Coverage

без фрустраций

Придерживаемся плана

- Глоссарий
- Контекст (где, как, почему)
- Что не работает
- Что работает
- Пишем код
- Интегрируемся с CI
- Визуализация отчетов
- Инсайды

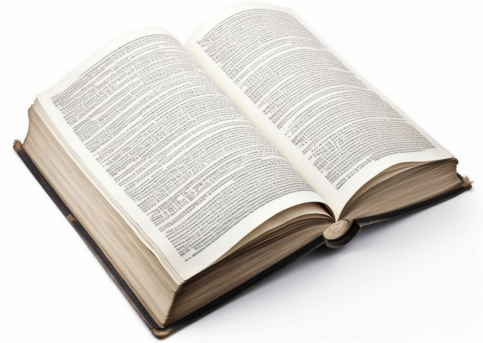
Глоссарий: Тесты

1. Модульные тесты
2. Интеграционные тесты
3. Функциональные тесты
4. Сквозные тесты
5. Приемочное тестирование
6. Тестирование производительности
7. Smoke-тестирование
8. …* тестирование

Глоссарий: Тесты

1. Модульные тесты - **UNIT**
2. Интеграционные тесты - INTEGRATION
3. Функциональные тесты - INTEGRATION
4. Сквозные тесты - INTEGRATION
5. Приемочное тестирование - INTEGRATION
6. Тестирование производительности - N/A
7. Smoke-тестирование - INTEGRATION

Глоссарий



- **unit** тест
  - тестирует логику функций, классов
  - snapshot тестирование React компонента

- **integration** тест
  - тестирует поведение группы компонентов (events, network, hooks)
  - тестирует какой-то функционал приложения (логин пользователя)

Про измерение тестового покрытия

Code Coverage -

- не панацея
- не заменяет осмысленное тестирование
- не говорит о безопасности кода

Про измерение тестового покрытия

Code Coverage -

- говорит нам о том, какие части кода выполняются в рамках тестирования
- уменьшает вероятность возникновения критических ошибок
- помогает увидеть корнер-кейсы

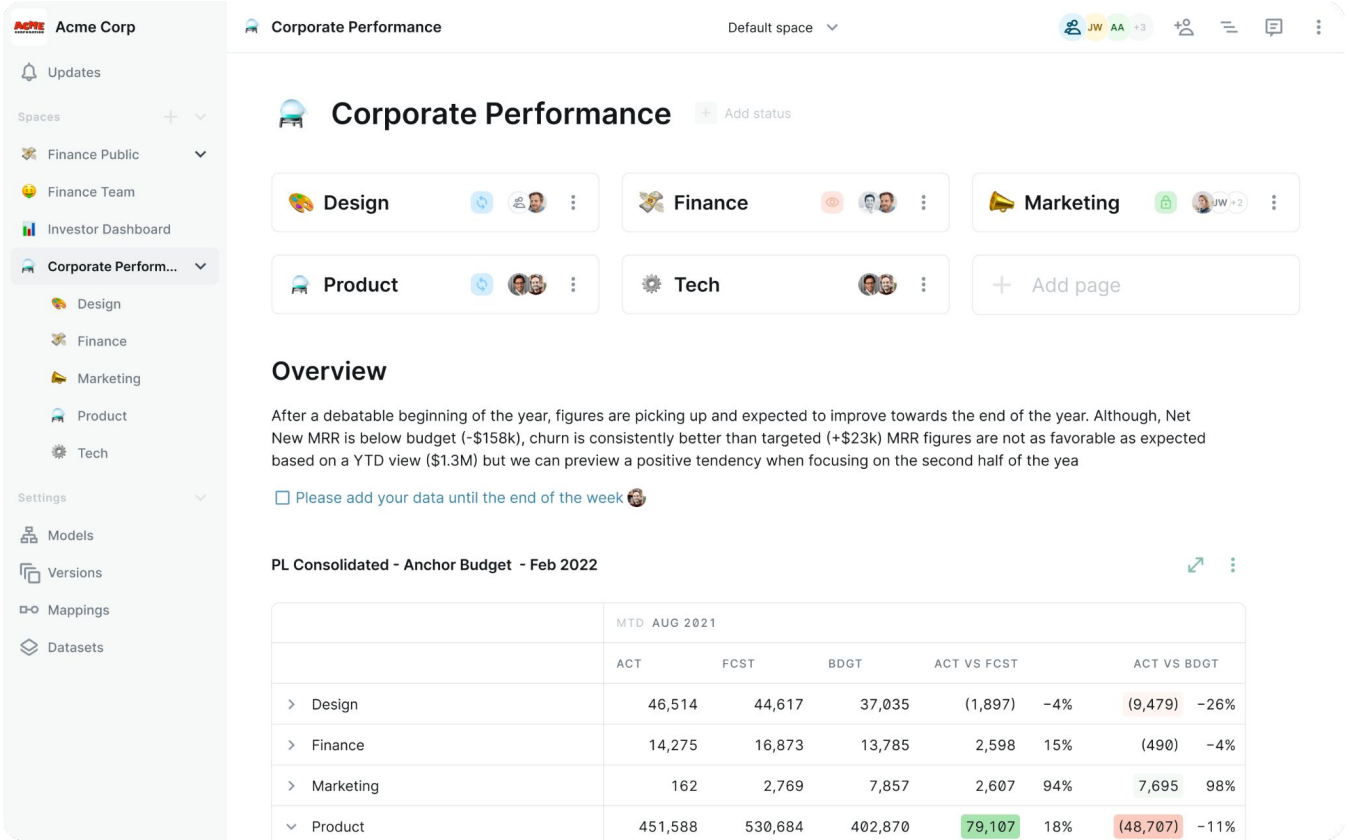
Тестовое покрытие в интеграционных тестах

- Интеграционный тест заходит в **80%** строк кода, охватывая большое количество файлов, тем самым экономит время на написание простых **unit** тестов.

- **20%** сложных кейсов, проще покрывать **unit** тестами

Тайминг

June - Dec 2022



О проекте

Abacum

React

Redux

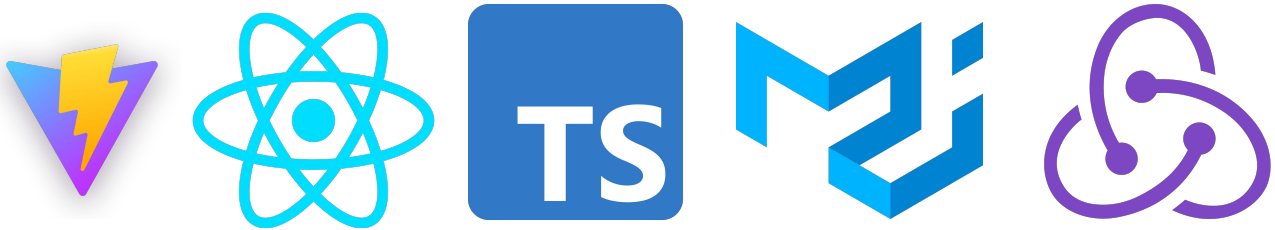
Vite

TypeScript

Material UI

/src

2 500 файлов
600 000 строк кода



Abacum

Зоопарк тестов

- Jest

  unit, integration

- Cypress
- Playwright

Что с покрытием?

Abacum

- Кавереджом покрывается только **Jest** часть
- **Playwright** & **Cypress** без покрытия
- Бесплатный кавередж репорт экшн для каждого пулл-реквеста
  github.com/vebr/jest-lcov-reporter

Предпосылки

- Быстрорастущий стартап
- Отсутствие QA специалистов
- Новый функционал должен хорошо работать

- Команда должна видеть что функционал протестирован

- Я не люблю суету
- Не хочется тратить время на багфиксы и дебаггинг

Проблема

- разработчики **страдают** когда пишут интеграционные тесты **React** компонентов с помощью **Jest**.

```
it('can render and update a counter', () => {
  // Test first render and componentDidMount
  act(() => {
    ReactDOM.createRoot(container).render(<Counter />);
  });
  const button = container.querySelector('button');
  const label = container.querySelector('p');
  expect(label.textContent).toBe('You clicked 0 times');
  expect(document.title).toBe('You clicked 0 times');

  // Test second render and componentDidUpdate
  act(() => {
    button.dispatchEvent(new MouseEvent('click', {bubbles: true}));
  });
  expect(label.textContent).toBe('You clicked 1 times');
  expect(document.title).toBe('You clicked 1 times');
});
```

Проблема

- разработчики **страдают** когда пишут интеграционные тесты **React** компонентов с помощью **Jest**.

- Очень сложно дебажить
- Нужно писать кучу кода / обёртки для того чтобы завёлся стейт
- Не потрогать в браузере
- Начинает страдать доступность и семантика тестов
- Трудно поддерживать тесты

Что я хотел / не хотел

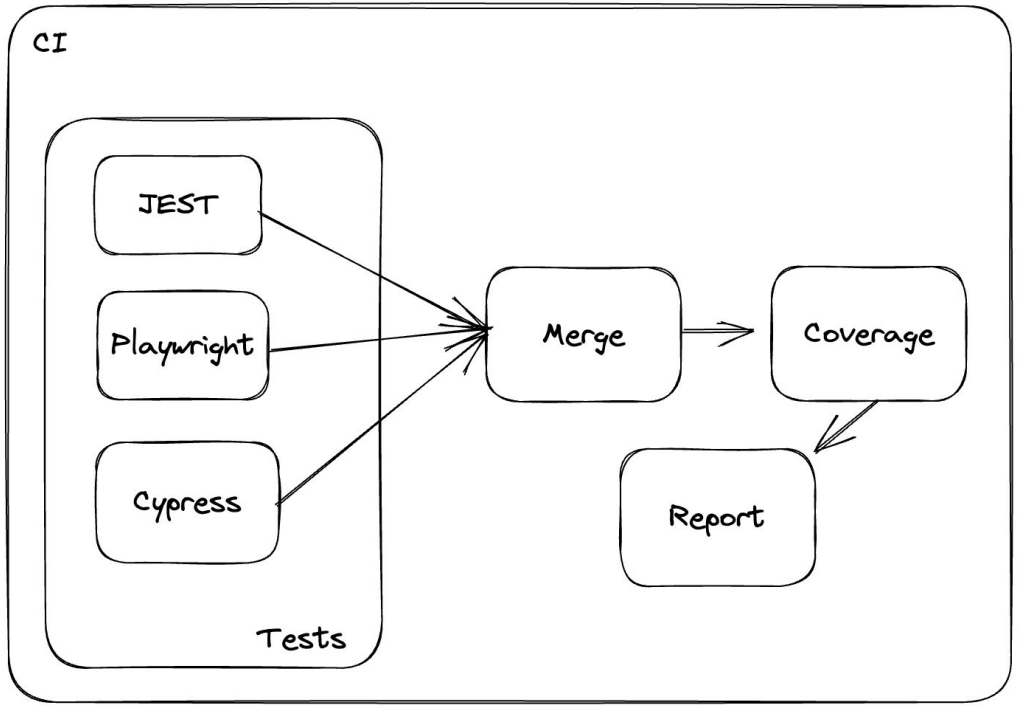
Abacum

- Я не хотел **писать** интеграционные **тесты** React компонентов с **Jest**
- **Не** хотелось **снижать** общий код-кавередж компании (считающийся через **Jest**) новыми фичами
- Улучшить **прозрачность** тестового **покрытия** кодовой базы (часть кода уже написана на **playwright** и **cypress**)

План

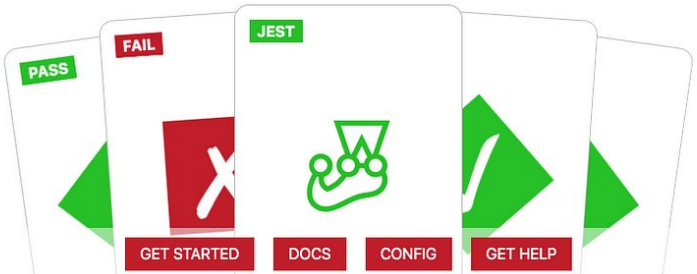
Что нужно было сделать

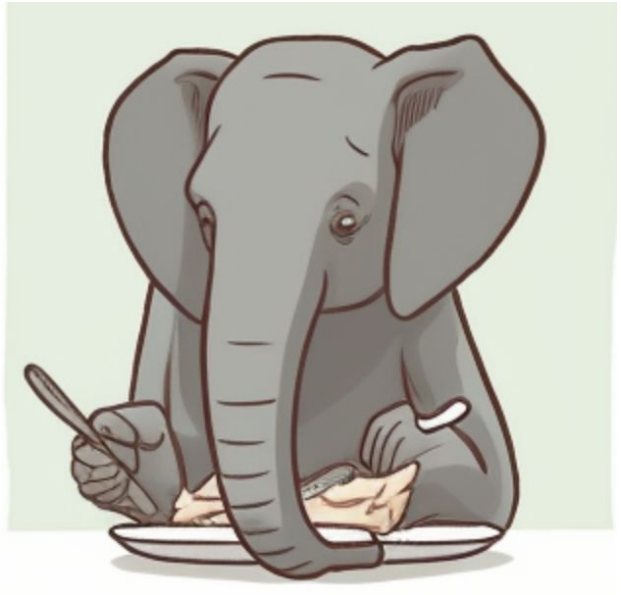
- Запустить CI тесты Cypress
- Запустить CI тесты Jest
- Запустить CI тесты Playwright
- Собрать результаты
- Смержить их
- Зарепортать кавередж



Attempt #0 - Едим слона по частям: Jest

- базируется на istanbul babel плагине
- под капотом сам добавляет плагин
- уже работает - не трогаем





Attempt #0 - Едим слона по частям: Cypress

[@cypress/code-coverage]

- базируется на istanbul babel плагине

Attempt #0 - Едим слона по частям: Playwright

playwright-test-coverage

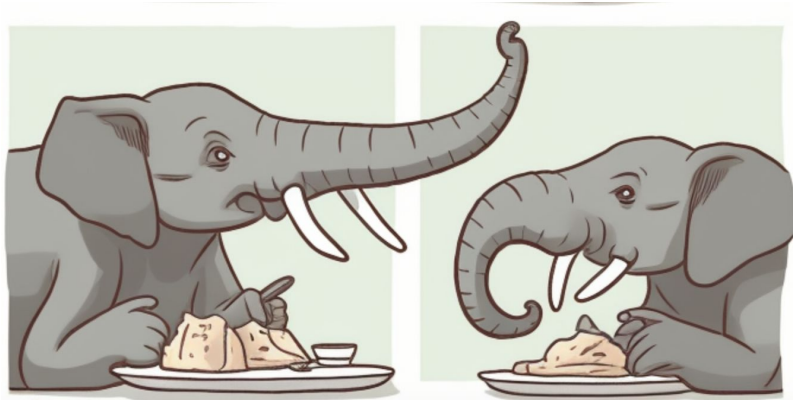
```
// tests/foo.spec.js
const { test, expect } = require("playwright-test-coverage");

// Use test and expect as usual
test("basic test", async ({ page }) => {
  await page.goto("https://playwright.dev/");
  const title = page.locator(".navbar__inner .navbar__title");
  await expect(title).toHaveText("Playwright");
});
```

Attempt #0 - Едим слона по частям: Babel plugin istanbul

- наш production target - **es2020**
- последний релиз был **2 года назад**
- под капотом использует **istanbul-lib-instrument**
- github.com/istanbuljs/istanbuljs/issues/614

- поддерживает только синтаксис **es6**



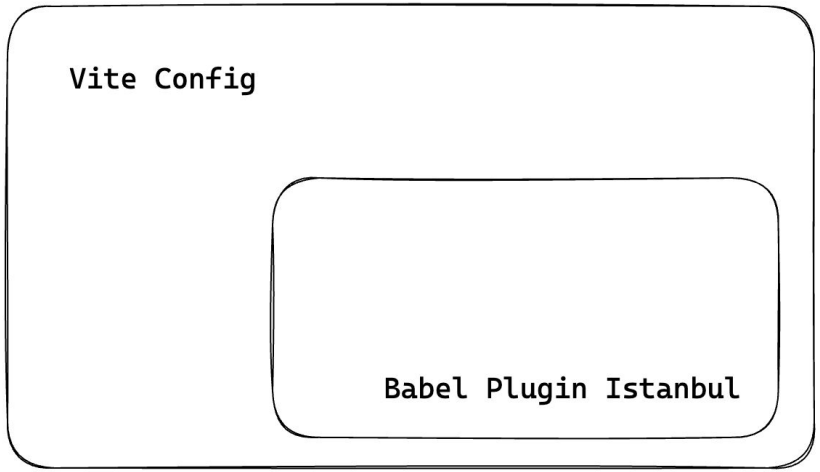
Attempt #0 - Едим слона по частям: Babel plugin istanbul

Что сделал:

- **понизить** compilation target до **es5**
- добавить плагин в **vite build** pipeline

Что получилось:

- иногда CI сборка **зависала**
- **пустое** покрытие для ряда тестов



Attempt #0 - Едим слона по частям: Babel plugin istanbul

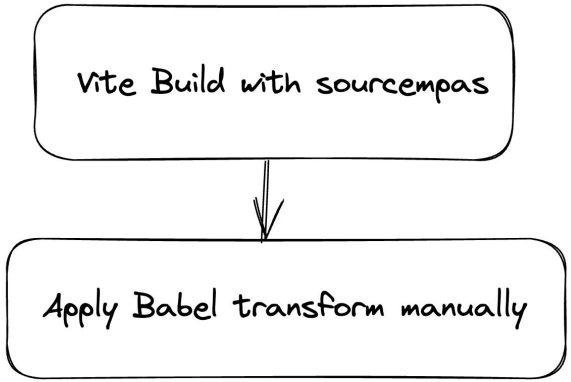
Что делаем:

- собираем код с **sourcemaps**
- применяем **istanbul-babel** plugin

gist.github.com/lifeart/c9c3e8a339dc57210cfdb8da5fbf99ac

Результат:

- пустые отчёты о покрытии для ряда тестов



Attempt #0 - Результаты

Istanbul - нам **не** подходит

- требует **понижения** compilation target
- сильно **усложняет** процесс сборки
- усложняет процесс **дебага**
- есть **вопросы** с поддержкой

Едим слона по частям: Рефлексируем

confirmation bias

Рефлексия: Babel plugin istambul

medium.com/@kuldeepkeshwar/code-coverage-directly-from-v8-3a4e86c2cdba

- They need to play catch up with the evolving JavaScript language and sometimes lag behind on language features; here's a pull request adding object-spread syntax to Istanbul from September 2017, several months after the feature became widely available.
- Introducing counters on every line of an application significantly impacts performance (Node.js' own test suite runs about 300% slower when instrumented).
- It's difficult to insert counters into a codebase without accidentally changing its behavior.

Рефлексия: Babel plugin istambul

```
let a = 23;
if (a !== 23) {
  alert("holy");
} else {
  alert("js");
}
```

```
__cov_a4da048374578.s["1"]++;
let a = 23;
__cov_a4da048374578.s["2"]++;
if (a !== 23) {
  __cov_a4da048374578.b["1"][0]++;
  __cov_a4da048374578.s["3"]++;
  alert("holy");
} else {
  __cov_a4da048374578.b["1"][1]++;
  __cov_a4da048374578.s["4"]++;
  alert("js");
}
```

Рефлексия: Почему мы ушли от Cypress

- Свой DSL
- Неудобно кастомизировать
- Непонятно где и как ждать (непрозрачный await)
- Медленный раннер
- Неприятно дебажить
- twitter.com/housecor/status/1597954687705104385

Cory House
@housecor · **Follow**

I'm a big Cypress fan, so I'm shocked to say this: I just switched to @playwrightweb.

Here are 16 reasons I switched:

1. WAY Faster. ~2X faster with 1 core. ~6x faster with multiple cores (uses multiple workers)

2. Tests multiple browsers in parallel.

1/5

Рефлексия: Playwright

playwright-test-coverage

- базируется на istanbul babel плагине
- реекспортирует playwright импорты
- не отключается
- работает не явно

```
// tests/foo.spec.js
const { test, expect } = require("playwright-test-coverage");

// Use test and expect as usual
test("basic test", async ({ page }) => {
  await page.goto("https://playwright.dev/");
  const title = page.locator(".navbar__inner .navbar__title");
  await expect(title).toHaveText("Playwright");
});
```

Next Attempt

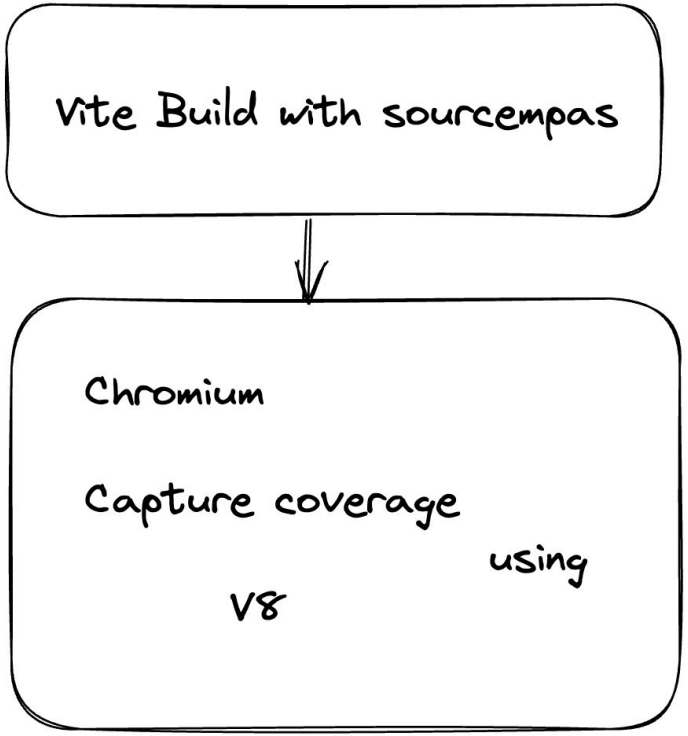
Attempt #1 - Едим слона по частям: v8

- Собирает реальный coverage
- Не требователен к версии compilation target
- Можно отключить
- Не устареет (поддерживается разработчиками браузера)

Attempt #1 - Едим слона по частям: v8

Что делаем:

- сгенерировать билд с sourcemaps
- запустить код в playwright



Attempt #1 - Едим слона по частям: v8

Что делаем:

- собираем проект с sourcemaps

vite.config.ts

```
{
    build: {
      sourcemap: true
    }
}
```

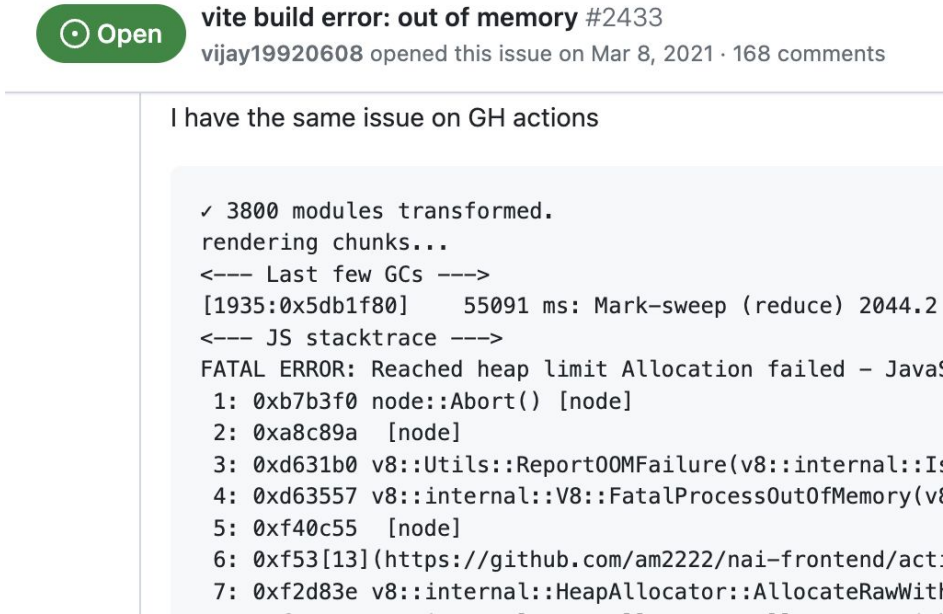
Attempt #1 - Едим слона по частям: v8

Что делаем:

- собираем проект с sourcemaps

github.com/vitejs/vite/issues/2433

```
NODE_OPTIONS=--max-old-space-size=16000
```

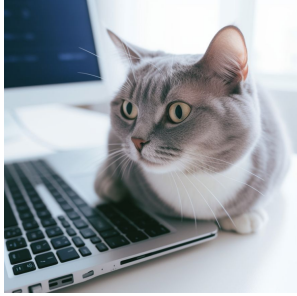


Playwright integration: Requirements

Требования:

- Кавередж должен явно включаться для каждого тест-файла (performance)
- Должна быть возможность выключать Coverage глобально
- Интеграция должна быть простой

Playwright integration: implementation



Имплементация:

```
await page.coverage.startJSCoverage(options);

                ТУТ ЛОГИКА ТЕСТА

await page.coverage.stopJSCoverage();
```

Playwright integration: implementation

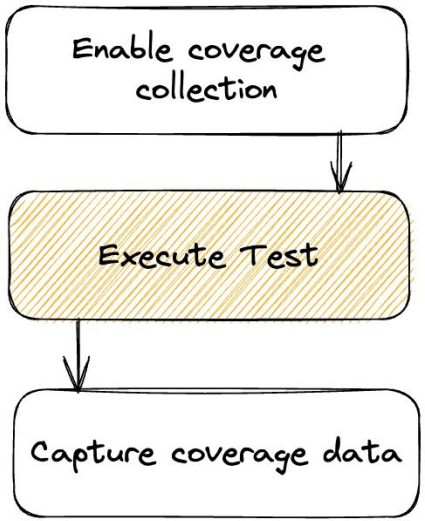
```
const jsCoverage = await page.coverage.stopJSCoverage();

for (const entry of jsCoverage) {
    const { functions, source, url } = entry;
    const fName = filePathFromUrl(url);
    const converter = v8toIstanbul(fName, 0, { source });
    await converter.load();
    converter.applyCoverage(functions);
    const result = converter.toIstanbul();
    saveCoverage(JSON.stringify(result));
}
```

Playwright integration: implementation

Имплементация:

github.com/lifeart/demo-ember-vite/blob/master/e2e/utils/index.ts




```
import { test, expect } from '@playwright/test';

import { captureCoverage } from './../utils/index.ts';

captureCoverage(test);

test.describe('Ember-Bootstrap route', () => {
  test('it loadable', async ({ page }) => {
    await page.goto('http://localhost:4200');
    await expect(page).not.toHaveTitle(/Bootstrap/);
    await page.click('a[href="/bootstrap"]');
    // Expect a title "to contain" a substring.
    await expect(page).toHaveTitle(/Bootstrap/);
  });
});
```

Playwright integration: details

Каждый запуск создаёт **n** файлов кавереджа, playwright может выполняться на нескольких раннерах, в каждом из которых путь до корня проекта будет разный

Нам нужно каким-то способом объединить эти файлы и исправить пути.

Playwright integration: paths

Исправляем пути:

github.com/lifeart/demo-ember-vite/blob/master/scripts/fix-paths-in-coverage-reports.cjs


```
"/home/runner/work/demo-ember-vite/demo-ember-vite/src/helpers/memory-usage.ts": {
  "path": "/home/runner/work/demo-ember-vite/demo-ember-vite/src/helpers/memory-usage.ts",
  "all": false,
  "statementMap": {
    "0": {
      "start": { "line": 1, "column": 0 },
      "end": { "line": 1, "column": 45 }
    },
    "1": {
      "start": { "line": 2, "column": 0 },
      "end": { "line": 2, "column": 0 }
    },
    "2": {
      "start": { "line": 3, "column": 0 },
      "end": { "line": 3, "column": 49 }
    },
```

Объединяем кавередж

- Формат
- Инструмент

Формат

LCOV / JSON?

LCOV

github.com/mweibel/lcov-result-merger

> When you have multiple test suites for the same application yo have the code coverage across all test suites.

```
TN:
SF:xxxx
FN: 2,func
....
FNF:3
FNH:2
BRDA:127,1,0,204
....
BRF:10
....
DA:122,148
....
LF:19
LH:13
end_of_record
```

LCOV

github.com/mweibel/lcov-result-merger

- **Empty file when merging lcov.files**
  #55 opened on Feb 3 by migheorghe
- **Branch mismatch**
  #42 opened on Aug 16, 2018 by rinick
- **Merge policy**
  #41 opened on Jul 13, 2018 by merlosy
- **Check if we should implement merging of FN, FNDA, FNF, FNH, BRF, BRH, LF, LH** enhancement
  #14 opened on May 4, 2016 by mweibel

LCOV

https://wiki.documentfoundation.org/Development/Lcov#Combine_lcov_tracefiles

Combine lcov tracefiles

Now combine the 'before tests' and 'after tests' snapshots.

```
lcov --add-tracefile /tmp/libreoffice_base.info --add-tracefile /tmp/libreoffice_test.info \
--output-file /tmp/libreoffice_total.info
```

LCOV

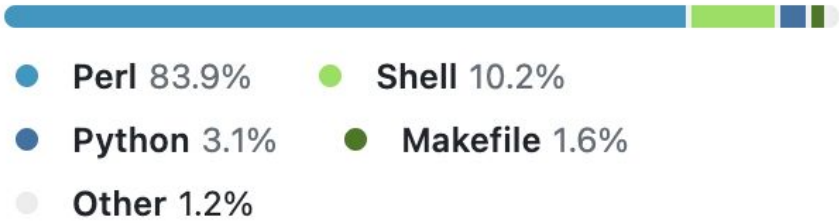
wiki.documentfoundation.org/Development/Lcov#Combine_lcov_tracefiles

Combine lcov tracefiles

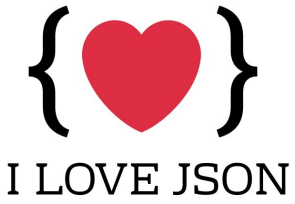
Now combine the 'before tests' and 'after tests' snapshots.

```
lcov --add-tracefile /tmp/libreoffice_base.info --add-tracefile /tmp/libreoffice_test.info \
--output-file /tmp/libreoffice_total.info
```

github.com/linux-test-project/lcov



JSON




```
"fnMap": {
  "0": {
    "name": "MemoryUsage",
    "decl": {
      "start": { "line": 7, "column": 2 },
      "end": { "line": 15, "column": 3 }
    },
    "loc": {
      "start": { "line": 7, "column": 2 },
      "end": { "line": 15, "column": 3 }
    },
    "line": 7
  },
```

Инструмент

nyc

Istanbul Command Line interface

Инструмент

nyc merge {input_folder} {output_file}

nyc merge .nyc_output ./coverage/playwright-final.json

Генерируем репорт

nyc report --reporter=**html** --reporter=**lcov** **--temp-dir** **./coverage**

CI

Как ждать?

Как загружать данные?

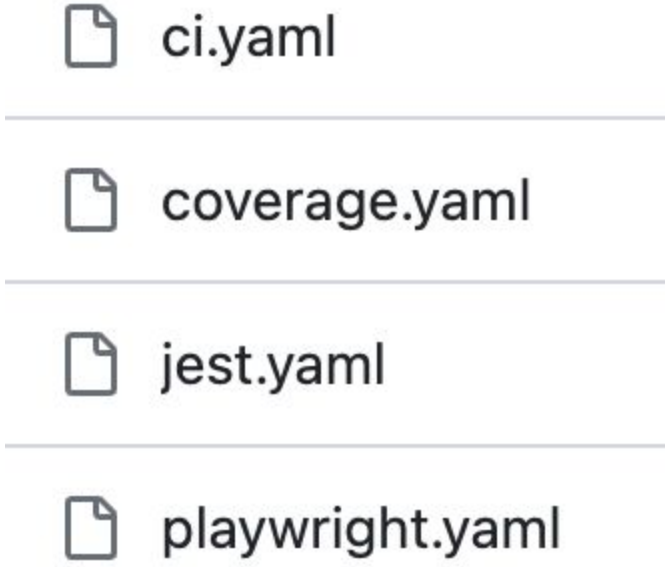
Как генерировать отчёты?

CI

Как ждать?

Как загружать данные?

Как генерировать отчёты?



CI

Как ждать?

fountainhead/action-wait-for-check

A GitHub Action that waits for another Check Run to have completed

CI

fountainhead/action-wait-for-check

playwright.yaml

```yaml
jobs:
  test_playwright:
    name: Playwright tests queue
    timeout-minutes: 60
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v3
    - uses: actions/setup-node@v3
      with:
        node-version: 16
```

coverage.yaml

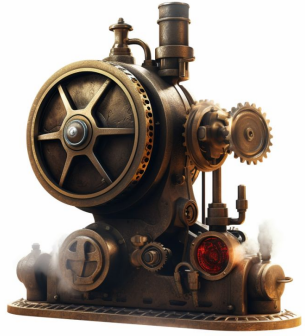
```yaml
- name: Wait for Playwright tests
  continue-on-error: false
  uses: fountainhead/action-wait-for-check@v1.1.0
  id: wait-for-playwright-tests
  with:
    token: ${{ secrets.GITHUB_TOKEN }}
    checkName: Playwright tests queue
    timeoutSeconds: 600
    intervalSeconds: 30
    ref: ${{ github.event.pull_request.head.sha || github.sha }}
```

CI

Как загружать данные?

dawidd6/action-download-artifact

A GitHub Action to download an artifact associated with given workflow and commit or other criteria



CI

dawidd6/action-download-artifact

playwright.yaml

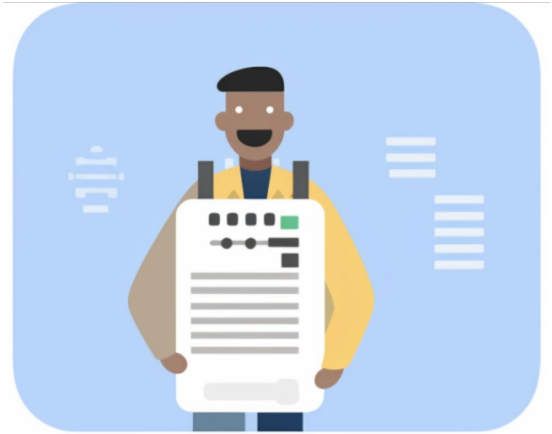
```yaml
- name: Upload Playwright coverage report
  uses: actions/upload-artifact@v3
  with:
    name: playwright-coverage
    path: ./coverage/playwright-final.json
    retention-days: 30
```

coverage.yaml

```yaml
- name: Download Playwright test coverage artifacts
  uses: dawidd6/action-download-artifact@v2
  with:
    github_token: ${{secrets.GITHUB_TOKEN}}
    path: ./artifacts
    workflow: playwright.yaml
    workflow_conclusion: success
    name: playwright-coverage
    pr: ${{github.event.pull_request.number}}
```

CI

Как генерировать отчёты?



CI: Report

- исправить пути
- смержить кавередж
- сгенерировать отчёт
- отобразить результаты

CI: Report

- исправить пути

```
- name: Fix paths inside coverage reports
  run: node ./scripts/fix-paths-in-coverage-reports.cjs
```

CI: Report

- смержить кавередж

```
- name: Merge all reported JSONs into one
  run: node ./node_modules/.bin/nyc merge ./coverage-artifacts ./merged-coverage/coverage.json
```

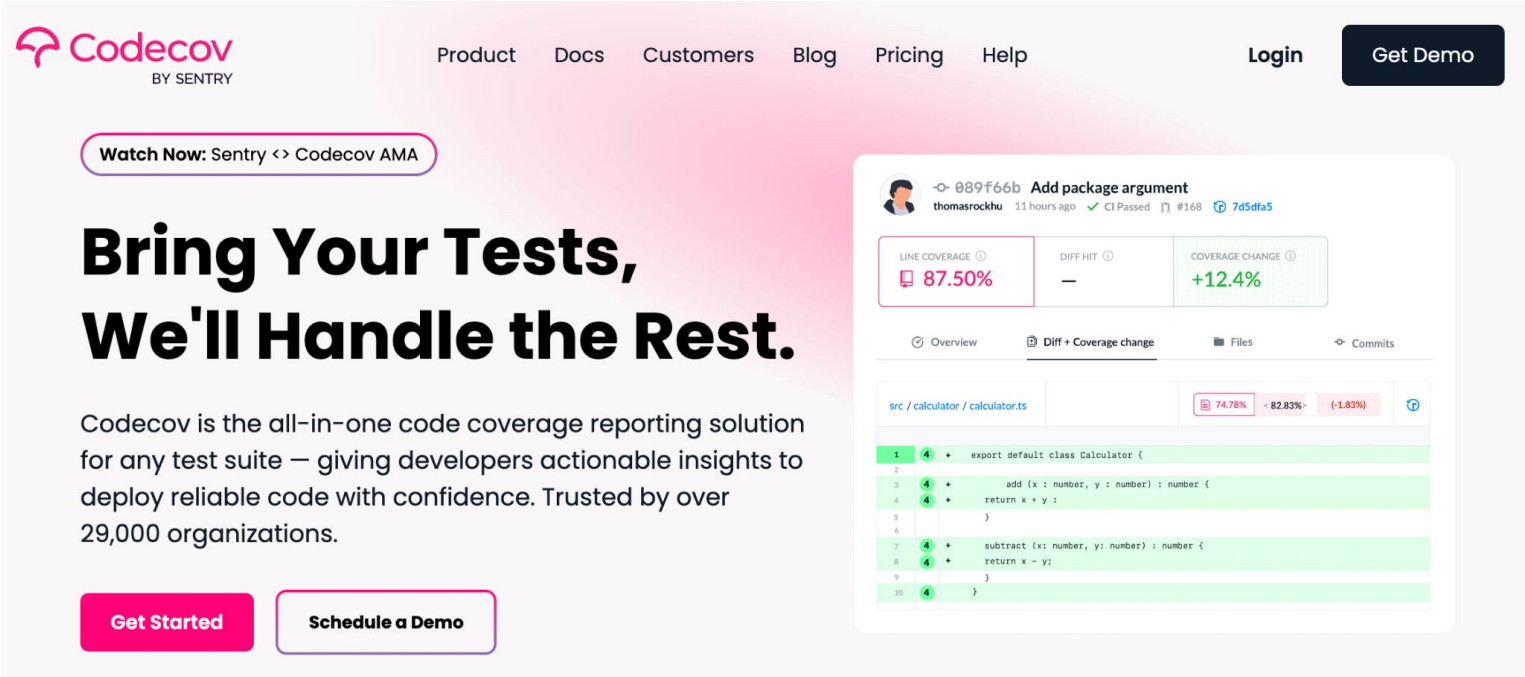
CI

Как генерировать отчёты?

```
- name: Generate report
  run: node ./node_modules/.bin/nyc report --reporter=html --reporter=lcov --temp-dir ./merged-coverage
```

CI

Как отобразить результаты?



CI

Как отобразить результаты?

lifeart/jest-lcov-reporter

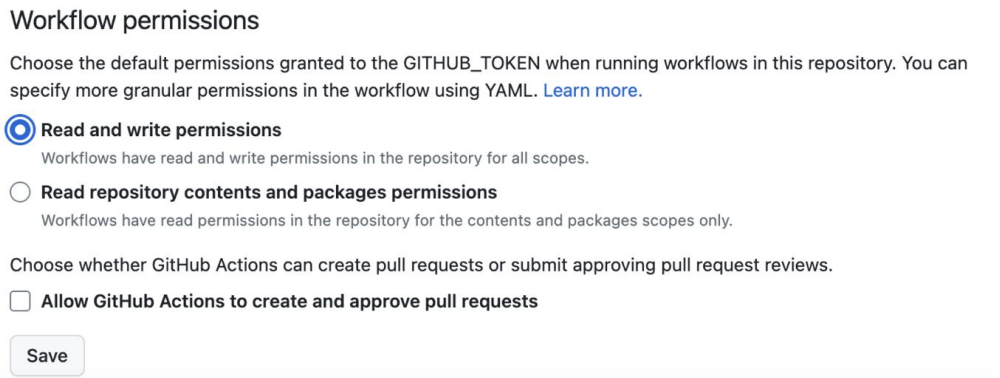
Экшн добавляет комментарий к пулл-реквесту с табличкой, содержащей список файлов и их покрытие тестами.

CI

lifeart/jest-lcov-reporter

workspace→settings→general

https://github.com/actions/first-interaction/issues/10#issuecomment-1475121828



CI

```yaml
- name: Jest Lcov Reporter v2
  uses: lifeart/jest-lcov-reporter@v0.3.6
  with:
    github-token: ${{ secrets.GITHUB_TOKEN }}
    lcov-file: ./coverage/lcov.info
    update-comment: true
    show-changed-files: false
    min-coverage: 90
```

CI

Coverage after merging **test-for-not-found-route** into **master**

95.89%

▼ Coverage Report

File	Lines (%)	Funcs (%)	Uncovered Lines
addons			
ember-simple-auth.ts			12, 16
authenticators			
custom.ts		**66.67**	
config			
🔴 inspector.ts ⬆️ 58.48%	**31.52**	0	80, 81, 82, 83, ...
registry.ts			70, 84
router.ts	98.08		97, 98

Зависимости

nyc

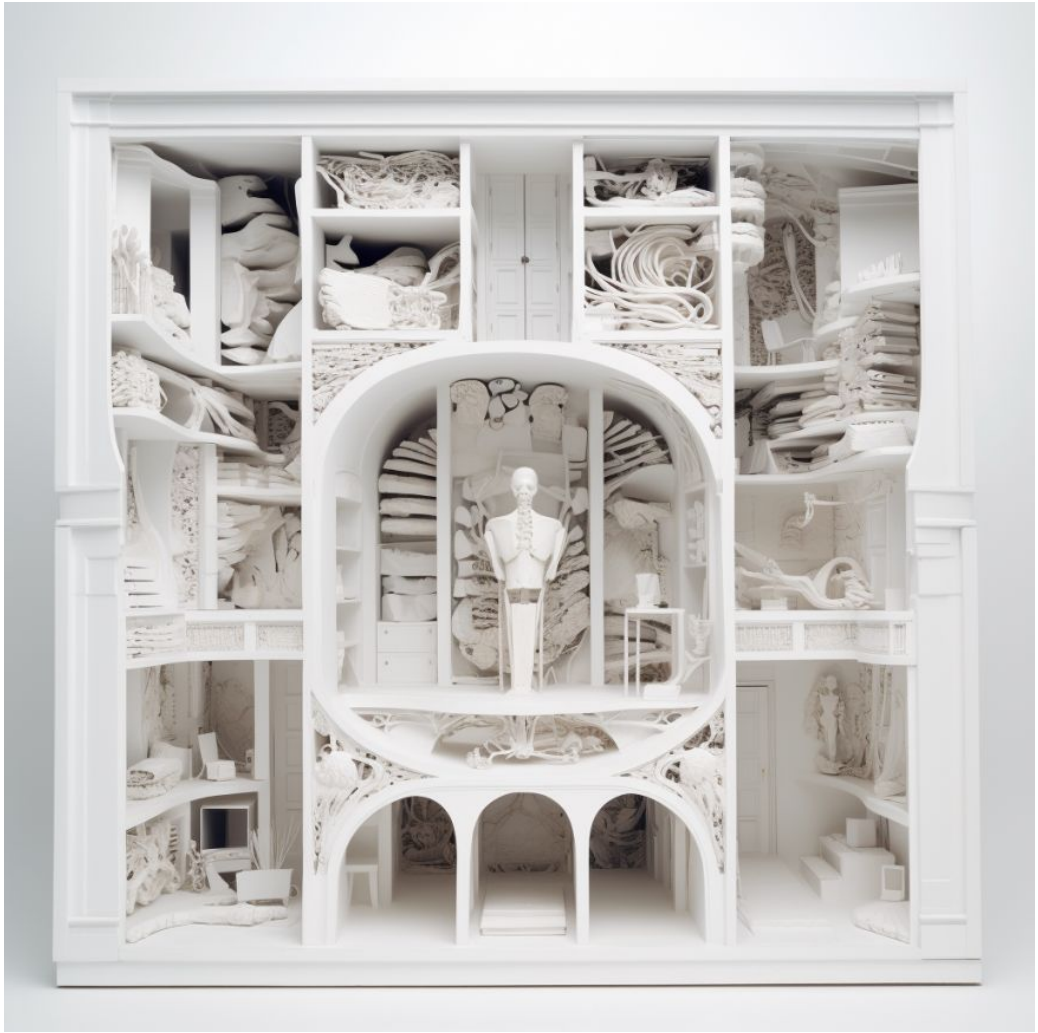
объединяем кавередж

генерируем отчёты

v8-to-istanbul

конвертируем данные **v8** в понятный для **Istambul** формат

Insides



Jest Insides

github.com/aelbore/esbuild-jest - A Jest transformer using esbuild

оказался самым быстрым из того что нам получилось попробовать,

я его даже форкнул,

большинство библиотек (включая реакт), имеют cjs версию, на этом можно сэкономить (скипая трансформ)

Jest Insides

custom transformer,

process - работает по умолчанию

processAsync - работает только для **es** модулей

Jest Insides: Custom transformer

```
// resolve prebuilded runtime helpers
if (filename.includes('@babel/runtime/helpers/esm/')) {
  return {
    code: fs.readFileSync(filename.replace('/esm/', '/'), 'utf-8'),
  };
}
```

Jest Insides: Custom transformer

```
// mock material ui icons
if (filename.includes('@mui/icons-material') && !filename.includes('/utils/')) {
  const iconName = filename.split('@mui/icons-material/')[1].split('/')[0].split('.')[0];
  return {
    code: muiMaterialIconMock(iconName),
    map: null,
  };
}
```

Jest Insides: Custom transformer

```
// if files looks like cjs, likely it's cjs
if (filename.includes('/cjs/') || filename.includes('.cjs.')) {
  return { code: content, map: null };
}
```

Jest Insides: Custom transformer

```
// if file has cjs things, count it as cjs
if (
  inNodeModules &&
  (content.includes('module.exports') || content.includes('exports.default'))
) {
  return { code: content, map: null };
}
```

Jest Insides

custom resolver,

по умолчанию читает всё дерево импортов, поэтому лучше использовать максимально специфичные импорты

MSW Insides

github.com/mswjs/msw

Работает на порядок медленнее в интеграционных тестах Playwright (из-за того, что нужно регистрировать сервис-воркер)

Playwright Insides

eslint-plugin-playwright - отличный eslint плагин, который позволяет не забывать **await**'ы. Связанное issue: https://github.com/playwright-community/eslint-plugin-playwright/issues/79#issuecomment-1218507062

github.com/microsoft/playwright/issues/16605 - TS Compilation problems (import from app namespace)

Включение измерения кавереджа замедляет тесты на ~ **30%**

Запуск playwright с пребилженного кода ускоряет тесты на **~ 10%**

Сёрвинг Gzip ассетов ускоряет тесты на **~ 10%**

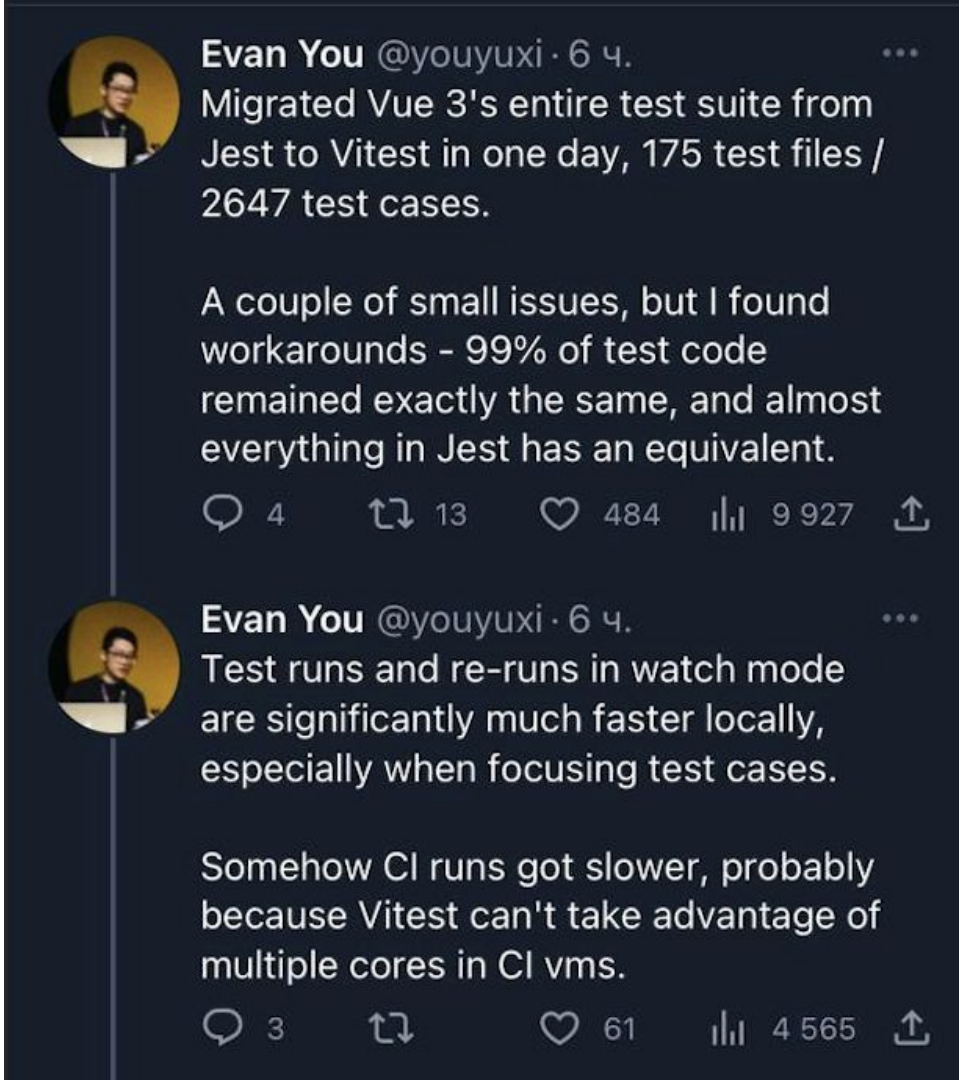
Включение роутинга эндпоинтов в playwright отключает кэширование по умолчанию.

Кэширование можно включить силой - https://github.com/microsoft/playwright/issues/7220#issuecomment-1323709726

Vitest

- Мы проверили, переписали один модуль на Vitest (порядка **250 тестов**), и **локально** оно работало **быстрее**, с оговорками на то, что пришлось отказаться от части моков, потому что Vitest их не поддерживает.
- Когда мы это всё запустили на **CI** - Vitest оказался **медленнее** Jest с включённым код-кавереджом
- Сам Эван Ю говорит что **Vitest** пока **медленнее** на **CI**

Vitest


Evan You @youyuxi · 6 ч.
Migrated Vue 3's entire test suite from Jest to Vitest in one day, 175 test files / 2647 test cases.
A couple of small issues, but I found workarounds - 99% of test code remained exactly the same, and almost everything in Jest has an equivalent.
4
13
484
9 927
Evan You @youyuxi · 6 ч.
Test runs and re-runs in watch mode are significantly much faster locally, especially when focusing test cases.
Somehow CI runs got slower, probably because Vitest can't take advantage of multiple cores in CI vms.
3
61
4 565

Vitest


Evan You @youyuxi · 6 ч
Vitest pro tip: if you have many test files and they share a lot of common modules, tests actually run faster with --no-threads because it avoids repeated module initialization costs. This made Vue 3 unit tests run 6x faster on CI 🤯
5
38
357
33,6 тыс.
Evan You @youyuxi · 6 ч
Thanks @johnsoncodehk for sharing this with me.
Note: this sacrifices test isolation to a degree since a test can mutate a stateful module. Also Vitest team is still actively looking to improve perf with threads enabled.
1
38
8 073

Insides: Cypress

V8

github.com/leftyio/v8-cypress-coverage-plugin

github.com/bahmutov/cypress-native-chrome-code-coverage-example

Результаты

Code coverage продукта: **80.34**%

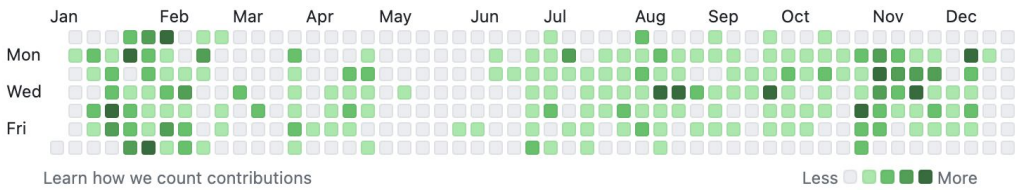
После "включения", добавилось **17**% кавереджа.

Команда:

- довольна

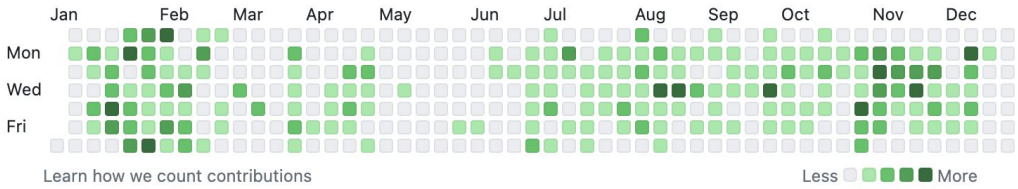
Александр Канунников

twitter.com/**vaier**
github.com/**lifeart**



Александр Канунников

twitter.com/**vaier**
github.com/**lifeart**



Playwright integration: wishes

Как было бы здорово:

Использовать глобальный **beforeAll** / **beforeEach** хук

github.com/microsoft/playwright/issues/9468

Но он не имплементирован,

поэтому нужно придумать свой способ