotlin)
- Join the dialogue on [#javascript](https://kotlinlang.slack.com)



Thanks!  
Have a nice Kotlin

