# Screenshot testing

Візуальне тестування є критичним компонентом забезпечення якості сучасних веб-додатків. Screenshot testing дозволяє автоматично виявляти непередбачені зміни в інтерфейсі користувача, які можуть виникнути через оновлення коду, зміни в залежностях або регресії. Цей підхід особливо цінний для команд, які працюють над складними інтерфейсами з багатьма візуальними станами.

Playwright надає потужні вбудовані можливості для screenshot testing, які інтегруються безпосередньо в тестову інфраструктуру. На відміну від зовнішніх інструментів, вбудований функціонал Playwright забезпечує швидке виконання, просту конфігурацію та надійне порівняння зображень з мінімальними налаштуваннями.

# Вбудована функція toHaveScreenshot

Playwright пропонує потужну assertion функцію `toHaveScreenshot(name)`, яка спрощує процес візуального тестування. Ця функція автоматично обробляє складну логіку очікування та порівняння скріншотів, забезпечуючи надійність ваших тестів.

Механізм роботи базується на інтелектуальному алгоритмі очікування: Playwright автоматично чекає, поки сторінка або елемент досягне стабільного візуального стану. Це означає, що анімації завершаться, зображення завантажаться, а динамічний контент відобразиться повністю. Тільки після цього відбувається фінальне порівняння з еталонним знімком.

Функція підтримує два режими роботи: скріншоти повної сторінки та скріншоти окремих елементів. Це дає гнучкість у виборі рівня деталізації вашого візуального тестування, дозволяючи фокусуватися на критичних компонентах інтерфейсу.

## Ключові переваги

- Автоматичне очікування стабільного стану
- Вбудоване порівняння зображень
- Підтримка повних сторінок та елементів
- Налаштовувані пороги відмінностей
- Кросплатформенна сумісність

## Параметри функції

Функція приймає ім'я файлу як обов'язковий параметр та опціональні налаштування для контролю процесу порівняння.

# Синтаксис та базове використання

01

### Імпорт залежностей

Переконайтеся, що у вашому тесті імпортовано об'єкти `page` та `expect` з Playwright

02

### Виклик assertion

Використовуйте функцію `toHaveScreenshot()` з унікальним ім'ям файлу для кожного тесту

03

### Обробка результату

Playwright автоматично порівняє поточний та еталонний скріншоти, повідомивши про будь-які відмінності

Базовий приклад використання демонструє мінімальний код, необхідний для впровадження візуального тестування:

```
await expect(page).toHaveScreenshot('image.png');
```

Цей простий рядок коду запускає складний процес: Playwright чекає на стабільність сторінки, робить скріншот, порівнює його з еталоном і повідомляє про результат. Ім'я файлу `'image.png'` має бути описовим та унікальним для кожного тестового сценарію, щоб уникнути конфліктів.

### Повна сторінка

expect(page).toHaveScreenshot()

### Окремий елемент

expect(locator).toHaveScreenshot()

# Організація та зберігання скріншотів


Playwright автоматично створює структуровану систему для зберігання скріншотів. При першому виконанні тесту зі screenshot assertion, фреймворк генерує нову папку поруч з вашим тестовим файлом. Ця папка слідує конвенції іменування `<назва-файлу-з-тестом>-snapshots`, забезпечуючи логічну організацію та легку навігацію.

Така структура допомагає підтримувати чистоту проекту, оскільки всі скріншоти для конкретного тесту зберігаються разом. Це особливо корисно в великих проектах з десятками або сотнями візуальних тестів, де організація стає критичним фактором підтримки.

Для більшого контролю над структурою зберігання, Playwright надає опцію `snapshotPathTemplate` у конфігураційному файлі. Ця налаштування дозволяє визначити власний шаблон шляху для скріншотів, використовуючи змінні як `{testDir}`, `{testFileName}`, `{arg}`, та інші.

```
// playwright.config.ts
export default {
  use: {
    snapshotPathTemplate: '{testDir}/__screenshots__/{testFilePath}/{arg}{ext}'
  }
}
```

### Стандартна структура

Автоматично створюється папка з суфіксом `-snapshots`

### Власна структура

Налаштовується через `snapshotPathTemplate` у конфігурації

# Перший запуск та baseline генерація

Важливою особливістю роботи зі screenshot testing є поведінка при першому запуску. Коли ви вперше виконуєте тест з `toHaveScreenshot()`, Playwright не має еталонних зображень для порівняння, тому тест технічно "провалюється". Проте це очікувана та нормальна поведінка, яка є частиною робочого процесу візуального тестування.

При цьому першому запуску Playwright автоматично створює baseline скріншоти - еталонні зображення, які будуть використовуватися для всіх наступних порівнянь. Фреймворк зберігає ці зображення у відповідну папку snapshots, і саме ці файли стають "джерелом правди" для майбутніх тестових прогонів. Кожен наступний запуск буде порівнювати нові скріншоти з цими збереженими еталонами.

### Перший запуск

Тест провалюється, створюються baseline скріншоти

### Наступні запуски

Порівняння з baseline, тест проходить при збігу

Критично важливо зберігати baseline скріншоти у вашому репозиторії системи контролю версій (Git). Багато розробників інтуїтивно додають зображення до `.gitignore`, але у випадку з тестовими скріншотами це помилковий підхід. Ці файли є невід'ємною частиною вашого тестового набору - без них тести не можуть виконуватися на інших машинах або у CI/CD pipeline.

Коли ви працюєте в команді або використовуєте continuous integration, кожен член команди та CI сервер повинні мати доступ до тих самих baseline зображень. Тому переконайтеся, що папки з суфіксом `-snapshots` включені у ваш Git репозиторій. Це забезпечує консистентність візуального тестування у всій команді та інфраструктурі.

**Важливо:** Завжди додавайте baseline скріншоти до репозиторію. Вони необхідні для роботи тестів на інших машинах та в CI/CD. Не додавайте папки `*-snapshots` до `.gitignore`!

# Кросплатформенні особливості


Однією з найважливіших особливостей screenshot testing у Playwright є платформо-залежність згенерованих зображень. Різні операційні системи рендерять веб-контент з тонкими відмінностями у шрифтах, згладжуванні, субпіксельному рендерингу та інших візуальних аспектах. Це означає, що скріншот, створений на macOS, буде відрізнятися від скріншота тієї ж сторінки на Windows або Linux.

Playwright автоматично управляє цією складністю, зберігаючи окремі набори baseline скріншотів для кожної платформи. Коли ви запускаєте тести, фреймворк визначає поточну ОС і використовує відповідні еталонні зображення для порівняння. Це відбувається прозоро, без додаткової конфігурації з вашого боку.

Проте ця особливість створює важливі наслідки для робочого процесу розробки. Якщо ви створюєте baseline скріншоти на macOS, але ваш CI/CD pipeline працює на Linux, ваші тести провалюватимуться через відсутність відповідних baseline файлів для Linux платформи. Аналогічно, якщо різні члени команди працюють на різних операційних системах, кожна ОС потребує власного набору baseline зображень.

### macOS

Унікальний рендеринг шрифтів та згладжування

### Windows

Відмінності в субпіксельному рендерингу

### Linux

Варіації залежно від дистрибутива

Для вирішення цієї проблеми рекомендується генерувати baseline скріншоти на тій самій платформі, де виконуються ваші основні тести, особливо у CI/CD середовищі. Якщо ваш CI використовує Linux, найкраще створювати baseline на Linux або використовувати Docker контейнери для консистентності. Деякі команди використовують спеціалізовані Docker образи з Playwright для гарантії однакового середовища на всіх етапах розробки та тестування.

**Best Practice:** Використовуйте ту саму ОС для генерації baseline та виконання тестів. Для команд з різними ОС розгляньте використання Docker або виділеного CI сервера для створення та підтримки baseline скріншотів.