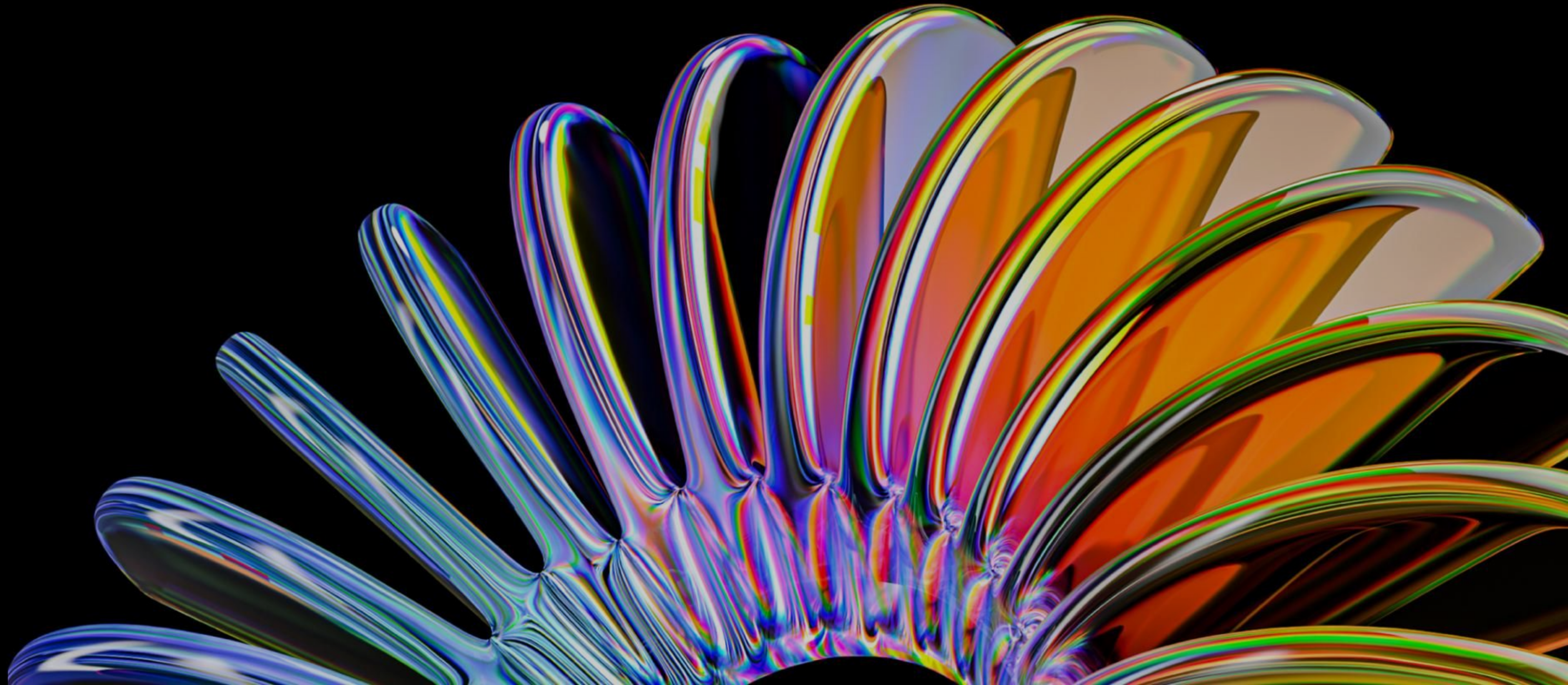
Kotlin in GitHub Actions



Макс Качинкин

Android Tech Lead

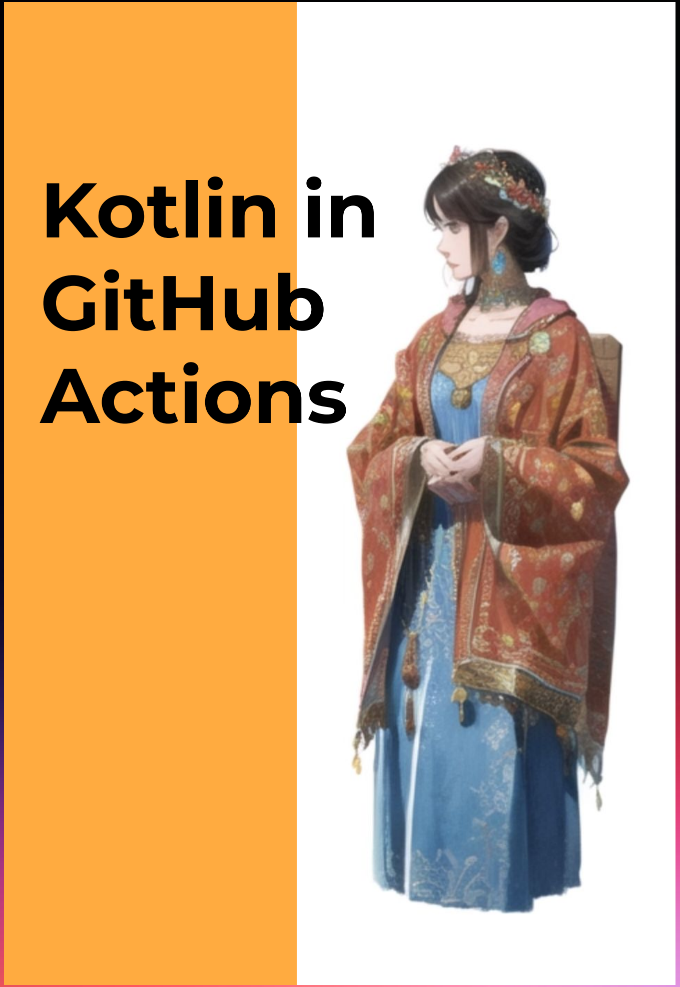
- Разрабатываю под Android 10+ лет
- Dodo Engineering, Android Tech Lead
- Выступаю, пишу статьи, преподаю
- ТГ: “Мобильное Чтиво” @mobilefiction

42k
сотрудников

330
команда ИТ

23
страны

- GitHub Actions 🫶
- Kotlin 🫶
- KMP 🫶
- GHA + Kotlin + KMP = 🫶🫶🫶



Kotlin in GitHub Actions

Для кого доклад?

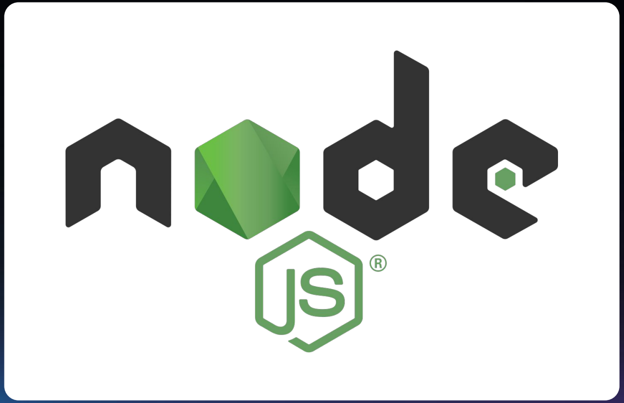
🔥 Для всех, кто любит GitHub Actions

🔥 Для всех, кто любит KMP во всех проявлениях, в том числе в таких необычных как Kotlin/JS.

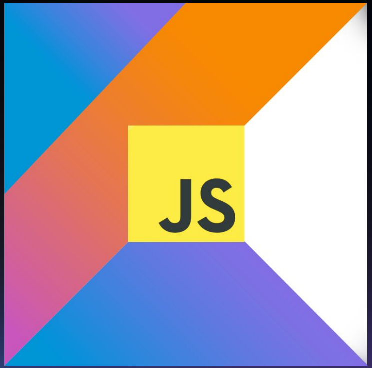
🧠 Для всех остальных будет тоже интересно

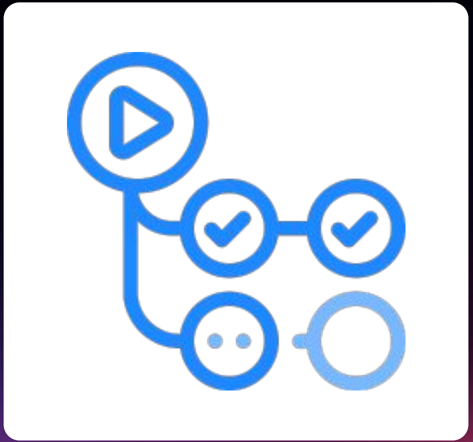
Спойлеры

Node.js, NPM

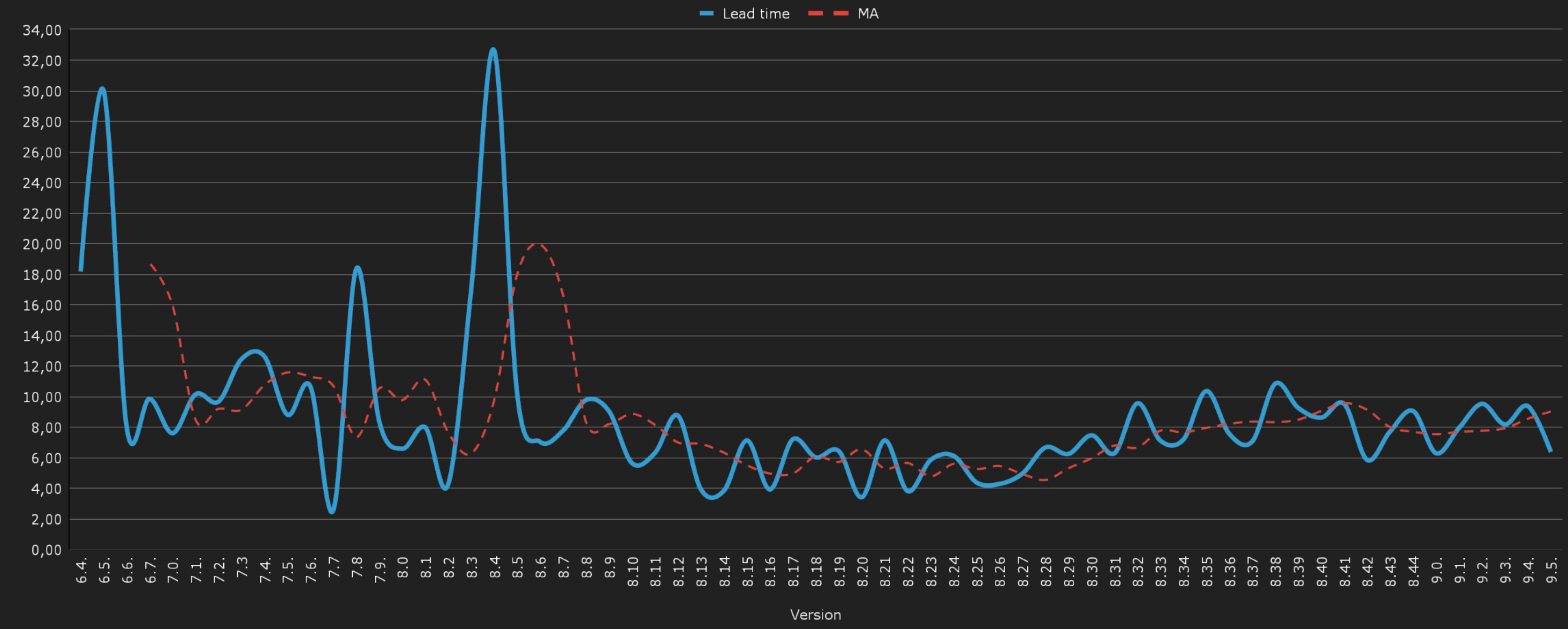


KMP Kotlin/JS, GHA

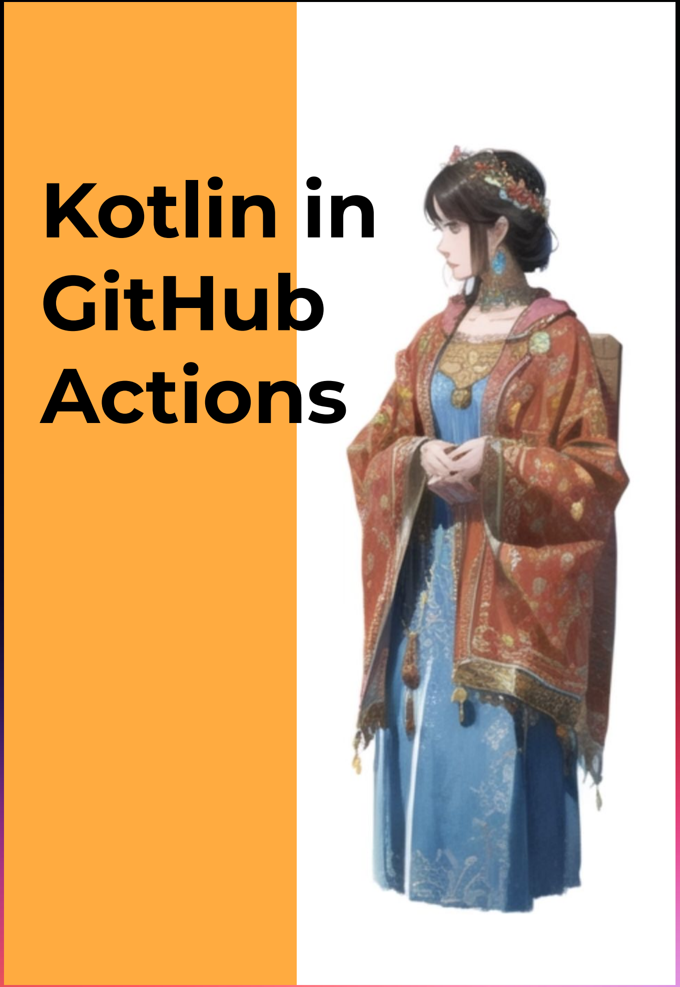




Commit lead time

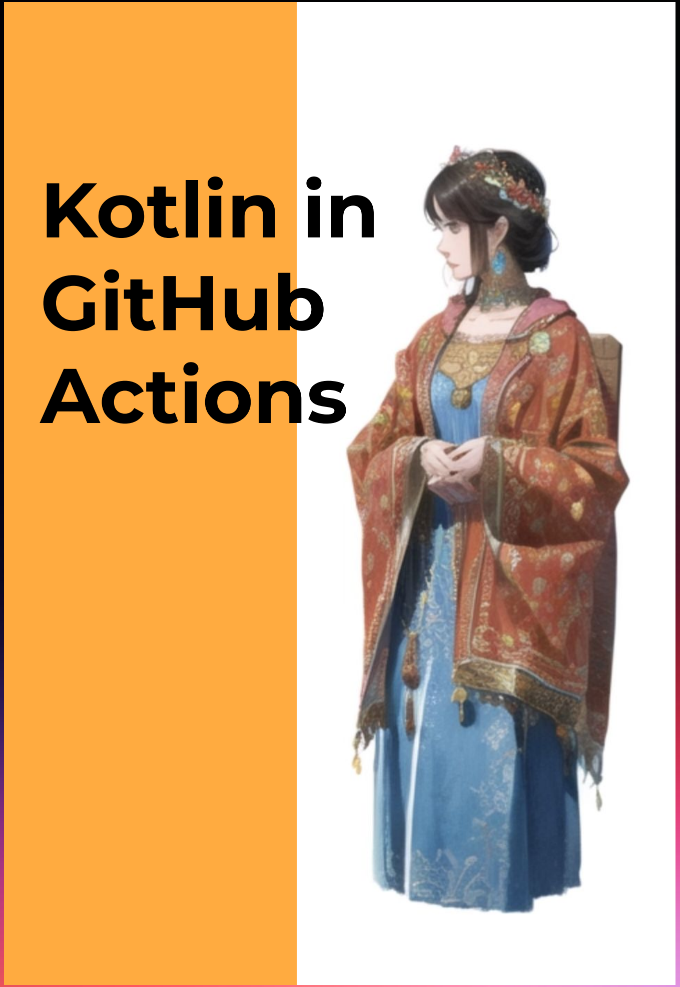


Поехали

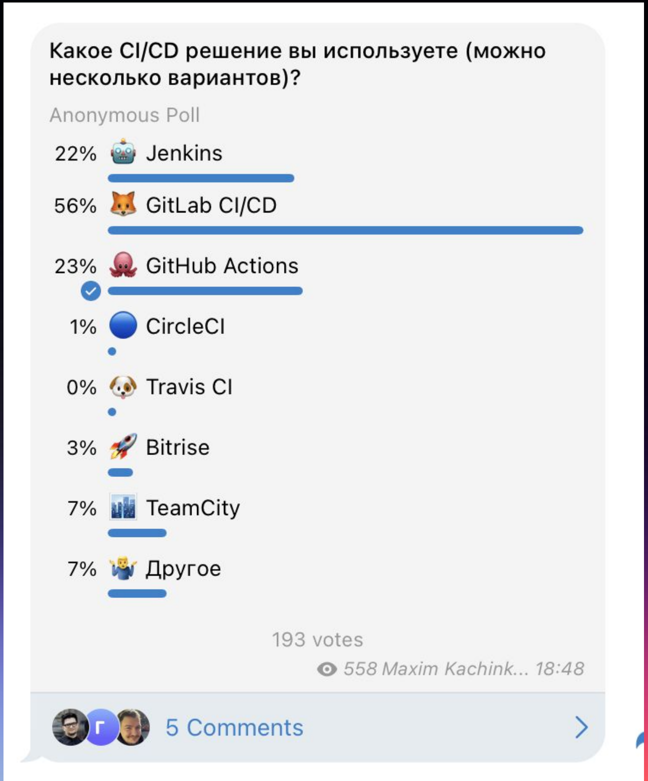


Поехали

Оценили шутку? Сам придумал!



Для тех, кто не пользуется GHA



Какое CI/CD решение вы используете (можно несколько вариантов)?

Anonymous Poll

22%	🤖 Jenkins
56%	🦊 GitLab CI/CD
23% ✓	🐙 GitHub Actions
1%	🔵 CircleCI
0%	🐶 Travis CI
3%	🚀 Bitrise
7%	🏙 TeamCity
7%	🤷 Другое

193 votes

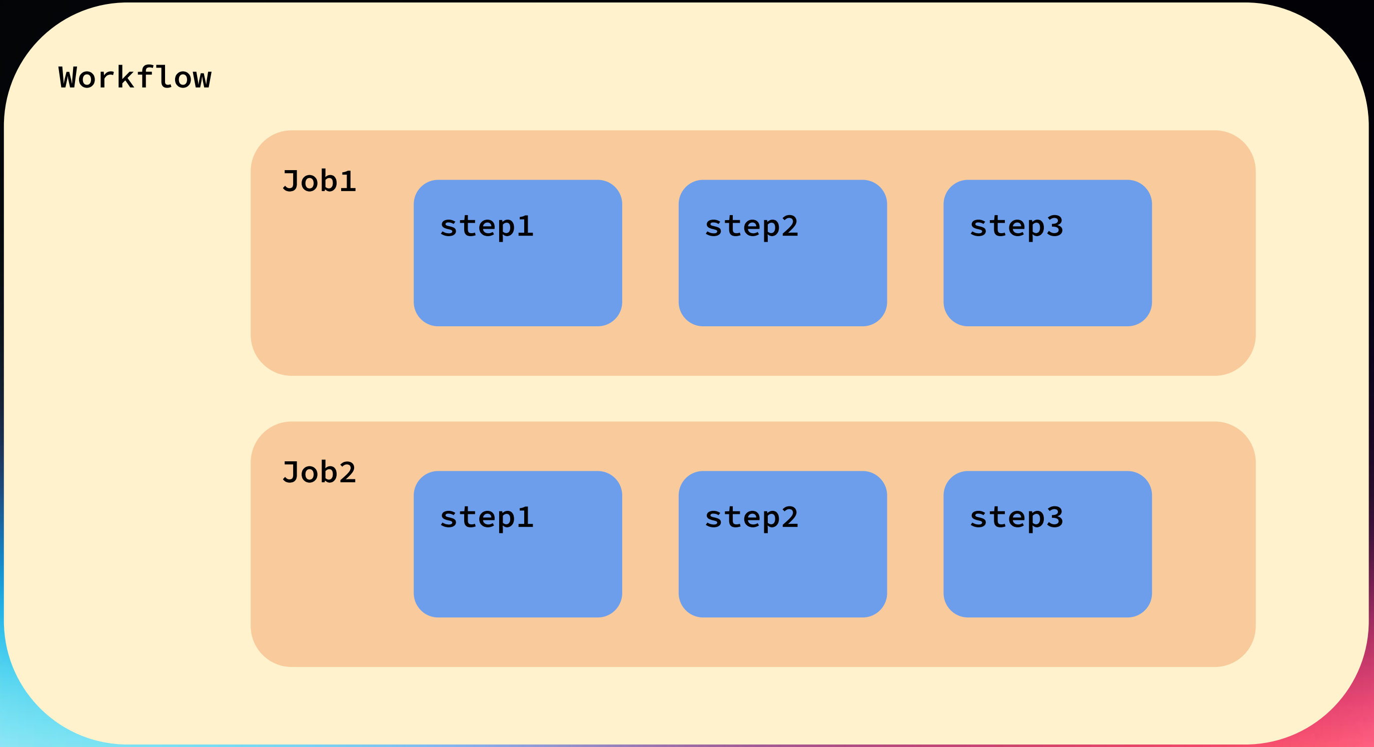
558 Maxim Kachink... 18:48

5 Comments

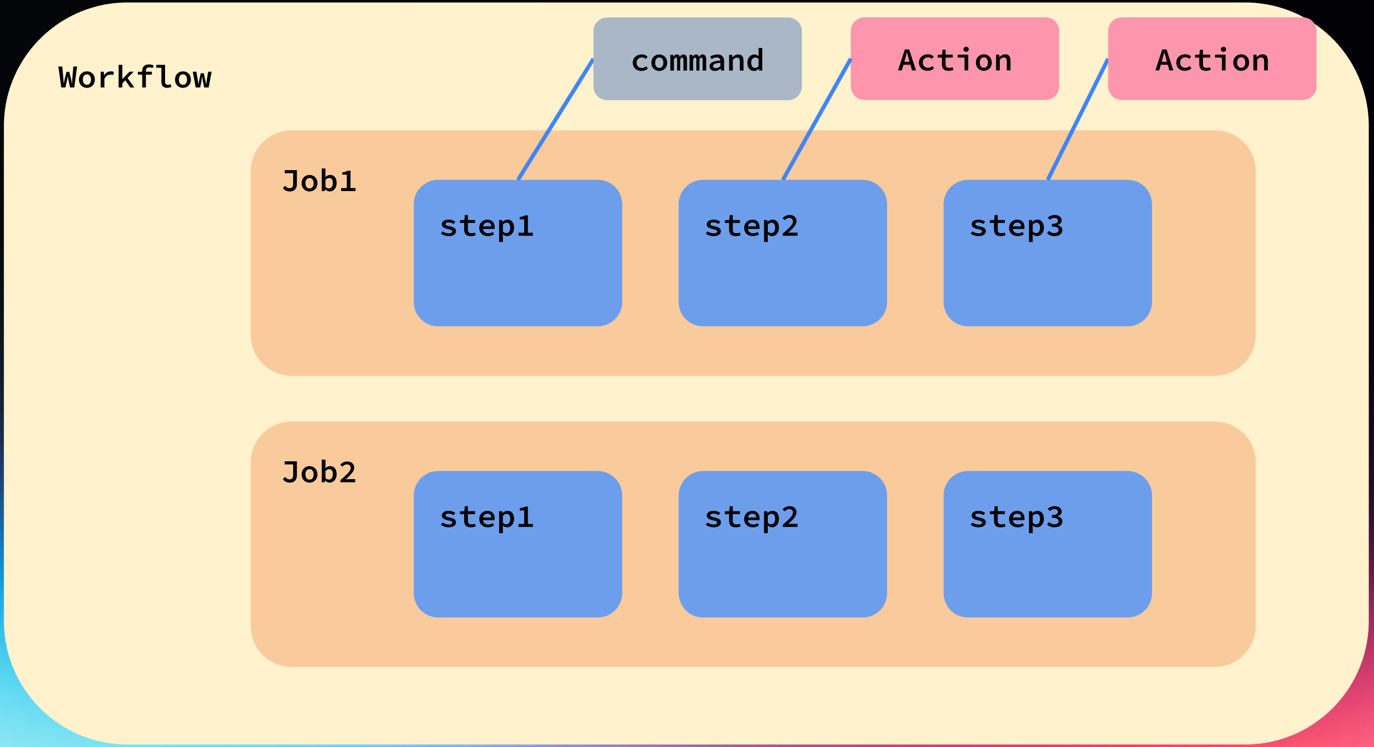
Для тех, кто не пользуется GHA

- Автоматизация сборки
- Прогон тестов
- Бета релизы (Firebase App Distribution, TestFlight)
- Заливка в сторы
- Статический анализ кода
- Репорты метрик, алертинг
- Автоматические действия (автомержи, комменты, и т.д.)

Для тех, кто не пользуется GHA



Для тех, кто не пользуется GHA



Проблема

- Есть Marketplace
- Иногда этого недостаточно
- Иногда хочется создать кастомное решение
  - мой кейс: тулза для подсчета Lead time
  - уже готовый код на Kotlin
  - другие причины

Я решил написать свой GHA, что делать?

Типы GitHub Actions

- Composite
- Docker
    - Linux only
    - more flexible
- JS
    - faster
    - @/actions/core, @/actions/github

Типы GitHub Actions: Composite

```
...

runs:
 using: "composite"
 steps:
   - name: My First Step
     id: my-first-step
     shell: bash
     run: |
       echo ...
```

Типы GitHub Actions

- Composite
- Docker
  - Linux only
  - more flexible
- JS
  - faster
  - @/actions/core, @/actions/github

Типы GitHub Actions

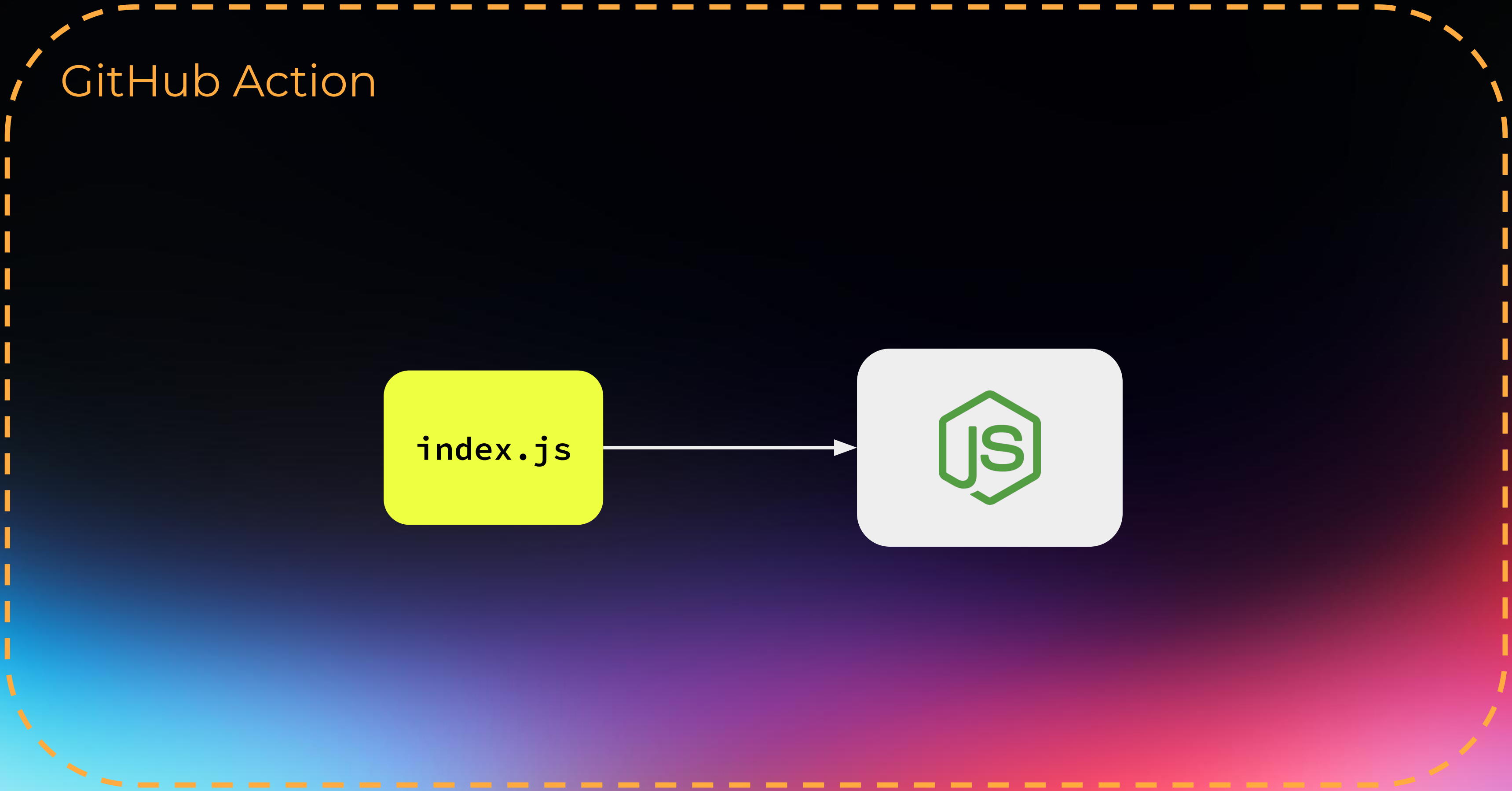
- Composite
- Docker
  - Linux only
  - more flexible
- JS
  - faster
  - @/actions/core, @/actions/github

JS GitHub Actions

- Рантайм Node.js
- Core libs
- Упаковка ncc (webpack)

Node.js

- Runtime, в котором можно запускать JavaScript
- Не в браузере.
- Построен на V8 JavaScript engine (такой же, как в Google Chrome)
- Single-Threaded Execution с неблокирующим I/O

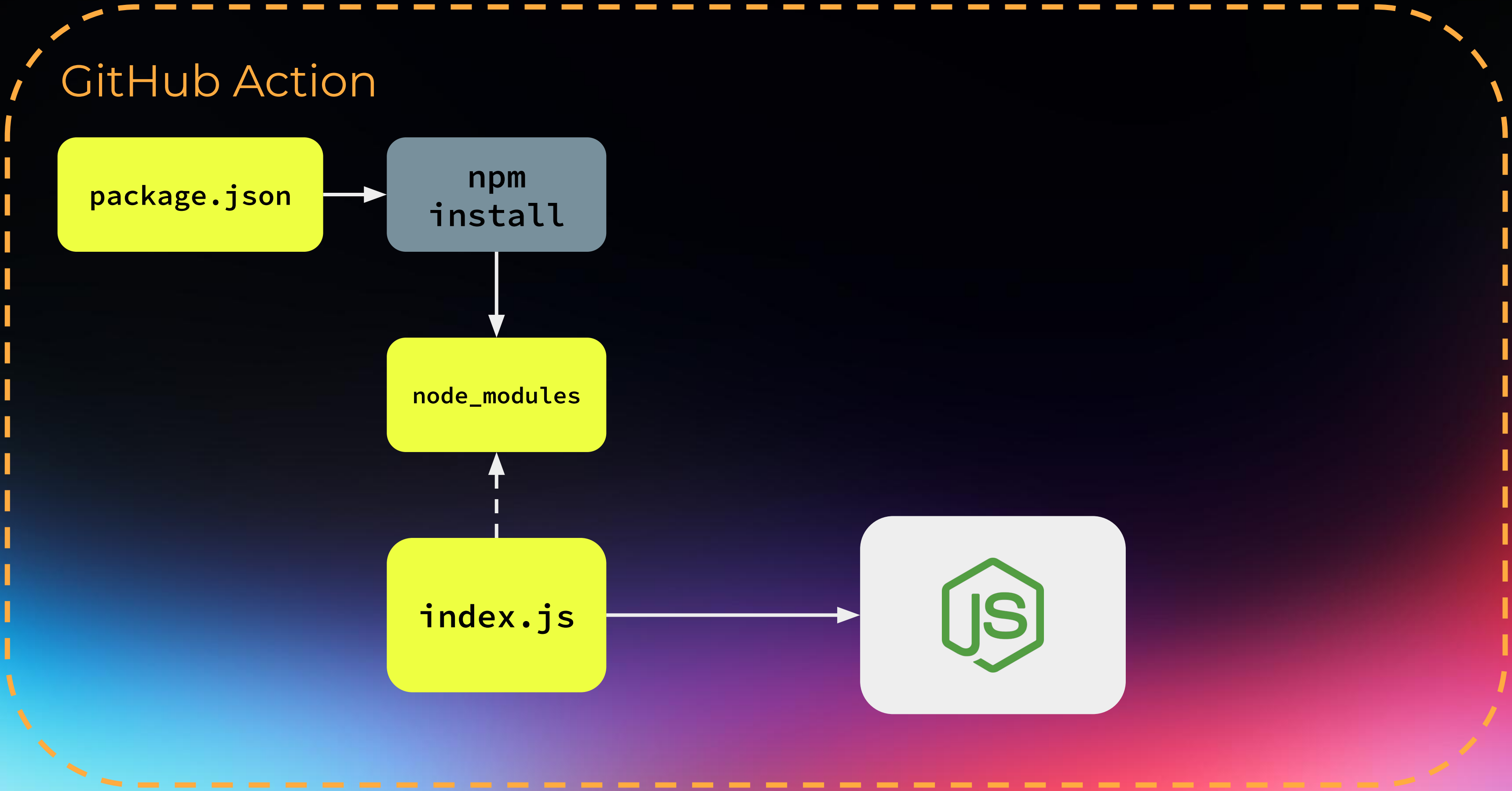


GitHub Action

index.js

NPM (Node Package Manager)

- Для Node.js куча библиотек, где они все?
- Доступны через NPM


GitHub Action
package.json
npm install
node_modules
index.js
JS

Напишем простейший GHA

- Напишем GHA на JS
- Потом разберемся, как то же самое сгенерировать с помощью Kotlin/JS

Напишем простейший GHA

Задаем зависимости, которые нам нужны:

```
// Это создаст нам пустой package.json
npm init -y

// Скачиваем и устанавливаем зависимость @actions/core
npm install @actions/core

node_modules
├── @actions
├── @fastify
├── tunnel
├── undici
└── uuid
```

Напишем простейший GHA

Создаем ./action.yml

```yaml
name: 'Roll a dice'
description: 'Simple Github Action for roll a dice'
inputs:
 number-of-sides:
   description: 'How many sides the dice has'
   required: true
   default: '6'
outputs:
 concat:
   description: 'Result of rolling the dice'
runs:
 using: 'node20'
 main: 'index.js'
```

Напишем простейший GHA

Создаем ./action.yml

```yaml
name: 'Roll a dice'
description: 'Simple Github Action for roll a dice'
inputs:
 number-of-sides:
   description: 'How many sides the dice has'
   required: true
   default: '6'
outputs:
 result:
   description: 'Result of rolling the dice'
runs:
 using: 'node20'
 main: 'index.js'
```

Напишем простейший GHA

Создаем ./action.yml

```yaml
name: 'Roll a dice'
description: 'Simple Github Action for roll a dice'
inputs:
 number-of-sides:
   description: 'How many sides the dice has'
   required: true
   default: '6'
outputs:
 concat:
   description: 'Result of rolling the dice'
runs:
 using: 'node20'
 main: 'index.js'
```

Напишем простейший GHA

.github/workflows/main.yml

```yaml
on: [push]

jobs:
 roll_the_dice_job:
   runs-on: ubuntu-latest
   name: Roll the dice
   steps:
     - name: Checkout
       uses: actions/checkout@v4
     - name: Roll the dice step
       id: roll
       uses: ./
       with:
         number-of-sides: '12'
     - name: Show the result step
       run: echo "The die rolled at ${{ steps.roll.outputs.result }}"
```

Напишем простейший GHA

.github/workflows/main.yml

```yaml
on: [push]

jobs:
 roll_the_dice_job:
   runs-on: ubuntu-latest
   name: Roll the dice
   steps:
     - name: Checkout
       uses: actions/checkout@v4
     - name: Roll the dice step
       id: roll
       uses: ./
       with:
         number-of-sides: '12'
     - name: Show the result step
       run: echo "The die rolled at ${{ steps.roll.outputs.result }}"
```

Напишем простейший GHA

.github/workflows/main.yml

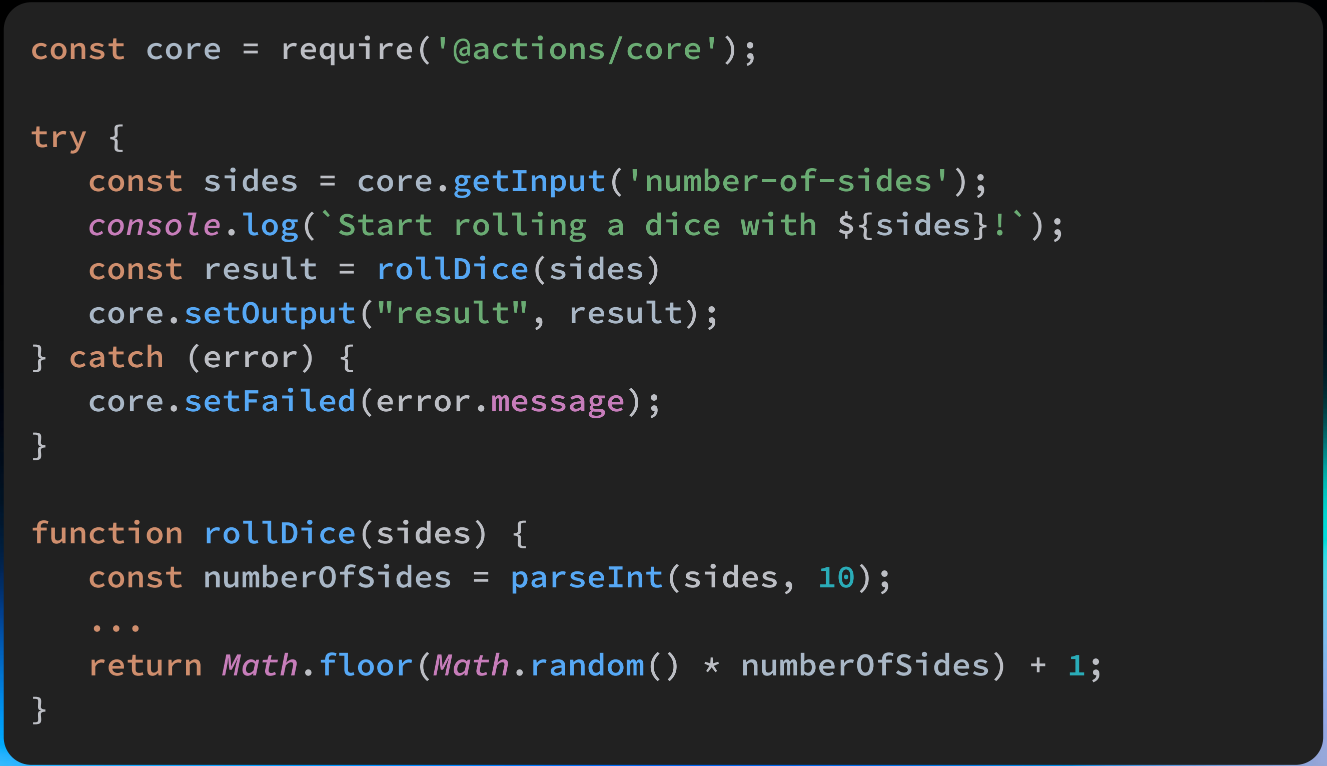
```
on: [push]

jobs:
 roll_the_dice_job:
   runs-on: ubuntu-latest
   name: Roll the dice
   steps:
     - name: Checkout
       uses: actions/checkout@v4
     - name: Roll the dice step
       id: roll
       uses: ./
       with:
         number-of-sides: '12'
     - name: Show the result step
       run: echo "The die rolled at ${{ steps.roll.outputs.result }}"
```

Напишем простейший GHA

.github/workflows/main.yml

```yaml
on: [push]

jobs:
 roll_the_dice_job:
   runs-on: ubuntu-latest
   name: Roll the dice
   steps:
     - name: Checkout
       uses: actions/checkout@v4
     - name: Roll the dice step
       id: roll
       uses: ./
       with:
         number-of-sides: '12'
     - name: Show the result step
       run: echo "The die rolled at ${{ steps.roll.outputs.result }}"
```

index.js



```
const core = require('@actions/core');

try {
   const sides = core.getInput('number-of-sides');
   console.log(`Start rolling a dice with ${sides}!`);
   const result = rollDice(sides)
   core.setOutput("result", result);
} catch (error) {
   core.setFailed(error.message);
}

function rollDice(sides) {
   const numberOfSides = parseInt(sides, 10);
   ...
   return Math.floor(Math.random() * numberOfSides) + 1;
}
```

index.js

```
const core = require('@actions/core');

try {
   const sides = core.getInput('number-of-sides');
   console.log(`Start rolling a dice with ${sides}!`);
   const result = rollDice(sides)
   core.setOutput("result", result);
} catch (error) {
   core.setFailed(error.message);
}

function rollDice(sides) {
   const numberOfSides = parseInt(sides, 10);
   ...
   return Math.floor(Math.random() * numberOfSides) + 1;
}
```

index.js

```
const core = require('@actions/core');

try {
   const sides = core.getInput('number-of-sides');
   console.log(`Start rolling a dice with ${sides}!`);
   const result = rollDice(sides)
   core.setOutput("result", result);
} catch (error) {
   core.setFailed(error.message);
}

function rollDice(sides) {
   const numberOfSides = parseInt(sides, 10);
   ...
   return Math.floor(Math.random() * numberOfSides) + 1;
}
```

index.js

```
const core = require('@actions/core');

try {
   const sides = core.getInput('number-of-sides');
   console.log(`Start rolling a dice with ${sides}!`);
   const result = rollDice(sides)
   core.setOutput("result", result);
} catch (error) {
   core.setFailed(error.message);
}

function rollDice(sides) {
   const numberOfSides = parseInt(sides, 10);
   ...
   return Math.floor(Math.random() * numberOfSides) + 1;
}
```

Запуск

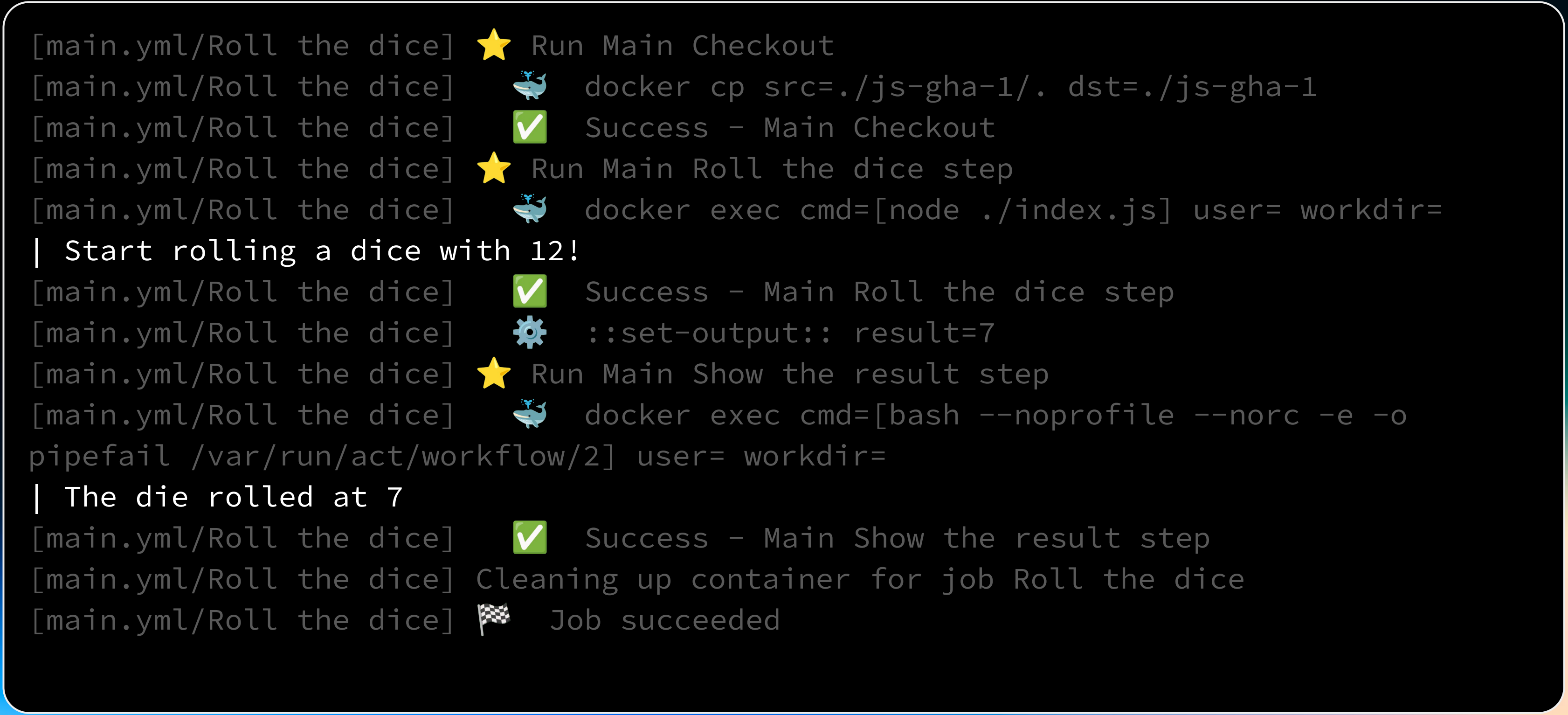
- Запушить в репозиторий
- Проверить локально через *act*

  https://nektosact.com/

Запуск

```
[main.yml/Roll the dice] ⭐ Run Main Checkout
[main.yml/Roll the dice]   🐳  docker cp src=./js-gha-1/. dst=./js-gha-1
[main.yml/Roll the dice]   ✅  Success - Main Checkout
[main.yml/Roll the dice] ⭐ Run Main Roll the dice step
[main.yml/Roll the dice]   🐳  docker exec cmd=[node ./index.js] user= workdir=
| Start rolling a dice with 12!
[main.yml/Roll the dice]   ✅  Success - Main Roll the dice step
[main.yml/Roll the dice]   ⚙️  ::set-output:: result=7
[main.yml/Roll the dice] ⭐ Run Main Show the result step
[main.yml/Roll the dice]   🐳  docker exec cmd=[bash --noprofile --norc -e -o
pipefail /var/run/act/workflow/2] user= workdir=
| The die rolled at 7
[main.yml/Roll the dice]   ✅  Success - Main Show the result step
[main.yml/Roll the dice] Cleaning up container for job Roll the dice
[main.yml/Roll the dice] 🏁  Job succeeded
```

Запуск

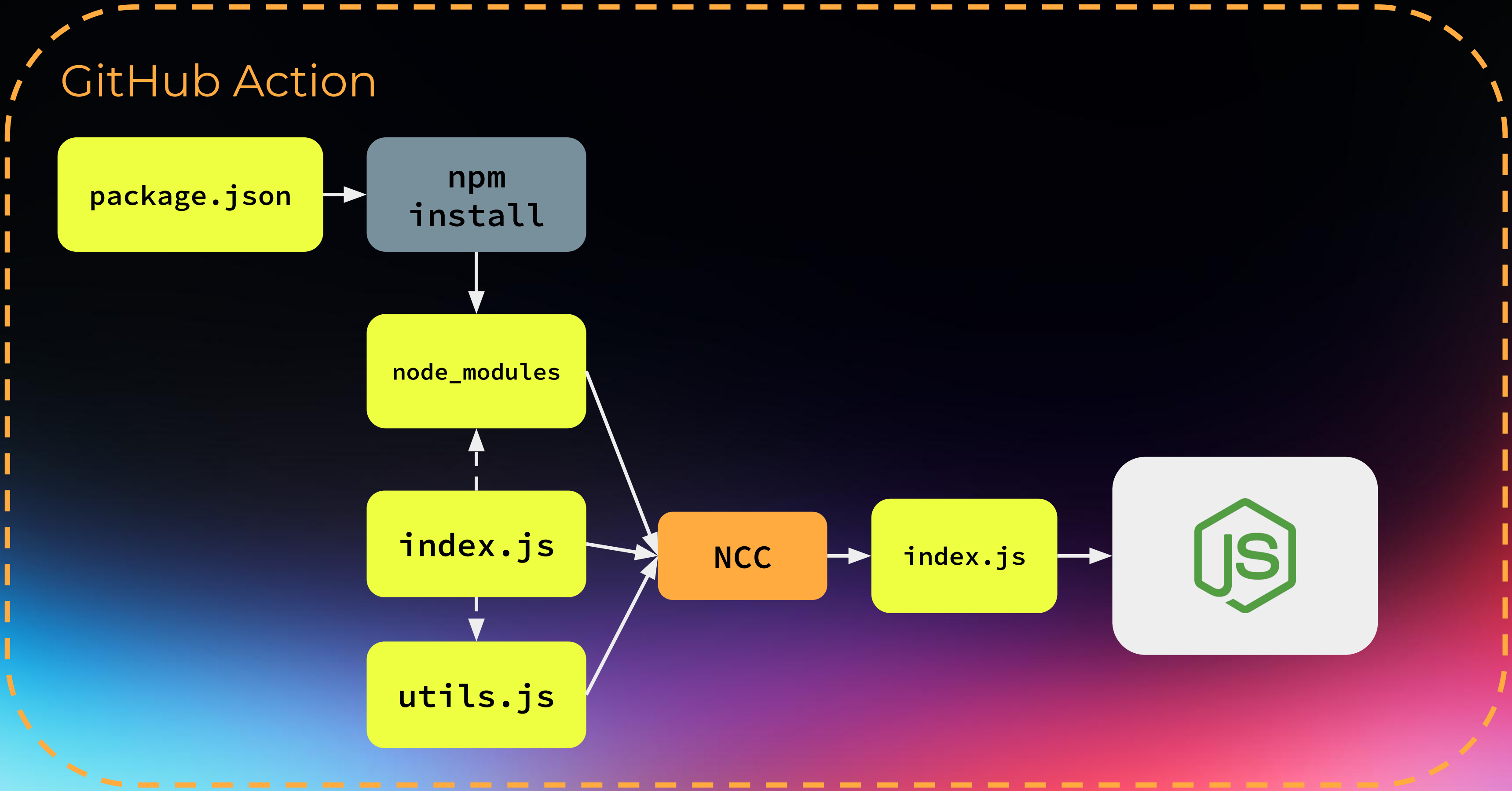


```
[main.yml/Roll the dice] ⭐ Run Main Checkout
[main.yml/Roll the dice]   🐳  docker cp src=./js-gha-1/. dst=./js-gha-1
[main.yml/Roll the dice]   ✅  Success - Main Checkout
[main.yml/Roll the dice] ⭐ Run Main Roll the dice step
[main.yml/Roll the dice]   🐳  docker exec cmd=[node ./index.js] user= workdir=
| Start rolling a dice with 12!
[main.yml/Roll the dice]   ✅  Success - Main Roll the dice step
[main.yml/Roll the dice]   ⚙️  ::set-output:: result=7
[main.yml/Roll the dice] ⭐ Run Main Show the result step
[main.yml/Roll the dice]   🐳  docker exec cmd=[bash --noprofile --norc -e -o
pipefail /var/run/act/workflow/2] user= workdir=
| The die rolled at 7
[main.yml/Roll the dice]   ✅  Success - Main Show the result step
[main.yml/Roll the dice] Cleaning up container for job Roll the dice
[main.yml/Roll the dice] 🏁  Job succeeded
```

NCC (Node Compiler Collection)

- NPM устанавливает зависимости в папку node_modules
- Не удобно их загружать в git
- Для этого есть паккеры, например, NCC

```
node_modules
├── @actions
├── @fastify
├── tunnel
├── undici
└── uuid
```


GitHub Action
package.json
npm install
node_modules
index.js
utils.js
NCC
index.js
JS

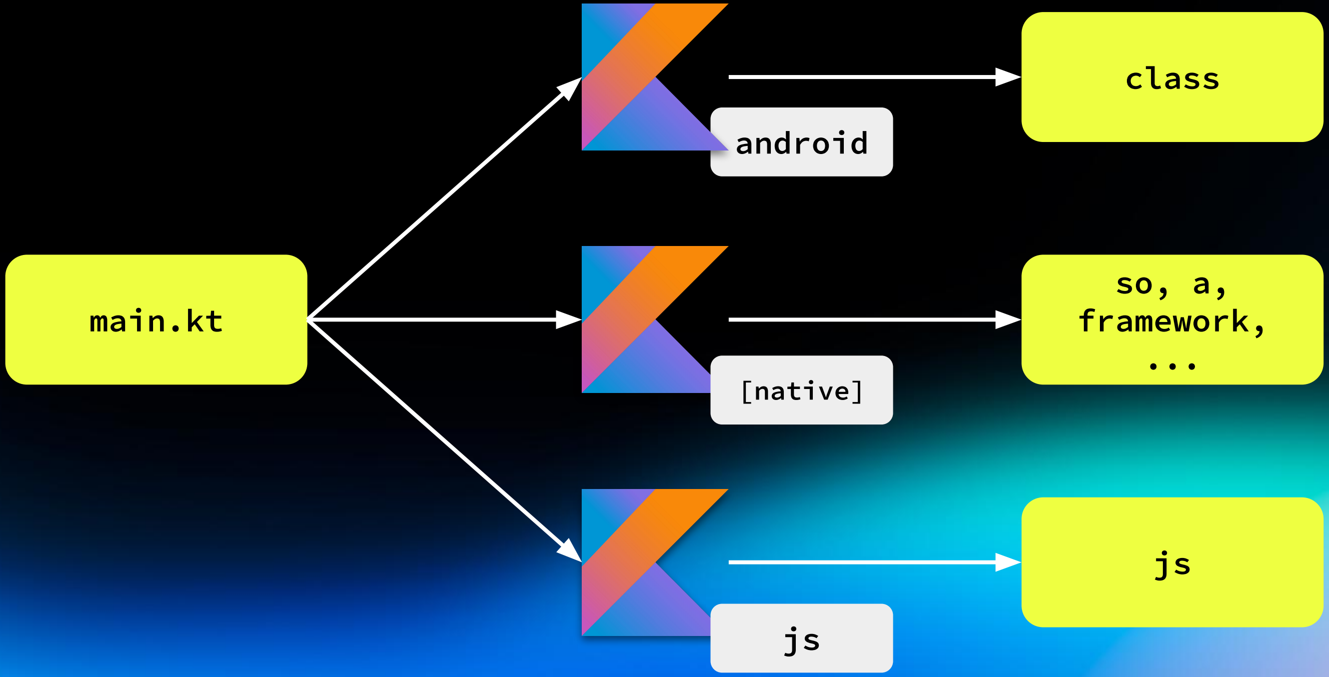
NCC (Node Compiler Collection)

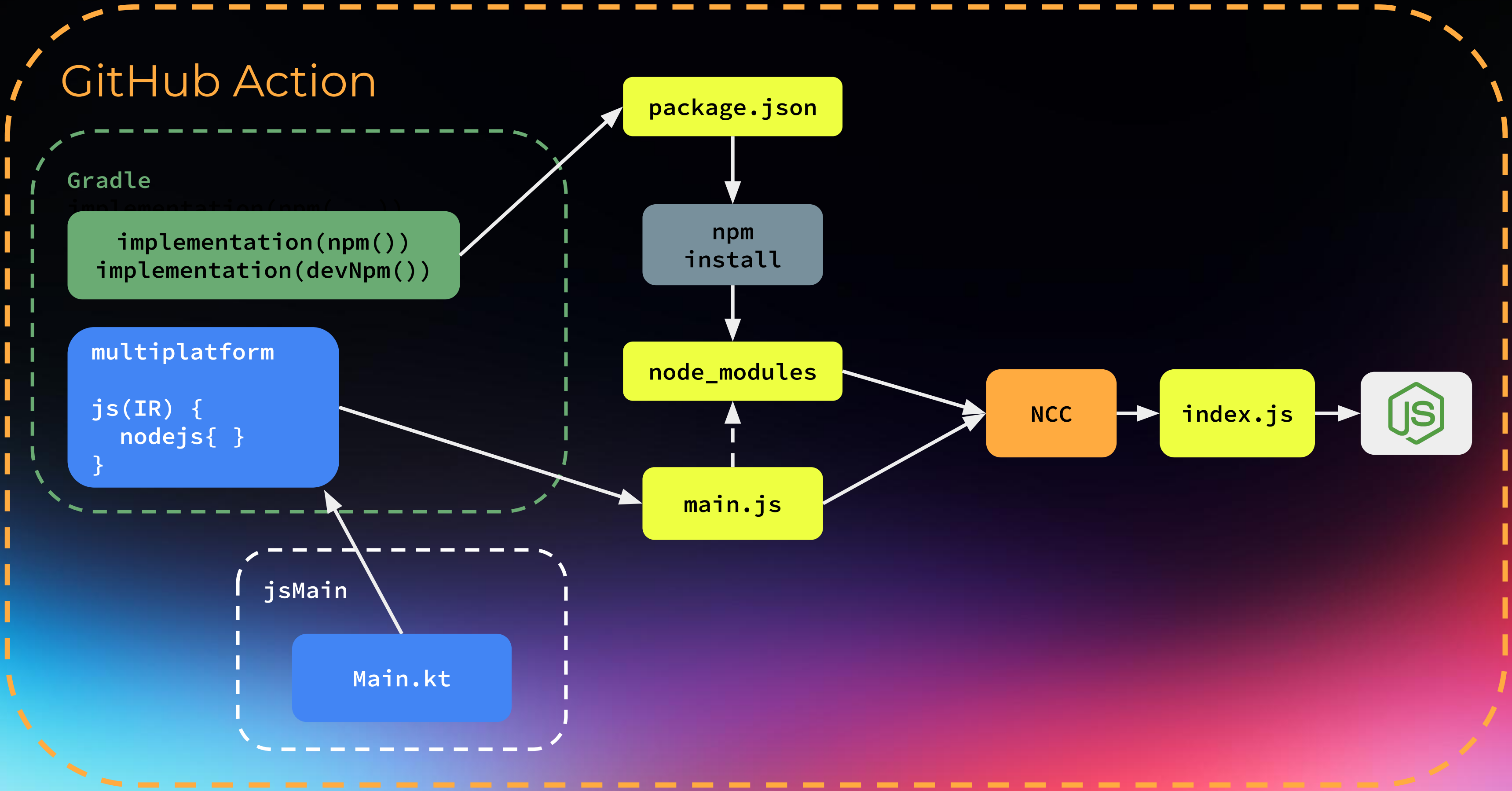
- Устанавливаем NCC
- Запускаем
- Получаем boundled index.js

```
npm i -g @vercel/ncc
ncc build index.js --license licenses.txt

dist
├── index.js
└── licenses.txt
```

Создание простого Kotlin/JS Action

Создание простого Kotlin/JS Action





GitHub Action

Gradle

```
implementation(npm())
implementation(devNpm())
```

```
multiplatform

js(IR) {
  nodejs{ }
}
```

jsMain

Main.kt

package.json

npm install

node_modules

main.js

NCC

index.js

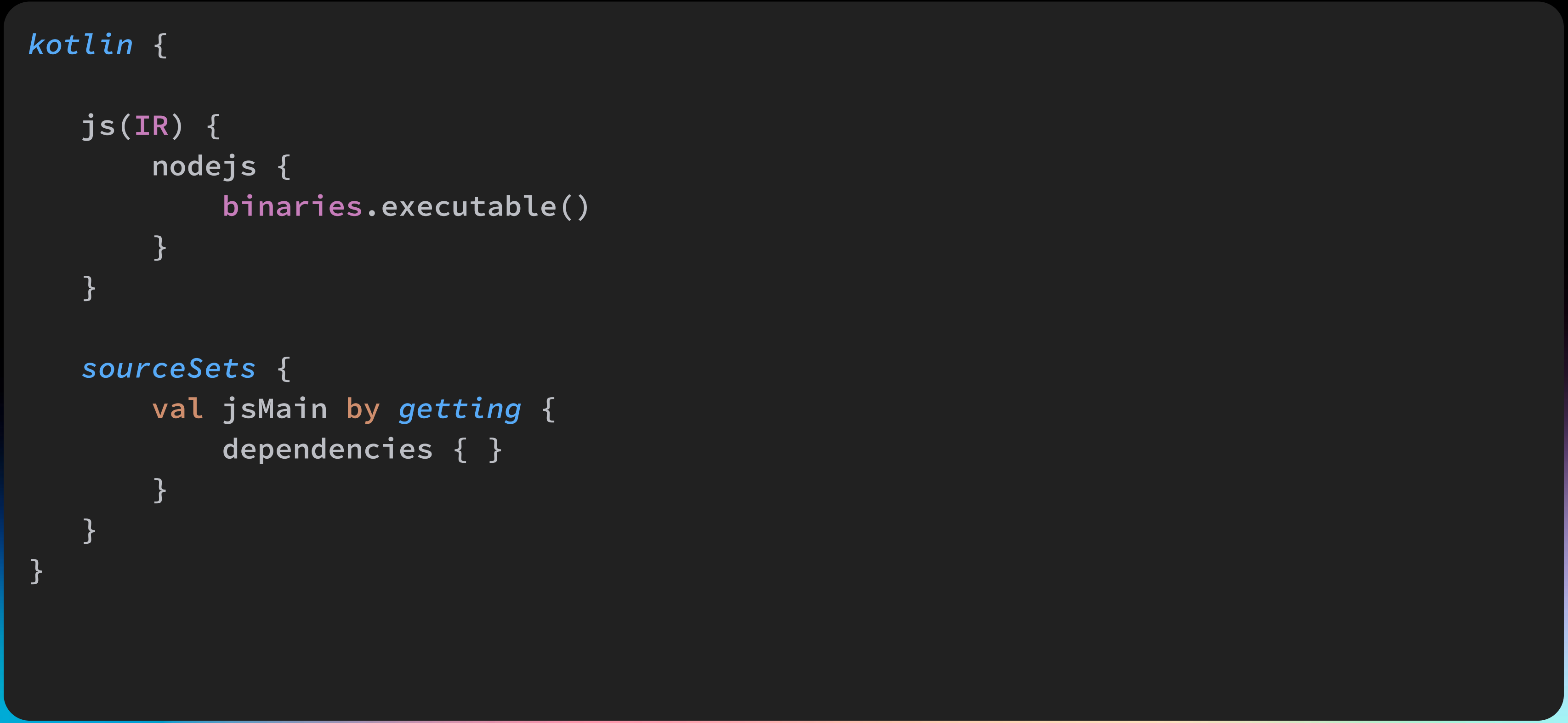
Настраиваем KMP

```
plugins {
    kotlin("js") version "2.0.0"
}
```

или

```
plugins {
    kotlin("multiplatform") version "2.0.0"
}
```

Конфигурируем



```
kotlin {

   js(IR) {
       nodejs {
           binaries.executable()
       }
   }

   sourceSets {
       val jsMain by getting {
           dependencies { }
       }
   }
}
```

action.yml

```yaml
name: 'Test GHA 1'
description: 'This action for learning purposes'
outputs:
 result:
   description: 'Result of GHA'
runs:
 using: 'node20'
 main: 'dist/index.js'
```

main.yml

```
on: [push]

jobs:
 hello_world_job:
   runs-on: ubuntu-latest
   name: Launch my custom action
   steps:
     - name: Checkout
       uses: actions/checkout@v4
     - name: Launch my custom kotlin gha
       uses: ./
       id: custom-gha
     - name: Print the result of GHA
       run: echo "Result is - ${{ steps.custom-gha.outputs.result }}"
```

dependencies

```
js(IR) {
   useCommonJs()
   nodejs {
       binaries.executable()
   }
}

sourceSets {
   val jsMain by getting {
       dependencies {
           implementation(npm("@actions/core", "1.4.0"))
       }
   }
}
```

ActionsCore.kt

```kotlin
@file:JsModule("@actions/core")
package com.example.utils.actions


external fun setOutput(name: String, value: Any)
external fun setFailed(message: String)
```

Dukat

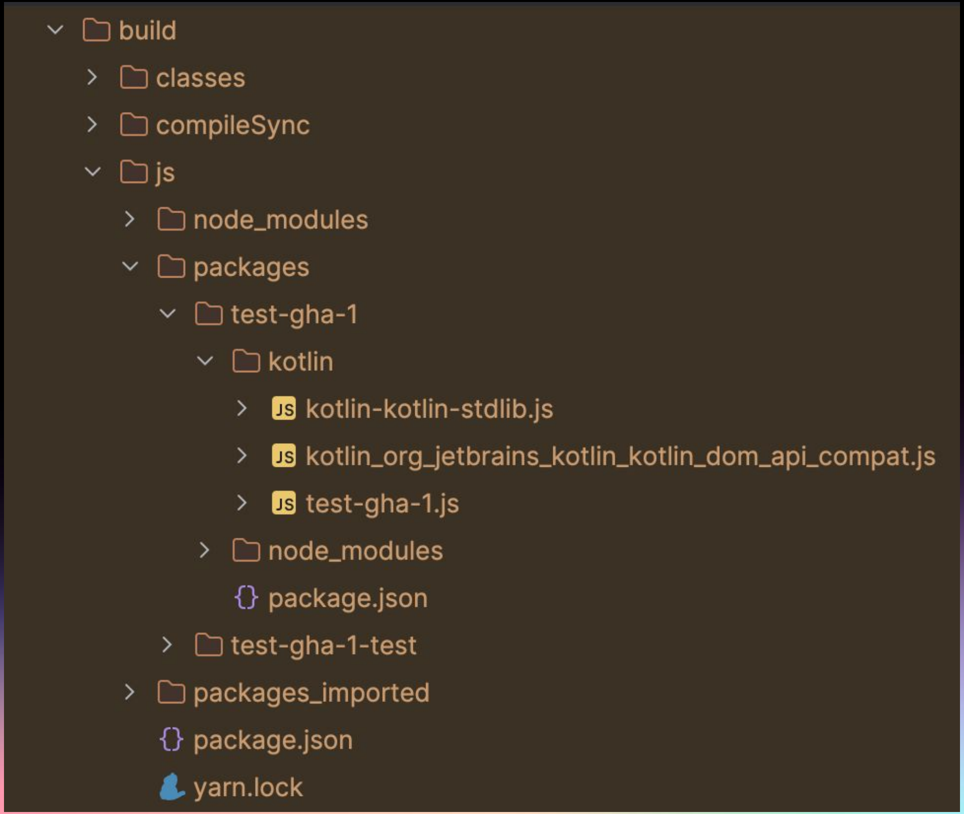
- Есть тулза Dukat, которая упрощает жизнь
- Но у нее есть жалобы (долго не исправляют issue)

App.kt

```kotlin
import com.example.utils.actions.*


suspend fun main() {
    setOutput("failed", false)
    try {
        val result = "Custom String! Congratulations!"
        setOutput("result", result)
        print(result)
    } catch (ex: Exception) {
        setFailed("Error while performing GHA")
    }
}
```

Что сгенерировал KMP

- ./gradlew build



```
build
  classes
  compileSync
  js
    node_modules
    packages
      test-gha-1
        kotlin
          kotlin-kotlin-stdlib.js
          kotlin_org_jetbrains_kotlin_kotlin_dom_api_compat.js
          test-gha-1.js
        node_modules
        package.json
      test-gha-1-test
    packages_imported
    package.json
    yarn.lock
```

Дальше надо сделать шаги:

- Найти скомпиленные файлы

  `./build/js/packages/test-gha-1/kotlin`
- Выполнить ncc

  `ncc build test-gha-1.js --license licenses.txt`
- получить результат в

  `./build/js/packages/test-gha-1/kotlin/dist`
- скопировать оттуда в

  `./dist`

Запихиваем ncc в Gradle таску

```kotlin
tasks.register<Exec>("buildAndPackWithInstalledNCC") {
    dependsOn("build")
    commandLine("npx", "ncc", "build",
        "${layout.buildDirectory.get()}/js/packages/${project.name}
        /kotlin/${project.name}.js",
        "--license", "licenses.txt", "-o", "dist")
}
```

Запихиваем ncc в Gradle таску

```
tasks.register<Exec>("buildAndPackWithInstalledNCC") {
   dependsOn("build")
   commandLine("npx", "ncc", "build",
        "${layout.buildDirectory.get()}/js/packages/${project.name}
        /kotlin/${project.name}.js",
        "--license", "licenses.txt", "-o", "dist")
}
```

- Но NCC должен быть установлен!

Устанавливаем npm автоматически

build.gradle.kts:

```
sourceSets {
    val jsMain by getting {
        dependencies {
            implementation(npm("@actions/core", "1.4.0"))
            implementation(devNpm("@vercel/ncc", "0.38.1"))
        }
    }
}
```

Устанавливаем npm автоматически

package.json:

```
{
...
 "devDependencies": {
   "@vercel/ncc": "0.38.1",
   "typescript": "5.4.3",
   "source-map-support": "0.5.21"
 },
 "dependencies": { ... },
}
```

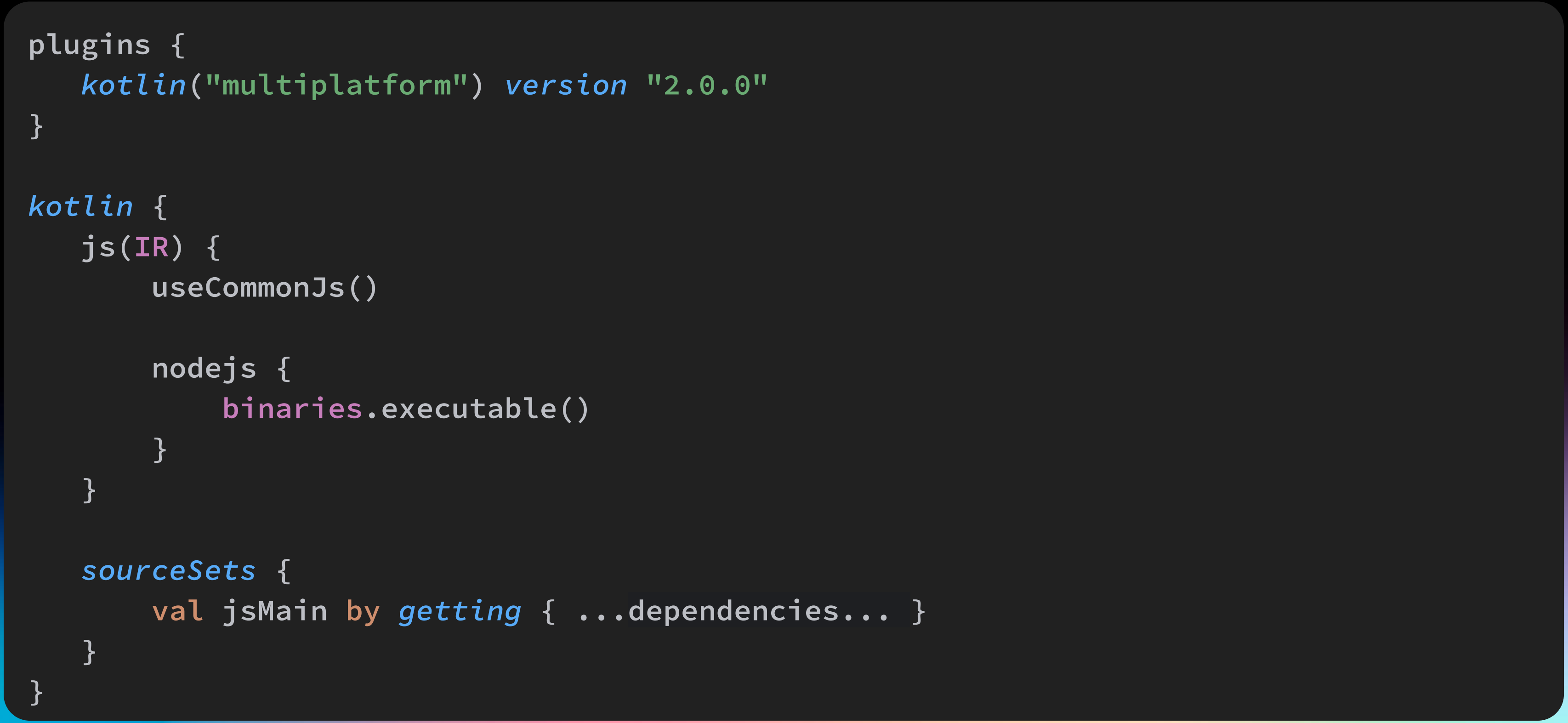
Устанавливаем npm автоматически

```
tasks.register<Exec>("installNodeModules") {
    dependsOn("build")
    workingDir = file("${layout.buildDirectory.get()}/js/packages/${project.name}/")
    commandLine("npm", "install")
}

tasks.register<Exec>("buildAndInstallAndPackWithNCC") {
    dependsOn("installNodeModules")
    workingDir = file("${layout.buildDirectory.get()}/js/packages/${project.name}/")
    commandLine("npx", "ncc", "build", "./kotlin/${project.name}.js", "--license",
                "licenses.txt", "-o", "${layout.projectDirectory}/dist")
}
```

Устанавливаем npm автоматически

- Будет каждый раз скачивать зависимость и устанавливать её
- Сделаем более круто — воспользуемся API NCC

Создаем новый gradle модуль

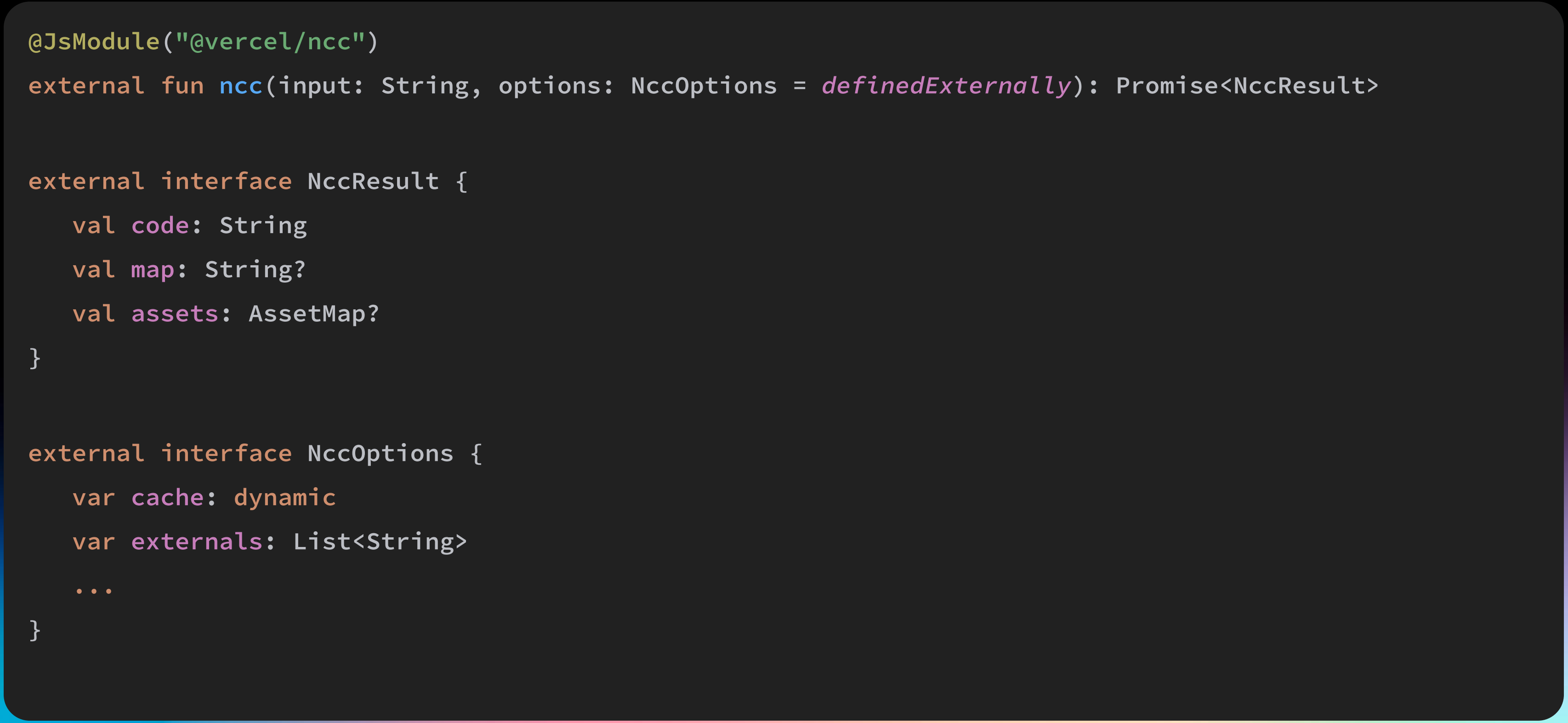


```
plugins {
    kotlin("multiplatform") version "2.0.0"
}

kotlin {
    js(IR) {
        useCommonJs()

        nodejs {
            binaries.executable()
        }
    }

    sourceSets {
        val jsMain by getting { ...dependencies... }
    }
}
```

Добавляем зависимости

```
dependencies {
    implementation("org.jetbrains.kotlinx:kotlinx-coroutines-core:1.5.0")
    implementation("org.jetbrains.kotlinx:kotlinx-nodejs:0.0.7")
    implementation("org.jetbrains.kotlin-wrappers:kotlin-js:1.0.0-pre.785")
    implementation(npm("@vercel/ncc", "0.38.1", generateExternals = false))
}
```

Делаем интероп



```kotlin
@JsModule("@vercel/ncc")
external fun ncc(input: String, options: NccOptions = definedExternally): Promise<NccResult>

external interface NccResult {
    val code: String
    val map: String?
    val assets: AssetMap?
}

external interface NccOptions {
    var cache: dynamic
    var externals: List<String>
    ...
}
```

Главный метод: ncc(...)

```kotlin
val nccResult = ncc(
   input = inputPath,
   options = jsObject {
       sourceMap = true
       license = "LICENSES"
   }
).await()

external interface NccResult {
   val code: String
   val map: String?
   val assets: AssetMap?
}
```

main метод нового модуля

```kotlin
suspend fun main() {
   runCatching {
       val (inputPath, outputPath) = readArgs(process.argv)

       val combinedCode = combineCode(
           inputPath = inputPath,
           outputPath = outputPath,
           fileName = "index.js"
       )

       createOutputFolder(outputPath = outputPath)

       with(combinedCode) {
           copyCode()
           copyMapping()
           copyAssets()
       }
   }.onFailure { throwable ->
       console.error(throwable)
       process.exit(1)
   }
}
```

main метод нового модуля

```kotlin
suspend fun main() {
   runCatching {
       val (inputPath, outputPath) = readArgs(process.argv)

       val combinedCode = combineCode(
           inputPath = inputPath,
           outputPath = outputPath,
           fileName = "index.js"
       )

       createOutputFolder(outputPath = outputPath)

       with(combinedCode) {
           copyCode()
           copyMapping()
           copyAssets()
       }
   }.onFailure { throwable ->
       console.error(throwable)
       process.exit(1)
   }
}
```

main метод нового модуля

```
suspend fun main() {
    runCatching {
        val (inputPath, outputPath) = readArgs(process.argv)

        val combinedCode = combineCode(
            inputPath = inputPath,
            outputPath = outputPath,
            fileName = "index.js"
        )

        createOutputFolder(outputPath = outputPath)

        with(combinedCode) {
            copyCode()
            copyMapping()
            copyAssets()
        }
    }.onFailure { throwable ->
        console.error(throwable)
        process.exit(1)
    }
}
```

main метод нового модуля

```kotlin
suspend fun main() {
    runCatching {
        val (inputPath, outputPath) = readArgs(process.argv)

        val combinedCode = combineCode(
            inputPath = inputPath,
            outputPath = outputPath,
            fileName = "index.js"
        )

        createOutputFolder(outputPath = outputPath)

        with(combinedCode) {
            copyCode()
            copyMapping()
            copyAssets()
        }
    }.onFailure { throwable ->
        console.error(throwable)
        process.exit(1)
    }
}
```

main метод нового модуля

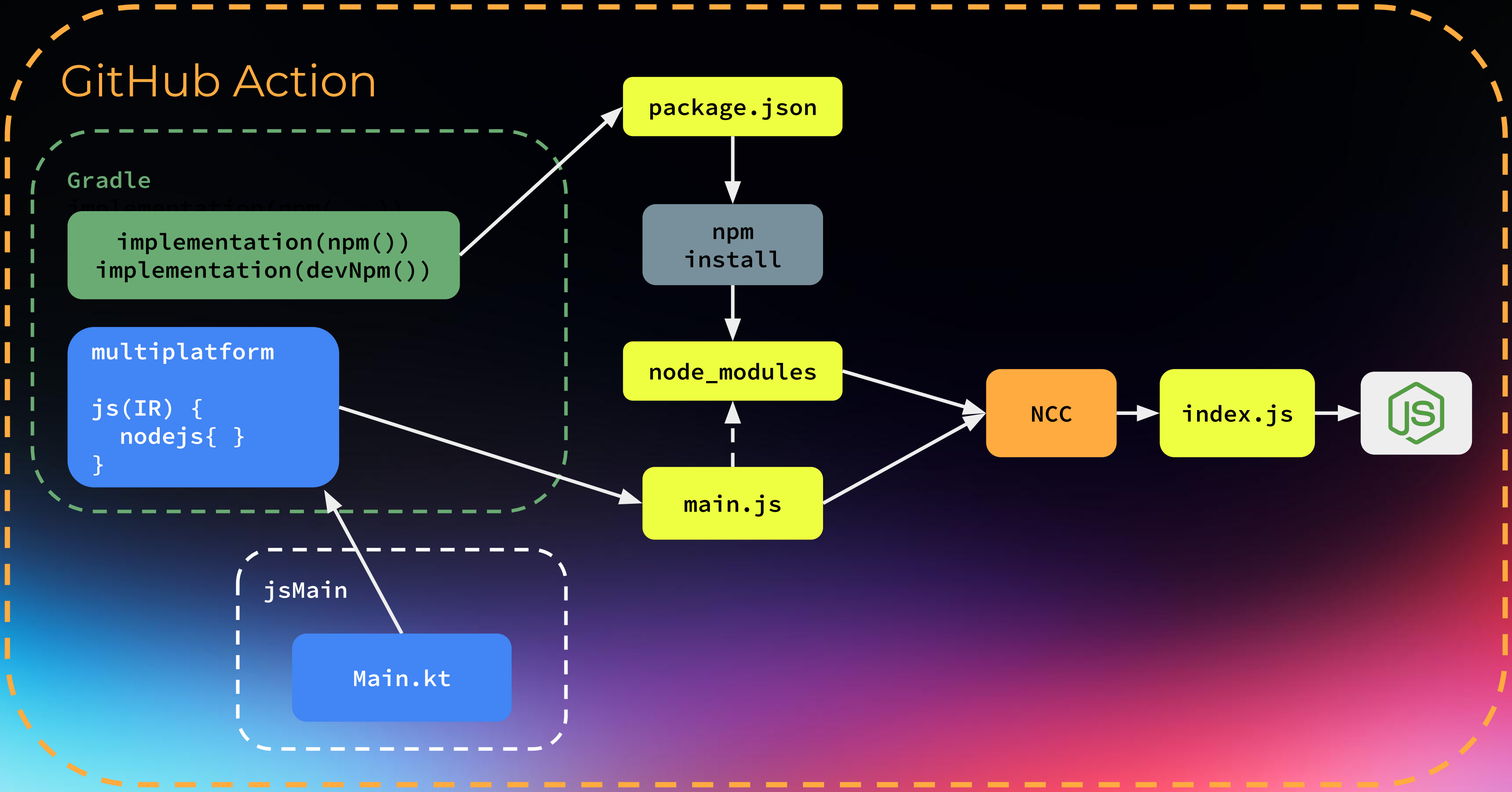
```
suspend fun main() {
    runCatching {
        val (inputPath, outputPath) = readArgs(process.argv)

        val combinedCode = combineCode(
            inputPath = inputPath,
            outputPath = outputPath,
            fileName = "index.js"
        )

        createOutputFolder(outputPath = outputPath)

        with(combinedCode) {
            copyCode()
            copyMapping()
            copyAssets()
        }
    }.onFailure { throwable ->
        console.error(throwable)
        process.exit(1)
    }
}
```

Передаем аргументы в модуль

```
tasks.named<NodeJsExec>("jsNodeProductionRun") {
    val inputPath =
        "${rootProject.layout.buildDirectory.get()}/js/packages/${rootProject.name}/"
    val outputPath = "${rootProject.layout.projectDirectory}/dist/"
    args(inputPath, outputPath)
}
```

Добавляем запуск модуля после build

```
./gradlew build :ncc:jsNodeProductionRun
```



GitHub Action

Gradle

```
implementation(npm())
implementation(devNpm())
```

```
multiplatform

js(IR) {
  nodejs{ }
}
```

jsMain

Main.kt

package.json

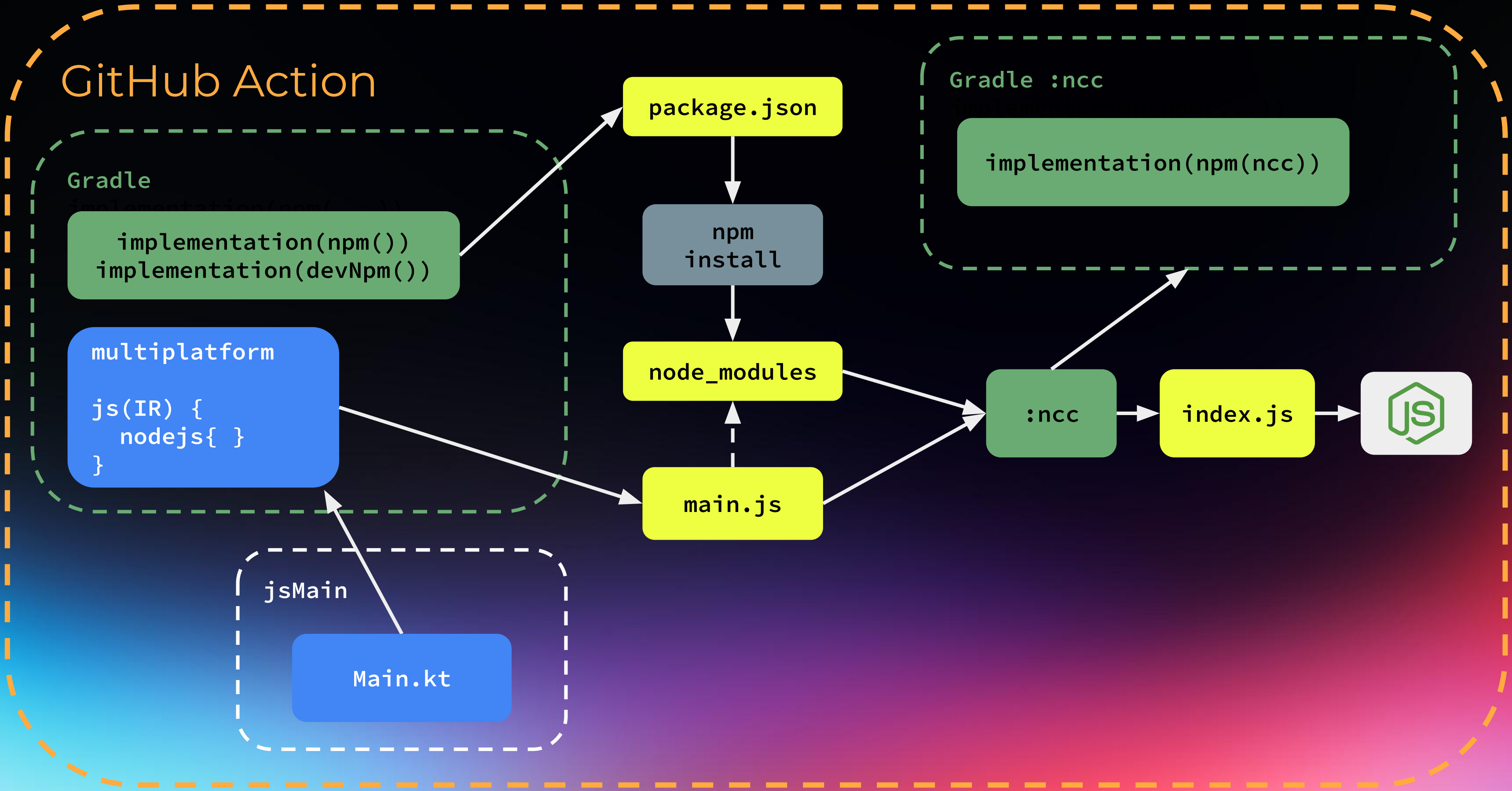
npm
install

node_modules

main.js

NCC

index.js



GitHub Action

Gradle

```
implementation(npm())
implementation(devNpm())
```

```
multiplatform

js(IR) {
  nodejs{ }
}
```

jsMain

`Main.kt`

`package.json`

npm install

`node_modules`

`main.js`

Gradle :ncc

`implementation(npm(ncc))`

`:ncc`

`index.js`

GHA по подсчету Lead Time

GHA по подсчету Lead Time

- Commit Lead time to release
- Commit Lead time to develop

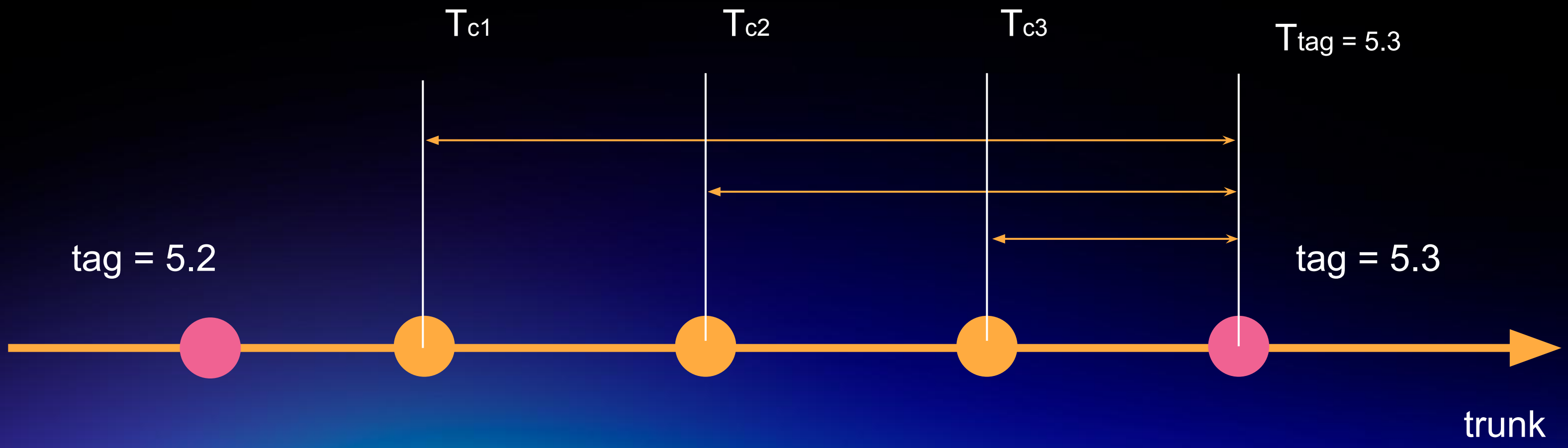
Commit Lead time to release

Commit Lead time to release

Как считали:

- `GET /repos/$owner/$repo/git/refs/tags`
- `GET /repos/$owner/$repo/compare/$from...$to`
- Проставили для каждого коммита дату релиза (тега)
- Подсчитали разницу между датой тега и датой создания коммита

Commit Lead time to release

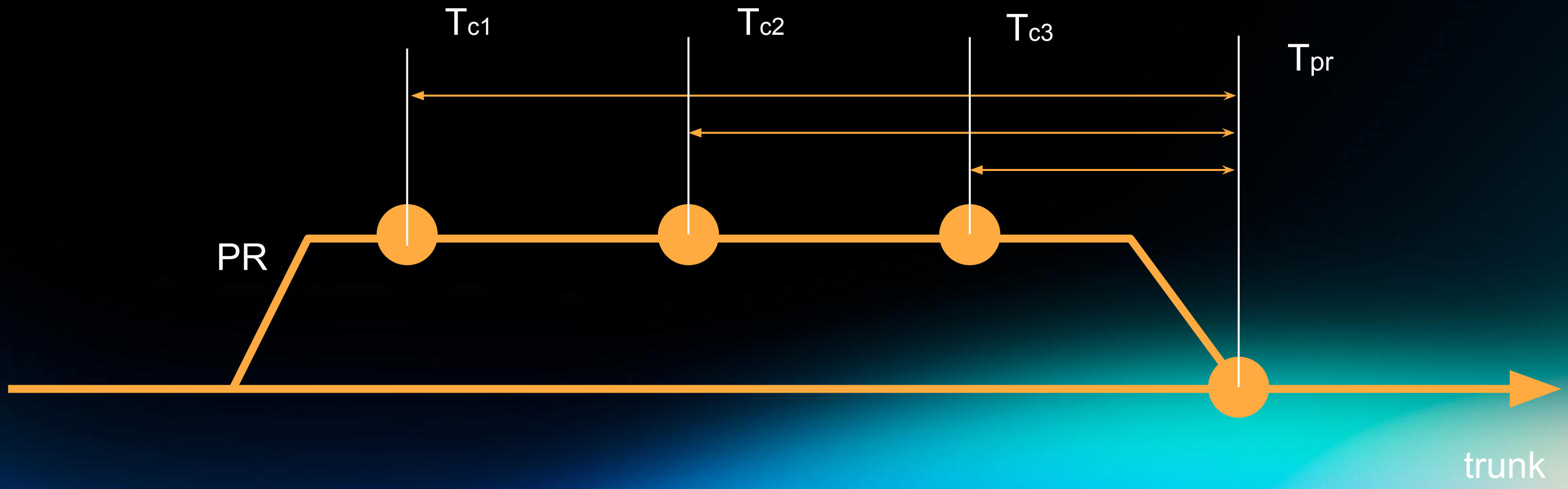


Commit Lead time to develop

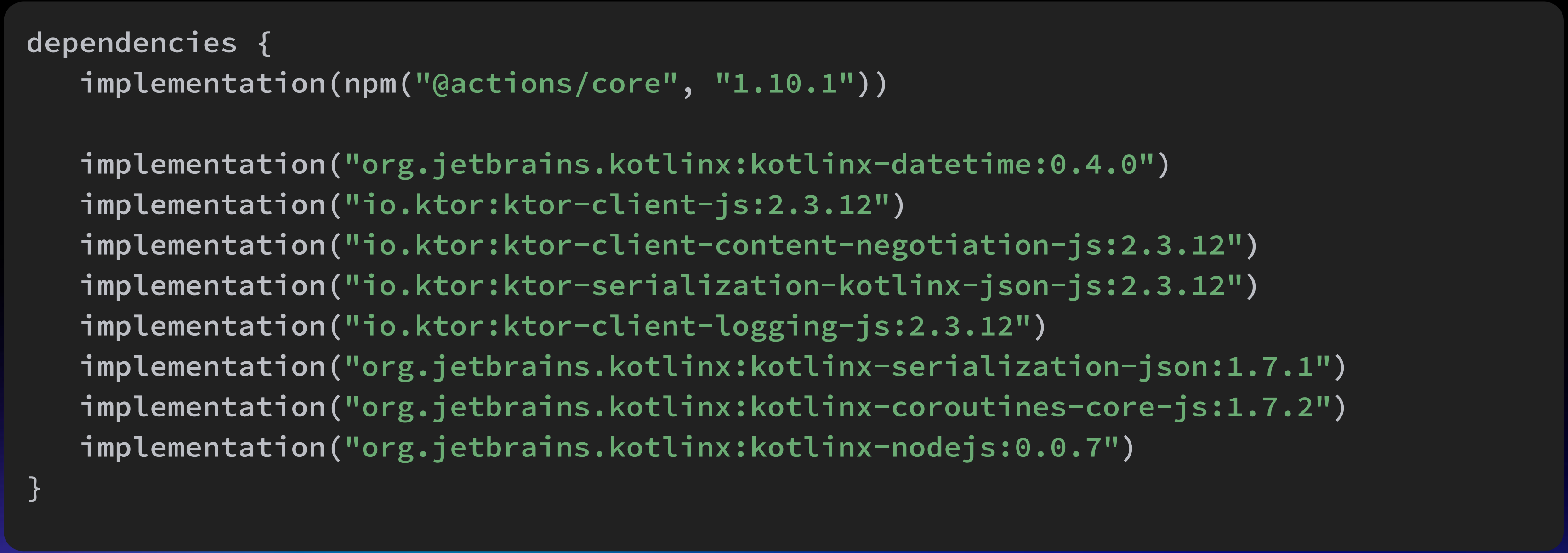
Как считали:

- `GET /repos/{owner}/{repo}/pulls`
- `GET /repos/{owner}/{repo}/pulls/{pull_number}/commits`
- Проставили для каждого коммита дату мерджа PR
- Подсчитали разницу между датой мерджа и датой создания коммита

Commit Lead time to develop



GHA по подсчету Lead Time: Gradle



```
dependencies {
    implementation(npm("@actions/core", "1.10.1"))

    implementation("org.jetbrains.kotlinx:kotlinx-datetime:0.4.0")
    implementation("io.ktor:ktor-client-js:2.3.12")
    implementation("io.ktor:ktor-client-content-negotiation-js:2.3.12")
    implementation("io.ktor:ktor-serialization-kotlinx-json-js:2.3.12")
    implementation("io.ktor:ktor-client-logging-js:2.3.12")
    implementation("org.jetbrains.kotlinx:kotlinx-serialization-json:1.7.1")
    implementation("org.jetbrains.kotlinx:kotlinx-coroutines-core-js:1.7.2")
    implementation("org.jetbrains.kotlinx:kotlinx-nodejs:0.0.7")
}
```

GHA по подсчету Lead Time: main

```kotlin
suspend fun main() {
    // read inputs
    ...

    // setup
    ...

    try {
        // do action
        ...
    } catch (e: Exception) {
        setFailed("Error while performing GitHub Action: ${e.message}")
    }
}
```

GHA по подсчету Lead Time: inputs

```
external fun getInput(name: String, ...): String

fun buildGHAInput(): GHAInput = group("Reading input values") {
   val repoAsList = Env.GITHUB_REPOSITORY.split("/")

   return@group GHAInput(
       token = getInput("Token").ifEmpty { Env.GITHUB_TOKEN },
       owner = getInput("Owner").ifEmpty { repoAsList.first() },
       repo = getInput("Repo").ifEmpty { repoAsList[1] },
       fromTag = getInput("FromTag"),
       toTag = getInput("ToTag"),
       excludeBranches = getInput("ExcludeBranches").split(",")
           .map { it.trim() },
       debug = getInput("Debug").ifEmpty { null }.toBoolean() ?: false,
   )
}
```

GHA по подсчету Lead Time: setup, DI

```
suspend fun main() {
    // read inputs
    ...

    val router = Router()
    val di = DI.create(input)
    router.registerRoute(LEAD_TIME_TO_RELEASE, di.provideReleaseControllerFactory())
    router.registerRoute(LEAD_TIME_TO_MAIN, di.provideMainControllerFactory())

    // do action
    ...
}
```

GHA по подсчету Lead Time: action

```
suspend fun main() {
    ...

    try {
        group("Calculate commit lead time") {
            val result = router.route(command, input)
            if (result.success) {
                setOutput(LeadTime.value, result.convertToCSV())
            } else {
                setFailed(result.errorText.orEmpty())
            }
        }
    } catch (e: Exception) {
        setFailed("Error while performing GitHub Action: ${e.message}")
    }
}
```

GHA по подсчету Lead Time: use

```
jobs:
 calc-lead-time-job:
  ...
  steps:
   - name: Get source code
     uses: actions/checkout@v4
   - name: Launch Lead Time Action
     uses: dodobrands/commitleadtime-action@v0.0.6
     id: lead-time
     with:
      command: toMain
      fromDate: ${{ inputs.fromDate }}
      excludeBranches: master, master*, release/*, test/*
      debug: true
     env:
      GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
```

GHA по подсчету Lead Time: use

```
jobs:
 calc-lead-time-job:
   ...
   steps:
     - name: Get source code
       uses: actions/checkout@v4
     - name: Launch Lead Time Action
       uses: dodobrands/commitleadtime-action@v0.0.6
       id: lead-time
       with:
         command: toMain
         fromDate: ${{ inputs.fromDate }}
         excludeBranches: master, master*, release/*, test/*
         debug: true
       env:
         GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
```

GHA по подсчету Lead Time: use

```
jobs:
 calc-lead-time-job:
   ...
   steps:
     - name: Get source code
       uses: actions/checkout@v4
     - name: Launch Lead Time Action
       uses: dodobrands/commitleadtime-action@v0.0.6
       id: lead-time
       with:
         command: toMain
         fromDate: ${{ inputs.fromDate }}
         excludeBranches: master, master*, release/*, test/*
         debug: true
       env:
         GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
```

GHA по подсчету Lead Time: use

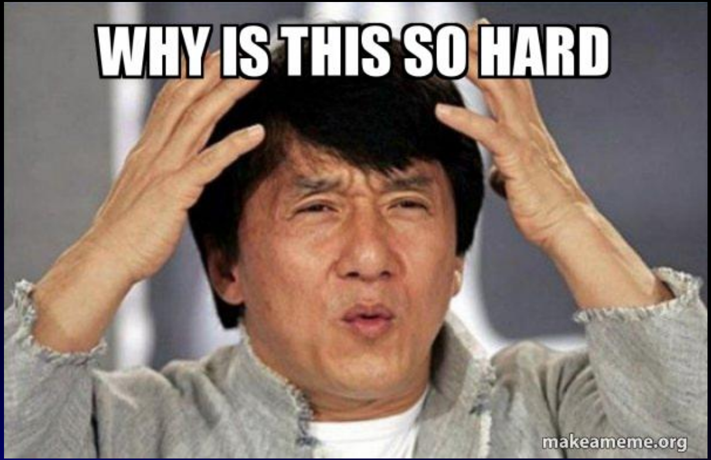
```
steps:
  ...
  - name: Convert CSV to JSON
    ...

  - name: Update Google Sheets
    id: update-google-sheets
    uses: jroehl/gsheet.action@v2.1.1
    with:
      spreadsheetId: ${{ vars.GOOGLE_SPREADSHEET_ID }}
      commands: ... appendData ...
    env:
      ...
```

GHA по подсчету Lead Time: use

```yaml
steps:
  ...
  - name: Convert CSV to JSON
    ...

  - name: Update Google Sheets
    id: update-google-sheets
    uses: jroehl/gsheet.action@v2.1.1
    with:
      spreadsheetId: ${{ vars.GOOGLE_SPREADSHEET_ID }}
      commands: ... appendData ...
    env:
      ...
```

Template

- Вроде не сложно, но много мелких деталей



Template

- Вроде не сложно, но много мелких деталей
- Я сделал прототип

  https://github.com/makzimi/kotlin-github-action

Template: App.kt

```kotlin
suspend fun main() {
    val input = buildGHAInput()

    try {
        group("Action body") {
            val output = runAction(input = input)

            if (output.success) {
                setOutput(Result.value, output.result)
            } else {
                setFailed(output.errorText.orEmpty())
            }
        }
    } catch (e: Exception) {
        setFailed("Error while performing GitHub Action: ${e.message}")
    }
}
```

Template: App.kt

```kotlin
suspend fun main() {
    val input = buildGHAInput()

    try {
        group("Action body") {
            val output = runAction(input = input)

            if (output.success) {
                setOutput(Result.value, output.result)
            } else {
                setFailed(output.errorText.orEmpty())
            }
        }
    } catch (e: Exception) {
        setFailed("Error while performing GitHub Action: ${e.message}")
    }
}
```

Template: App.kt

```kotlin
suspend fun main() {
    val input = buildGHAInput()

    try {
        group("Action body") {
            val output = runAction(input = input)

            if (output.success) {
                setOutput(Result.value, output.result)
            } else {
                setFailed(output.errorText.orEmpty())
            }
        }
    } catch (e: Exception) {
        setFailed("Error while performing GitHub Action: ${e.message}")
    }
}
```

Template: App.kt

```
suspend fun main() {
    val input = buildGHAInput()

    try {
        group("Action body") {
            val output = runAction(input = input)

            if (output.success) {
                setOutput(Result.value, output.result)
            } else {
                setFailed(output.errorText.orEmpty())
            }
        }
    } catch (e: Exception) {
        setFailed("Error while performing GitHub Action: ${e.message}")
    }
}
```

Template: структура

```
./
├── ncc
├── webpack
└── src
    └── jsMain
```

Template: структура

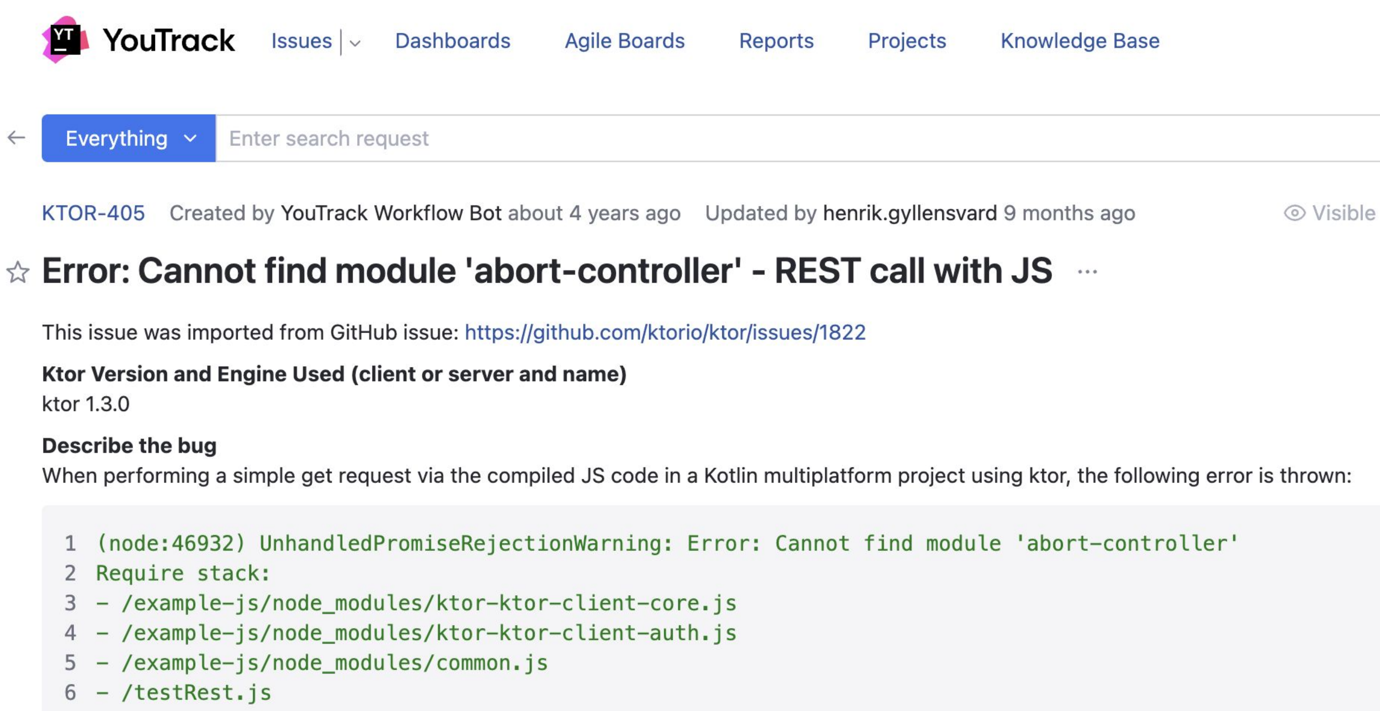
```
./
├── ncc
├── webpack ❓
└── src
    └── jsMain
```

Template: Webpack

```
https://api.github.com/repos/[...]/[...]/git/refs/tags
failed with exception: Error: Cannot find module 'abort-controller' |
  Require stack: | -
    ./dist/index.js
```

Template: Webpack

https://youtrack.jetbrains.com/issue/KTOR-405



KTOR-405 Created by YouTrack Workflow Bot about 4 years ago Updated by henrik.gyllensvard 9 months ago

Error: Cannot find module 'abort-controller' - REST call with JS

This issue was imported from GitHub issue: https://github.com/ktorio/ktor/issues/1822

Ktor Version and Engine Used (client or server and name)
ktor 1.3.0

Describe the bug
When performing a simple get request via the compiled JS code in a Kotlin multiplatform project using ktor, the following error is thrown:

```
(node:46932) UnhandledPromiseRejectionWarning: Error: Cannot find module 'abort-controller'
Require stack:
- /example-js/node_modules/ktor-ktor-client-core.js
- /example-js/node_modules/ktor-ktor-client-auth.js
- /example-js/node_modules/common.js
- /testRest.js
```

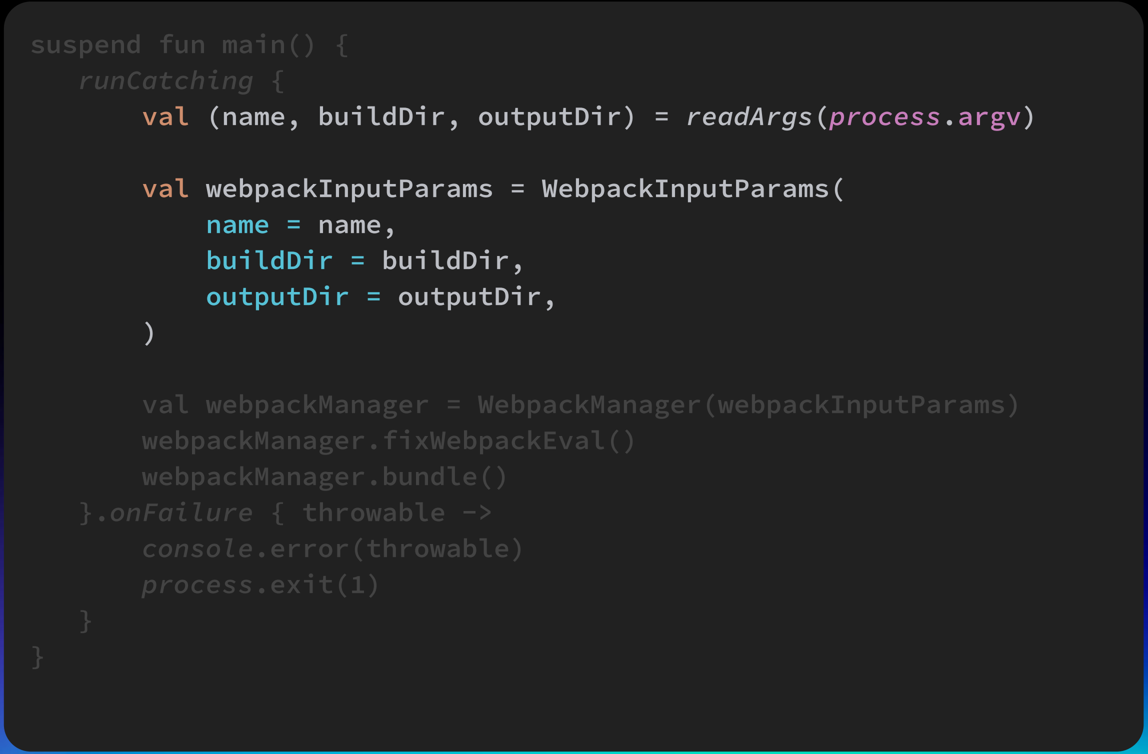
Template: Webpack

```
private fun WebpackInputParams.toWebpackConfig(): WebpackConfig {
    return WebpackConfig(
        projectName = name,
        inputFilePath = "$buildDir/js/packages/$name/kotlin/$name.js",
        outputDirPath = outputDir,
        outputFileName = "index.js",
        modules = listOf(...),
        aliases = mapOf(
            "node-fetch$" to "node-fetch/lib/index.js",
            "abort-controller$" to "abort-controller/dist/abort-controller.js",
        )
    )
}

...

if (content.contains("eval('require')")) {
    val fixedContent = content.replace("eval('require')", "require")
    writeFileSync(path, fixedContent)
}
```

Template: Webpack

```kotlin
suspend fun main() {
   runCatching {
       val (name, buildDir, outputDir) = readArgs(process.argv)

       val webpackInputParams = WebpackInputParams(
           name = name,
           buildDir = buildDir,
           outputDir = outputDir,
       )

       val webpackManager = WebpackManager(webpackInputParams)
       webpackManager.fixWebpackEval()
       webpackManager.bundle()
   }.onFailure { throwable ->
       console.error(throwable)
       process.exit(1)
   }
}
```

Template: Webpack



```
suspend fun main() {
    runCatching {
        val (name, buildDir, outputDir) = readArgs(process.argv)

        val webpackInputParams = WebpackInputParams(
            name = name,
            buildDir = buildDir,
            outputDir = outputDir,
        )

        val webpackManager = WebpackManager(webpackInputParams)
        webpackManager.fixWebpackEval()
        webpackManager.bundle()
    }.onFailure { throwable ->
        console.error(throwable)
        process.exit(1)
    }
}
```

Template: Webpack

```
suspend fun main() {
   runCatching {
       val (name, buildDir, outputDir) = readArgs(process.argv)

       val webpackInputParams = WebpackInputParams(
           name = name,
           buildDir = buildDir,
           outputDir = outputDir,
       )

       val webpackManager = WebpackManager(webpackInputParams)
       webpackManager.fixWebpackEval()
       webpackManager.bundle()
   }.onFailure { throwable ->
       console.error(throwable)
       process.exit(1)
   }
}
```

Template: запускаем 🚀

```
./gradlew build :ncc:jsNodeProductionRun

./gradlew build :webpack:jsNodeProductionRun
```

Выводы

Выводы

- В Marketplace есть много, но не всё

Выводы

- В Marketplace есть много, но не всё
- **Написать кастомный GHA — просто**

Выводы

- В Marketplace есть много, но не всё
- Написать кастомный GHA — просто
- **Кажется Docker проще, но Kotlin/JS не сложнее**

Выводы

- В Marketplace есть много, но не всё
- Написать кастомный GHA — просто
- Кажется Docker проще, но Kotlin/JS не сложнее
- **Пользуйтесь Template**

Kotlin in GitHub Actions

Макс Качинкин
Dodo Engineering, Android Tech Lead

Спасибо за внимание!

ТГ: “Мобильное Чтиво”
@mobilefiction