1. 개요

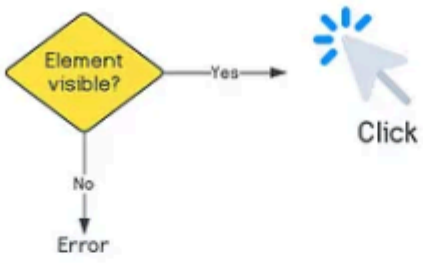
Selenium VS Playwright

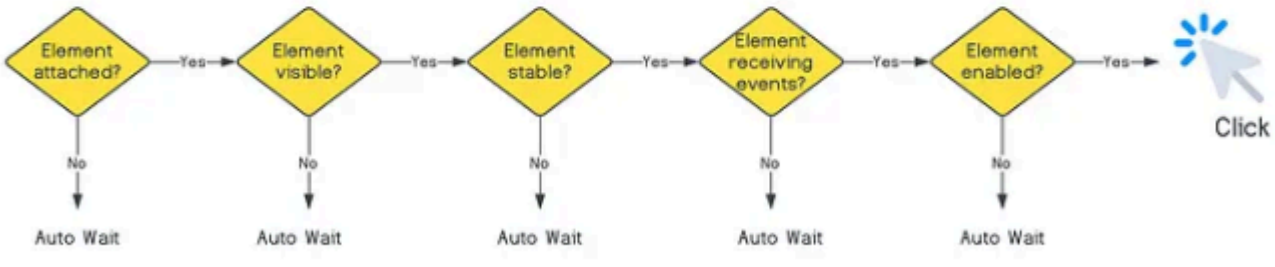
	Selenium	**Playwright**
브라우저 제어 구조	WebDriver 기반	WebSocket 기반
대기 처리	**수동 대기**	동작에 **자동 대기** 내장 타이밍 이슈 감소
테스트 안정성	대기 누락 등으로 **Flaky** 테스트 발생	기본 동작이 안정적이어서 **Flaky** 현상이 줄어듬

Selenium VS Playwright 브라우저 제어 방식

- **Selenium**
  테스트 코드 → Selenium(WebDriver) 클라이언트 → 브라우저 드라이버 → 브라우저
- **Playwright**
  테스트 코드 → Playwright 드라이버 → 브라우저

Selenium VS Playwright 대기 처리 및 안정성





- Ataached : Dom에 생성되었는가
- Visible : CSS 기준으로 보이는 상태인가 (display : none, width: 0px, height: 0px)
- Stable : 요소가 움직이거나 크기가 변하지 않는 안정적인 상태인가
- Receving event : 다른 요소(일반적으로 오버레이)가 해당 지점의 이벤트를 막고 있는지 (https://getbootstrap.com/docs/4.4/components/modal/#static-backdrop)
- Enabled : 활성화된 상태인가
- https://www.cuketest.com/playwright/python/docs/actionability/

```python
"""Selenium V1"""
def test_login():
  # 프로 스토어 접속
  driver = webdriver.Chrome()
  driver.get("https://store.frommyarti.com/arti")

  # 더보기 메뉴 클릭
  menu = driver.find_element(By.CSS_SELECTOR, 'img[alt="Main Menu"]')
  menu.click()
```

```
  # 요소가 보일때 까지 대기
  time.sleep(n)

  # "로그인 후 이용해주세요." 문구 클릭
  log_in = driver.find_element(By.CSS_SELECTOR, 'a[href="/signin"]')
  log_in.click()
```

```
"""Selenium V2"""
# 요소가 보일때 까지 대기
def wait_and_find(driver, by, locator):
  # 10초 동안 요소가 보일때까지 기다림
  return WebDriverWait(driver, 10).until(
    EC.visibility_of_element_located((by, locator))
  )

def test_login():
  # 프로 스토어 접속
  driver = webdriver.Chrome()
  driver.get("https://store.frommyarti.com/arti")

  # 더보기 메뉴 클릭
  menu = wait_and_find(driver,By.CSS_SELECTOR, 'img[alt="Main Menu"]')
  menu.click()

  # "로그인 후 이용해주세요." 문구 클릭
  log_in = wait_and_find(driver,By.CSS_SELECTOR, 'a[href="/signin"]')
  log_in.click()
```

```
"""Selenium V3"""
# 요소가 보일때 까지 대기
def safe_click(driver, locator, timeout=10):
  wait_driver = WebDriverWait(driver, timeout)
  # 10초 동안 요소가 존재하는지 기다림
  wait_driver.until(EC.presence_of_element_located(locator))
  # 10초 동안 요소가 보일때까지 기다림
  wait_driver.until(EC.visibility_of_element_located(locator))
  # 10초 동안 요소가 클릭가능 할 때 까지 기다림
  element = wait_driver.until(EC.element_to_be_clickable(locator))
  element.click()

def test_login():
  # 프로 스토어 접속
  driver = webdriver.Chrome()
  driver.get("https://store.frommyarti.com/arti")

  # 더보기 메뉴 클릭
  safe_click(driver,(By.CSS_SELECTOR, 'img[alt="Main Menu"]'))

  # "로그인 후 이용해주세요." 문구 클릭
  safe_click(driver,(By.CSS_SELECTOR, 'a[href="/signin"]'))
```

```
"""Playwright"""
# 요소가 보일때 까지 대기

def test_login(page: Page):
    # 프로 스토어 접속
    page.goto("https://store.frommyarti.com/arti")

    # 더보기 메뉴 클릭
    menu=page.locator('img[alt="전체메뉴"]')
    menu.click()

    # "로그인 후 이용해주세요." 문구 클릭
    log_in=page.locator('a[href="/signin"]')
    log_in.click()
```

F.I.R.S.T 원칙

테스트를 구성하기 위해서 F.I.R.S.T 원칙을 따른다.
항목에 대한 내용은 F.I.R.S.T Principles 문서를 참고한다.

FAST

- **테스트는 빠르게 실행되어야 합니다.**
- **Selenium VS Playwright 같은 테스트 실행 시간 차이**

```python
"""Selenium"""
def test_has_title():
    driver = webdriver.Chrome()
    driver.get("https://playwright.dev/")
    assert re.search(r"Playwright", driver.title)
    #     test_example.py::test_has_title PASSED [100%] Selenium
    #     ============== 1 passed in 1.14s ==============

"""Playwright"""
def test_has_title(page: Page):
    page.goto("https://playwright.dev/")
    expect(page).to_have_title(re.compile("Playwright"))
    #     test_example.py::test_has_title[chromium] PASSED [100%] Playwright
    #     ============== 1 passed in 0.73s ==============
```

Isolated

- **다른 테스트 케이스에 종속적인 테스트는 절대 작성하지 마세요.**

```
Feature: FrommStore SignOut
  Background:
    When Store_더보기 [☰] 버튼 클릭
    Then Store_더보기 페이지 노출
    When Store_[로그인 후 이용해주세요.] 항목 클릭
    Then Store_로그인 페이지 노출
     And Store_[로그인] 버튼 비활성화 상태로 노출
    When Store_유효한 아이디 입력
     And Store_올바른 비밀번호 입력
    Then Store_[로그인] 버튼 활성화 상태로 노출
    When Store_[로그인] 버튼 클릭
     And Store_더보기 [☰] 버튼 클릭
    Then Store_로그인한 아이디 노출

  @SST@Regression@Store
  Scenario: FrommStore 로그아웃
    When Store_[로그아웃] 항목 클릭
    Then Store_아티관 페이지 노출
    When Store_더보기 [☰] 버튼 클릭
    Then Store_더보기 페이지 노출
     And Store_[로그인 후 이용해주세요.] 항목 노출
```

Repeatable

- **실행할 때마다 동일한 결과를 생성하는 테스트를 말하며 반복가능해야합니다.**

```python
pytestmark = pytest.mark.order(1)

@allure.title("Store 회원가입")
@scenario( feature_name: "../features/store/store_SignUp.feature", scenario_name: "FrommStore 회원가입")
def test_fromm_store_signup():
    pass
```

```python
@allure.title("Store BFF 구매")
@scenario( feature_name: "../features/store.feature", scenario_name: "FrommStore BFF 구매")
def test_fromm_store_buy_bff():
    pass
```

Self-validating

- **결과를 수동으로 해석해서는 안 됩니다.**

```python
@then(parsers.parse('Store_내 콘텐츠 페이지 내 "{content_name}"이 "{status}" 상태로 노출'))
@allure.step('Store_내 콘텐츠 페이지 내 "{content_name}"이 "{status}" 상태로 노출')
def content_status_enabled(web_function_driver:Page,content_name:str,status:str):
    target = web_function_driver.locator( selector: "p",has_text=content_name)
    target_container = target.locator('../..')
    if status == '비활성화':
        expect(target_container).to_be_disabled(timeout=10000)
    elif status == '활성화':
        expect(target_container).to_be_enabled(timeout=10000)
```

Timely

- **적절한 시기에 테스트를 작성하는 데 집중하는 것이 더 좋습니다.**