

Локатори та фільтрація елементів

Локатори є фундаментальною концепцією у Playwright, що дозволяє розробникам автоматизованих тестів точно та надійно взаємодіяти з елементами веб-сторінок. Ця документація розкриває всі аспекти роботи з локаторами, від базових CSS та XPath селекторів до складних фільтрів та операцій над групами елементів. Розуміння цих механізмів критично важливе для створення стабільних та підтримуваних автоматизованих тестів.



Requirements Table

TEST CASES	CASE TYPE	TASTING TIME	TOTAL PASSED	FAILED
Report one	Manual	1h 20m	25 (29%)	15 (67%)
Report two	Manual	1h 23m	129 (69%)	8 (19%)
Report three	Automatic	1h 21m	111 (59%)	5 (15%)
Report four	Manual	1h 23m	99 (45%)	1 (9%)

CSS and XPath support

CSS селектори

Playwright підтримує повний спектр CSS селекторів, надаючи потужний інструмент для точного визначення елементів. CSS селектори забезпечують високу читабельність коду та широку підтримку браузерами. Вони є найпопулярнішим вибором серед розробників завдяки своїй інтуїтивності та гнучкості.

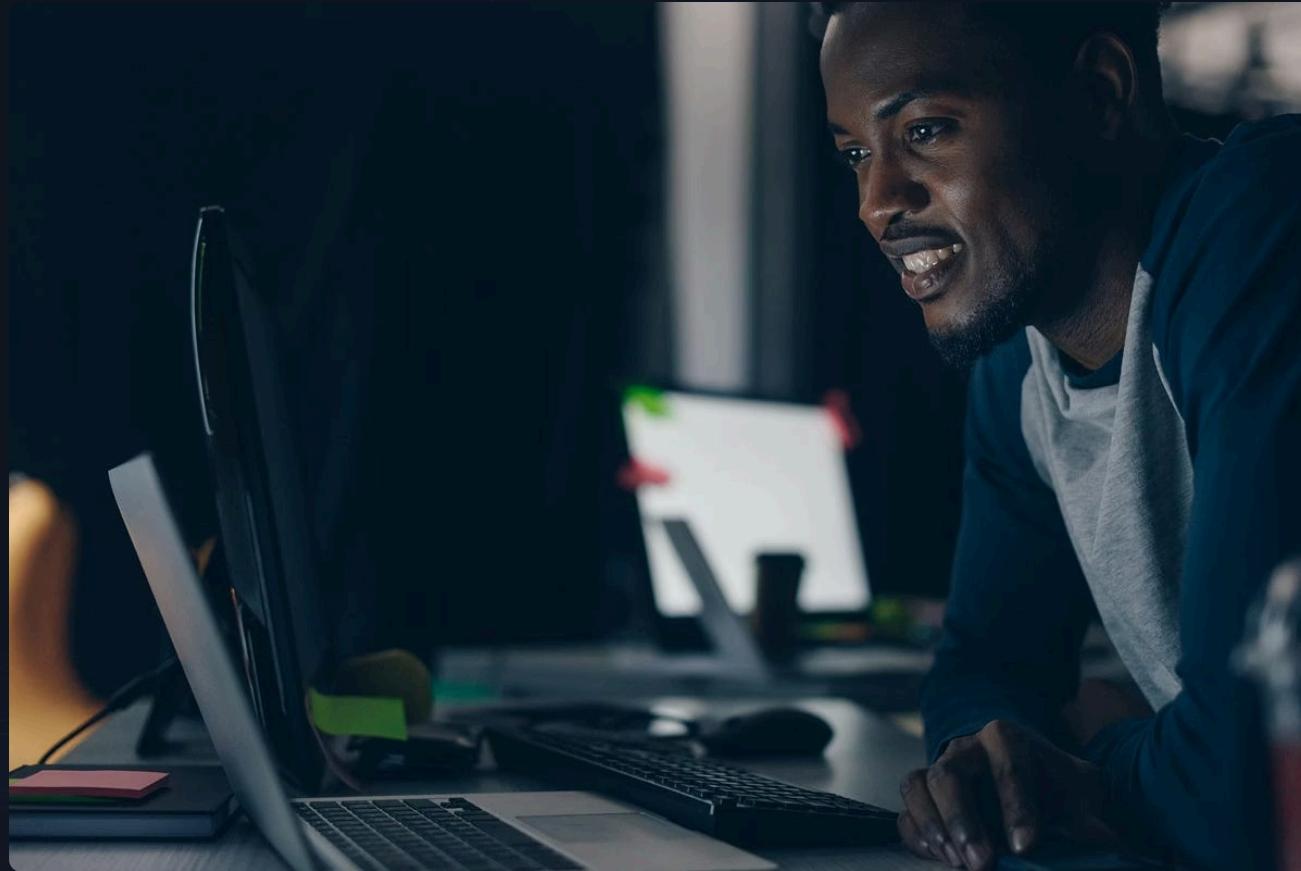
Важливою перевагою CSS є можливість проникнення в Shadow DOM, що відкриває доступ до компонентів сучасних веб-фреймворків.

```
await page.locator('button').click();
```

Цей простий приклад демонструє базове використання CSS селектора для взаємодії з кнопкою на сторінці. Метод locator() приймає селектор і створює об'єкт локатора, на якому можна викликати різні дії.



XPATH селектори



Можливості XPath

XPath надає альтернативний підхід до пошуку елементів, особливо корисний для складних навігаційних шляхів у DOM-структурі. Playwright автоматично розпізнає XPath вирази: будь-який рядок селектора, що починається з `//` або `..` вважається селектором `xpath`.

```
await  
page.locator('//button').click();
```

Наприклад, Playwright перетворює `//html/body` на `xpath='//html/body'`. Важливо пам'ятати про обмеження: **XPath не проникає в Shadow DOM**, що може бути критичним фактором при виборі стратегії селекції елементів у сучасних веб-додатах з компонентною архітектурою.

Locator

Локатори є центральним елементом в Playwright і представляють собою найважливішу концепцію для взаємодії з веб-сторінками. На короткий лад, локатори представляють собою спосіб знаходження елементів на сторінці в будь-який момент часу. Вони забезпечують автоматичне очікування та повторні спроби, що робить тести надійнішими та стійкішими до тимчасових затримок завантаження.

Динамічність

Локатори оцінюються лінно, що означає пошук елемента тільки коли це потрібно

Надійність

Автоматичні повторні спроби та очікування забезпечують стабільність тестів

Гнучкість

Підтримка фільтрації та ланцюжкових операцій для точного таргетування

Пошук елементів

Локатор можна створити за допомогою методу `page.locator()`. Це базовий метод, який приймає селектор і повертає об'єкт локатора, готовий до взаємодії:

```
const input = page.locator('input.required')
await input.fill("Hello Playwright!")
```

Вбудовані методи пошуку

Playwright надає потужний набір спеціалізованих методів для пошуку елементів, які орієнтовані на доступність та семантику HTML. Ці методи є рекомендованими розробниками фреймворку, оскільки вони створюють більш читабельні та надійні тести, які краще відповідають реальній взаємодії користувачів з інтерфейсом.



getByRole()

Знаходить за явними та неявними атрибутами доступності, ідеальний для семантичних елементів

```
await page  
  .getByRole('button',  
    { name: /submit/i })  
  .click();
```



getByText()

Шукає елементи за їх текстовим вмістом, підтримує точний та частковий збіг

```
const text = await page  
  .getByText('Welcome,  
John')  
  .innerText()
```

getByLabel()

Знаходить елемент управління форми за текстом асоційованої мітки

```
await page  
  .getByLabel('Password')  
  .fill('secret');
```



getByPlaceholder()

Знаходить поля вводу за атрибутом placeholder

```
await page  
  .getByPlaceholder(  
    'name@example.com')  
  .fill('test@test.com');
```

Додаткові методи пошуку

1

getByAltText()

Шукає елемент, зазвичай зображення, за альтернативним текстом атрибута alt

```
await page  
  .getByAltText('playwright  
logo')  
  .click();
```

2

getTitle()

Знаходить елемент за атрибутом title, корисно для підказок та додаткової інформації

```
await expect(page  
  .getTitle('Issues count'))  
  .toHaveText('25 issues');
```

3

getById()

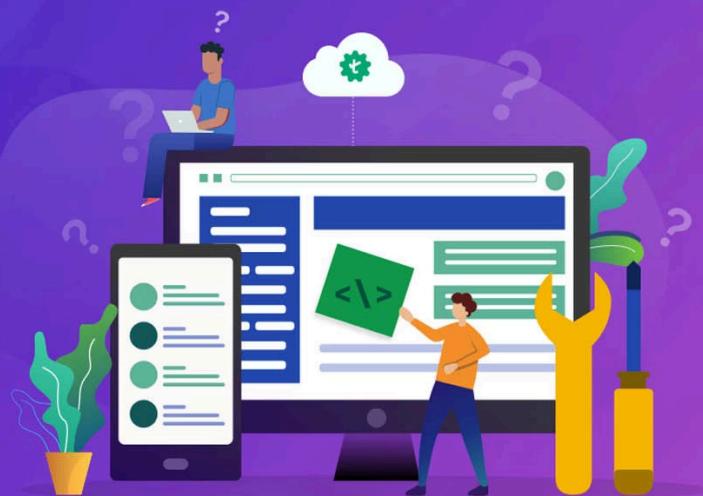
Шукає елемент за його атрибутом data-testid, найнадійніший метод для автотестів

```
await page  
  .getById('directions')  
  .click();
```



What is the
difference between

**desktop
application testing
and web testing?**



- **Порада:** Метод getById() є найнадійнішим для автоматизованого тестування, оскільки не залежить від змін у тексті інтерфейсу або структури DOM. Ви можете налаштовувати інші атрибути замість data-testid згідно зі стандартами вашого проекту.

Вибіркові локатори та фільтрація

Playwright надає потужні механізми для роботи з альтернативними варіантами елементів та уточнення пошуку. Метод `locator.or(locator)` створює локатор, який відповідає одному з двох або більше локаторів, що особливо корисно для обробки варіативних інтерфейсів.

Логіка OR для локаторів

Уявіть ситуацію, де ви хочете натиснути кнопку "Новий лист", але іноді з'являється діалогове вікно з налаштуваннями безпеки. У цьому випадку ви можете очікувати або кнопку, або діалогове вікно і діяти відповідно:

```
const newEmail = page.getByRole('button',  
  { name: 'New' });  
const dialog = page.getText(  
  'Confirm security settings');  
  
await expect(newEmail.or(dialog))  
  .toBeVisible();  
  
if (await dialog.isVisible())  
  await page.getByRole('button',  
    { name: 'Dismiss' }).click();  
await newEmail.click();
```

Також ви можете передати декілька селекторів у локатор розділяючи їх комою, і Playwright поверне той який зміг зарезолвити першим.

Фільтрація has/hasText

Ви можете уточнити ваш пошук використовуючи опції `has`, `hasText` або `hasNot`, `hasNotText`:

```
await page.locator('button',  
  {hasText: "Submit"})  
  .click()
```

```
const text = await page  
  .locator('p.info',  
  {has: page.locator('span.userInfo')})  
  .innerText()
```

Ці опції дозволяють створювати складні умови пошуку без необхідності писати довгі XPath вирази.

Метод filter() для локаторів

Метод filter() дозволяє уточнити локатори вже після їх знаходження, приймаючи ті самі опції has/hasText або hasNot/hasNotText. Це надзвичайно потужний інструмент для роботи зі складними структурами, такими як таблиці або списки з повторюваними елементами.

Зайдіть батьківський локатор

Спочатку створіть широкий локатор, який захоплює групу елементів

```
const rowLocator = page.locator('tr');
```

Додайте додаткові фільтри

Ланцюжком додавайте додаткові умови пошуку

```
.filter({ has: page.getByRole('button',  
{ name: 'column 2 button' } ) })
```

Цей підхід дозволяє створювати дуже точні селектори навіть для найскладніших DOM-структур, зберігаючи при цьому читабельність та підтримуваність коду.

Застосуйте перший фільтр

Уточніть пошук за текстовим вмістом

```
.filter({ hasText: 'text in column 1' })
```

Виконайте дію

Застосуйте потрібну операцію на відфільтрованому локаторі

```
.screenshot();
```

Операції над групами локаторів

Коли локатор посилається на декілька елементів, Playwright надає ефективні способи обробки всієї колекції.

Розуміння цих методів критично важливе для роботи зі списками, таблицями та іншими повторюваними структурами.

Спосіб 1: count() + nth()

```
const tableRows = page.locator('tr')
const count = await
tableRows.count()

for (let i = 0; i < count; i++) {
  const text = await tableRows
    .nth(i).innerText()
  console.log(text)
}
```



Спосіб 2: all() + for...of

```
const tableRows = page.locator('tr')

for (const row of await
tableRows.all()) {
  const text = await row.innerText()
  console.log(text)
}
```

Обидва варіанти принесуть вам бажаний результат, просто останній варіант є більш сучасним та ідіоматичним для JavaScript.

Поради щодо локаторів

Ваші селектори мають бути максимально короткими та надійними. Бажано орієнтуватися на атрибути елементів. Пам'ятайте, що ви можете шукати локатори всередині іншого локатора для складних структур:

```
const parentLocator =
page.locator('div.parent');
const childLocator =
parentLocator.locator('span.child');
```

Дії над локаторами

Playwright надає багатий набір методів для взаємодії з елементами через локатори. Ці методи охоплюють всі типові сценарії взаємодії користувача з веб-інтерфейсом, від простих кліків до складних маніпуляцій з фокусом та прокруткою.



`click()` / `hover()`

Клацання та наведення курсору для імітації базової взаємодії з елементами



`check()` / `uncheck()`

Керування станом чекбоксів та радіо-кнопок у формах



`focus()` / `scrollIntoView()`

Керування фокусом та видимістю елементів на сторінці



`fill()` / `press()`

Заповнення полів введення текстом та натискання клавіш для симуляції вводу



`selectOption()`

Вибір опцій у випадаючих списках та мультиселектах



`innerText()`

Отримання текстового вмісту елементів для верифікації та аналізу

- Додаткова інформація:** Повний список методів локатора з детальним описом параметрів та прикладами використання можна знайти в [офіційній документації Playwright](#). Ці методи постійно оновлюються та розширяються зожною новою версією фреймворку.