

Selenium

Auto Mobile Robot

Exported on 06/07/2024

Table of Contents

1	Selenium 설치.....	4
1.1	Beautiful Soup 만으로 해결할 수 없는 것	4
1.2	Selenium.....	4
1.3	Selenium 을 사용하기 위해서는.....	4
1.4	Chrome 브라우저 버전 확인	4
1.5	Chrome Driver 다운로드 페이지로 가서.....	5
1.6	Chrome 과 같은 버전의 링크를 클릭하고.....	5
1.7	chromedriver_linux64.zip 우클릭 후 링크 주소 복사	6
1.8	터미널에서 크롬 드라이버 파일을 다운로드 받고.....	6
1.9	압축 해제	6
1.10	잠깐.. 작업 폴더 점검	7
1.11	크롬 드라이버 파일은 driver 폴더로 이동	7
1.12	이번에는 Selenium 설치.....	7
1.13	간단한 테스트	8
1.14	결과 확인	8
1.15	사용이 끝나면 반드시 닫아주자.....	9
2	Selenium 맛보기	10
2.1	브라우저 주의.....	10
2.2	일단 다시 페이지를 열고.....	10
2.3	URL 가져오기	10
2.4	맨 밑으로 스크롤	10
2.5	맨 위로 스크롤	11
2.6	특정 Element 로 이동하려면	12
2.7	일단 위치를 찾고	13
2.8	확인 - find_element.....	13
2.9	아.. 공백	14
2.10	다시 확인	14

2.11 자식 엘리먼트의 TEXT	14
2.12 다시.. News 로 스크롤.....	14
2.13 겨우 보임. ㅋ	15
2.14 웹페이지의 스크린 샷도 저장할 수 있다	15
2.15 확인	16
2.16 XPath	16
2.17 클릭!	17
2.18 Search 버튼을 찾아서 - XPATH	18
2.19 Search 버튼 클릭.....	19
2.20 검색어 입력창을 찾고 - CSS Selector	19
2.21 검색어도 타이핑 해보자 - send_keys	20
2.22 엔터도 쳐야지 - Keys.ENTER	21
2.23 우린 Python 예제가 필요하니까	22
2.24 이번에는 ID 로 가져와서 클릭 - By.ID	22
2.25 예제를 찾아서	23
2.26 출력도 해보자	23
2.27 다시 이전 페이지로 이동하고 - back().....	24
2.28 이번엔 메뉴 목록을 찾아서	24
2.29 메뉴 목록을 가져오자 - find_elements.....	25
2.30 About 의 DropDown 메뉴 리스트	25
2.31 그중 History	26
2.32 클릭하면 이동한다.....	26
2.33 다시 돌아와서	27
2.34 이제 여러가지를 연속으로 - Action Chain.....	28
2.35 Selenium 에서 Attribute 값은 어떻게? - get_attribute	29
2.36 현재 페이지의 HTML 가져오기	29
2.37 이걸 다 외워야하나?	30
2.38 잊지말고 종료~	30

1 Selenium 설치

1.1 BeautifulSoup 만으로 해결할 수 없는 것

- 접근할 웹 주소를 알수 없을 때
- 자바 스크립트를 사용하는 웹페이지의 경우
- 웹 브라우저로 접근하지 않으면 안될 때

1.2 Selenium

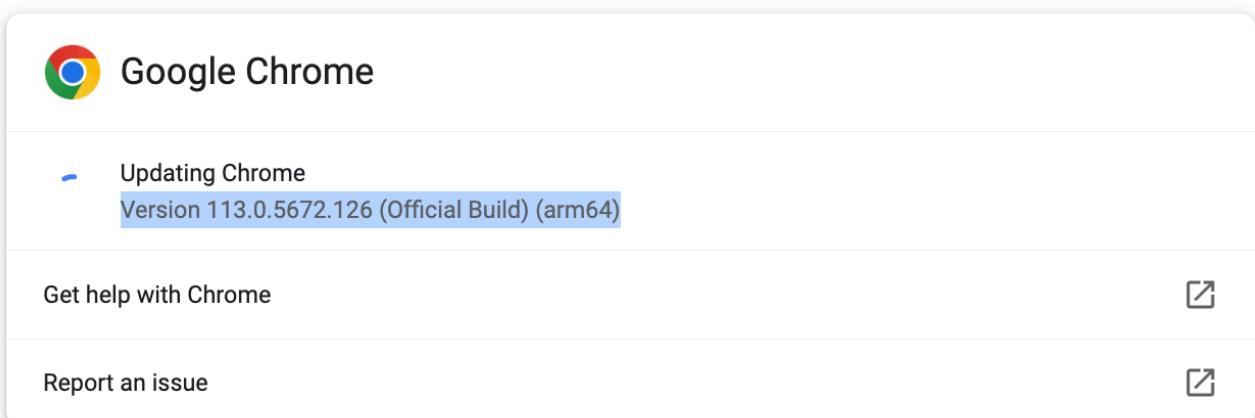
- 웹 브라우저를 원격 조작하는 도구
- 자동으로 URL을 열고 클릭하는 동작 등이 가능
- 스크롤, 문자 입력, 화번 캡쳐, 버튼 클릭 등

1.3 Selenium 을 사용하기 위해서는

- Python 모듈도 설치하고 크롬 브라우저와 크롬 드라이버가 필요하다.
- 우리는 이미 크롬 브라우저를 설치하여 사용중이다.

1.4 Chrome 브라우저 버전 확인

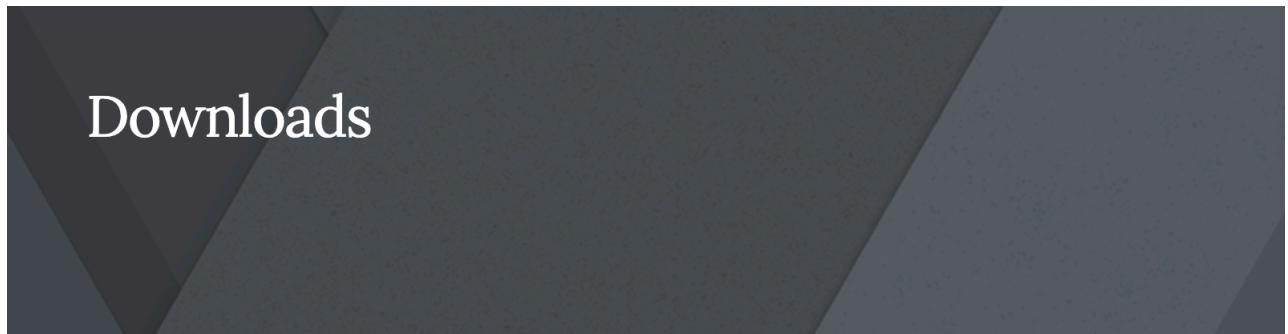
- Chrome > About Google Chrome



1.5 Chrome Driver 다운로드 페이지로 가서

<https://chromedriver.chromium.org/downloads>

1.6 Chrome 과 같은 버전의 링크를 클릭하고..



Current Releases

- If you are using Chrome version 114, please download [ChromeDriver 114.0.5735.90](#)
- If you are using Chrome version 113, please download [ChromeDriver 113.0.5672.63](#)
- If you are using Chrome version 112, please download [ChromeDriver 112.0.5615.49](#)
- For older version of Chrome, please see below for the version of ChromeDriver that supports it.

If you are using Chrome from Dev or Canary channel, please follow the instructions on the [ChromeDriver Canary](#) page.

For more information on selecting the right version of ChromeDriver, please see the [Version Selection](#) page.

1.7 chromedriver_linux64.zip 우클릭 후 링크 주소 복사

Name	Last modified	Size	ETag
Parent Directory		-	-
chromedriver_linux64.zip	2022-05-03 00:24:07	6.08MB	b64b1d3b8fe10d635dc5775965ca8048
chromedriver_mac64.zip			597f9231804384ee2345bba18584cca
chromedriver_mac_arm64			91bb4f92a44e26af2c38d71b1876bfa
chromedriver_win32.zip			7853f34740941971c153b11365716ef
notes.txt			a6ae80cd5a17d104faa80e953a140bf
Save Link As...			
Copy Link Address			

1.8 터미널에서 크롬 드라이버 파일을 다운로드 받고

```
wget https://chromedriver.storage.googleapis.com/113.0.5672.63/chromedriver_linux64.zip
```

```
noma@ubuntu:~$ wget https://chromedriver.storage.googleapis.com/106.0.5249.61/chromedriver_linux64.zip
--2022-10-13 03:37:55-- https://chromedriver.storage.googleapis.com/106.0.5249.61/chromedriver_linux64.zip
Resolving chromedriver.storage.googleapis.com (chromedriver.storage.googleapis.com)... 34.64.4.112, 2404:f340:10:1803::2010
Connecting to chromedriver.storage.googleapis.com (chromedriver.storage.googleapis.com)|34.64.4.112|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 6657315 (6.3M) [application/zip]
Saving to: 'chromedriver_linux64.zip'

chromedriver_linux64.zip      100%[=====] 6.35M  9.62MB/s   in 0.7s

2022-10-13 03:37:56 (9.62 MB/s) - 'chromedriver_linux64.zip' saved [6657315/6657315]

noma@ubuntu:~$
```

1.9 압축 해제

```
unzip chromedriver_linux64.zip
rm chromedriver_linux64.zip
```

```
noma@ubuntu:~$ unzip chromedriver_linux64.zip
Archive:  chromedriver_linux64.zip
  inflating: chromedriver
noma@ubuntu:~$ ls
chromedriver  chromedriver_linux64.zip  Desktop  Documents  Downloads  Music  Pictures  Public  snap  Templates  venv  Videos  ws
noma@ubuntu:~$ rm chromedriver_linux64.zip
noma@ubuntu:~$ ls
chromedriver  Desktop  Documents  Downloads  Music  Pictures  Public  snap  Templates  venv  Videos  ws
noma@ubuntu:~$ █
```

1.10 잠깐.. 작업 폴더 점검

- driver 폴더도 추가

```
Last login: Fri Jun 2 02:18:08 on ttys003
noma@noma-macbook ~ % cd dev/eda
noma@noma-macbook eda % ls
data      src
noma@noma-macbook eda % mkdir driver
noma@noma-macbook eda % ls
data      driver  src
noma@noma-macbook eda % █
```

1.11 크롬 드라이버 파일은 driver 폴더로 이동

```
noma@noma-macbook ~ % ls chromedriver
chromedriver
noma@noma-macbook ~ % cp chromedriver ~/dev/eda	driver
noma@noma-macbook ~ % ls ~/dev/eda	driver
chromedriver
noma@noma-macbook ~ %
```

1.12 이번에는 Selenium 설치

```
(eda) noma@noma-macbook dev % pip install selenium
Requirement already satisfied: selenium in /Users/noma/venv/eda/lib/python3.10/site-packages (4.9.1)
Requirement already satisfied: urllib3[socks]<3,>=1.26 in /Users/noma/venv/eda/lib/python3.10/site-packages (from selenium) (2.0.2)
Requirement already satisfied: trio~=0.17 in /Users/noma/venv/eda/lib/python3.10/site-packages (from selenium) (0.22.0)
Requirement already satisfied: trio-websocket~=0.9 in /Users/noma/venv/eda/lib/python3.10/site-packages (from selenium) (0.10.2)
Requirement already satisfied: certifi>=2021.10.8 in /Users/noma/venv/eda/lib/python3.10/site-packages (from selenium) (2023.5.7)
```

1.13 간단한 테스트 ..

```
from selenium import webdriver
from selenium.webdriver.chrome.service import Service

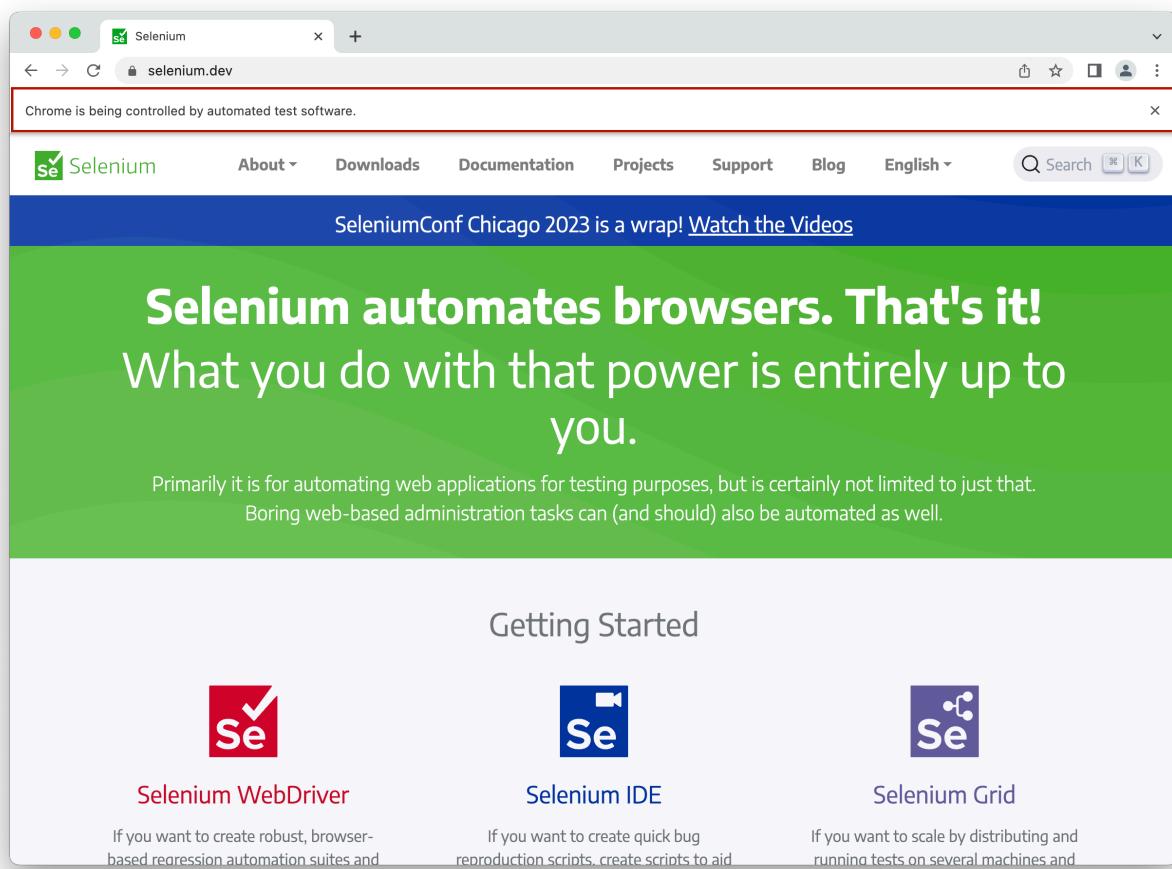
driver = webdriver.Chrome(service=Service("../driver/chromedriver"))
driver.get("https://www.selenium.dev/")

✓ 2.2s
```

Python

1.14 결과 확인

- 브라우저가 아래와 같이 실행되고 크롬 상단 부분에 'Chrome 이 자동화된 테스트 소프트웨어에 의해 제어되고 있습니다.'라는 문구를 확인할 수 있다.



1.15 사용이 끝나면 반드시 닫아주자.

```
driver.close()
```

✓ 0.1s

Python

2 Selenium 맛보기

2.1 브라우저 주의

- 구글 개발자툴로 구조를 분석하기 위한 브라우저와
- 셀레니움으로 자동 실행된 브라우저를 잘 구분하자

2.2 일단 다시 페이지를 열고

```
from selenium import webdriver
from selenium.webdriver.chrome.service import Service

driver = webdriver.Chrome(service=Service("../driver/chromedriver"))
driver.get("https://www.selenium.dev/")

✓ 2.2s
```

Python

2.3 URL 가져오기

```
driver.current_url

'https://www.selenium.dev/'
```

Python

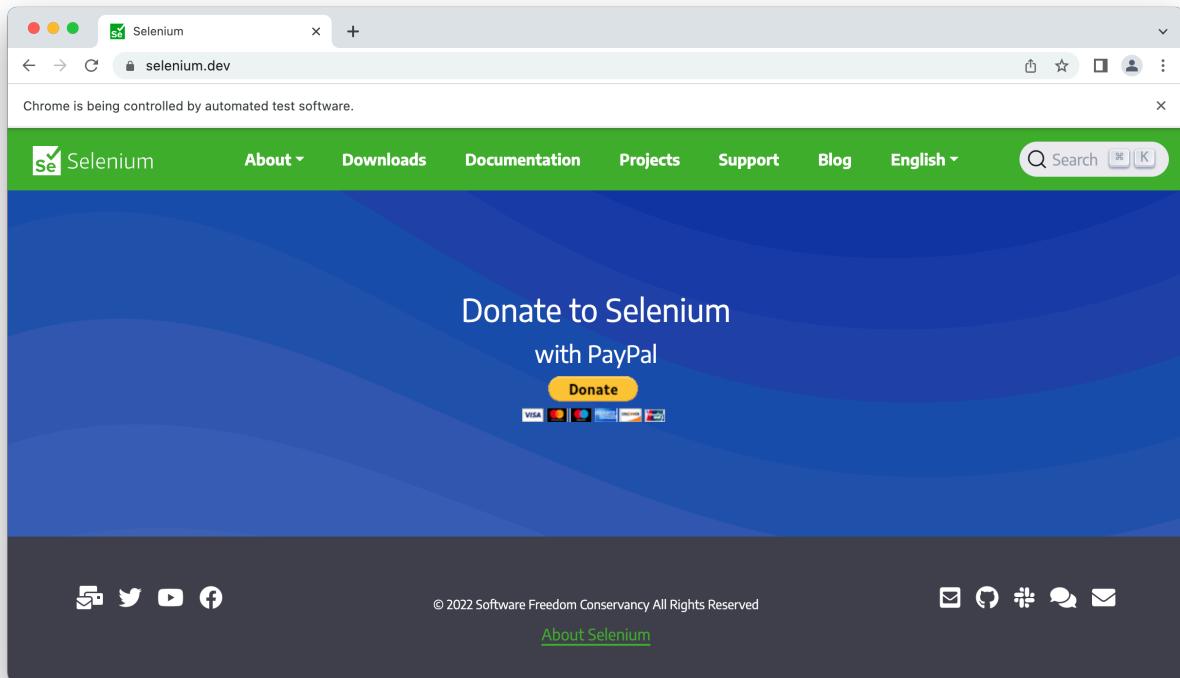
2.4 맨 밑으로 스크롤

- 스크롤 가능한 페이지 높이 : `document.body.scrollHeight`

```
driver.execute_script("window.scrollTo(0, document.body.scrollHeight)")
```

✓ 0.0s

Python



2.5 맨 위로 스크롤

```
driver.execute_script("window.scrollTo(0, 0)")  
✓ 0.0s
```

Python

The screenshot shows the official Selenium website (selenium.dev) displayed in a web browser. The page has a green header with the Selenium logo and navigation links for About, Downloads, Documentation, Projects, Support, Blog, and English. A search bar is also present. The main content area features a large heading "Selenium automates browsers. That's it!" followed by the subtext "What you do with that power is entirely up to you." Below this, a note states that primarily it is for automating web applications for testing purposes, but is certainly not limited to just that. It also mentions that boring web-based administration tasks can (and should) also be automated as well. A yellow banner at the bottom of the main content area announces the "Selenium Conference Chicago 2023" and encourages users to "Submit A Talk".

2.6 특정 Element로 이동하려면

The screenshot shows the Selenium homepage again, but this time the "News" section is highlighted with a red border. The news item reads: "InvalidSelectorException has \triangle InvalidSelectorException now extends from WebDriverException".

2.7 일단 위치를 찾고

```
<div class="d-flex justify-content-center p-5 td-box--100">
<h2 class="selenium">News</h2>
</div>
```

2.8 확인 - find_element

```
from selenium.webdriver.common.by import By

element = driver.find_element(By.CLASS_NAME, "d-flex justify-content-center p-5 td-box--100")
```

NoSuchElementException
Cell In[34], line 3
 1 from selenium.webdriver.common.by import By
----> 3 element = driver.find_element(By.CLASS_NAME, "d-flex justify-content-center p-5 td-box--100")

2.9 아.. 공백

```
from selenium.webdriver.common.by import By  
element = driver.find_element(By.CLASS_NAME, "d-flex.justify-content-center.p-5.td-box--100")
```

Python

2.10 다시 확인

```
from selenium.webdriver.common.by import By  
  
element = driver.find_element(By.CLASS_NAME, "d-flex.justify-content-center.p-5.td-box--100")  
print(element.get_attribute('innerHTML'))
```

Python

```
<h2 class="selenium">News</h2>
```

2.11 자식 엘리먼트의 TEXT

```
from selenium.webdriver.common.by import By  
  
element = driver.find_element(By.CLASS_NAME, "d-flex.justify-content-center.p-5.td-box--100")  
child = element.find_element(By.CLASS_NAME, "selenium")  
  
print(child.text)
```

Python

```
News
```

2.12 다시.. News 로 스크롤

```
driver.execute_script("arguments[0].scrollIntoView(false);", child)  
✓ 0.0s
```

Python

2.13 겨우 보임. ㅋ



Selenium Level Sponsors



News

2.14 웹페이지의 스크린샷도 저장할 수 있다

```
driver.save_screenshot("../data/image_01.png")
```

✓ 0.2s

Python

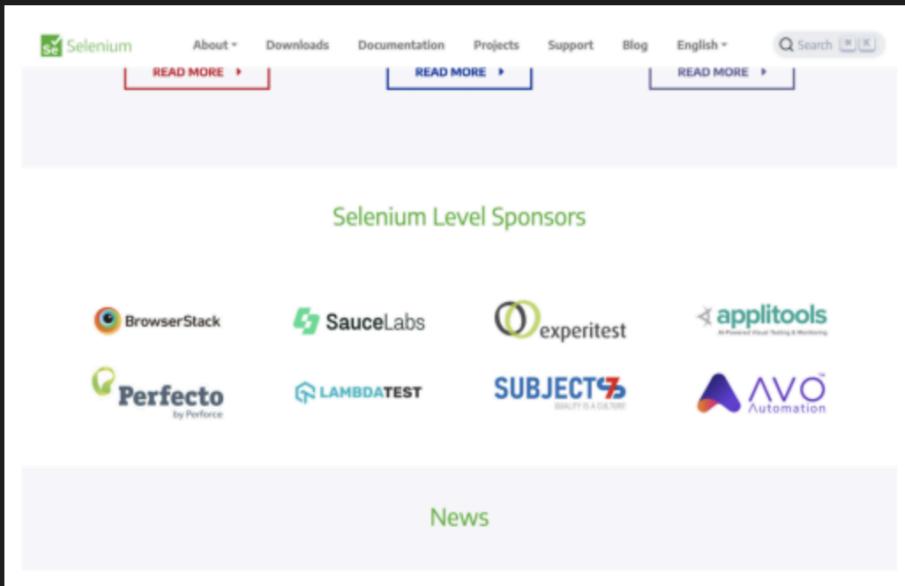
True

2.15 확인

```
from matplotlib import pyplot as plt
from matplotlib import image as mpimg

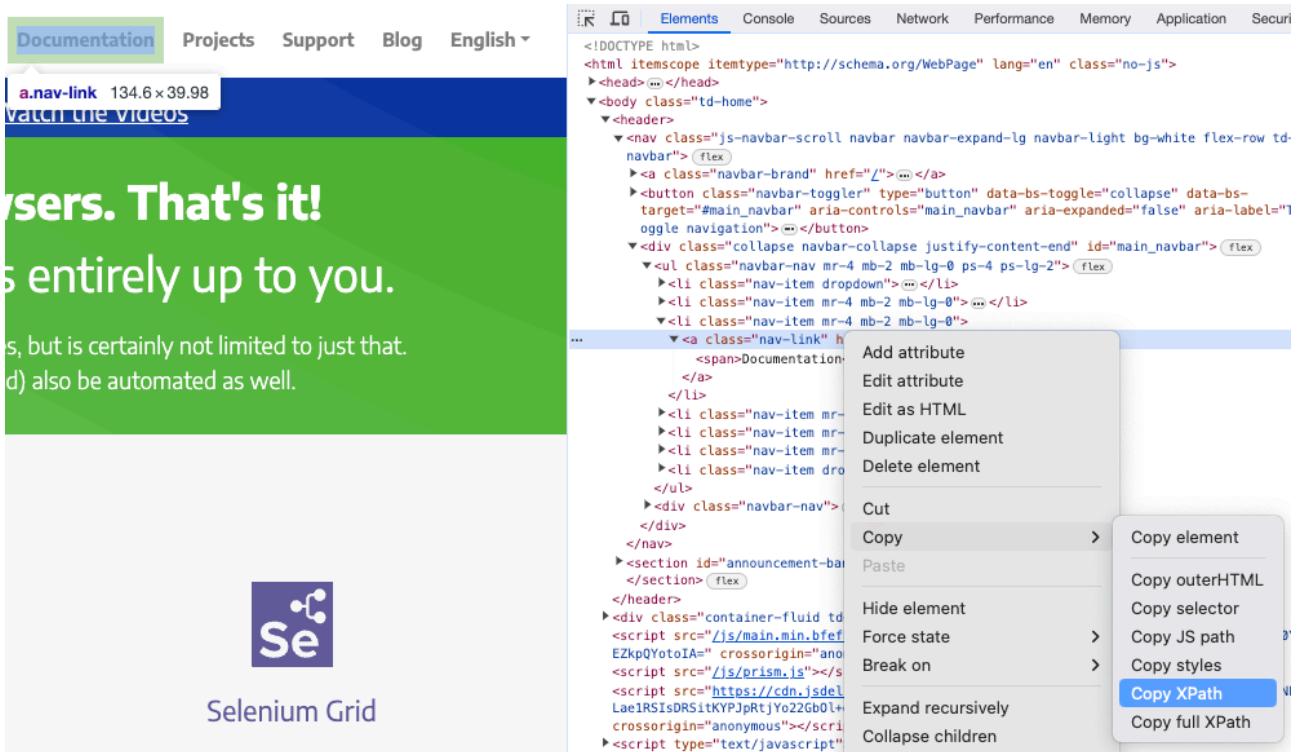
image = mpimg.imread("../data/image_01.png")
plt.imshow(image)
plt.axis('off')
plt.show()
```

✓ 0.3s Python



2.16 XPath

- Full XPATH - 절대 경로
- XPATH - 상대 경로



2.17 클릭!

- Documentation 을 XPath 로 찾아서 클릭할수있다.

```
doc_link = driver.find_element(By.XPATH, '//*[@id="main-navbar"]/ul/li[3]/a')
doc_link.click()
```

Python

Chrome is being controlled by automated test software.

Selenium

About Downloads Documentation Projects Support Blog English ▾

Search K

SeleniumConf Chicago 2023 is a wrap! Watch the Videos

Documentation

Documentation v4.0

The Selenium Browser Automation Project

Selenium is an umbrella project for a range of tools and libraries that enable and support the automation of web browsers.

It provides extensions to emulate user interaction with browsers, a distribution server for scaling browser allocation, and the infrastructure for implementations of the [W3C WebDriver specification](#) that lets you write interchangeable code for all major web browsers.

Edit this page Create documentation issue Create project issue Print entire section

2.18 Search 버튼을 찾아서 - XPATH

Blog English ▾

button.DocSearch.DocSearch arch-Button 153.59 x 35.99

tomation Project

able and support the automation of web

tribution server for scaling browser allocation, [specification](#) that lets you write interchangeable

ousands of hours of their own time, and rove.

to further an open discussion around [reference](#) to teach and nurture the community.

sets that can be run interchangeably in many t you inside a browser. You can find a more

Elements Console Sources Network Performance Memory Application Security

```
<nav class="js-navbar-scroll navbar navbar-expand-lg navbar-light bg-white flex-row td-navbar">
  <a class="navbar-brand" href="#">@@</a>
  <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#main_navbar" aria-controls="main_navbar" aria-expanded="false" aria-label="Toggle navigation">@@</button>
  <div class="collapse navbar-collapse justify-content-end" id="main_navbar">flex
    <ul class="navbar-nav mr-4 mb-2 mb-lg-0 ps-4 ps-lg-2">flex
      <li class="nav-item dropdown">@@</li>
      <li class="nav-item mr-4 mb-2 mb-lg-0">@@</li>
      <li class="nav-item mr-4 mb-2 mb-lg-0">@@</li>
      <li class="nav-item mr-4 mb-2 mb-lg-0">@@</li>
      <li class="nav-item dropdown d-none d-lg-block">@@</li>
    </ul>
    <div class="navbar-nav">flex
      <div class="td-search">
        <div class="td-search--algolia" id="docsearch-1">
          ... <button type="button" class="DocSearch DocSearch-Button" aria-label="Search">flex == $0
            <span class="D" width="20">@@</span>
            <span class="D">@@</span>
            <span class="D">@@</span>
          </button>
        </div>
      </div>
    </div>
  </div>
  <section id="announcement">flex
    <script src="https://main.min.js" crossorigin="EZkpQYotoIA=">@@</script>
    <script src="https://prism.js">@@</script>
    <script src="https://cdn.js">@@</script>
</header>
```

Add attribute Edit as HTML Duplicate element Delete element

Cut Copy Paste

Hide element Force state Break on

Expand recursively Collapse children

Copy element Copy outerHTML Copy selector Copy JS path Copy styles Copy XPath Copy full XPath

2.19 Search 버튼 클릭

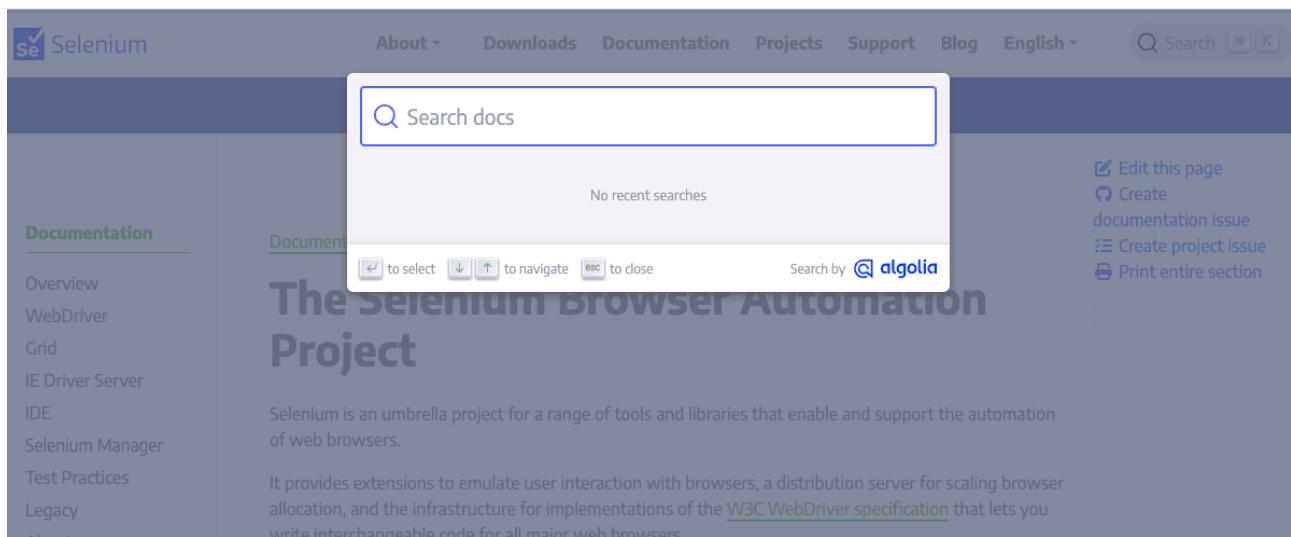
```
search_btn = driver.find_element(By.XPATH, '//*[@id="docsearch-1"]/button')
search_btn.click()
```

✓ 0.0s

Python

Chrome is being controlled by automated test software.

X



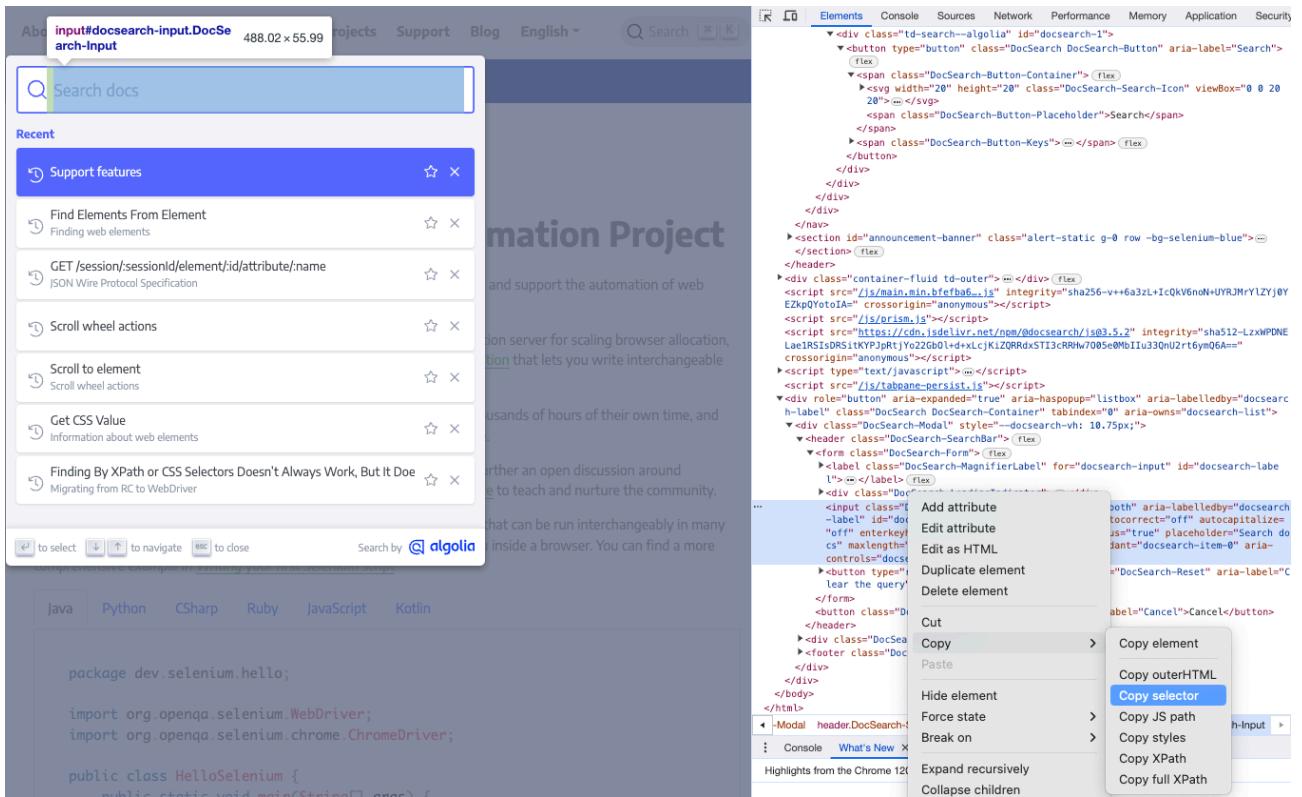
2.20 검색어 입력창을 찾고 - CSS Selector

- HTML에서 특정 요소를 선택하여 스타일을 적용하는데, 이때 선택을 해주는 요소

```
from selenium.webdriver.common.keys import Keys

search_doc = driver.find_element(By.CSS_SELECTOR, "#docsearch-input")
```

Python



2.21 검색어도 타이핑 해보자 - send_keys

```
from selenium.webdriver.common.keys import Keys

search_doc = driver.find_element(By.CSS_SELECTOR, "#docsearch-input")
search_doc.send_keys('find elements')

✓ 0.1s
```

Python

Chrome is being controlled by automated test software.

The screenshot shows the Selenium documentation website. A search bar at the top contains the query "find elements". Below the search bar, a sidebar on the left lists various documentation categories like Overview, WebDriver, Grid, etc. The main content area displays search results for "Find Elements From Element". The first result is "Finding web elements", which has a detailed description and a link to "FAQs for Selenium 2". Other results include "Q: My XPath finds elements in one browser, but not ...", "Q: WebDriver fails to find elements / Does not block ...", "Find element(s) utility methods in Java", and "5. Find an element". On the right side, there's a sidebar with options to edit the page, create issues, and print sections. At the bottom, there are navigation keys and a search bar.

2.22 엔터도 쳐야지 - Keys.ENTER

```
search_doc.send_keys(Keys.ENTER)
```

✓ 0.5s

Python

Chrome is being controlled by automated test software.

The screenshot shows the SeleniumConf Chicago 2023 wrap-up page. At the top, a banner says "SeleniumConf Chicago 2023 is a wrap! Watch the Videos". The main content area features a large heading "Finding web elements" with a sub-section "Locating the elements based on the provided locator values". Below this, there's a paragraph about Selenium's locator strategies and a snippet of HTML code. On the left, a sidebar lists documentation categories like Overview, WebDriver, and Elements. On the right, there's a sidebar with links to "Edit this page", "Create documentation issue", "Create project issue", and "Print entire section". A vertical sidebar on the far right lists various locator-related terms such as First matching element, Evaluating entire DOM, etc.

2.23 우린 Python 예제가 필요하니까

Find Elements From Element

It is used to find the list of matching child WebElements within the context of parent element. To achieve this, the parent WebElement is chained with 'findElements' to access child elements

Move C button#tabs-09-02-tab.nav-link 84.48 x 41.09

```
Java Python CSharp Ruby JavaScript Kotlin

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import java.util.List;

public class findElementsFromElement {
    public static void main(String[] args) {
        WebDriver driver = new ChromeDriver();
        try {
            driver.get("https://example.com");
        }
    }
}
```

```
</div>
<h2 id="find-elements-from-element">...</h2>
<p>...</p>
<ul class="nav nav-tabs" id="tabs-9" role="tablist">
  <li class="nav-item" role="presentation">...</li>
  <li class="nav-item" role="presentation">...</li>
  ...
  <li class="nav-item" role="presentation">
    <button class="nav-link" id="tabs-09-02-tab" data-bs-toggle="tab" data-bs-target="#tabs-09-02" role="tab" data-td-persist="python" aria-controls="tabs-09-02" aria-selected="false" tabindex="-1"> Python</button> == $0
  </li>
  <li class="nav-item" role="presentation">...</li>
  <li class="nav-item" role="presentation">...</li>
  <li class="nav-item" role="presentation">...</li>
  <li class="nav-item" role="presentation">...</li>
</ul>
<div class="tab-content" id="tabs-9-content">...
</div>
<h2 id="get-active-element">...</h2>
<p>...</p>
<p>...</p>
<ul class="nav nav-tabs" id="tabs-10" role="tablist">
  ...
  <li class="nav-item" role="presentation">
    <div class="tab-content" id="tabs-10-content">...
    </div>
    <div class="text-muted mt-5 pt-3 border-top">...</div>
  </li>
</ul>
</main>
</div>
```

2.24 이번에는 ID로 가져와서 클릭 - By.ID¹

```
python_tab = driver.find_element(By.ID, "tabs-09-02-tab")
driver.execute_script("arguments[0].scrollIntoView(false);", python_tab)
✓ 0.0s                                         Python

python_tab.click()
✓ 0.0s                                         Python
```

Find Elements From Element

It is used to find the list of matching child WebElements within the context of parent element. To achieve this, the parent WebElement is chained with 'findElements' to access child elements

Move Code

Java Python CSharp Ruby JavaScript Kotlin

¹ <http://By.ID>

2.25 예제를 찾아서

Find Elements From Element

It is used to find the list of matching child WebElements within the context of parent element. To achieve this, the parent WebElement is chained with 'findElements' to access child elements

[Move Code](#)

div#tabs-09-02.tab-pane.fade.a 791.68 × 656 JavaScript Kotlin

```
from selenium import webdriver
from selenium.webdriver.common.by import By

driver = webdriver.Chrome()
driver.get("https://www.example.com")

# Get element with tag name 'div'
element = driver.find_element(By.TAG_NAME, 'div')

# Get all the elements available with tag name 'p'
elements = element.find_elements(By.TAG_NAME, 'p')
for e in elements:
    print(e.text)
```

```
:before
"Find Elements From Element"
><a aria-hidden="true" href="#find-elements-from-element" style="visibility: hidden;">...<./span>
    "webdriver"
    <span class="token punctuation"><./span>
    "common"
    <span class="token punctuation"><./span>
    "by "
```

2.26 출력도 해보자

```
python_panel = driver.find_element(By.ID, "tabs-09-02")
python_code = python_panel.find_element(By.TAG_NAME, "code")

print(python_code.text)
```

✓ 0.0s

Python

```
from selenium import webdriver
from selenium.webdriver.common.by import By

driver = webdriver.Chrome()
driver.get("https://www.example.com")

# Get element with tag name 'div'
element = driver.find_element(By.TAG_NAME, 'div')

# Get all the elements available with tag name 'p'
elements = element.find_elements(By.TAG_NAME, 'p')
for e in elements:
    print(e.text)
```

2.27 다시 이전 페이지로 이동하고 - back()

driver.back()

✓ 0.0s

Python

Chrome is being controlled by automated test software.

2.28 이번엔 메뉴 목록을 찾아서

2.29 메뉴 목록을 가져오자 - find_elements

```

menu = driver.find_element(By.ID, "main-navbar")
li_list = menu.find_elements(By.TAG_NAME, "li")
for idx, li in enumerate(li_list):
    print("[ " + str(idx) + " ] =====")
    print(li.get_attribute('innerHTML'))
    ✓ 0.0s
[ 0 ] =====
<a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button" data-bs-toggle="dropdown">
<a class="dropdown-item" href="/project">Structure and Governance</a>
<a class="dropdown-item" href="/events">Events</a>
<a class="dropdown-item" href="/ecosystem">Ecosystem</a>
<a class="dropdown-item" href="/history">History</a>
<a class="dropdown-item" href="/getinvolved">Get Involved</a>
<a class="dropdown-item" href="/sponsors">Sponsors</a></div>
[ 1 ] =====
<a class="nav-link" href="/downloads"><span>Downloads</span></a>
[ 2 ] =====
  
```

2.30 About 의 DropDown 메뉴 리스트

```

about = li_list[0]
about.click()
link_list = about.find_elements(By.CLASS_NAME, "dropdown-item")
for link in link_list:
    print(link.text)
    ✓ 0.0s
  
```

About Selenium
 Structure and Governance
 Events
 Ecosystem
 History
 Get Involved
 Sponsors

The screenshot shows the official Selenium Project website. At the top, there's a navigation bar with links for About, Downloads, Documentation, Projects, Support, Blog, and English. A search bar is also present. On the left, a sidebar titled "Documentation" contains links to About Selenium, Structure and Governance, Events, Ecosystem, History, Get Involved, and Sponsors. The main content area features a banner with the text "2023 is a wrap! Watch the Videos". Below the banner, the title "The Selenium Project" is displayed. A sidebar on the right provides options to edit the page, create documentation or project issues, and print the section.

2.31 그중 History

```
history = about.find_element(By.LINK_TEXT, "History")
print(history.text)
```

✓ 0.0s

Python

History

2.32 클릭하면 이동한다.

```
history.click()
```

✓ 0.0s

Python

The screenshot shows the Selenium website at selenium.dev/history/. The page has a blue header bar with the text "SeleniumConf Chicago 2023 is a wrap! Watch the Videos". Below the header, there's a green section with the title "Selenium History". The main content area starts with a green sidebar containing the text "The story starts in 2004". The sidebar text describes Jason Huggins building the Core mode as "JavaScriptTestRunner" for testing an internal Time and Expenses application (Python, Plone). It notes that automatic testing of any applications is core to ThoughtWork's style, given the Agile leanings of this consultancy. It also mentions help from Paul Gross and Jie Tina Wang.

The story starts in 2004

The story starts in 2004 at ThoughtWorks in Chicago, with Jason Huggins building the Core mode as "JavaScriptTestRunner" for the testing of an internal Time and Expenses application (Python, Plone). Automatic testing of any applications is core to ThoughtWork's style, given the Agile leanings of this consultancy. He has help from Paul Gross and Jie Tina Wang. For them, this was a day job.

Jason started demoing the test tool to various colleagues. Many were excited about its immediate and intuitive visual feedback, as well as its potential to grow as a reusable testing framework for other web applications.

2.33 다시 돌아와서

The screenshot shows the Selenium documentation page for the `driver.back()` method. The code snippet is shown in Python, with a checkmark indicating it runs in 0.0 seconds. The URL is selenium.dev/documentation/.

The main content area features a large heading "The Selenium Browser Automation Project". A sidebar on the left lists various documentation sections: Overview, WebDriver, Grid, IE Driver Server, IDE, Selenium Manager, Test Practices, Legacy, and About. A right sidebar contains links for editing the page, creating documentation issues, creating project issues, and printing the entire section.

2.34 이제 여러가지를 연속으로 - Action Chain

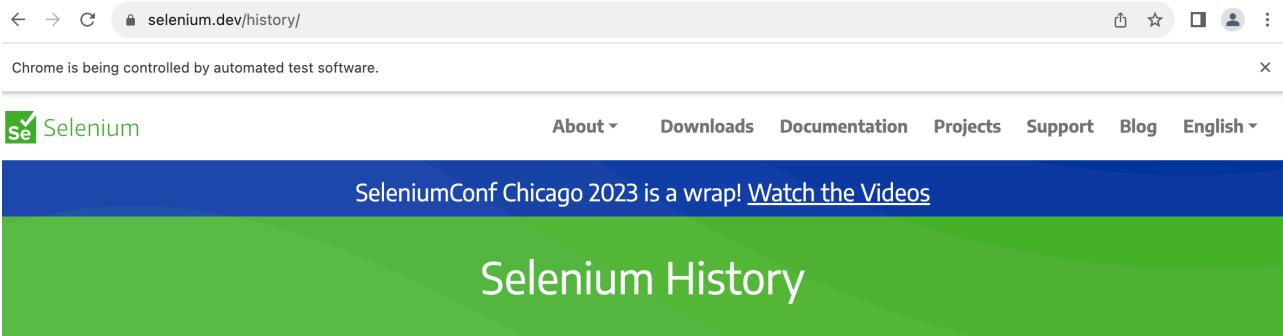
- Action Chain - 마우스, 키보드 입력 등 연속 동작 실행
- About 메뉴를 클릭하고, History 를 클릭해보자

```
from selenium.webdriver import ActionChains
import time

actions = ActionChains(driver)
actions.click(about)
actions.click(history)
actions.perform()
```

✓ 0.6s

Python



The story starts in 2004

The story starts in 2004 at ThoughtWorks in Chicago, with Jason Huggins building the Core mode as "JavaScriptTestRunner" for the testing of an internal Time and Expenses application (Python, Plone). Automatic testing of any applications is core to ThoughtWork's style, given the Agile leanings of this consultancy. He has help from Paul Gross and Jie Tina Wang. For them, this was a day job.

Jason started demoing the test tool to various colleagues. Many were excited about its immediate and intuitive visual feedback, as well as its potential to grow as a reusable testing framework for other web applications.

2.35 Selenium에서 Attribute 값은 어떻게? - get_attribute

```
driver.back()

about = li_list[0]
link_list = about.find_elements(By.CLASS_NAME, "dropdown-item")
for link in link_list:
    print(link.get_attribute("href"))

✓ 0.0s
```

Python

```
https://www.selenium.dev/about
https://www.selenium.dev/project
https://www.selenium.dev/events
https://www.selenium.dev/ecosystem
https://www.selenium.dev/history
https://www.selenium.dev/getinvolved
https://www.selenium.dev/sponsors
```

2.36 현재 페이지의 HTML 가져오기

- Selenium으로 원하는 페이지로 이동한 이후 이전처럼 BeautifulSoup으로 작업해도 된다.

```
from bs4 import BeautifulSoup

page = driver.page_source
soup = BeautifulSoup(page, "html.parser")
print(soup.prettify())

✓ 0.1s
```

Python

```
<html class="no-js" lang="en">
<head>
<meta charset="utf-8"/>
<meta content="width=device-width,initial-scale=1,shrink-to-fit=no" name="viewport"/>
<link href="/documentation/_print/" rel="alternate" type="text/html"/>
<meta content="index, follow" name="robots"/>
<link href="/favicons/favicon.ico" rel="shortcut icon"/>
<link href="/favicons/apple-touch-icon-180x180.png" rel="apple-touch-icon" sizes="180x180"/>
<link href="/favicons/favicon-16x16.png" rel="icon" sizes="16x16" type="image/png"/>
<link href="/favicons/favicon-32x32.png" rel="icon" sizes="32x32" type="image/png"/>
<link href="/favicons/android-36x36.png" rel="icon" sizes="36x36" type="image/png"/>
<link href="/favicons/android-48x48.png" rel="icon" sizes="48x48" type="image/png"/>
```

2.37 이걸 다 외워야하나?

- 예제 코드까지 볼수있어요. 필요할때 뒤적뒤적...

<https://www.selenium.dev/documentation/>

The screenshot shows the Selenium documentation website at <https://www.selenium.dev/documentation/>. The page title is "Mouse actions". The left sidebar has a "Documentation" section with links to Overview, WebDriver, Getting Started, Drivers, Browsers, Waits, Elements, Interactions, Actions API, Keyboard, Mouse (which is currently selected), Pen, Wheel, BiDirectional, Support Features, Troubleshooting, Grid, IE Driver Server, IDE, Selenium Manager, and Test Practices. The main content area shows the "Mouse actions" page with the URL [Documentation / WebDriver / Actions API / Mouse v4.0](#). It includes a brief description of mouse actions and a code example for Python:

```
clickable = driver.find_element(By.ID, "clickable")
ActionChains(driver)\n    .click_and_hold(clickable)\n    .perform()
```

Below the code example is a link to "View full example on GitHub". The right sidebar contains a vertical list of related topics: Click and hold, Click and release, Alternate Button Clicks, Context Click, Back Click, Forward Click, Double click, Move to element, Move by offset, Offset from Element, Offset from Viewport, Offset from Current Pointer Location, Drag and Drop on Element, and Drag and Drop by Offset.

2.38 잊지말고 종료~

- close() vs quit()

```
driver.close()
```

✓ 0.1s

Python