



파이썬 크롤링 입문

03 Selenium 심화: 브라우저 제어



목차

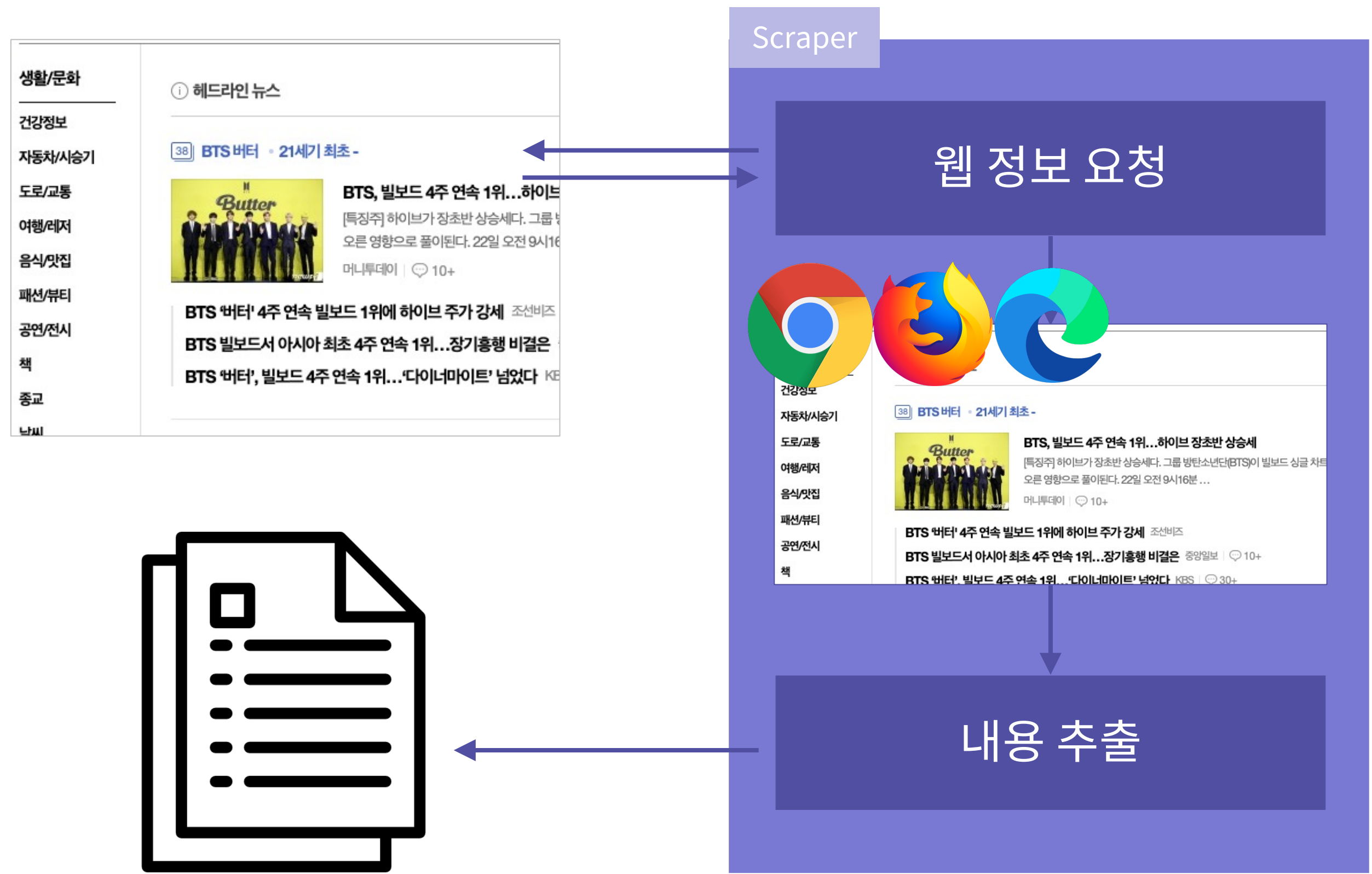
- 01. Selenium을 이용한 브라우저 제어
- 02. 브라우저 로딩 기다리기
- 03. 키보드/마우스 입력
- 04. 다양한 입력, ActionChains
- 05. 맺으며

01

Selenium을 이용한 브라우저 제어

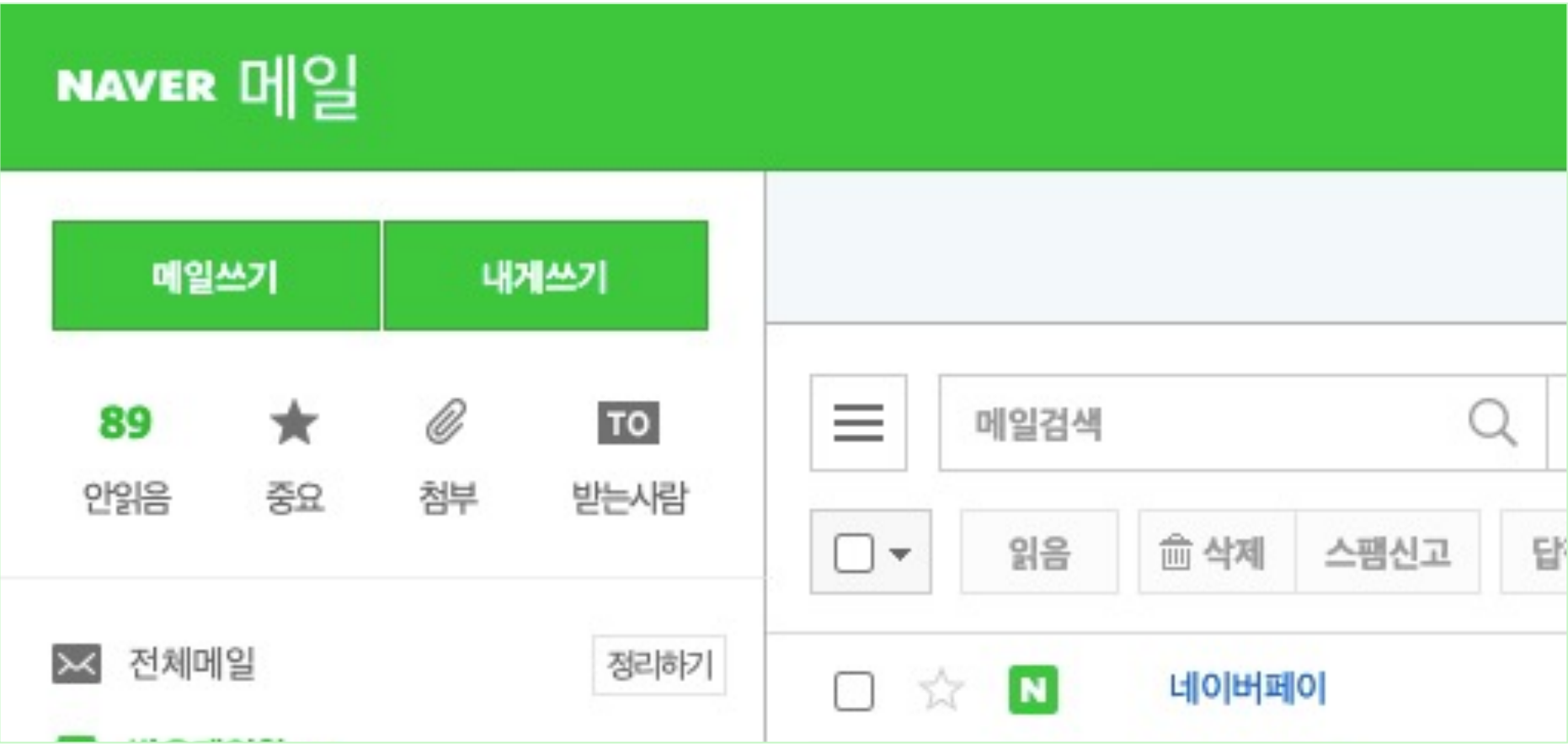


✓ Selenium을 통한 웹 스크래핑 과정 모식도



웹의 정보를 받아와서, 해석하고, 추출한다.

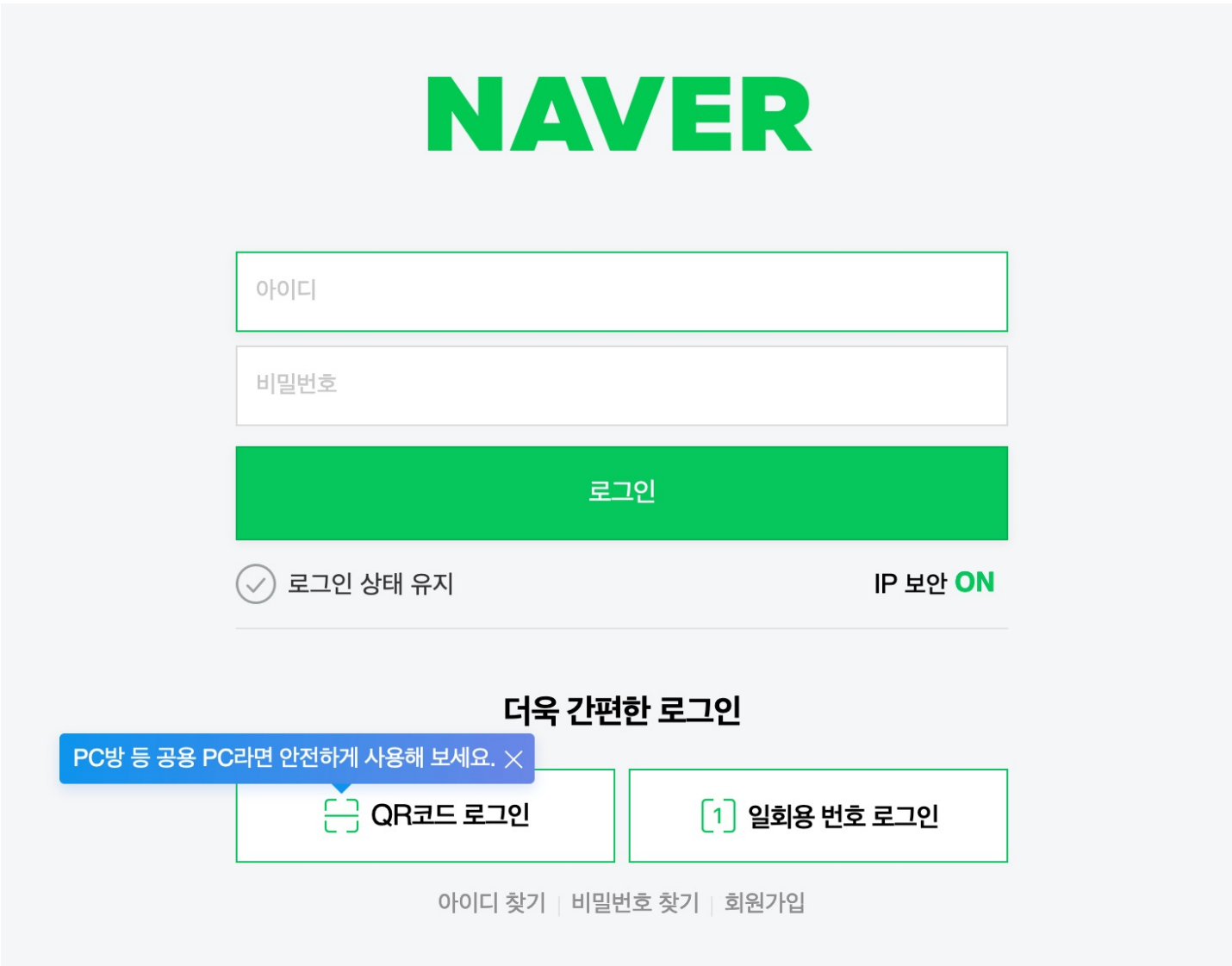
✔ 로그인에 필요한 페이지



```
driver.get('https://mail.naver.com/')
e_list = driver. \
    find_elements_by_xpath('//div/p')

for e in e_list:
    print(e.text)
```

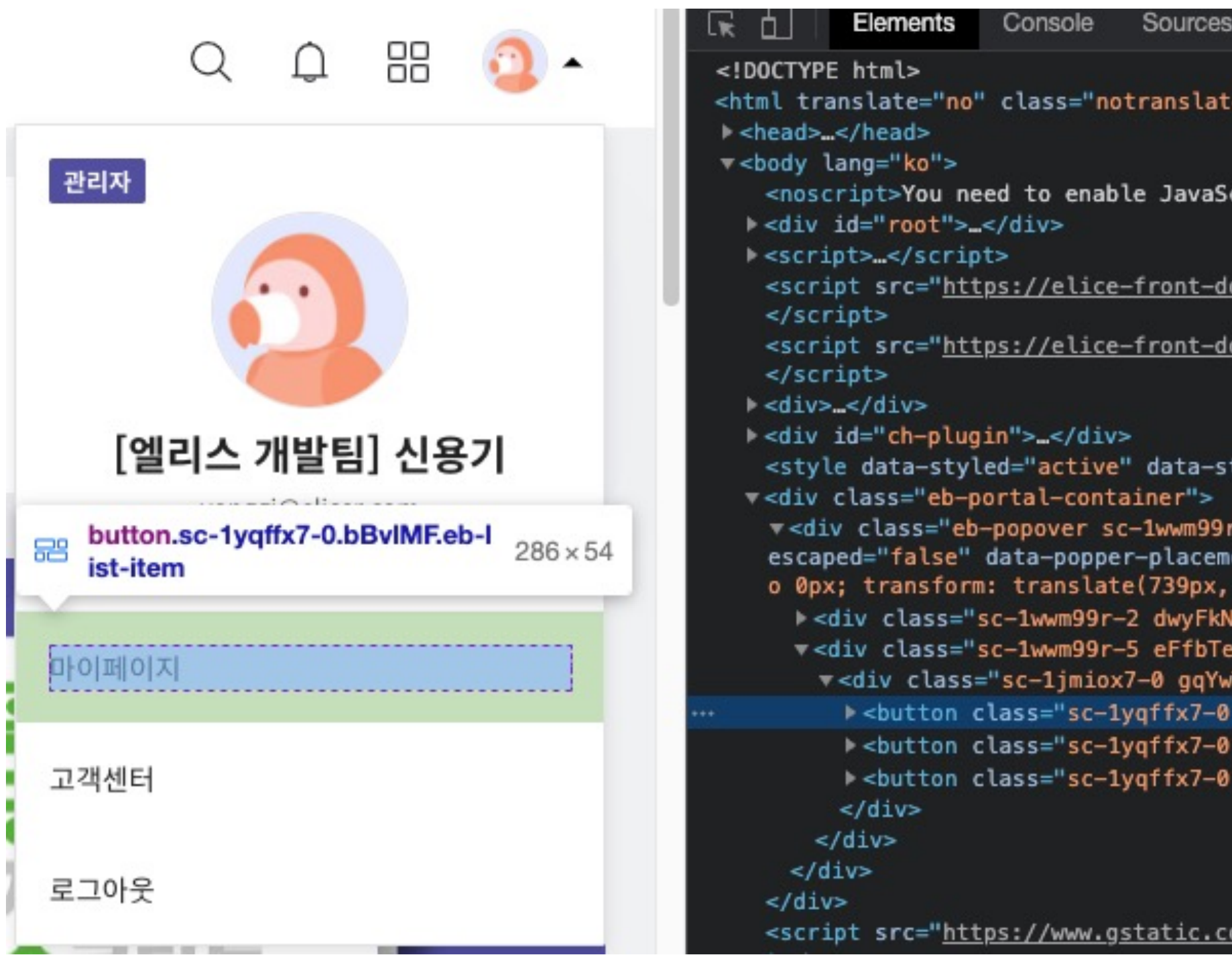
받은 메일함의 제목을 추출해야지!



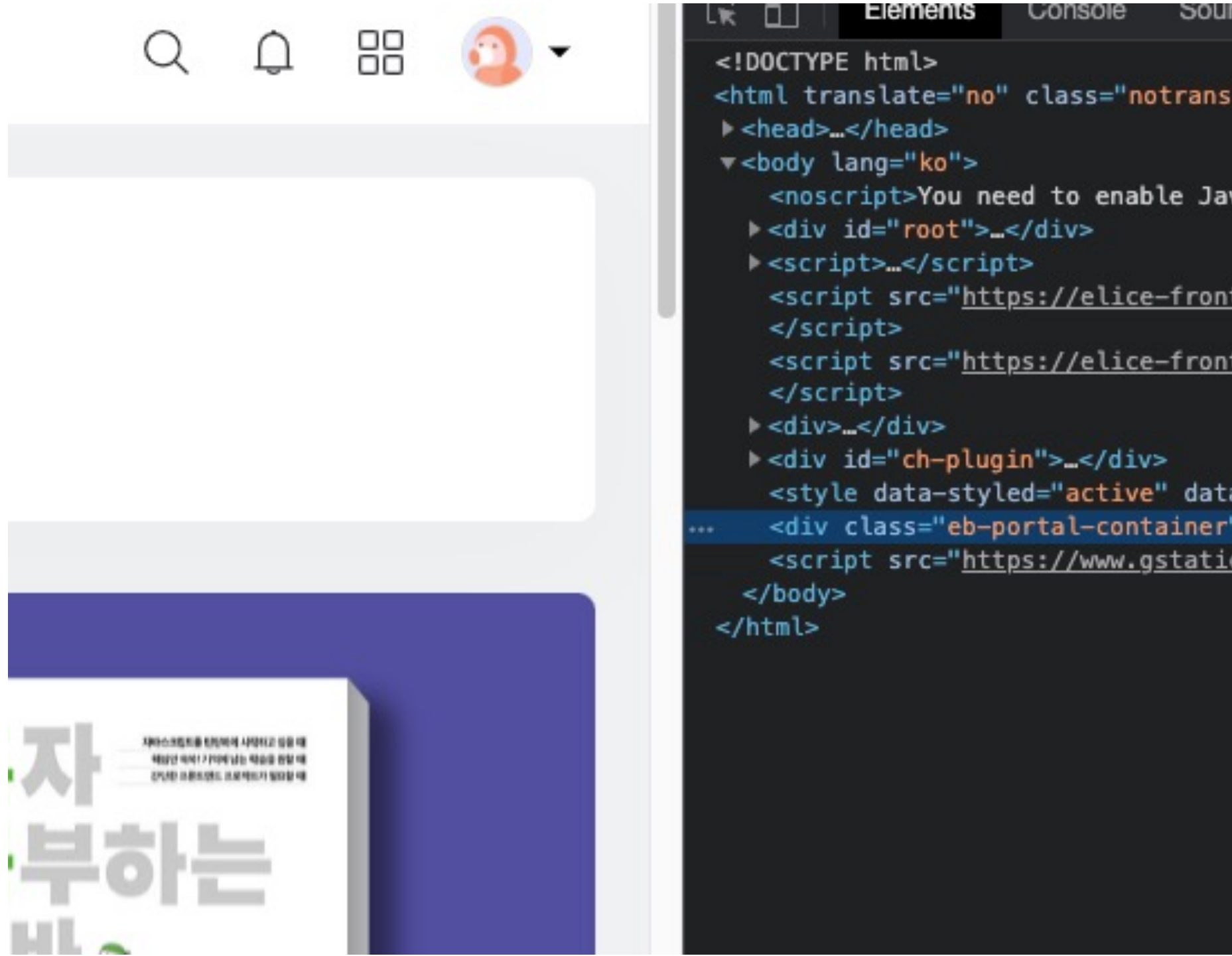
[실행 결과]
더욱 간편한 로그인

로그인이 필요

✓ 동적으로 렌더링되는 페이지



드롭다운 내리면 html에 있지만



닫으면 사라짐

✓ 브라우저 제어

1) 로그인 자동화

- 로그인 후에만 나오는 웹 페이지 분석을 위해
- ID와 비밀번호 입력 후 로그인 버튼 클릭 (또는 엔터키 입력)

=> 어떻게?

2) 드롭다운 버튼 클릭

- 드롭다운 버튼을 클릭해야만 나오는 요소의 추출을 위해
- 드롭다운 버튼을 찾아서 클릭

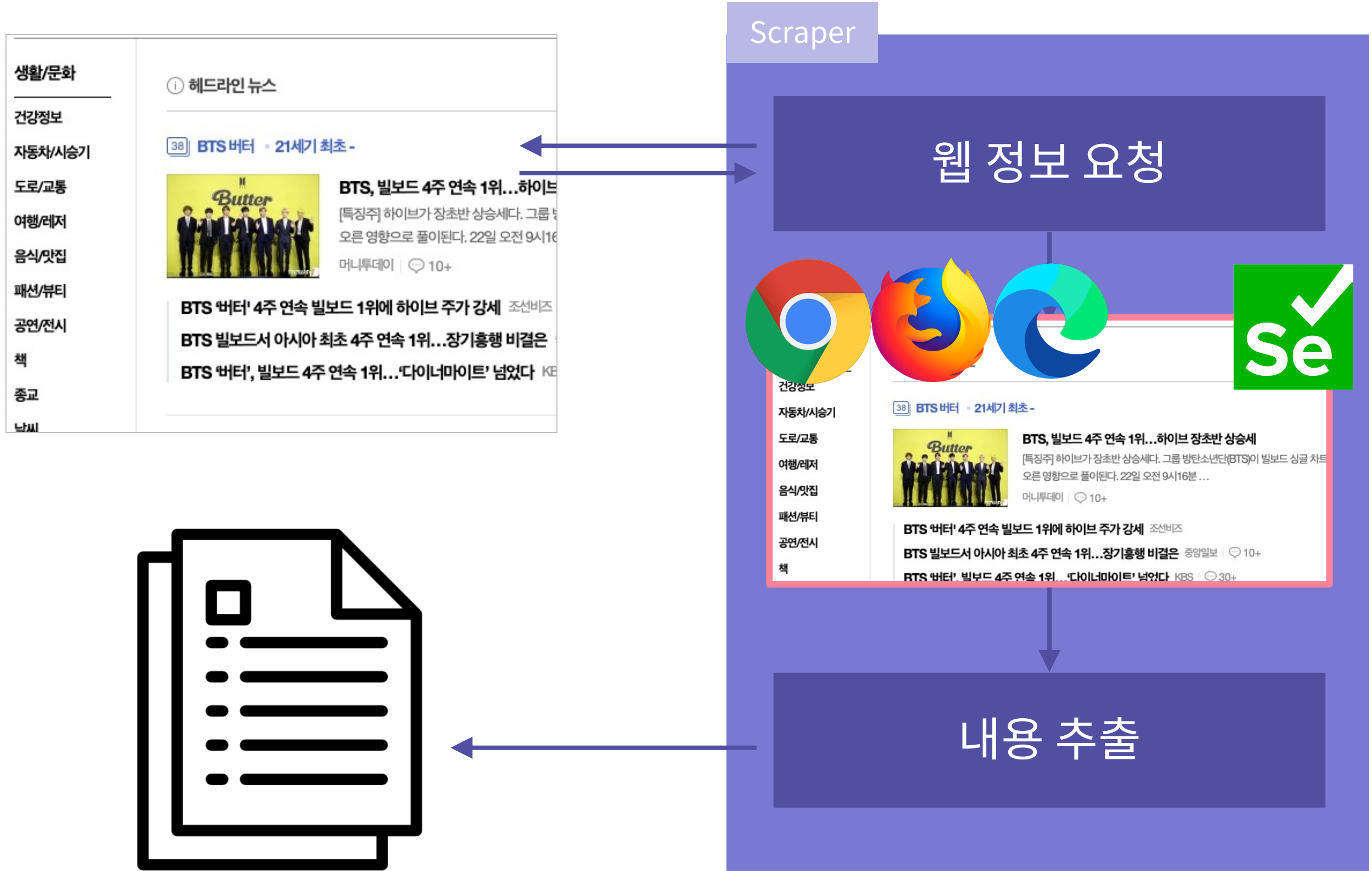
=> 어떻게?

✓ 브라우저 제어



답은 Selenium이다.

✔ 브라우저 제어



Selenium을 이용해 브라우저를 제어

02

브라우저 로딩 기다리기



✓ Scraping 동작 순서

Python

```
with webdriver.Firefox() as driver:  
    driver.get(url) # 웹 페이지 로딩  
  
    e = driver.find_element()  
    ...
```

< 이상적인 경우 >

with webdriver.Firefox() as driver

driver.get(url)

해당 웹 페이지 로딩

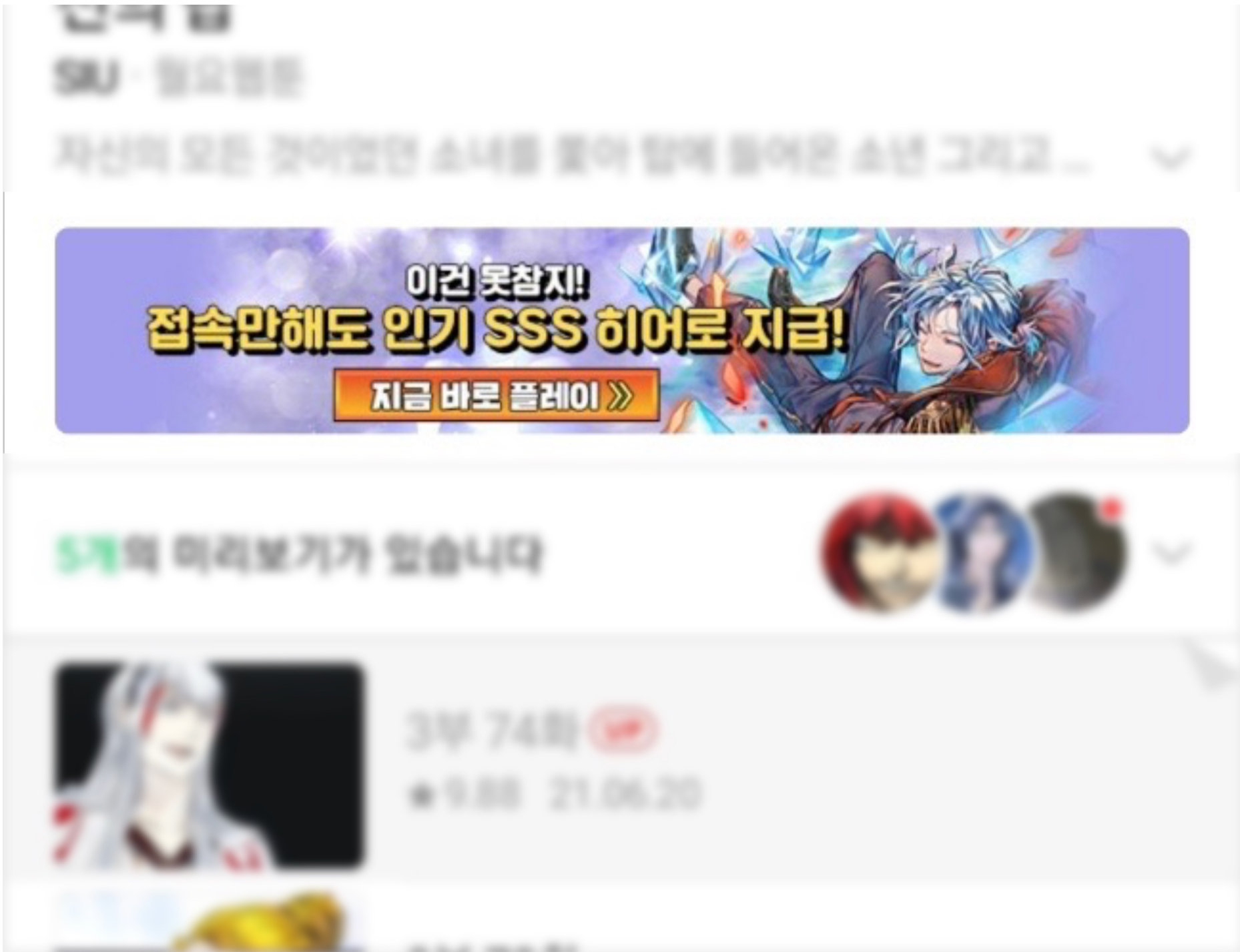
e = driver.find_element()

...

✔ 자연된 로딩 예시



페이지 로딩이 끝남



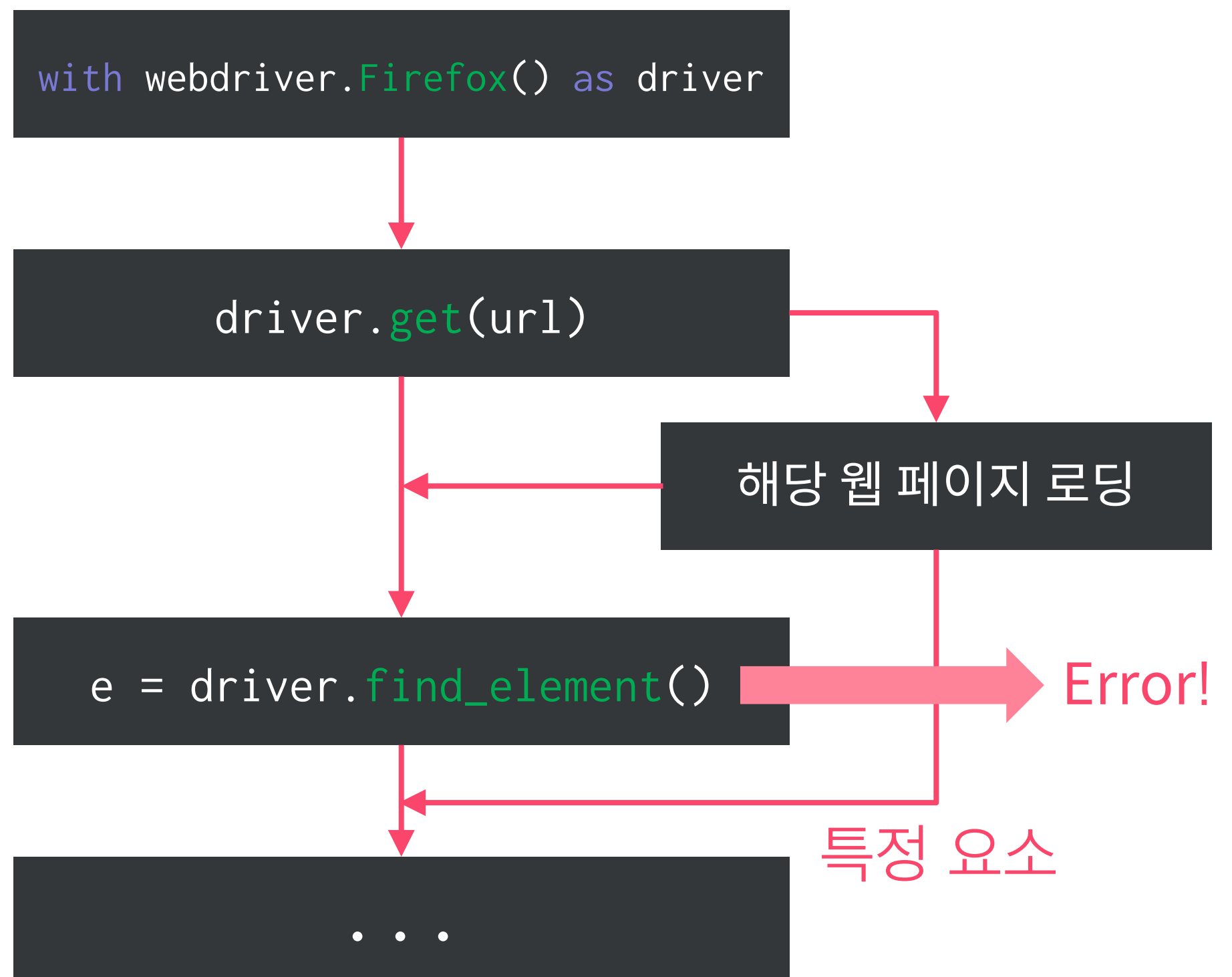
잠시 후에 광고가 보인다.

✓ Scraping 동작 순서: 지연된 로딩

Python

```
with webdriver.Firefox() as driver:  
    driver.get(url) # 웹 페이지 로딩  
  
    e = driver.find_element() # error!  
    ...
```

< JS로 인해 특정 요소의 로딩이 늦을 경우 >



✓ 무조건 기다리기

Python

```
import time

with webdriver.Firefox() as driver:
    driver.get(url) # 웹 페이지 로딩
    time.sleep(10) # 10초간 기다리기

    e = driver.find_element()
    ...
```

time.sleep(n)

- python 내장 라이브러리
- **n**초만큼 **무조건** 기다림
- 내가 원하는 요소가 **불러와 졌어도** 주어진 시간을 계속 기다림

✓ 무조건 기다리기

Python

```
import time

with webdriver.Firefox() as driver:
    driver.get(url) ←
    time.sleep(10)

    e = driver.find_element()
    ...
```



✔ 무조건 기다리기

Python

```
import time

with webdriver.Firefox() as driver:
    driver.get(url)
    time.sleep(10) ←
    e = driver.find_element()
    ...
```

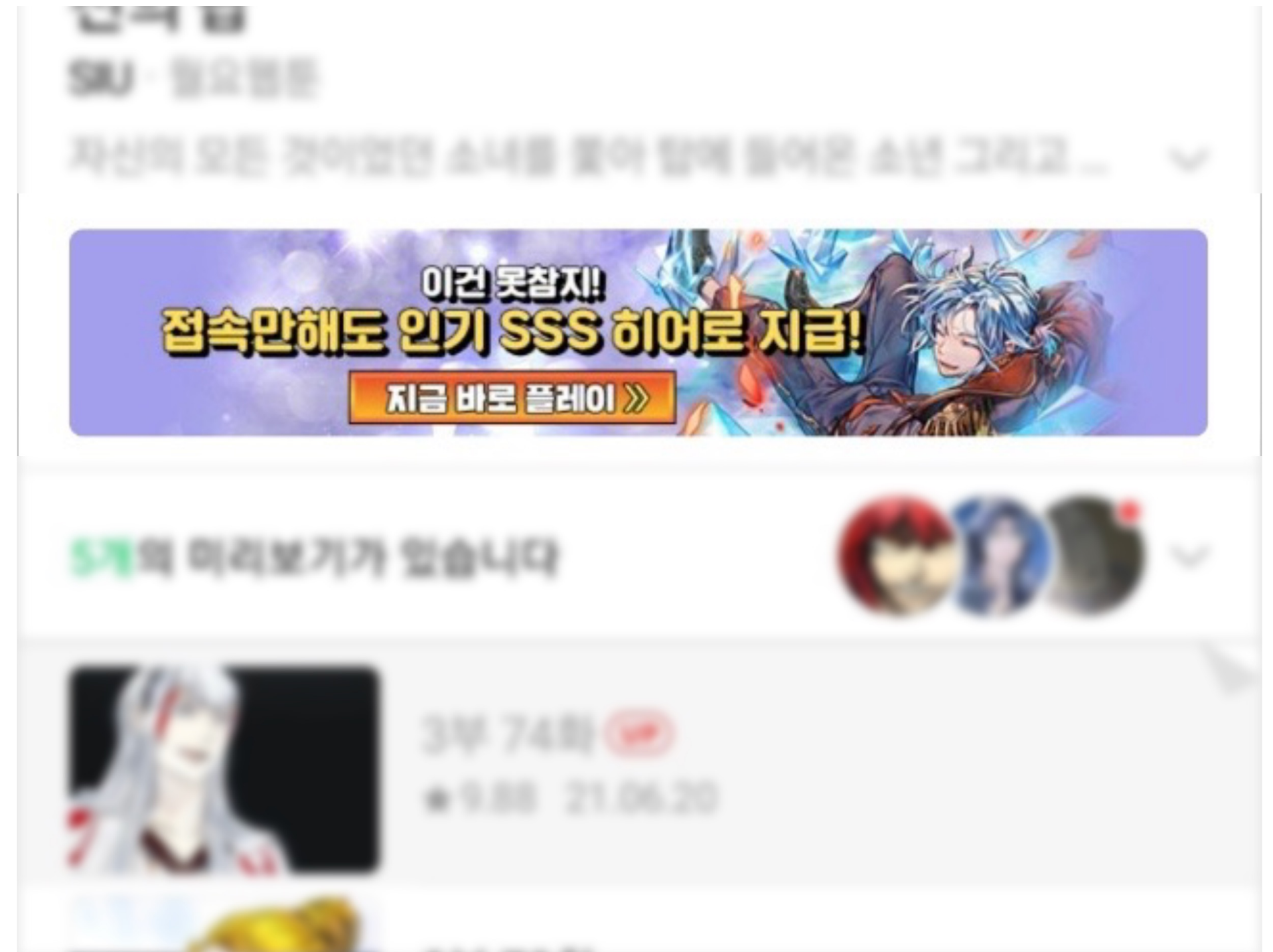


✓ 무조건 기다리기

Python

```
import time

with webdriver.Firefox() as driver:
    driver.get(url)
    time.sleep(10) ←
    e = driver.find_element()
    ...
```



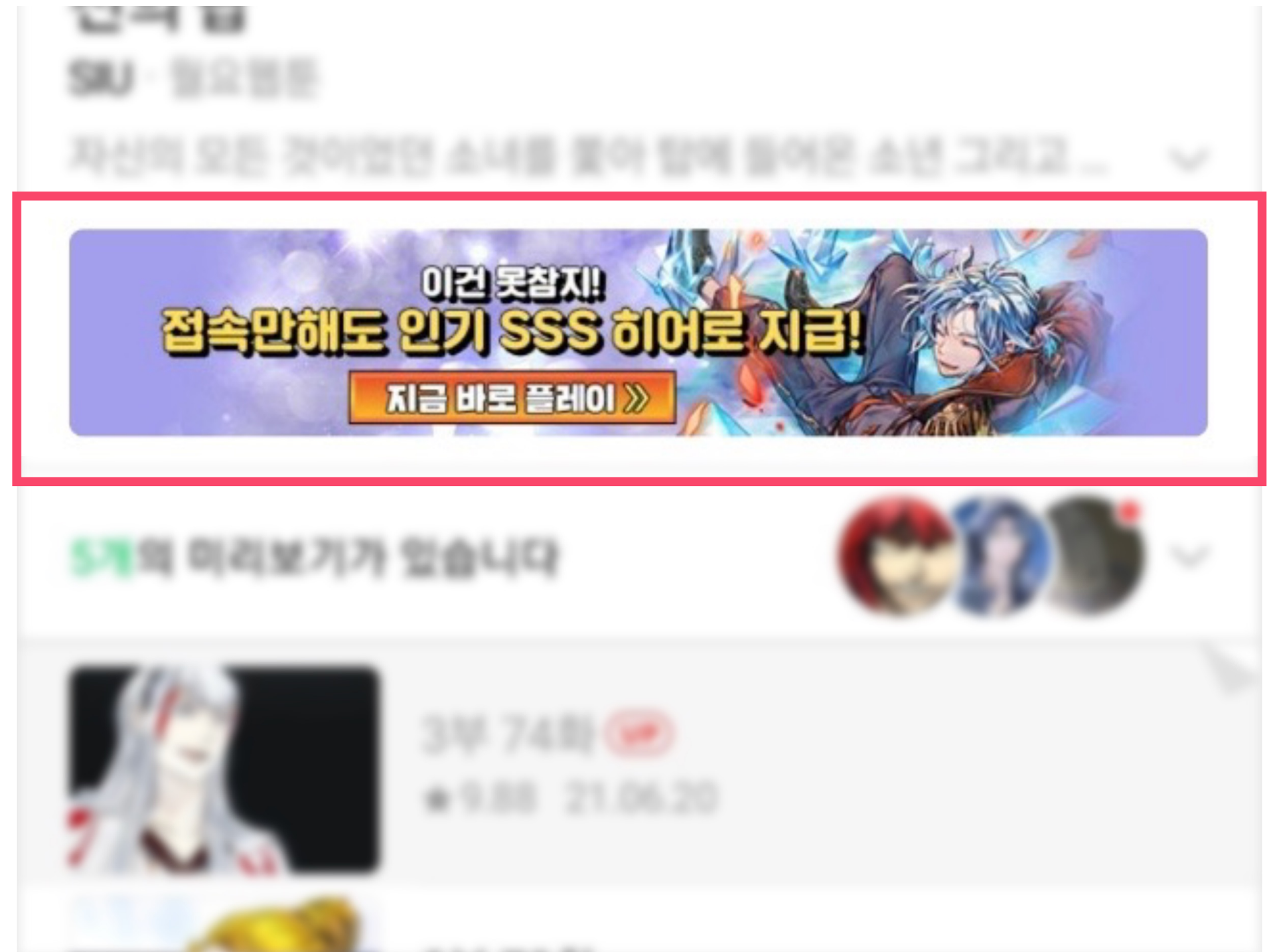
✔ 무조건 기다리기

Python

```
import time

with webdriver.Firefox() as driver:
    driver.get(url)
    time.sleep(10)

    e = driver.find_element() ←
    ...
```



✓ 암시적 기다리기

Python

```
with webdriver.Firefox() as driver:  
    driver.implicitly_wait(10)  
  
    driver.get(url) # 웹 페이지 로딩  
  
    e = driver.find_element()  
    ...
```

implicitly_wait(n)

- 암시적으로 기다림 수행
- 앞으로 요소를 추출할 때 (find_element)
 최대 n초까지 기다림
- 해당 요소의 로딩이 끝나면 즉시 기다리기를
 종료하고 코드를 수행
- 한 번 설정해주면 해당 브라우저에 계속해서 적용

✓ 암시적 기다리기

Python

```
with webdriver.Firefox() as driver:  
    driver.implicitly_wait(10)  
  
    driver.get(url) ←  
  
    e = driver.find_element()  
    ...
```



✓ 암시적 기다리기

Python

```
with webdriver.Firefox() as driver:  
    driver.implicitly_wait(10)
```

```
    driver.get(url)
```

```
    e = driver.find_element() ← 
```

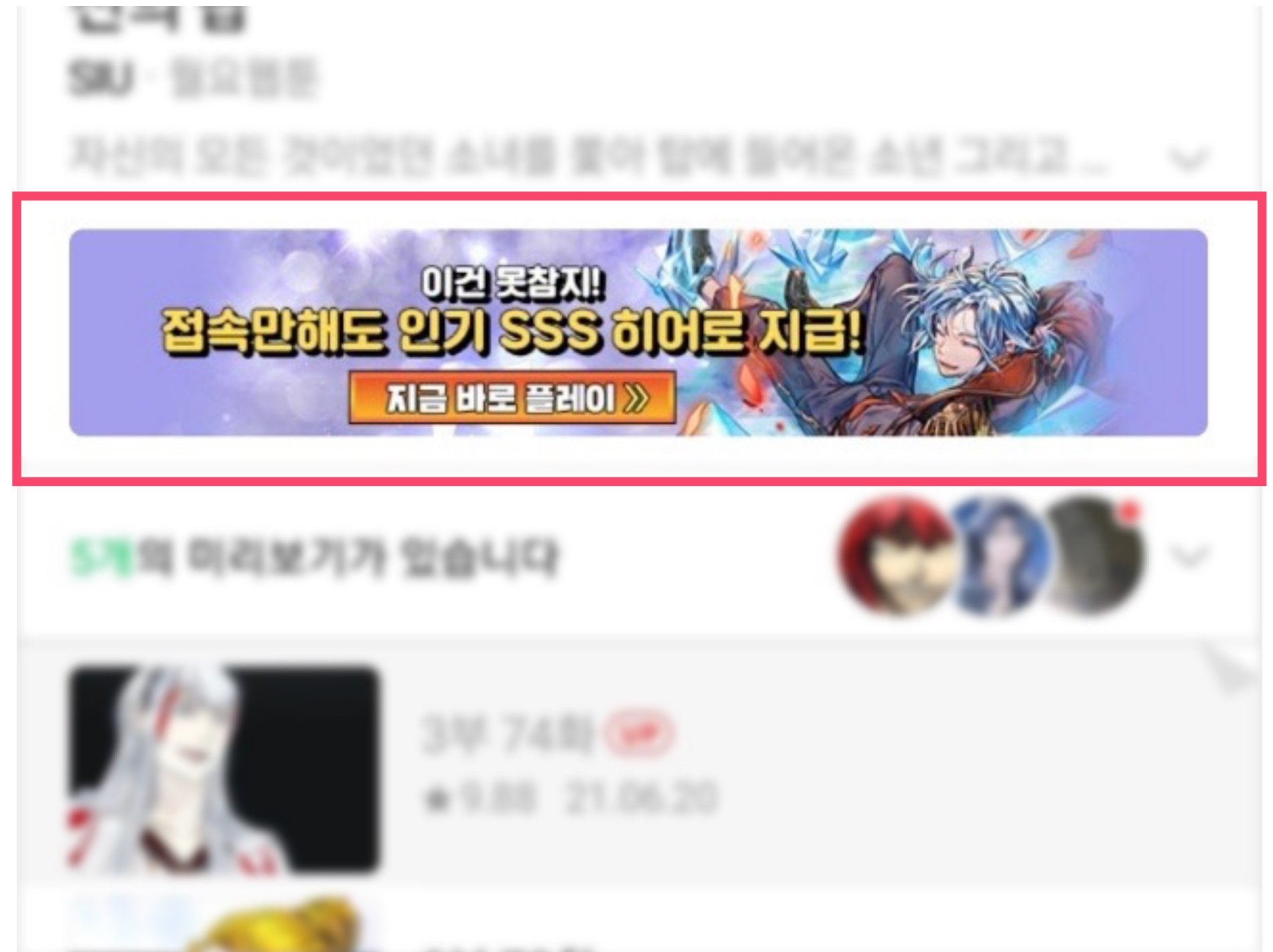
```
    ...
```



✓ 암시적 기다리기

Python

```
with webdriver.Firefox() as driver:  
    driver.implicitly_wait(10)  
  
    driver.get(url)  
  
    e = driver.find_element()  
    ...
```



✓ 명시적 기다리기

Python

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

with webdriver.Firefox() as driver:
    driver.implicitly_wait(10)
    driver.get(url) # 웹 페이지 로딩

    e = WebDriverWait(driver, 30).until(
        EC.presence_of_element_located((By.ID, "id_name"))
    ) # 해당 요소는 불러올 때까지 30초 기다림
    ...
```

명시적 기다리기

- 기다릴 요소를 **명시**
- 명시된 요소가 해당 방식으로 불러와질 때까지 **최대 n초** 기다림
- 요소는 **class_name, xpath** 등 다양한 방법으로 찾을 수 있음

presence_of_element_located

- 괄호 안의 요소가 **나타날 때까지** 기다림

✓ 명시적 기다리기

Python

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

with webdriver.Firefox() as driver:
    driver.implicitly_wait(10)
    driver.get(url) # 웹 페이지 로딩

    e = WebDriverWait(driver, 30).until(
        EC.presence_of_element_located((By.ID, "id_name"))
    ) # 해당 요소는 불러올 때까지 30초 기다림
    ...
```

(By.ID, "id_name")

- 요소를 찾는 방법과 파라미터를 의미
- id 외에 대표적으로 사용하는 것
 - (By.XPATH, "xpath")
 - (By.CLASS, "class_name")

element_to_be_clickable

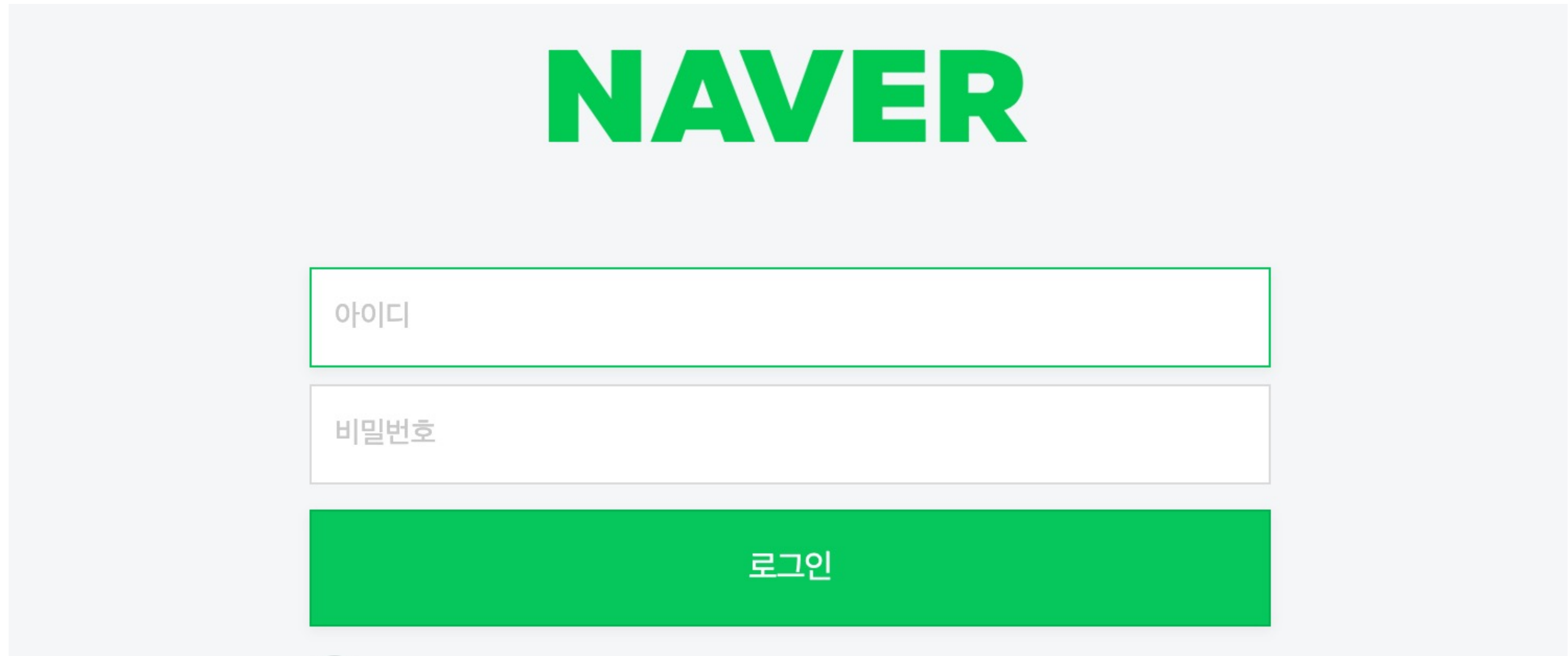
- 해당 요소가 **클릭 가능해질 때까지**
기다림

03

키보드/마우스 입력



✓ 로그인해야 볼 수 있는 페이지

A mockup of the Naver login page. At the top center is the word "NAVER" in a large, bold, green font. Below it are two white input fields with green borders. The first field contains the placeholder text "아이디" (ID) and the second field contains "비밀번호" (Password). Below these fields is a large green rectangular button with the white text "로그인" (Login) centered on it.

NAVER

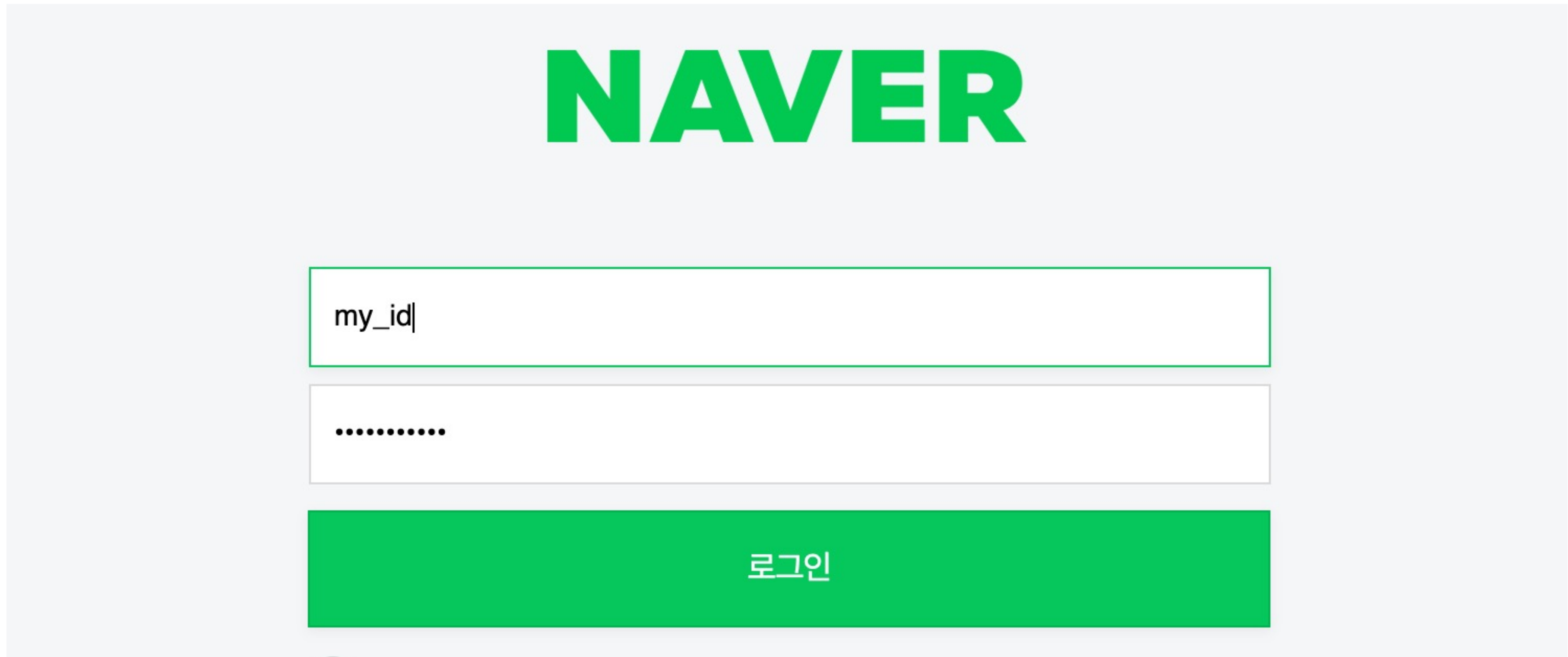
아이디

비밀번호

로그인

로그인이 필요한 서비스는 어김없이 해당 페이지가 뜬다.

✓ 로그인해야 볼 수 있는 페이지

A mockup of a Naver login page. At the top center is the word "NAVER" in large, bold, green capital letters. Below it are two input fields. The first field contains the text "my_id|". The second field contains a series of dots ".....". Below these fields is a large green rectangular button with the white Korean text "로그인" (Login) centered on it.

NAVER

my_id|

.....

로그인

키보드/마우스 입력으로 로그인 자동화

✓ 참고

NAVER

my_id|

.....

로그인

XPath

<input id="id" ...>

<input id="pw" ...>

<input id="log.login" ...>

✓ 키보드 입력

Python

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

# id 입력
driver \
    .find_element_by_xpath('//*[@id="id"]') \
    .send_keys('my_id')

# password 입력 + 엔터
driver \
    .find_element_by_xpath('//*[@id="pw"]') \
    .send_keys('my_password' + Keys.ENTER)
```

send_keys

- 해당 요소에 나열된 **key sequence**를 순차적으로 입력
- 예시에선 **m, y, _, l, d**를 **순차적**으로 입력

Keys

- **특수 키**를 쓸 수 있게 해주는 class
- **ENTER, CONTROL, SHIFT** 등 모든 특수 키가 있으며, 항상 대문자로 작성

✓ 마우스 입력

Python

```
# id 입력
driver \
    .find_element_by_xpath('//*[@id="id"]') \
    .send_keys('my_id')

# password 입력 + 엔터
driver \
    .find_element_by_xpath('//*[@id="pw"]') \
    .send_keys('my_password')

# 로그인 버튼 클릭
driver \
    .find_element_by_xpath('//*[@id="log.login"]') \
    .click()
```

click

- 해당 요소를 클릭하는 효과

04

다양한 입력, ActionChains



✓ 기본 입력의 한계

Python

```
# id 입력
driver \
    .find_element_by_xpath('//*[@id="id"]') \
    .send_keys('my_id')

# password 입력 + 엔터
driver \
    .find_element_by_xpath('//*[@id="pw"]') \
    .send_keys('my_password')

# 로그인 버튼 클릭
driver \
    .find_element_by_xpath('//*[@id="log.login"]') \
    .click()
```

각 요소에 **key sequence**를 입력하거나,
마우스 **클릭** 등의 액션을 할 수 있지만,
더 이상의 **복잡한** 액션은 할 수 없음

✓ ActionChains

여러 가지의 **action**을 **chain**처럼 엮어 수행하는 기능
이를 통해 좀 더 **복잡한 액션**을 할 수 있음
(**더블 클릭**, **ctrl + 클릭** 등)

✓ ActionChains 사용법

Python

```
webdriver.ActionChains(driver) \  
    .action1() \  
    .action2() \  
    ... \  
    .perform() # 1st action chains  
  
webdriver.ActionChains(driver) \  
    .action1() \  
    .action2() \  
    ... \  
    .perform() # 2nd action chains
```

Python

```
chains = webdriver.ActionChains(driver)  
  
chains.action1()  
chains.action2()  
...  
chains.perform() # 1st action chains  
  
chains.action1()  
chains.action2()  
...  
chains.perform() # 2nd action chains
```

✓ ActionChains의 키보드 입력

Python

```
# 요소 찾기
_id = driver.find_element_by_xpath('//*[@id="id"]')
_pw = driver.find_element_by_xpath('//*[@id="pw"]')

# 액션 수행
webdriver.ActionChains(driver) \
    .send_keys_to_element(_id, 'my_id') \
    .send_keys_to_element(_pw, 'my_pw') \
    .send_keys(Keys.ENTER) \
    .perform()
```

send_keys_to_element

- 주어진 요소에 주어진 key sequence를 입력
- 하나의 ActionChains 내에서 **여러 요소**에 액션을 수행할 수 있음

send_keys

- 요소가 주어지지 않기 때문에, **현재 커서**가 위치한 곳에서 key sequence를 입력
- 예시에선 **_pw**(비밀번호 입력 칸) 에서 **ENTER**를 입력

✓ 기본 입력과의 차이

Python

```
# 요소 찾기
_id = driver.find_element_by_xpath('//*[@id="id"]')
_pw = driver.find_element_by_xpath('//*[@id="pw"]')

# 액션 수행
webdriver.ActionChains(driver) \
    .send_keys_to_element(_id, 'my_id') \
    .send_keys_to_element(_pw, 'my_pw') \
    .send_keys(Keys.ENTER) \
    .perform()
```

Python

```
# 요소 찾기
_id = driver.find_element_by_xpath('//*[@id="id"]')
_pw = driver.find_element_by_xpath('//*[@id="pw"]')

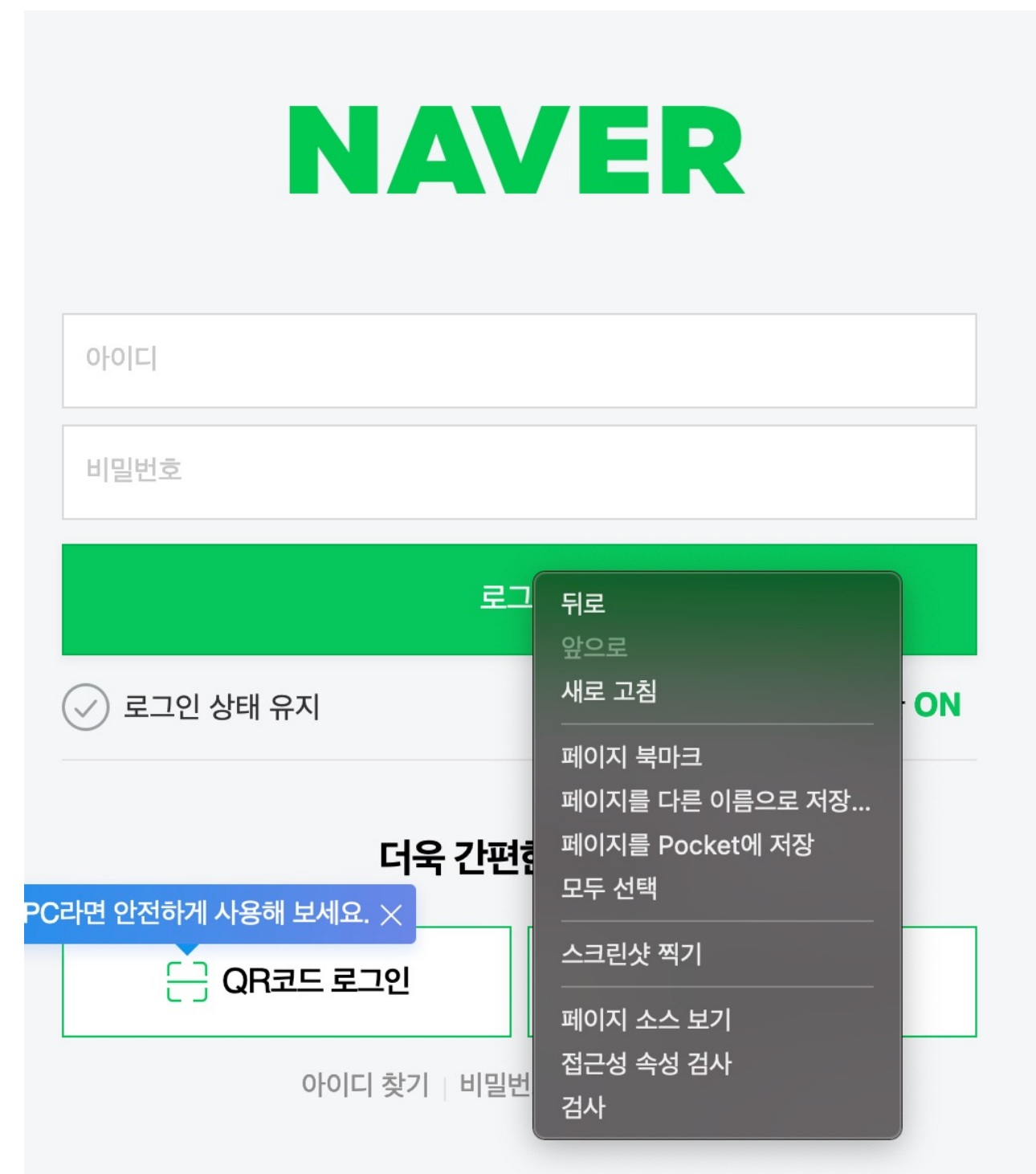
# 액션 수행
_id.send_keys('my_id')
_pw.send_keys('my_pw' + Keys.ENTER)
```

✓ ActionChains의 다양한 method 조합

Python

```
# 컨트롤 + click (우클릭과 같은 효과)
_button = driver.find_element_by_xpath('//button')
chains = webdriver.ActionChains(driver)

chains.key_down(Keys.CONTROL) # 컨트롤 버튼 누른 채로
chains.click(_button) # 버튼 요소를 클릭
chains.perform()
```



✓ ActionChains의 다양한 method 조합

Python

```
# 마우스 커서를 올렸다, 3초 뒤에 떼기
_button = driver.find_element_by_xpath('//button')
chains = webdriver.ActionChains(driver)

chains.move_to_element(_button) \
    .pause(3) \
    .move_by_offset(100, 100) \
    .perform()
```

move_to_element

- 주어진 요소로 마우스 커서를 이동

pause

- n초만큼 멈췄다 뒤의 액션을 수행

move_by_offset

- x, y축으로 주어진 수치만큼 이동

05

맺으며



✓ 웹 페이지 로딩을 기다리는 방법 3가지

1) 무조건 기다리기

- `time.sleep(n)` / python 내장 라이브러리
- **n초**만큼 **무조건** 기다림
- 내가 원하는 요소가 **불러와 졌어도** 주어진 시간을 계속 기다림

2) 암시적 기다리기

- `driver.implicitly_wait(n)`
- 앞으로 요소를 추출할 때 (`find_element`) **최대 n초**까지 기다림
- 해당 요소의 로딩이 **끝나면** 즉시 기다리기를 **종료**하고 코드를 수행

3) 명시적 기다리기

- 기다릴 요소를 **명시**
- 명시된 요소가 해당 방식으로 불러와질 때까지 **최대 n초** 기다림
- 요소는 **class_name, xpath** 등 다양한 방법으로 찾을 수 있음

✓ 키보드, 마우스 입력

입력 기본

```
# id 입력
driver \
    .find_element_by_xpath('//*[@id="id"]') \
    .send_keys('my_id')

# password 입력 + 엔터
driver \
    .find_element_by_xpath('//*[@id="pw"]') \
    .send_keys('my_password')

# 로그인 버튼 클릭
driver \
    .find_element_by_xpath('//*[@id="log.login"]') \
    .click()
```

ActionChains

```
# 요소 찾기
_id = driver.find_element_by_xpath('//*[@id="id"]')
_pw = driver.find_element_by_xpath('//*[@id="pw"]')




# 액션 수행
webdriver.ActionChains(driver) \
    .send_keys_to_element(_id, 'my_id') \
    .send_keys_to_element(_pw, 'my_pw') \
    .send_keys(Keys.ENTER) \
    .perform()
```

✓ 다음 시간 예고

Countries of the World: A Simple Example 250 items

A single page that lists information about all the countries in the world. Good for those just getting started looking for patterns in the HTML that will allow you to extract information about each country. The scraper makes a request to this page, parses the HTML and prints out each country's name.

📺 There are 4 video lessons that show you how to scrape this page.

 Andorra Capital: Andorra la Vella Population: 84000 Area (km²): 468.0	 United Arab Emirates Capital: Abu Dhabi Population: 4975593 Area (km²): 82880.0	 Afghanistan Capital: Kabul Population: 31100000 Area (km²): 652230.0
 Antigua and Barbuda Capital: St. John's Population: 86754 Area (km²): 443.0	 Anguilla Capital: The Valley Population: 13254 Area (km²): 102.0	 Argentina Capital: Buenos Aires Population: 41343201 Area (km²): 2766890.0
 Armenia Capital: Yerevan Population: 2968000 Area (km²): 29800.0	 Angola Capital: Luanda Population: 13068161 Area (km²): 1246700.0	 Australia Capital: Canberra Population: 21515754 Area (km²): 7686850.0
 Argentina Capital: Buenos Aires Population: 41343201 Area (km²): 2766890.0	 American Samoa Capital: Pago Pago Population: 57881 Area (km²): 199.0	 Austria Capital: Vienna Population: 8540000 Area (km²): 83859.0
 Australia Capital: Canberra Population: 21515754 Area (km²): 7686850.0	 Aruba Capital: Oranjestad Population: 71566 Area (km²): 193.0	 Azerbaijan Capital: Baku Population: 9700000 Area (km²): 86600.0

간단한 페이지

Forms: Forms, Searching and Pagination 25 items

A page of NHL team stats since 1990. Practice building a scraper that handles pagination. Look at how pagination and search elements change the URL as your browser paginates through the results.

📺 show you how to scrape this page.

Data via

Search for Teams: <input type="text" value="Search for Teams"/> <input type="button" value="Search"/>			
Winnipeg Jets	1990	26	43
Boston Bruins	1991	36	32
Buffalo Sabres	1991	31	37
Calgary Flames	1991	31	37
Chicago Blackhawks	1991	36	29

1

2

3

4

5

6

7

8

9

10

11

12

13

14

검색 & Pagination

Winning Films: AJAX and Javascript 87 items

A page of a bunch of great films. Learn how content is added to the page asynchronously with JavaScript. Look for ways that you can tell visually when a site is loading content with AJAX. Then, build a scraper that makes AJAX requests and scrape them.

📺 30 lessons that show you how to scrape this page.

Data via https://en.wikipedia.org/wiki/List_of_Academy_Award-winning_best_picture_film

Choose a Year to View Films					
2015	2014	2013	2012	2011	2010
<div></div>					
Lessons and Videos © Hartley Brody 2018					

JS를 통한 동적 렌더링

크레딧

/* elice */

코스 매니저

임승연

콘텐츠 제작자

신용기

강사

신용기

감수자

장석준

디자이너

강혜정

연락처

TEL

070-4633-2015

WEB

<https://elice.io>

E-MAIL

contact@elice.io

