Налаштування jest-html-reporters для генерації HTML звітів

У цій презентації розглянемо процес налаштування та використання репортера jest-html-reporters для генерації детальних HTML звітів про виконання JavaScript тестів. Цей інструмент дозволяє отримувати зручні для читання звіти з результатами автоматизованих тестів.

Встановлення пакета jest-html-reporters

Команда встановлення

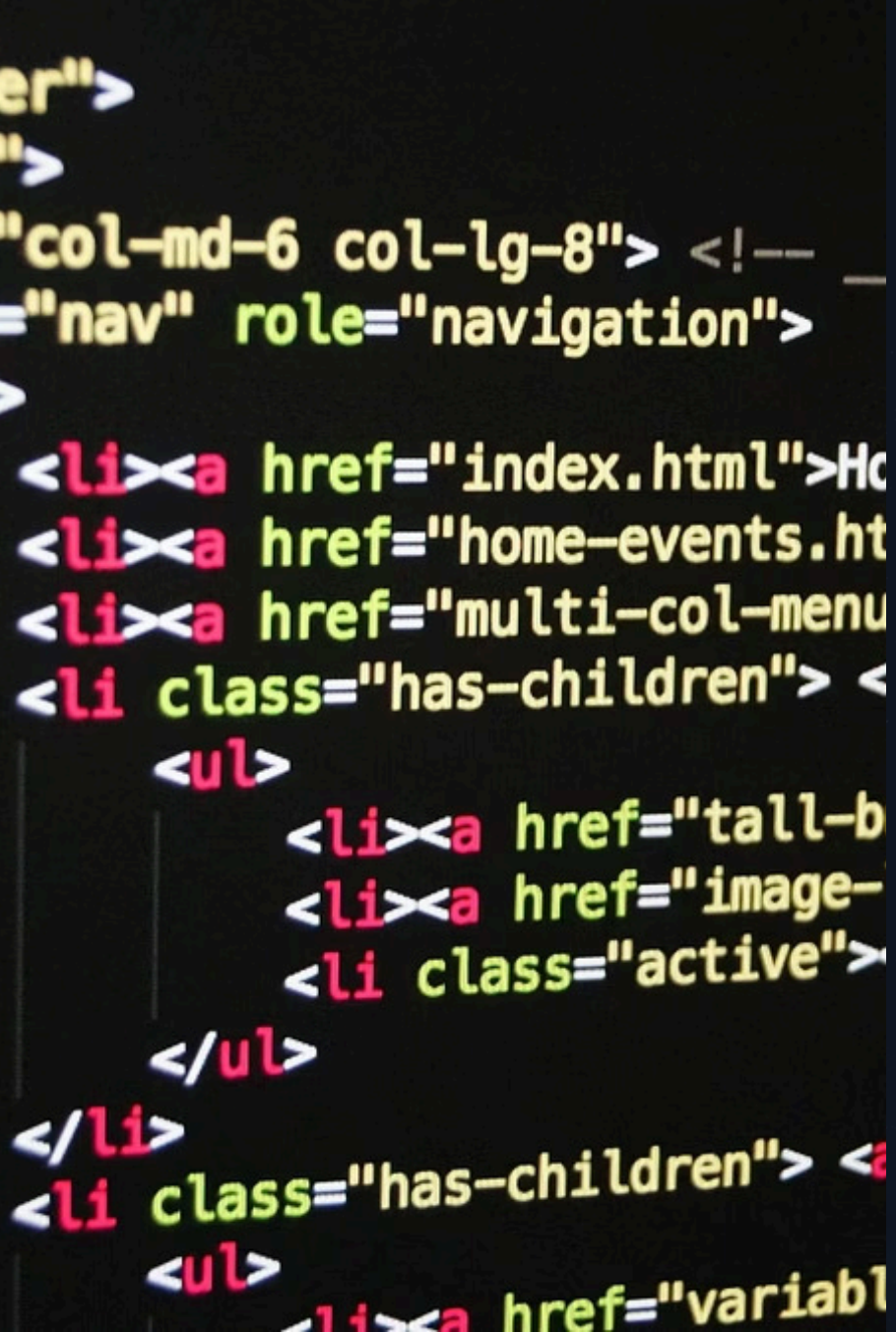
```
npm install jest-html-reporters --save-dev
```

Використовуйте цю команду в терміналі для додавання пакета до залежностей розробки

Перевірка встановлення

Після виконання команди пакет автоматично додається в секцію devDependencies файлу package.json вашого проекту

Важливо використовувати флаг --save-dev, щоб пакет був доданий саме як залежність для розробки, а не для продакшн середовища



Конфігурація репортера в Jest

Базові налаштування

Для активації генерації HTML звітів необхідно додати відповідні параметри в конфігураційний файл Jest. Це може бути jest.config.js або секція jest в package.json.

Репортер додається до масиву reporters, де "default" забезпечує стандартний консольний вивід, а "jest-html-reporters" активує генерацію HTML файлів.

Код конфігурації

```
reporters: [
  "default",
  "jest-html-reporters"
],
```

Конфігурація підключає обидва репортери одночасно для максимальної інформативності

Перевірка налаштувань репортера

Запуск тестів

Виконайте тести стандартною командою, яку використовували раніше для вашого проекту

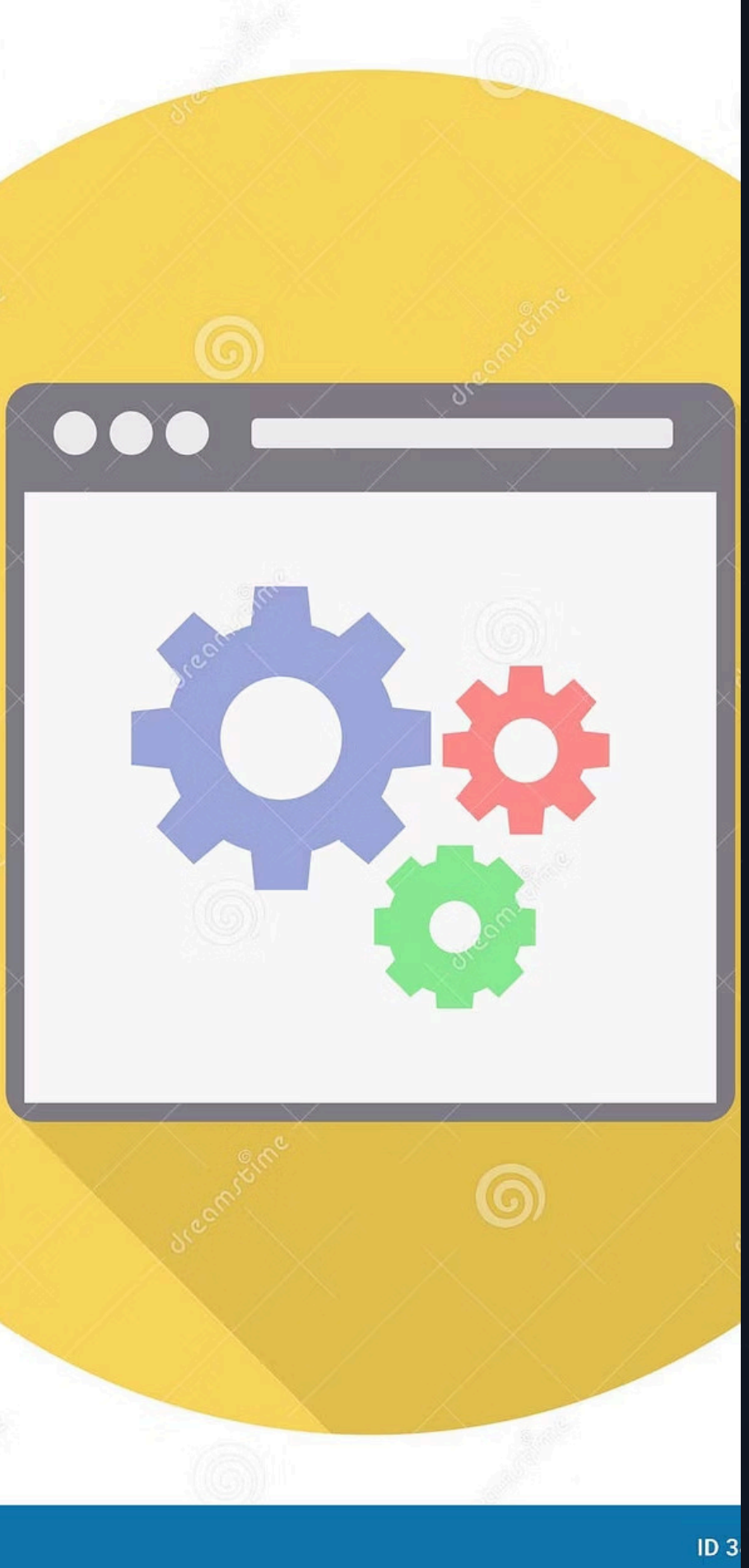
Генерація звіту

Після завершення тестів ви побачите повідомлення про успішну генерацію HTML репорту в консолі

Збереження результатів

Звіт автоматично зберігається у вказаній директорії проекту для подальшого аналізу

Якщо налаштування виконані коректно, ви побачите інформацію про створений файл звіту з вказанням шляху до нього в консольному виводі.



Розширена конфігурація параметрів звітності

Jest-html-reporters надає гнучкі можливості для налаштування параметрів генерації звітів. Ви можете кастомізувати назву файлу, розташування та поведінку після генерації.

publicPath

Визначає директорію для збереження звіту

```
"publicPath": "./html-report"
```

filename

Встановлює назву HTML файлу звіту

```
"filename": "report.html"
```

openReport

Автоматично відкриває звіт у браузері після генерації

```
"openReport": true
```

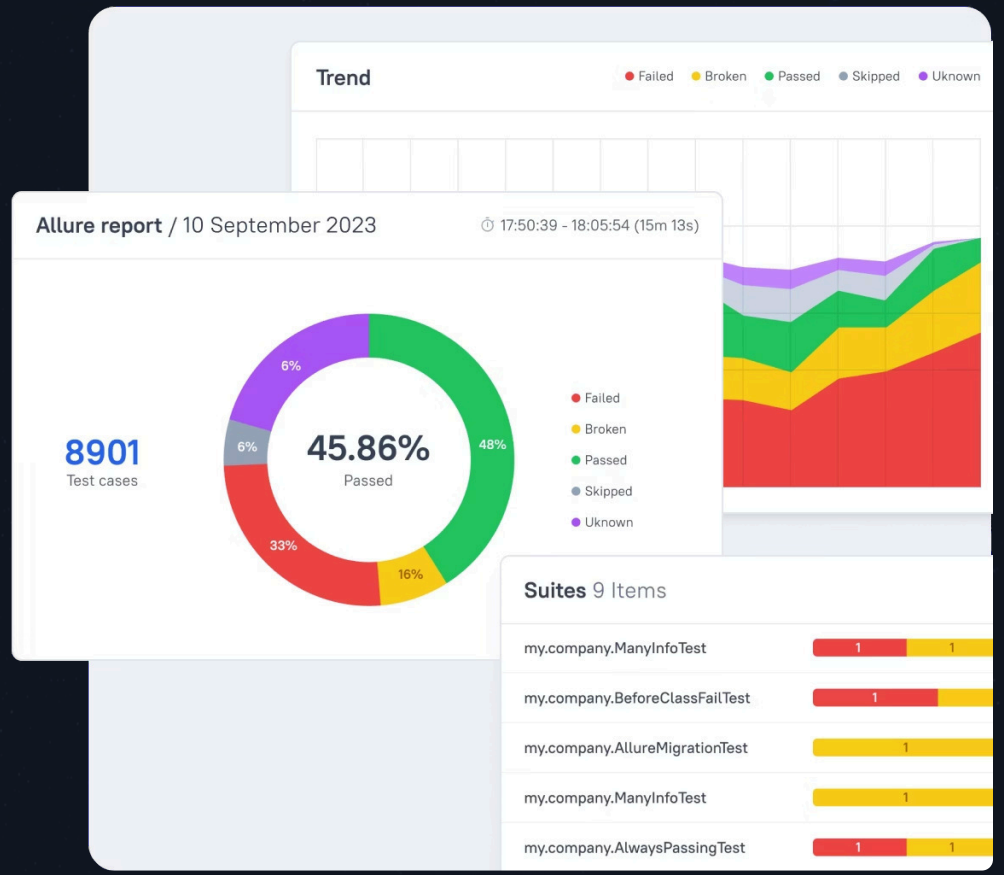
```
reporters: [
 "default",
 ["jest-html-reporters", {
 "publicPath": "./html-report",
 "filename": "report.html",
 "openReport": true
 }]
],
```

Перегляд згенерованого HTML звіту

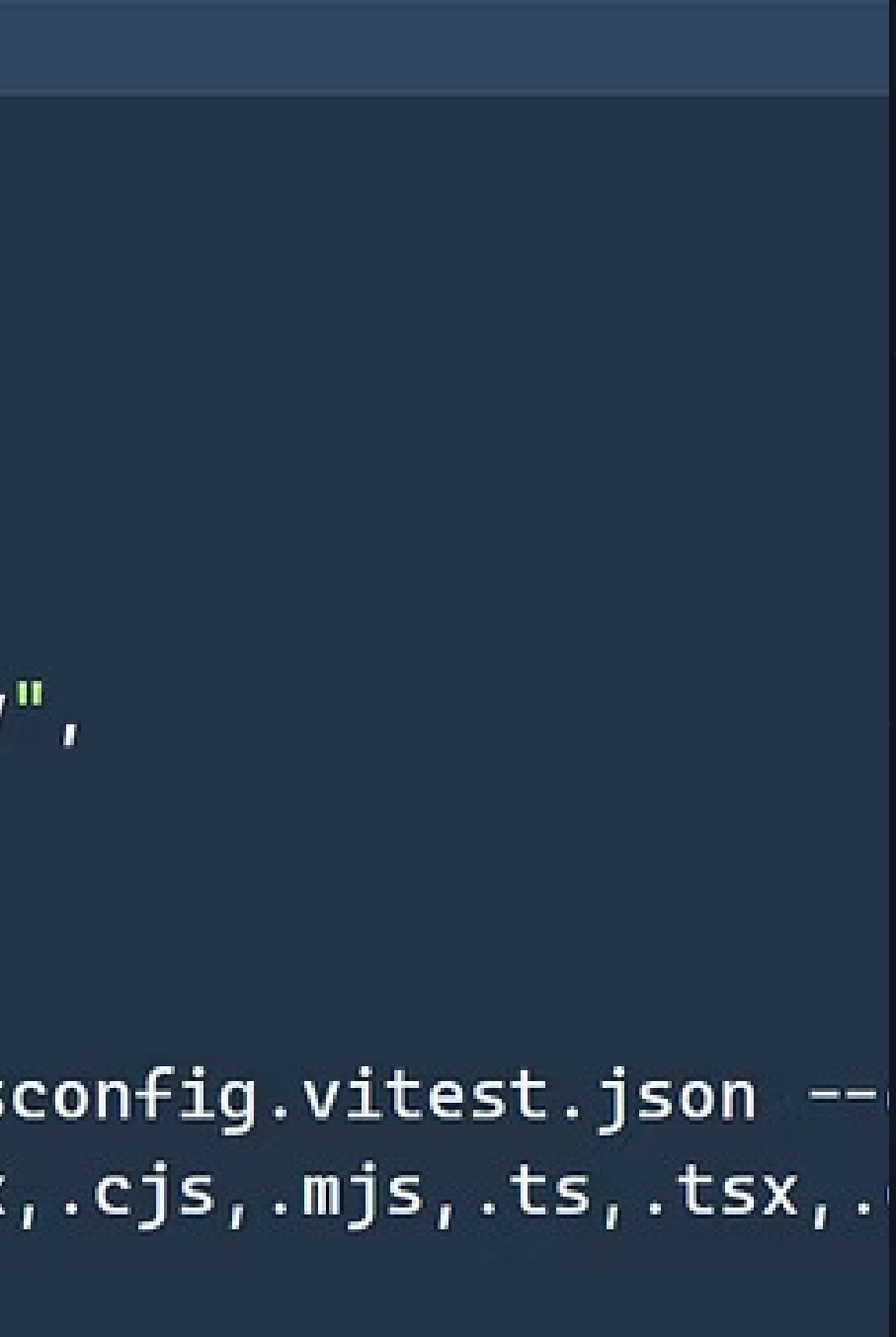
Відкриття звіту в редакторі

1. Знайдіть папку з репортом у провіднику VS Code
2. Відкрийте HTML файл, який був згенерований
3. Використайте функцію "Open with Live Server" або відкрийте файл безпосередньо в браузері

Звіт містить детальну інформацію про виконані тести, включаючи статистику успішних та невдалих тестів, час виконання та трасування помилок.



Для зручного перегляду HTML звітів рекомендується встановити розширення Live Server для VS Code або використовувати вбудовану функцію попереднього перегляду браузера



Створення спеціалізованої команди в package.json

01

Проблема

Довгі команди з багатьма параметрами важко запам'ятати і легко помилитися при введенні

02

Рішення

Створення кастомного скрипта в секції scripts файлу package.json спрощує виконання складних команд

03

Реалізація

Додайте новий запис з описовою назвою, яка відображає функціональність команди

Використання іменованих скриптів покращує читабельність коду, спрощує onboarding нових членів команди та зменшує ймовірність помилок при запуску тестів.

Структура команди api_tests

```
"api_tests": "npx jest api_requests/login.test.js && npx jest api_requests/actions.test.js"
```

Виконання логін тестів

Спочатку запускаються тести автентифікації з файлу login.test.js

Виконання action тестів

Після успішної автентифікації виконуються основні функціональні тести з actions.test.js

Оператор &&

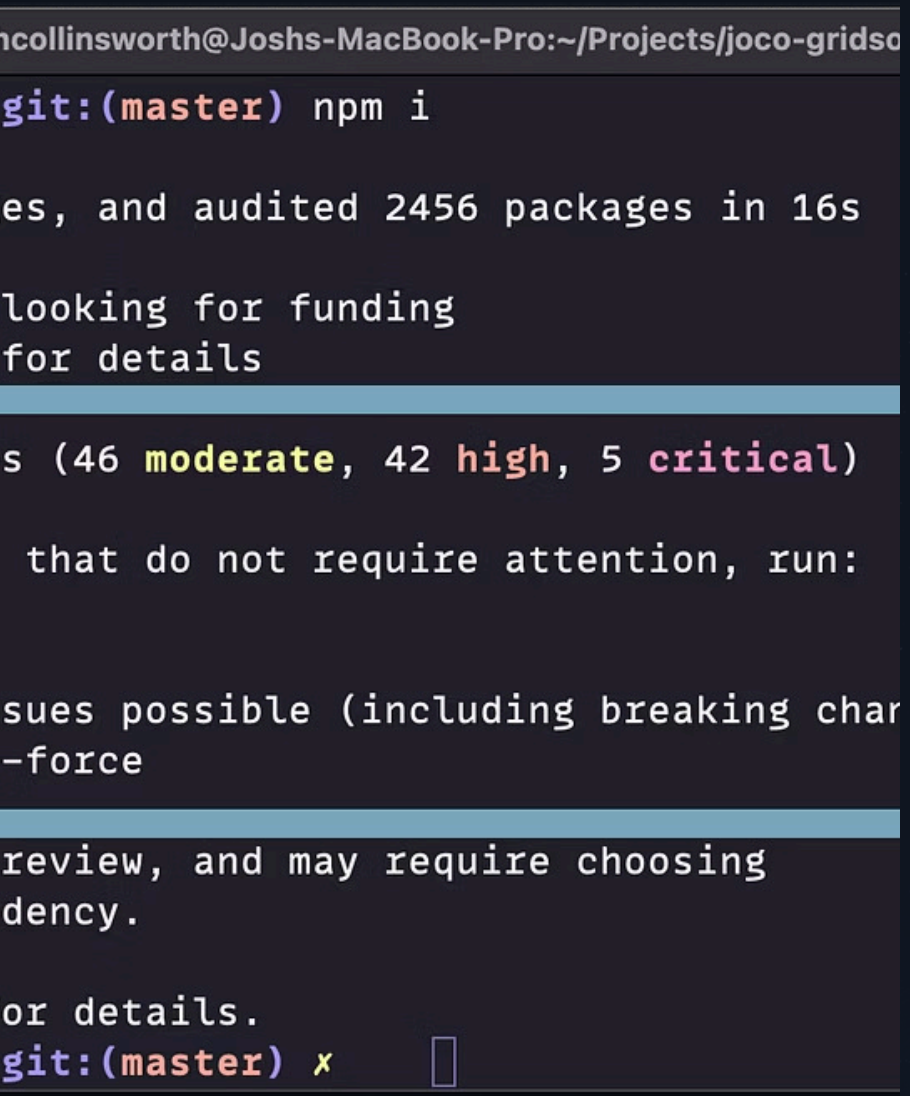
Забезпечує послідовне виконання команд - друга команда запускається тільки після успішного завершення першої

Переваги підходу

- Контроль послідовності виконання
- Збереження токенів між тестами
- Автоматична генерація звіту після завершення

Альтернативи

Можна використовувати більш складні сценарії з додатковими перевірками або умовами, залежно від потреб проекту



Запуск тестів через npm скрипт

1 Команда для запуску

```
npm run api_tests
```

Використовуйте цю просту команду замість довгого рядка з параметрами

2 Послідовність виконання

Логін та збереження токену → Запуск всіх функціональних тестів → Автоматична генерація HTML репорту

3 Результат

Ви отримуєте детальний HTML звіт з результатами всіх тестів, готовий для аналізу та презентації команді

Корисна порада: Додайте документацію до README.md проекту з описом доступних npm скриптів та їх призначення для зручності всієї команди розробки