웹 크롤링2와 동적 데이터 수집

made by Jay Hong

Check

웹 크롤링과 동적 데이터 수집

1. 웹 크롤링2?

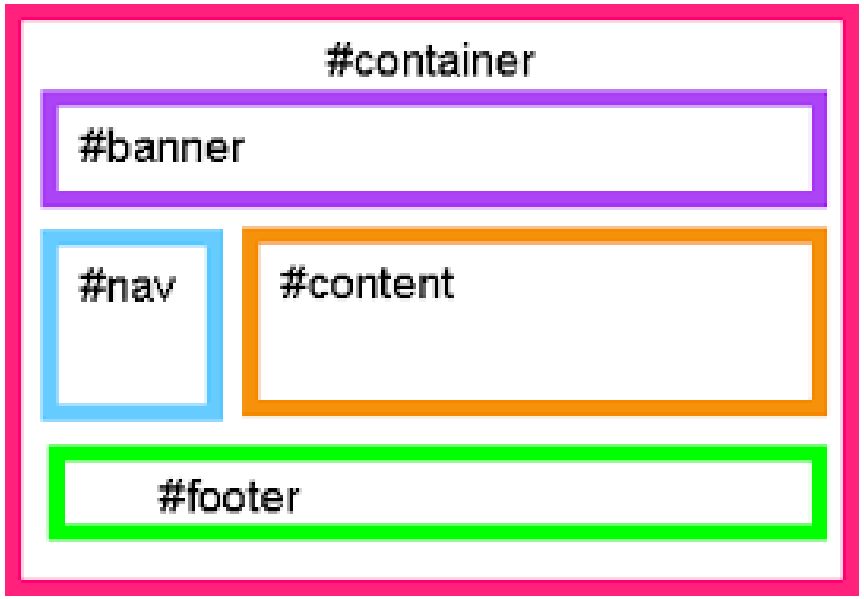
2. 동적 데이터 수집 방법

1. 웹 크롤링이란?

1. HTML의 구조 2

1) 컨테이너

✓ tag를 담고있는 html을 “컨테이너”라고 부릅니다.



1) 컨테이너

✓ 지난 시간에 실습했던 뉴스 기사를 떠올려 봅시다.

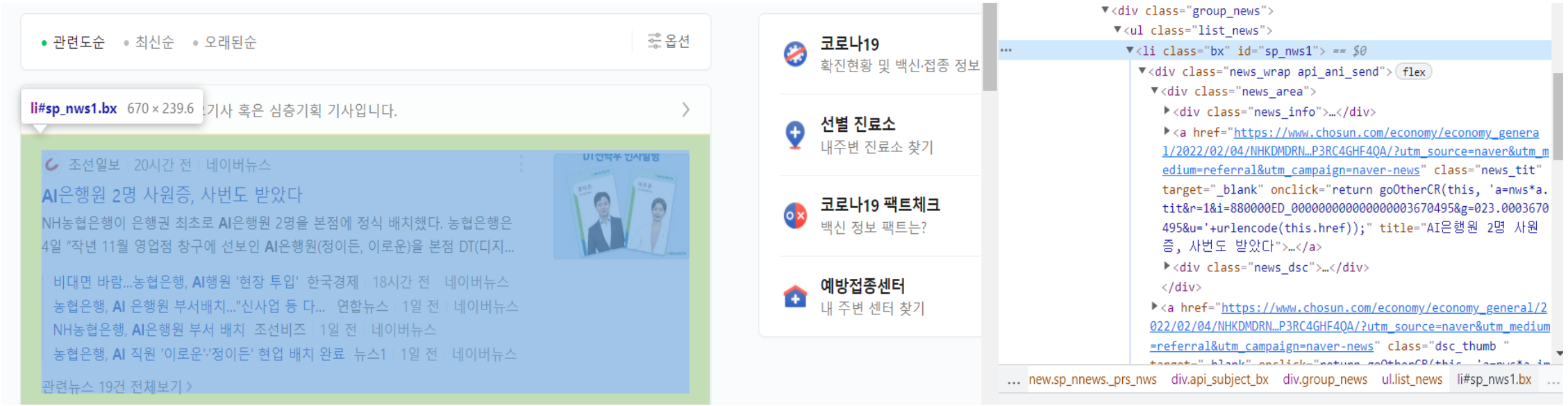
```
<ul class="list_news"> == $0
  <li class="bx" id="sp_nws1">…</li>
  <li class="bx" id="sp_nws6">…</li>
  <li class="bx" id="sp_nws11">…</li>
  <li class="bx" id="sp_nws16">…</li>
  <li class="bx" id="sp_nws19">…</li>
  <li class="bx" id="sp_nws21">…</li>
  <li class="bx" id="sp_nws26">…</li>
  <li class="bx" id="sp_nws27">…</li>
  <li class="bx" id="sp_nws28">…</li>
  <li class="bx" id="sp_nws31">…</li>
</ul>
```

✓ 은 리스트를 의미하고
 는 리스트의 원소를 의미합니다.

✓ 여기서 로 감싸진 부분이 컨테이너 입니다.

1) 컨테이너

- ✓ 또한 <li> 역시 다른 구조의 컨테이너 역할입니다.
- ✓ 위에 마우스를 올리면 다음과 같이 해당하는 부분에 색이 칠해집니다.
  이를 이용해서 원하는 부분을 정확히 찾아갈 수 있습니다.

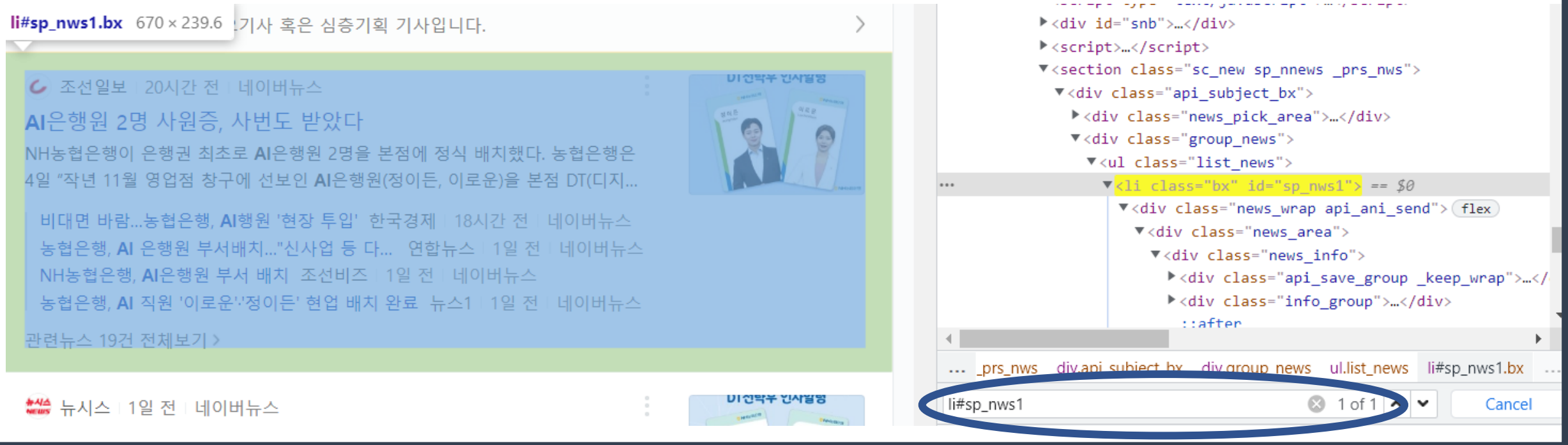


2) 원소 명칭 selector

- ✓ selctor에서는 다음과 같이 원소를 찾을 수 있습니다.
- ✓ tag.클래스_이름
  - ✓ example) li.bx : li태그에 class가 bx인 곳을 찾습니다.
  - ✓ 해당 이름을 가진 원소가 총 16개가 존재함을 알 수 있습니다.

2) 원소 명칭 selector

- ✓ selctor에서는 다음과 같이 원소를 찾을 수 있습니다.
- ✓ tag#id_이름
  - ✓ example) li#sp_nws1 : li태그에 id가 sp_nws1인 곳을 찾습니다.
  - ✓ 해당 이름을 가진 원소가 총 1개가 존재함을 알 수 있습니다.



Q) 어떻게 li를 이용해서 제목이 담긴 container를 찾을 수 있을까요?

- ✓ li의 class 이름을 사용하면 16개가 나오고
  li의 id 이름을 사용하면 1개만 나옵니다.

- ✓ 하지만 한 페이지당 뉴스는 10개씩 들어가 있기에 정확히 찾기가 어렵습니다.

A) container를 이용합시다.

- ✓ li는 상위 구조인 의 하위 구조입니다.
- ✓ 즉, 아래의 구조처럼 이루어져 있습니다.

```
<ul>
        <li>
        <li>
</ul>
```

- ✓ 뉴스 기사를 담고있는 container에 접근한 후에, li에 접근하면 우리가 원하는 부분만 찾을 수 있습니다.

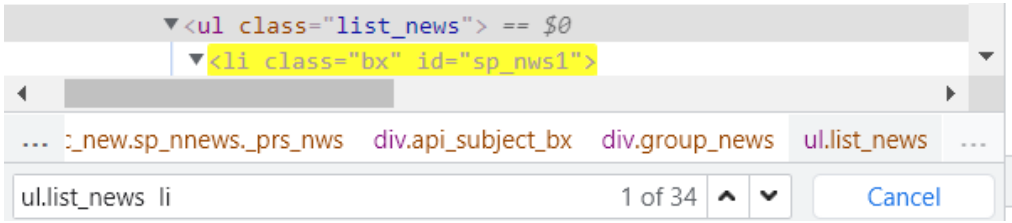
A) container를 이용합시다.

- ✓ 을 select해봅시다.

```
▼<ul class="list_news"> == $0
```

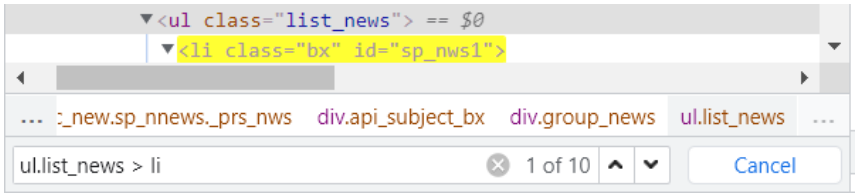
ul.list_news

- ✓ ul의 하부구조를 나타내는 방법은 두가지가 있습니다.



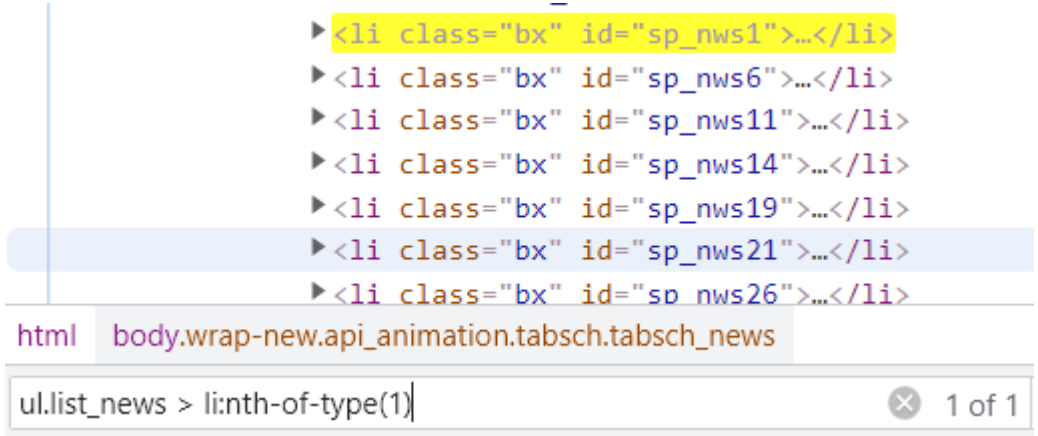
  - ✓ ul.list_news li
    이 방법은 ul.list_news 컨테이너 속에 li라는 tag를 모두 찾습니다.
    이를 사용할 때는 더 정확히 "ul.list_news li.bx" 라고 지정해서 찾아줍시다.

  - ✓ ul.list_news > li
    이 방법은 ul.list_news 컨테이너에 직속으로 연결된 li라는 tag를 모두 찾습니다.



Q) “첫번째“ 원소만 가져오는 방법은 무엇이 있을까요?

- ✓ 방법 1 : select를 이용해서 모두 가져온 후, indexing 을 사용한다.
- ✓ 방법 2 : selct_one을 이용해서 첫번째 만 가져온다.
- ✓ 방법 3 : nth-of-type(1)을 사용합니다.
  - ✓ nth-of-type은 형제 node, 즉 동일한 구조를 가진 node의 순서를 이용합니다.
  - ✓ 이 경우에는, ul.list_news > li:nth-of-type(1) 을 입력하면 됩니다.



2. 동적 데이터 수집 방법?

1. Selenium
2. Selenium으로 수집하기
3. 실습

1) Selenium이란?

- ✓ Web Browser를 제어하는 Library
- ✓ 실제 사람처럼 웹 브라우저를 제어하며 접속 가능
- ✓ request보다는 속도가 느리지만, 동적 데이터를 수집할 수 있다.

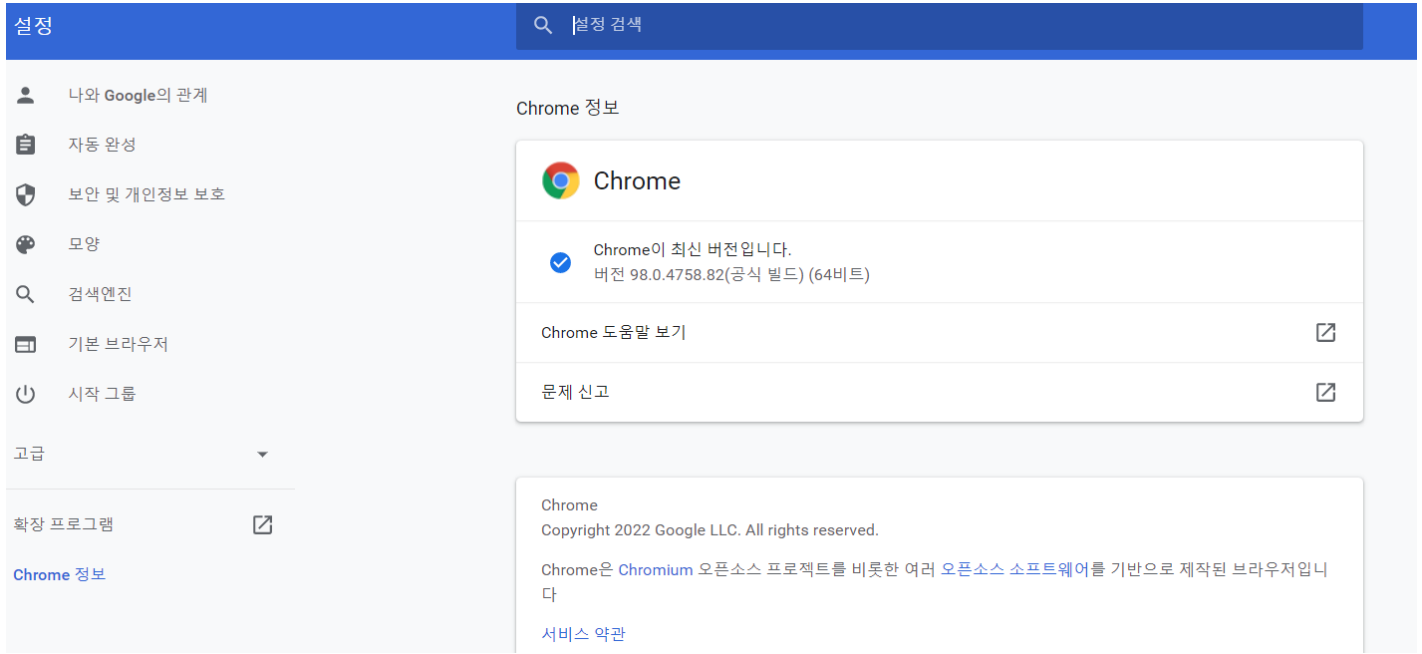
1) Selenium이란?

- ✓ Web Browser를 제어하는 Library
- ✓ 실제 사람처럼 웹 브라우저를 제어하며 접속 가능
- ✓ request보다는 속도가 느리지만, 동적 데이터를 수집할 수 있다.

2) Selenium 사용법 - 1

- ✓ Selenium을 이용해 Chrome Browser를 사용하기 위해서는 chromedriver.exe를 설치해야 한다.

- ✓ 이를 위해서는 자신의 chrome version을 체크해야 합니다
  크롬 우측 상단 톱니바퀴 -> 설정 -> 좌측 하단 Chrome 정보 -> 버전 확인



2) Selenium 사용법-1

- ✓ 설치 페이지
  https://chromedriver.chromium.org/downloads
  해당 페이지에 들어가서 버전에 맞는 chrome driver를 설치하고
  사용할 폴더에 추가해두면 된다.

- ✓ 장점
  - ✓ 한 번 설치해두면, 다시 다운로드 할 필요 없이 불러오기만 하면 된다.

- ✓ 단점
  - ✓ Chrome Version과 driver version이 다르면 불러와지지 않는다.
  - ✓ local에 자동화 프로그램을 만들어도, 버전이 다르면 돌아가지 않아 문제가 생길 수 있다.

2) Selenium 사용법-2

- ✓ 위의 단점을 보완하기 위해
  webdrivier_manager 라는 라이브러리를 사용

- ✓ 해당 라이브러리는 자신의 버전에 맞는 drivier를 자동으로 설치해준다.

- ✓ Selenium은 하나의 인터넷 브라우저를 사용하는 역할이다.
  인터넷 브라우저는 접속할 때 마다 캐시데이터가 쌓인다.
  인터넷 브라우저가 감당 가능한 캐시 데이터가 넘어가면, 로딩에 엄청난 시간이 걸린다.
  따라서, 특정 시간이나 횟수 반복마다 인터넷 브라우저를 껏다 켜야하는데
  이 과정에서 매번 다운로드해야하니 속도 측면에서의 아쉬움이 생긴다.

목표

- ✓ 네이버 웹툰의 회차별 제목 수집하기

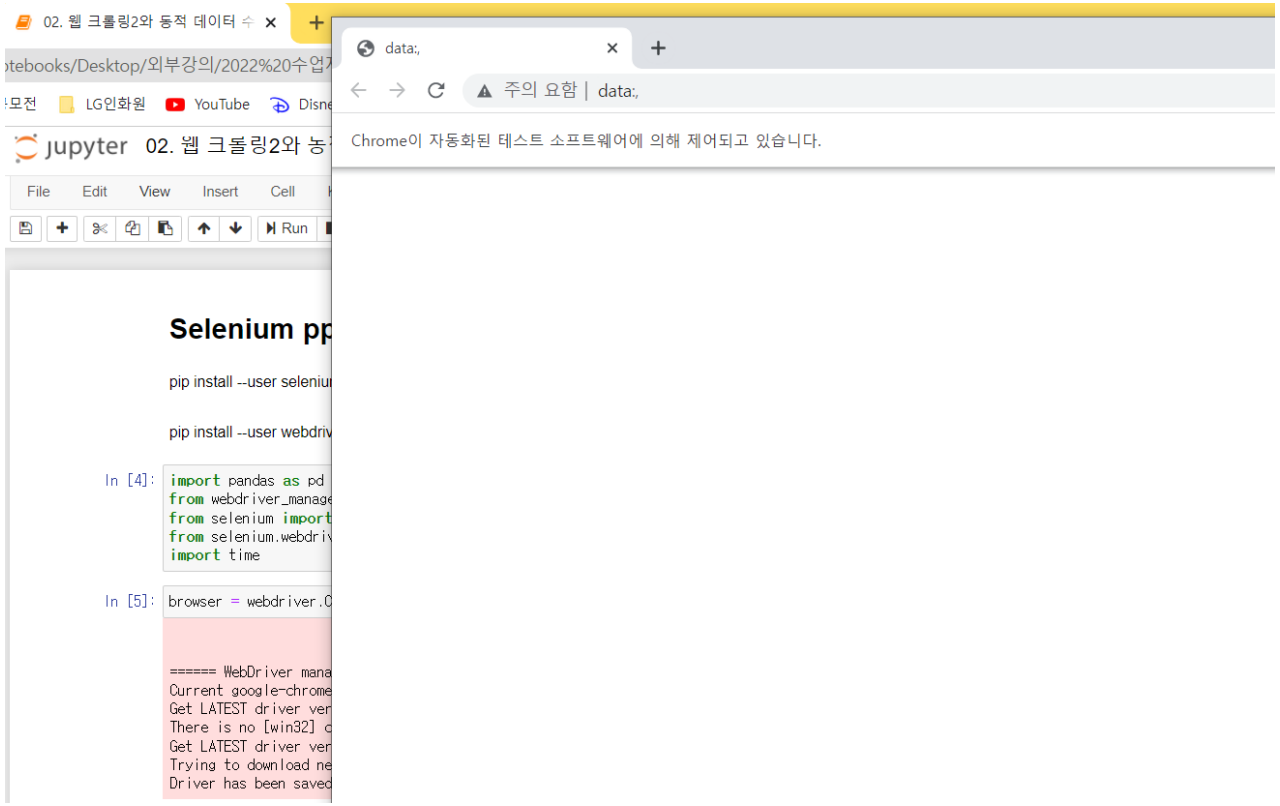
1) Library

```
import pandas as pd
from webdriver_manager.chrome import ChromeDriverManager
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
import time
```

- ✓ 위의 라이브러리를 사용할 예정이다.
  - ✓ pandas : 데이터 저장 목적
  - ✓ ChromeDriverManager : 크롬 드라이버 설치 목적
  - ✓ webdriver : 자동화된 웹페이지 탐색
  - ✓ Keys : 키보드나 클릭 등의 역할을 지원
  - ✓ time : 진행 시간 컨트롤
    - ✓ 너무 빠른 시간 내에 반복 수집을 하면, 해커 공격으로 오인 가능
    - ✓ 페이지를 로딩하는데 오랜 시간이 걸릴 수도 있어 time.sleep()으로 컨트롤

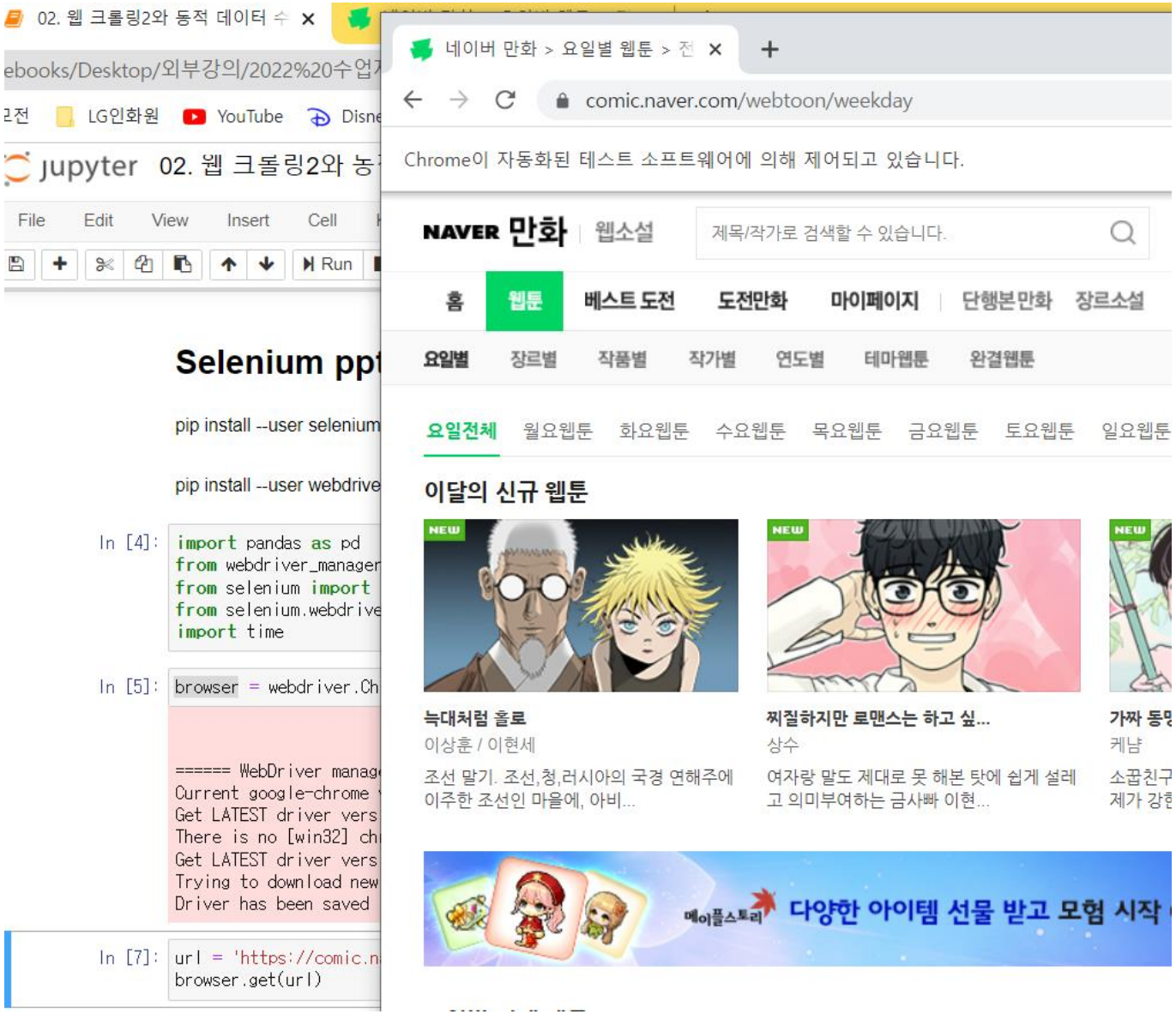
2) webdriver 연동

- ✓ browser = webdriver.Chrome(ChromeDriverManager().install())
- ✓ 크롬이 하나 자동으로 열리며, 이를 이용해서 자동화할 수 있다.



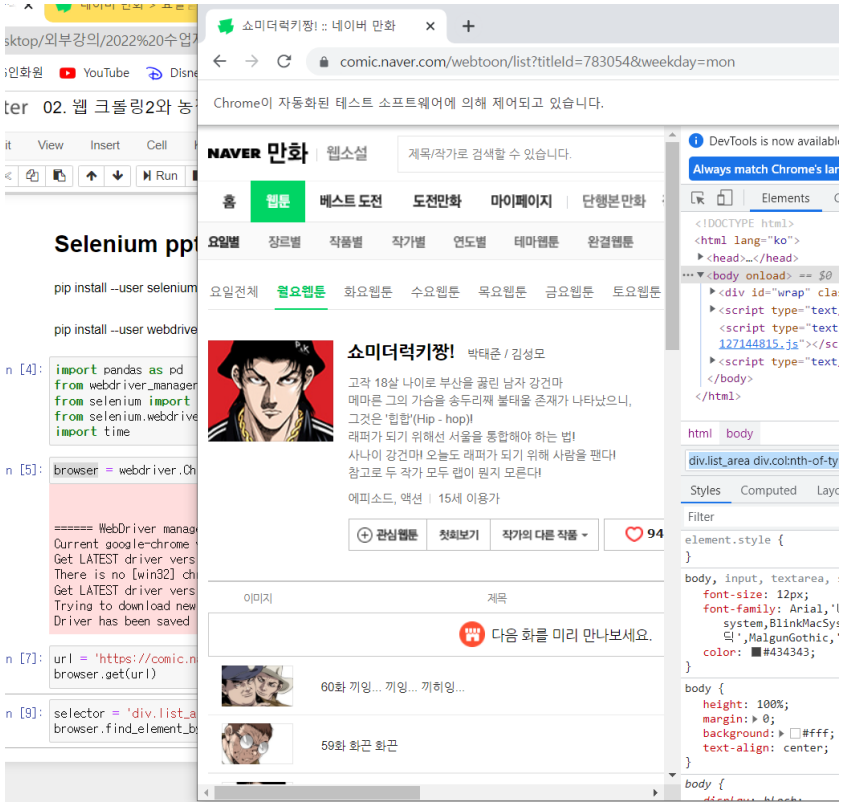
3) url로 이동

- ✓ browser.get(url)



4) 월요일 1등 웹툰으로 이동

- ✓ selector 찾은 후에
- ✓ browser.find_element_by_css_selector(selector).click()



5) 제목 수집

- ✓ selector 찾은 후에
- ✓ elements = browser.find_elements_by_css_selector(selector)
- ✓ 반복문을 이용해 elements 개별 원소에 .text로 제목을 받을 수 있다.
- ✓ 주의점)
  페이지에서 element를 저장한 후에,
  text를 받지 않고 다른 페이지로 넘어간다면
  나중에 이 element는 다시 불러올 수 없습니다.

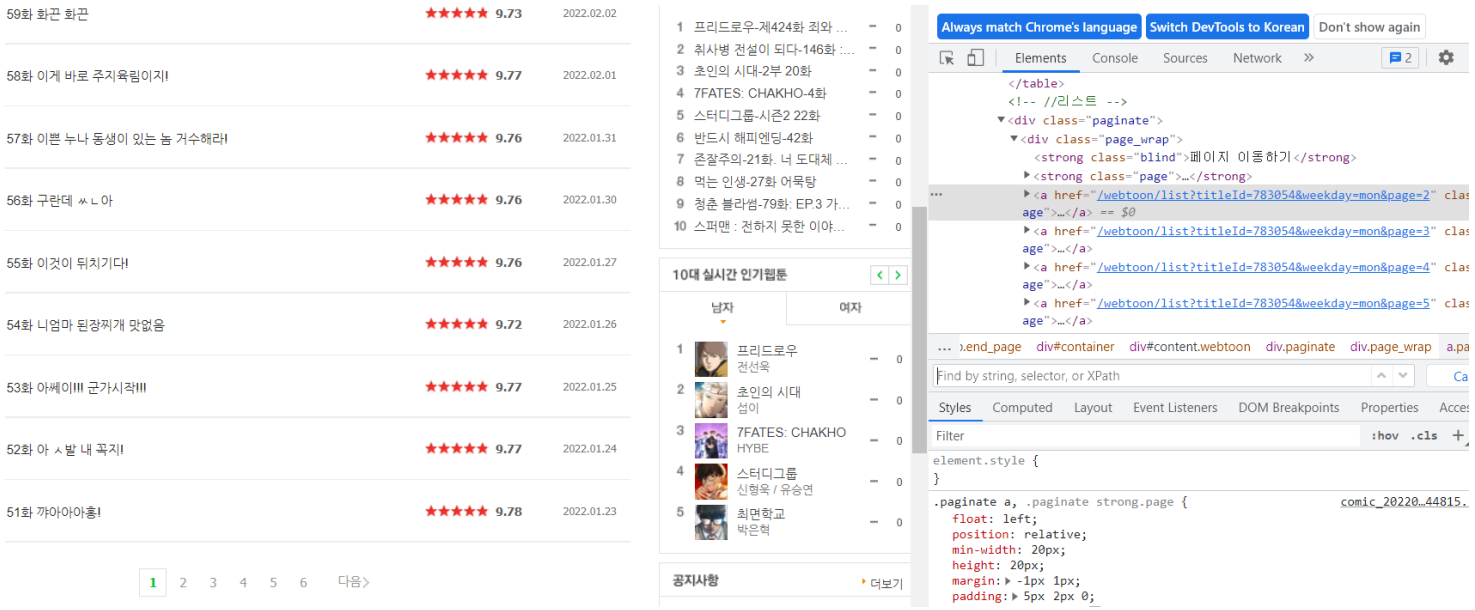
  example)
  1페이지에서 element만 가져온 후에
  2페이지로 넘어가면 text가 없다고 나옵니다.

```
title_path = 'td.title'
titles = browser.find_elements_by_css_selector(title_path)
titles
```

```
[<selenium.webdriver.remote.webelement.WebElement (session="988bcb7d16b2bd1af5cbcd6402accf97", element="06b54e81-aeac-450e-998c-83de037559dc")>,
 <selenium.webdriver.remote.webelement.WebElement (session="988bcb7d16b2bd1af5cbcd6402accf97", element="d624a8bd-17b3-47db-9b27-06ce22df6da2")>,
 <selenium.webdriver.remote.webelement.WebElement (session="988bcb7d16b2bd1af5cbcd6402accf97", element="51abf067-21c1-414b-aae4-4555708fe094")>,
 <selenium.webdriver.remote.webelement.WebElement (session="988bcb7d16b2bd1af5cbcd6402accf97", element="f3932045-de5a-4a8c-8c71-fb826a918320")>,
 <selenium.webdriver.remote.webelement.WebElement (session="988bcb7d16b2bd1af5cbcd6402accf97", element="eba1a40f-655f-447c-b0b6-5da2e7b496d2")>,
 <selenium.webdriver.remote.webelement.WebElement (session="988bcb7d16b2bd1af5cbcd6402accf97", element="189687f1-625e-47e0-b206-5c33f3b7267d")>,
 <selenium.webdriver.remote.webelement.WebElement (session="988bcb7d16b2bd1af5cbcd6402accf97", element="34cf296b-636d-4b2e-af11-39c8d0e6cb15")>,
 <selenium.webdriver.remote.webelement.WebElement (session="988bcb7d16b2bd1af5cbcd6402accf97", element="e5dbbcfc-1cfe-48d8-83e9-4670df093282")>,
 <selenium.webdriver.remote.webelement.WebElement (session="988bcb7d16b2bd1af5cbcd6402accf97", element="18fdcbb1-aaa9-492b-80d8-e1fee96e7bf2")>,
 <selenium.webdriver.remote.webelement.WebElement (session="988bcb7d16b2bd1af5cbcd6402accf97", element="e98efb65-55ea-4d4b-8b94-453f4270cf17")>]
```

6) 다음 페이지로 이동

- ✓ 첫 페이지의 데이터를 수집했으니, 다음 페이지로 이동해보겠습니다.
- ✓ 2 페이지의 버튼에 해당하는 selector를 찾은 후
- ✓ browser.find_element_by_css_selector(selector).click() 을 해주면 됩니다.



7) 반복문으로 모든 제목을 수집해봅시다.

- ✓ 네이버 웹툰은 url에 page_num이 있으니 selector가 아닌 url로 접근하며 찾아봅시다.
- ✓ 이 과정에서 try, except문 등 제어문을 사용하면 편합니다.

자동화

```
host_url = 'https://comic.naver.com/webtoon/list?titleId=783054&weekday=mon&page='

page_num = 1

korean_titles = []
start_length = len(korean_titles)
while True  :
    url = host_url + str(page_num)
    browser.get(url)
    time.sleep(0.5)
    title_path = 'td.title'
    titles = browser.find_elements_by_css_selector(title_path)
    titles = [title.text for title in titles]
    korean_titles.extend(titles)
    korean_titles = list(set(korean_titles))

    if start_length < len(korean_titles) :
        print(f'{page_num}페이지 수집 결과 {len(korean_titles)}개의 제목 수집')
        page_num += 1
        start_length = len(korean_titles)
    else :
        print(f'수집을 중단합니다.')
        print(f'최종 수집 결과 {len(korean_titles)}개의 제목 수집 완료.')
        break;
```

```
1페이지 수집 결과 10개의 제목 수집
2페이지 수집 결과 20개의 제목 수집
3페이지 수집 결과 30개의 제목 수집
4페이지 수집 결과 40개의 제목 수집
5페이지 수집 결과 50개의 제목 수집
6페이지 수집 결과 60개의 제목 수집
수집을 중단합니다.
최종 수집 결과 60개의 제목 수집 완료.
```

목표 : 서울과 부산의 인기음식점을 300개씩 수집합시다.

- ✓ https://www.tripadvisor.co.kr/
- ✓ 위의 페이지에 검색에 서울과 부산을 검색하고,
  하단의 인기음식점 (평점순)을 기준으로 페이지를 이동하며 음식점 이름 300개를 수집합시다.