Selenium WebDriver 빠른 참조

브라우저 자동화, 요소 상호작용, 대기, 어서션

설정

설치

```
pip install selenium webdriver-manager
# webdriver-manager가 브라우저 드라이버를 자동 다운로드
```

기본 드라이버 설정

```
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
driver = webdriver.Chrome(
    service=Service(ChromeDriverManager().install()))
```

헤드리스 모드

```
options = webdriver.ChromeOptions()
options.add_argument("--headless=new")
options.add_argument("--no-sandbox")
driver = webdriver.Chrome(options=options)
```

지원 브라우저

`webdriver.Chrome()`	Google Chrome / Chromium
`webdriver.Firefox()`	Mozilla Firefox (GeckoDriver)
`webdriver.Edge()`	Microsoft Edge (Chromium)
`webdriver.Safari()`	Apple Safari (macOS 전용)

브라우저 및 탐색

탐색

```
driver.get("https://example.com")
driver.back()                # 브라우저 뒤로
driver.forward()             # 브라우저 앞으로
driver.refresh()             # 페이지 새로고침
```

브라우저 속성

`driver.title`	현재 페이지 제목
`driver.current_url`	현재 페이지 URL
`driver.page_source`	전체 페이지 HTML 소스
`driver.get_cookies()`	모든 쿠키 목록

윈도우 관리

```
driver.set_window_size(1920, 1080)
driver.maximize_window()
driver.minimize_window()
driver.quit()                # 모든 윈도우 닫기, 세션 종료
```

요소 찾기

로케이터 전략

```
from selenium.webdriver.common.by import By
driver.find_element(By.ID, "login-btn")
driver.find_element(By.CLASS_NAME, "nav-item")
driver.find_element(By.CSS_SELECTOR, "div.card > h2")
driver.find_element(By.XPATH, "//input[@name='q']")
```

By 전략

`By.ID`	요소의 id 속성 일치
`By.NAME`	요소의 name 속성 일치
`By.CLASS_NAME`	CSS 클래스 일치 (단일 클래스)
`By.TAG_NAME`	HTML 태그명 일치
`By.CSS_SELECTOR`	CSS 셀렉터 (가장 유연)
`By.XPATH`	XPath 표현식
`By.LINK_TEXT`	정확한 앵커 텍스트
`By.PARTIAL_LINK_TEXT`	부분 앵커 텍스트 일치

여러 요소 찾기

```
items = driver.find_elements(By.CSS_SELECTOR, "li.item")
for item in items:
    print(item.text)
# 없으면 빈 리스트 반환 (예외 없음)
```

상호작용

클릭 및 입력

```
elem = driver.find_element(By.ID, "search")
elem.clear()                 # 기존 텍스트 지우기
elem.send_keys("selenium python")
elem.submit()                # 상위 폼 제출
```

드롭다운

```
from selenium.webdriver.support.ui import Select
select = Select(driver.find_element(By.ID, "country"))
select.select_by_visible_text("Canada")
select.select_by_value("ca")
select.select_by_index(2)
```

요소 속성

`.text`	표시되는 텍스트 내용
`.get_attribute('href')`	HTML 속성 값
`.is_displayed()`	요소가 보이면 True
`.is_enabled()`	요소가 상호작용 가능하면 True
`.is_selected()`	체크박스/라디오가 선택되면 True
`.tag_name`	HTML 태그 (예: 'input', 'div')
`.value_of_css_property('color')`	계산된 CSS 속성 값

대기

명시적 대기

```
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
elem = WebDriverWait(driver, 10).until(
    EC.presence_of_element_located((By.ID, "result")))
```

예상 조건

`presence_of_element_located`	DOM에 요소가 존재
`visibility_of_element_located`	페이지에서 요소가 보임
`element_to_be_clickable`	요소가 보이고 활성화됨
`text_to_be_present_in_element`	요소가 예상 텍스트 포함
`alert_is_present`	JavaScript 경고창이 표시됨
`staleness_of`	요소가 더 이상 DOM에 없음
`title_contains`	페이지 제목이 텍스트 포함

암묵적 대기

```
driver.implicitly_wait(10)  # 초 단위, 전역 적용
# 명시적 대기 권장 — 더 정밀한 제어
```

프레임 및 윈도우

프레임

```
driver.switch_to.frame("frame-name")    # 이름/id로
driver.switch_to.frame(0)                # 인덱스로
driver.switch_to.frame(elem)             # 요소로
driver.switch_to.default_content()       # 메인으로 돌아가기
```

윈도우 및 탭

```
original = driver.current_window_handle
driver.switch_to.new_window("tab")       # 새 탭 열기
driver.switch_to.window(original)        # 다시 전환
driver.close()                           # 현재 탭 닫기
```

알림창

```
alert = driver.switch_to.alert
print(alert.text)
alert.accept()     # 확인 클릭
alert.dismiss()    # 취소 클릭
alert.send_keys("input text")
```

스크린샷

스크린샷 캡처

```
driver.save_screenshot("page.png")       # 전체 페이지
elem = driver.find_element(By.ID, "chart")
elem.screenshot("chart.png")             # 단일 요소
```

Base64로 스크린샷

```
b64 = driver.get_screenshot_as_base64()
png = driver.get_screenshot_as_png()     # 바이트
```

액션

액션 체인

```
from selenium.webdriver.common.action_chains import ActionChains
actions = ActionChains(driver)
actions.move_to_element(menu).click().perform()
```

키보드 액션

```
from selenium.webdriver.common.keys import Keys
elem.send_keys(Keys.ENTER)
elem.send_keys(Keys.CONTROL, "a")        # 전체 선택
actions.key_down(Keys.SHIFT).click(elem).perform()
```

마우스 액션

`.click(elem)`	요소 클릭
`.double_click(elem)`	요소 더블클릭
`.context_click(elem)`	요소 우클릭
`.move_to_element(elem)`	요소 위에 마우스 올리기
`.drag_and_drop(src, dst)`	소스에서 목적지로 드래그
`.click_and_hold(elem)`	마우스 버튼 누르고 유지
`.release()`	마우스 버튼 해제

어서션

일반 어서션 (pytest)

```
assert "Dashboard" in driver.title
assert driver.find_element(By.ID, "msg").text == "Done"
assert driver.current_url.endswith("/home")
assert len(driver.find_elements(By.CSS_SELECTOR, "tr")) > 0
```

대기 기반 어서션

```
WebDriverWait(driver, 5).until(              # 나타남
    EC.visibility_of_element_located((By.ID, "success")))
WebDriverWait(driver, 5).until(              # 사라짐
    EC.invisibility_of_element_located((By.ID, "spinner")))
```

JavaScript 실행

```
result = driver.execute_script("return document.title")
driver.execute_script(
    "arguments[0].scrollIntoView(true);", elem)
```

Selenium WebDriver 빠른 참조

일반 패턴

페이지 객체 패턴

```
class LoginPage:
    URL = "/login"
    user_loc = (By.ID, "username")
    def login(self, drv, user, pwd):
        drv.find_element(*self.user_loc).send_keys(user)
```

컨텍스트 매니저

```
from selenium import webdriver
with webdriver.Chrome() as driver:
    driver.get("https://example.com")
    print(driver.title)
# driver.quit()이 자동으로 호출됨
```

재시도 및 정리

```
try:
    driver.get("https://example.com")
    WebDriverWait(driver, 10).until(
        EC.element_to_be_clickable((By.ID, "btn")))
finally: driver.quit()
```