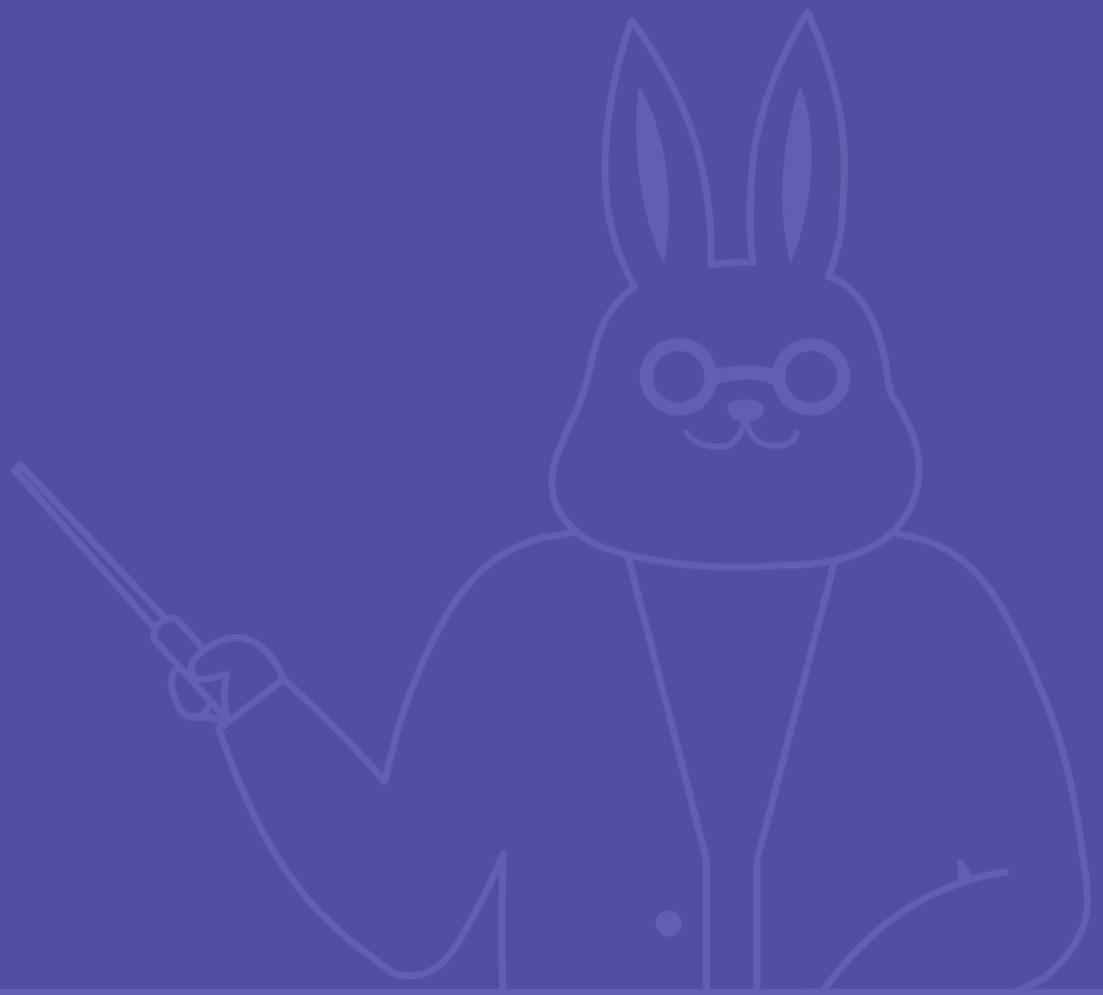
파이썬 크롤링 입문

02 Selenium 활용: 스크래핑

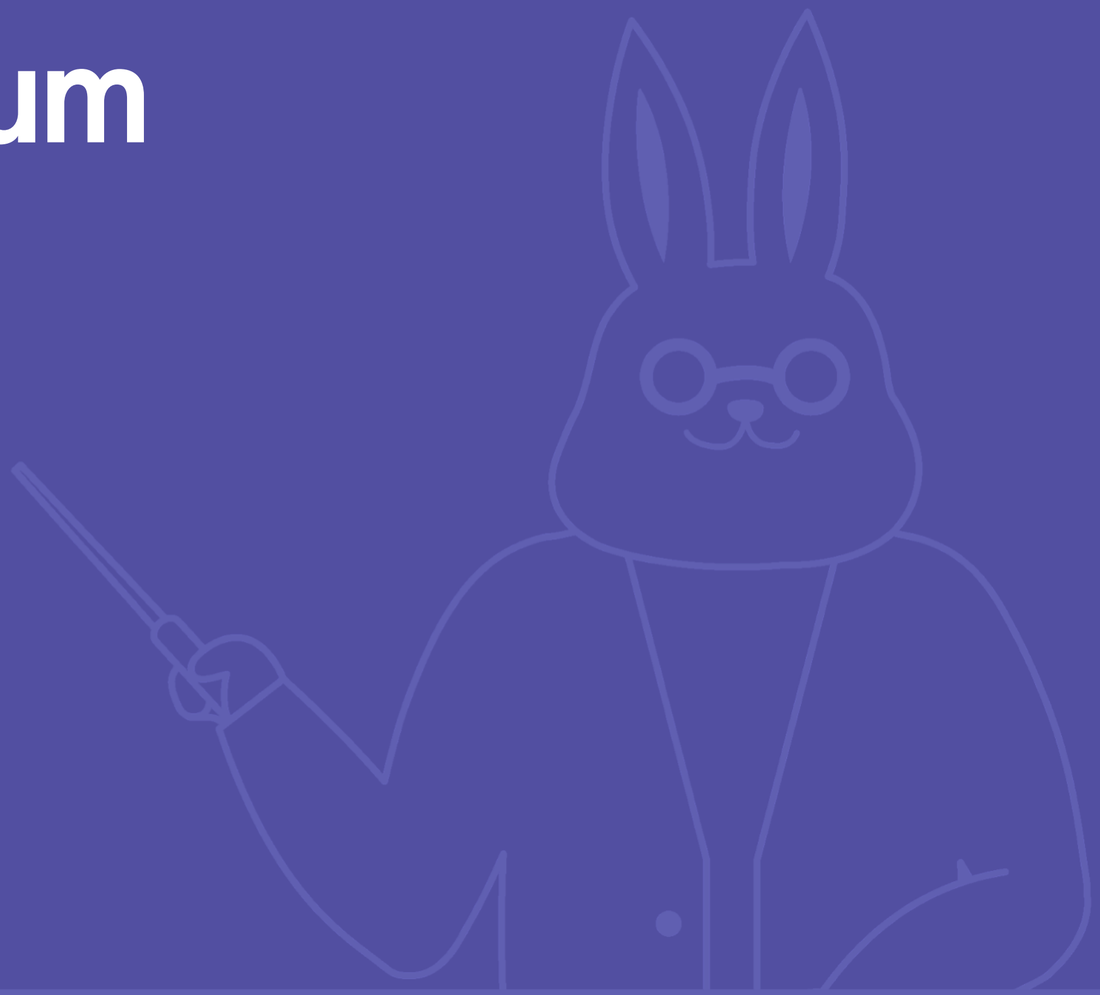


/* elice */

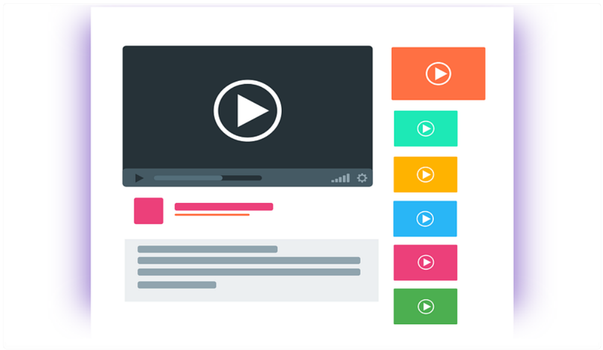
목차

01

웹 스크래핑과 Selenium

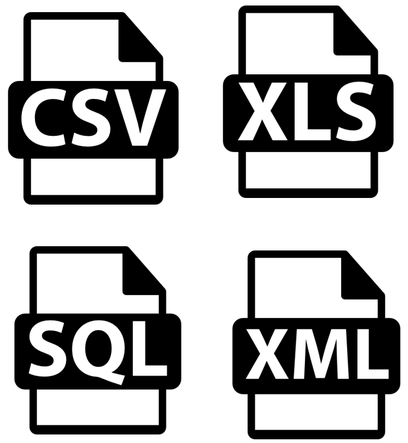


웹 스크래핑(Web Scraping)이란?



Website Pages,
Unstructured Data

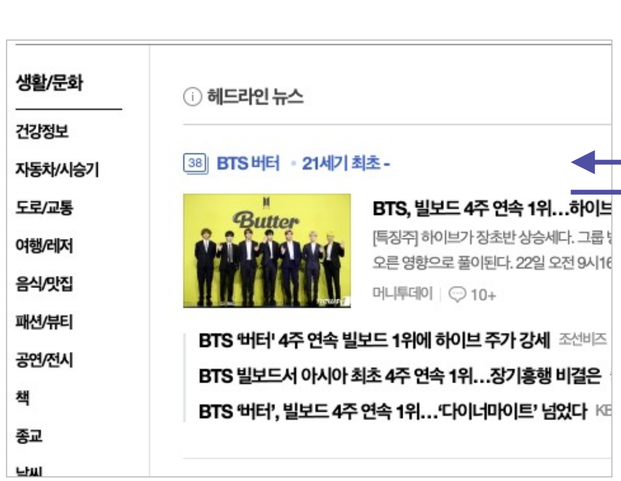
Web Scraping/
Data Extraction

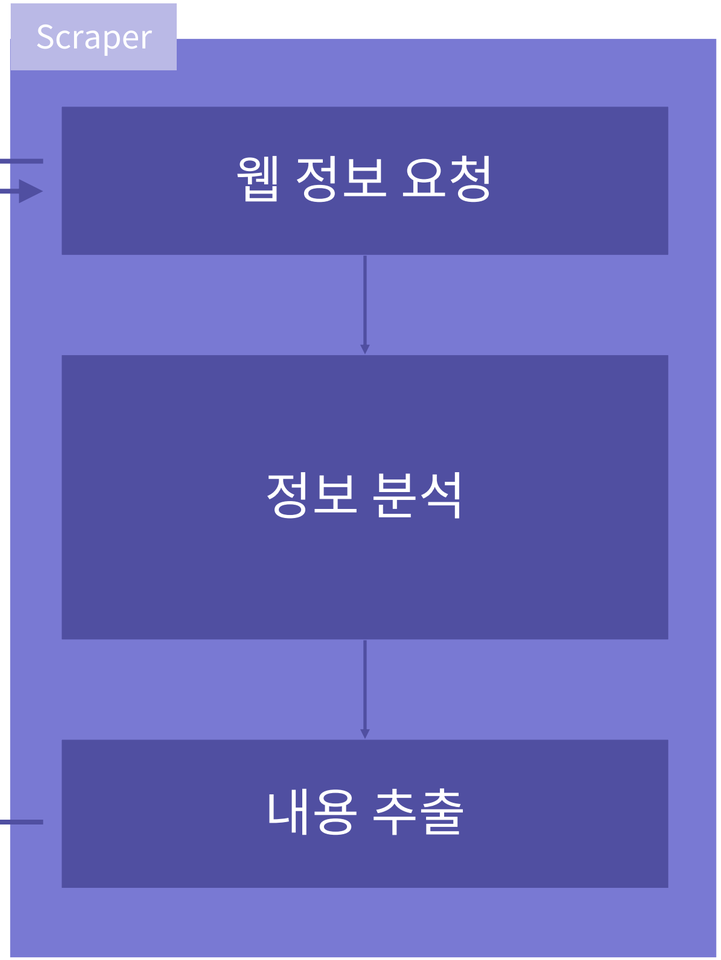


Structured Data

웹 사이트에서 **원하는** 데이터를 **수집**하고, **가공**하는 행위

웹 스크래핑 과정 모식도





웹의 정보를 **받아**와서, **분석**하고, **추출**한다.

Selenium이란?

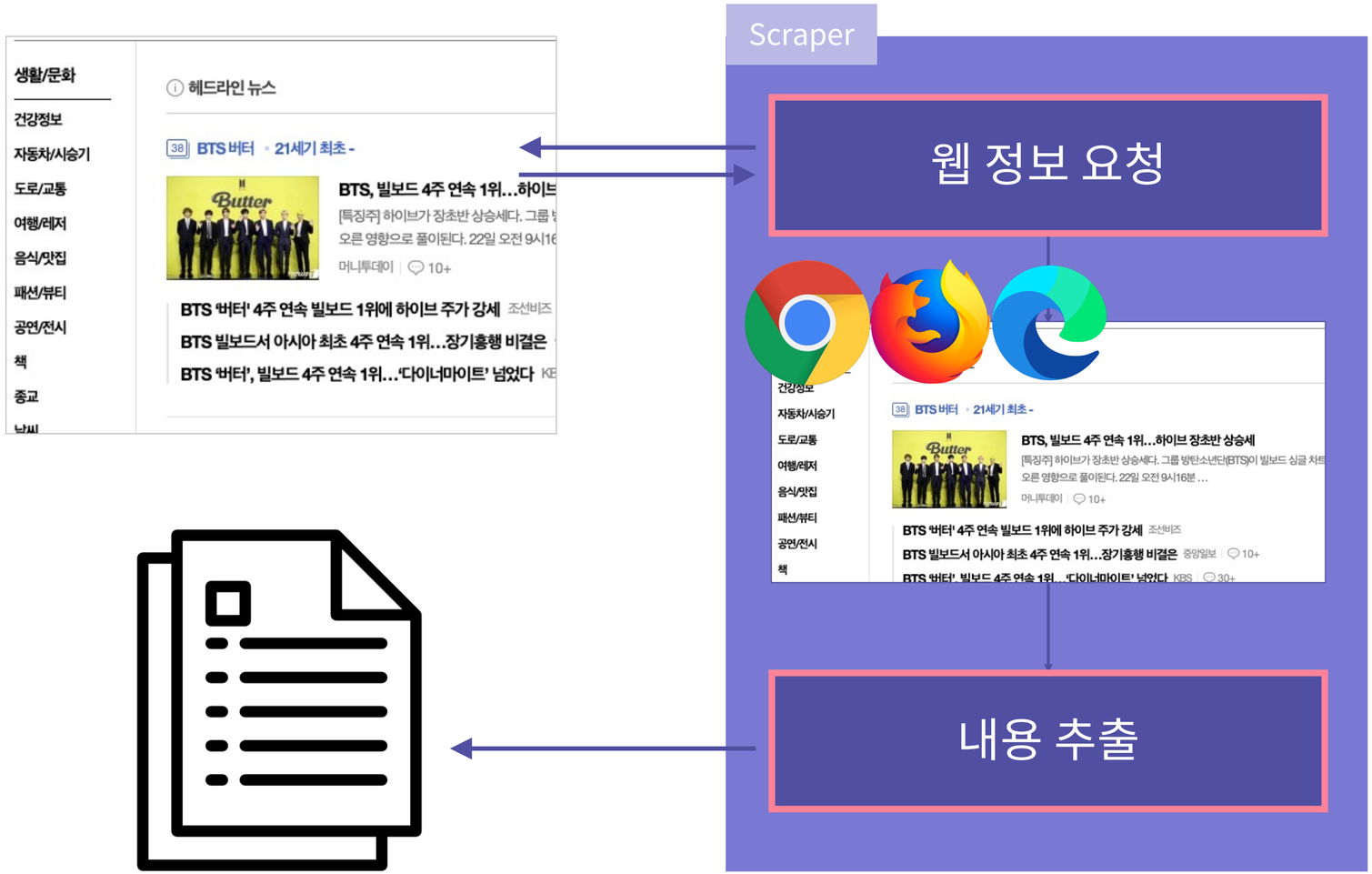
웹 테스트 자동화 도구로, 파이썬(또는 자바)에서 라이브러리로써 사용한다.

브라우저를 제어하는 기능이 있어 웹 스크래핑에 용이하다.

Selenium을 통한 웹 스크래핑 과정 모식도

Scraper 내에서 브라우저를 직접 **실행**하고, **제어**할 수 있음

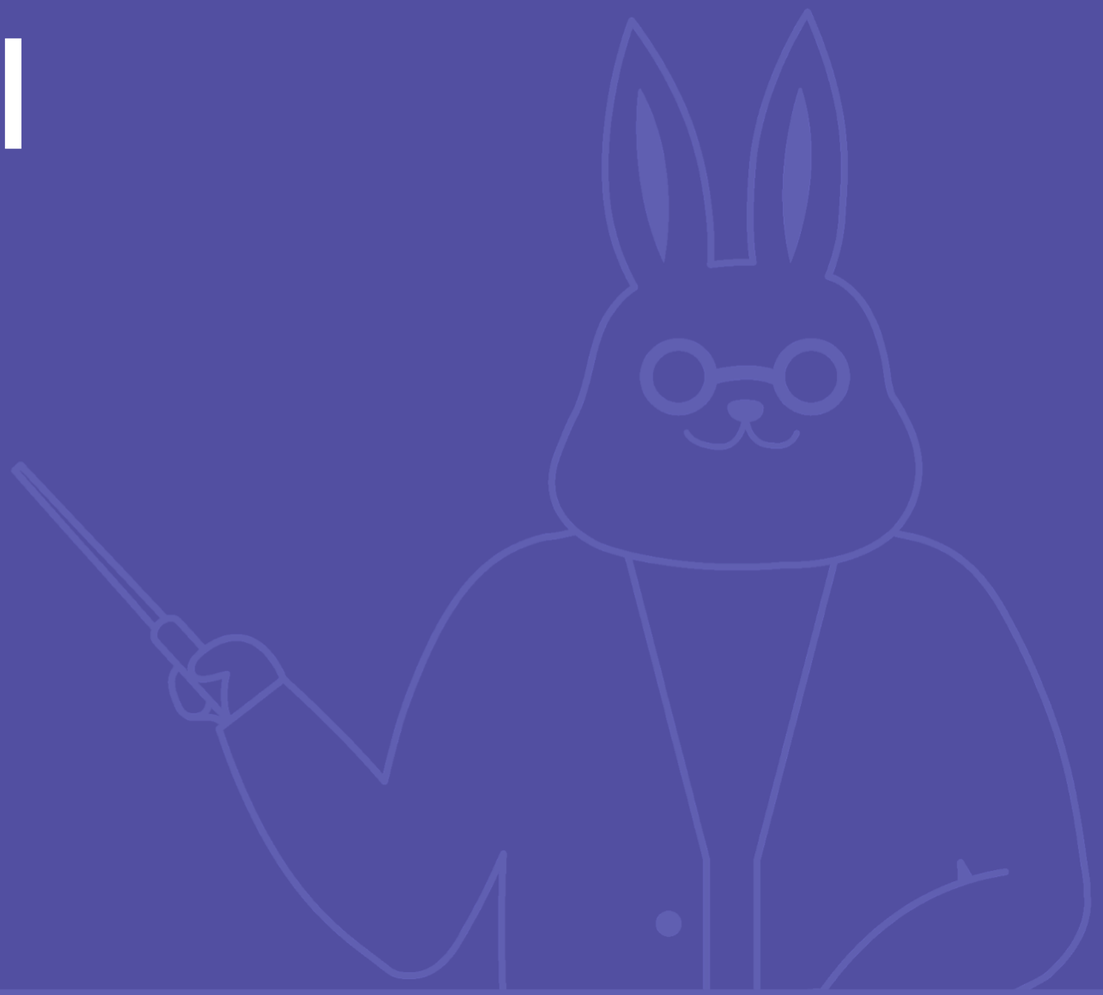
이번 수업에서 배울 것



웹 정보를 요청하고, 태그를 찾아 내용을 추출하는 방법을 배운다.

02

태그 이름으로 요소 찾기



How to Scrap?


```
{
  "category": {
    "name": "BTS 버터",
    "subName": "21세기 최초",
    "newsList": [
      {
        "title": "BTS, 빌보드 4주 연속 1위...하이브 장초반 상승세",
        "press": "머니투데이",
        "content": "[특징주] 하이브가 장초반 상승세다. 그룹 방탄소년단(BTS)이 빌보드 싱글 차트에서
      },
      {
        "title": "BTS '버터' 4주 연속 빌보드 1위에 하이브 주가 강세",
        "press": "조선비즈"
      },
      {
        "title": "BTS 빌보드서 아시아 최초 4주 연속 1위...장기흥행 비결은",
        "press": "중앙일보"
      },
      {
        "title": "BTS '버터', 빌보드 4주 연속 1위...'다이너마이트' 넘었다",
        "press": "KBS"
      }
    ]
  }
}
```


어느 **요소**에 있는지만 알면, 데이터를 **추출**할 수 있다.

Selenium 공통 설정

Python

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

driver = webdriver.Firefox()  # 실행 브라우저
driver.get('https://news.naver.com')
...
driver.quit()
```

해당 프로그램을 수행할 **브라우저를 선택**하여 실행
다양한 브라우저 (e.g. Android, Safari, Chrome, …) 를 지원함

Selenium 공통 설정 – with

Python

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

with webdriver.Firefox() as driver:  # 실행 브라우저
    driver.get('https://news.naver.com’)
    ...
```

코드가 정상적으로 작동하지 않고 **에러** 발생 시 **with** 표현식을 사용하면
꺼지지 않은 브라우저가 메모리를 점유하는 현상을 방지

원하는 요소 단일 선택

HTML

```
<body>
  <h1>Web Scraping</h1>
  <p>원하는 데이터를 수집한다.</p>
</body>
```

Python

```
e = driver.find_element(By.TAG_NAME,'p')

print(e.text)
```

실행결과

```
원하는 데이터를 수집한다.
```

원하는 요소 단일 선택 – 부모 요소 선택 시

HTML

```
<body>
  <h1>Web Scraping</h1>
  <p>원하는 데이터를 수집한다.</p>
</body>
```

Python

```
e = driver.find_element(By.TAG_NAME,'body')

print(e.text)
```

실행결과

```
Web Scraping
원하는 데이터를 수집한다.
```

원하는 요소 단일 선택 – 같은 요소가 여러 개일 때

HTML

```
<body>
  <h1>Web Scraping</h1>
  <div>
    <p>(한글) 웹 스크래핑</p>
  </div>
  <p>원하는 데이터를 수집한다.</p>
  <p>데이터를 가공한다.</p>
</body>
```

Python

```
e = driver.find_element(By.TAG_NAME,'p')

print(e.text)
```

실행결과

```
(한글) 웹 스크래핑
```

원하는 요소 단일 선택 – 정리

HTML

```
<body>
  <h1>Web Scraping</h1>
  <div>
    <p>(한글) 웹 스크래핑</p>
  </div>
  <p>원하는 데이터를 수집한다.</p>
  <p>데이터를 가공한다.</p>
</body>
```

Python

```
e = driver.find_element(By.TAG_NAME,'p')

print(e.text)
```

find_element(By.TAG_NAME, ‘tag_name’)

- 해당 태그를 가진 요소를 찾아 반환하는 method
- 여러 개 있다면 **맨 위**에 적힌 요소

원하는 요소 복수 선택 – 해당 요소를 전부 가져오고 싶을 때

HTML

```
<body>
  <h1>Web Scraping</h1>
  <div>
    <p>(한글) 웹 스크래핑</p>
  </div>
  <p>원하는 데이터를 수집한다.</p>
  <p>데이터를 가공한다.</p>
</body>
```

Python

```
e_list = driver.find_elements(By.TAG_NAME,'p')

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

실행결과

```
(한글) 웹 스크래핑
원하는 데이터를 수집한다.
데이터를 가공한다.
```

원하는 요소 복수 선택 – 단일 선택 method로 가져올 수 없는 요소

HTML

```
<body>
  <h1>Web Scraping</h1>
  <div>
    <p>(한글) 웹 스크래핑</p>
  </div>
  <p>원하는 데이터를 수집한다.</p>
  <p>데이터를 가공한다.</p>
</body>
```

Python

```
e_list = driver.find_elements(By.TAG_NAME,'p')
e = e_list[1]  # 2번째 p

print(e.text)
```

실행결과

```
원하는 데이터를 수집한다.
```

원하는 요소 복수 선택 – 정리

HTML

```
<body>
  <h1>Web Scraping</h1>
  <div>
    <p>(한글) 웹 스크래핑</p>
  </div>
  <p>원하는 데이터를 수집한다.</p>
  <p>데이터를 가공한다.</p>
</body>
```

Python

```
e_list = driver.find_elements(By.TAG_NAME,'p')
e = e_list[1]

print(e.text)
```

find_elements(By.TAG_NAME, ‘tag_name’)

- 해당 태그를 가진 요소를 전부 찾아 리스트로 반환하는 method

요소 내의 요소 선택할 때

HTML

```
<body>
  <h1>Web Scraping</h1>
  <p>원하는 데이터를 수집한다.</p>
  <p>데이터를 가공한다.</p>
  <div>
    <p>(한글) 웹 스크래핑</p>
  </div>
</body>
```

Python

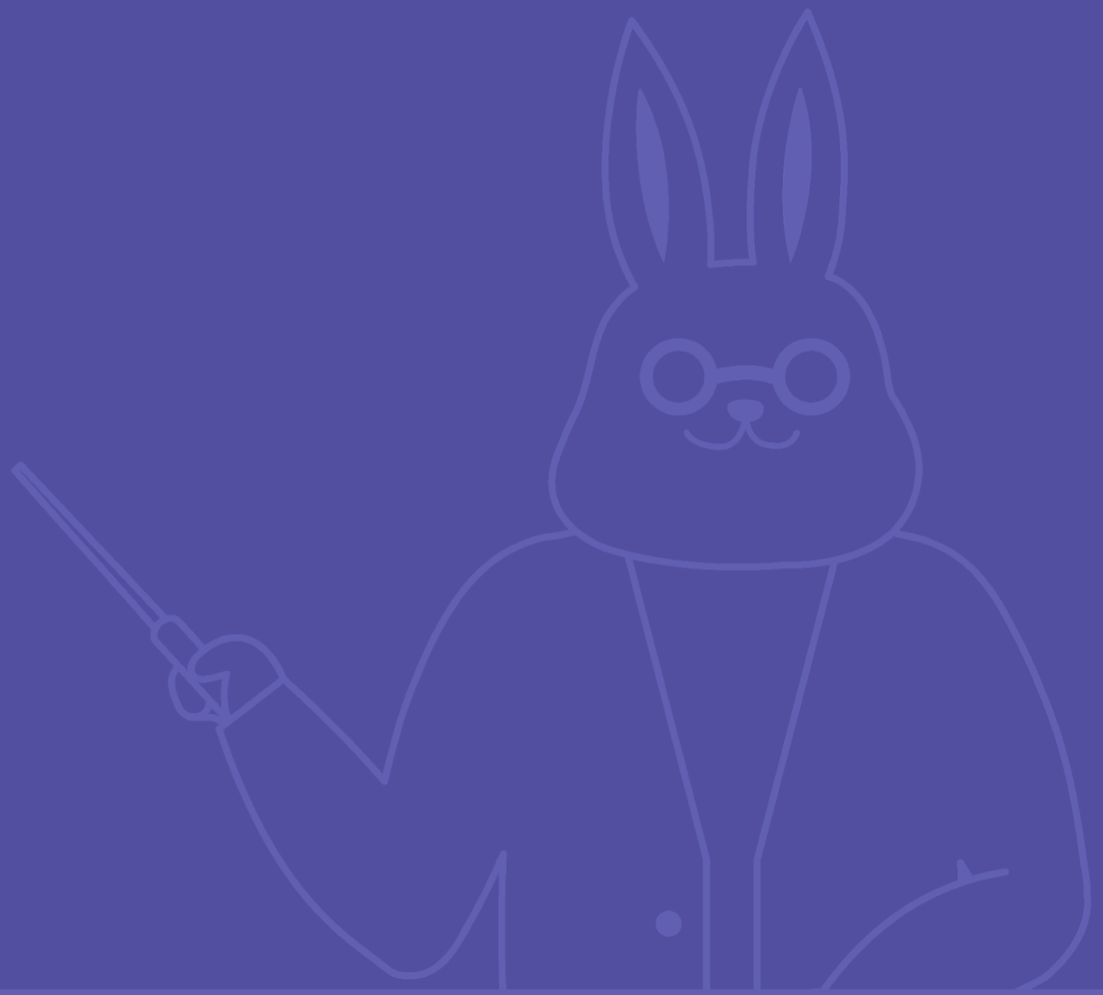
```
e = driver.find_element(By.TAG_NAME,'div') \
    .find_element(By.TAG_NAME, 'p')

print(e.text)
```

실행결과

```
(한글) 웹 스크래핑
```

03

class, id로 요소 찾기



find_element(By.TAG_NAME, ‘tag_name’)의 불편함

엘리스 웹 페이지의 HTML

Python

```python
e = driver.find_element(By.TAG_NAME,'div') \
    .find_element(By.TAG_NAME,'div') \
    .find_element(By.TAG_NAME,'div') \
    .find_element(By.TAG_NAME,'div') \
    .find_element(By.TAG_NAME,'div') \
    .find_element(By.TAG_NAME,'div') \
    ...
    .find_element(By.TAG_NAME,'div')

print(e.text)
```

class의 활용

엘리스 웹 페이지의 HTML

Python

```python
e = driver \
    .find_element(By.CLASS_NAME,'css-1ksqhvg')

print(e.text)
```

요소를 찾는 다른 방법

tag_name 만으로 요소를 찾기엔 무리가 있음

→ **class_name, id**를 활용해서 찾을 수 있음

class 이름으로 요소 찾기

HTML

```
<body>
  <h1>Web Scraping</h1>
  <p>원하는 데이터를 수집한다.</p>
  <p>데이터를 가공한다.</p>
  <div>
    <p class="bold">(한글) 웹 스크래핑</p>
  </div>
</body>
```

Python

```
e = driver \
    .find_element(By.CLASS_NAME,'bold')

print(e.text)
```

실행결과

```
(한글) 웹 스크래핑
```

class 이름으로 요소 찾기

HTML

```
<body>
  <h1 class="p-1 bold">Web Scraping</h1>
  <p>원하는 데이터를 수집한다.</p>
  <p class="bold">데이터를 가공한다.</p>
  <div>
    <p class="bold">(한글) 웹 스크래핑</p>
  </div>
</body>
```

Python

```
e_list = driver \
    .find_elements(By.CLASS_NAME,'bold')

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

실행결과

```
Web Scraping
데이터를 가공한다.
(한글) 웹 스크래핑
```

id로 요소 찾기

HTML

```
<body>
  <h1 class="p-1 bold">Web Scraping</h1>
  <p>원하는 데이터를 수집한다.</p>
  <p class="bold">데이터를 가공한다.</p>
  <div>
    <p id="kor-p" class="bold">
      (한글) 웹 스크래핑</p>
  </div>
</body>
```

Python

```
e = driver.find_element(By.ID, 'kor-p')

print(e.text)
```

실행결과

```
(한글) 웹 스크래핑
```

여러 method 같이 사용하기

HTML

```
<body>
  <div class="middle">
    <p>Web Scripaing</p>
  </div>
  <div class="small">
    <p>(한글) 웹 스크래핑</p>
  </div>
</body>
```

Python

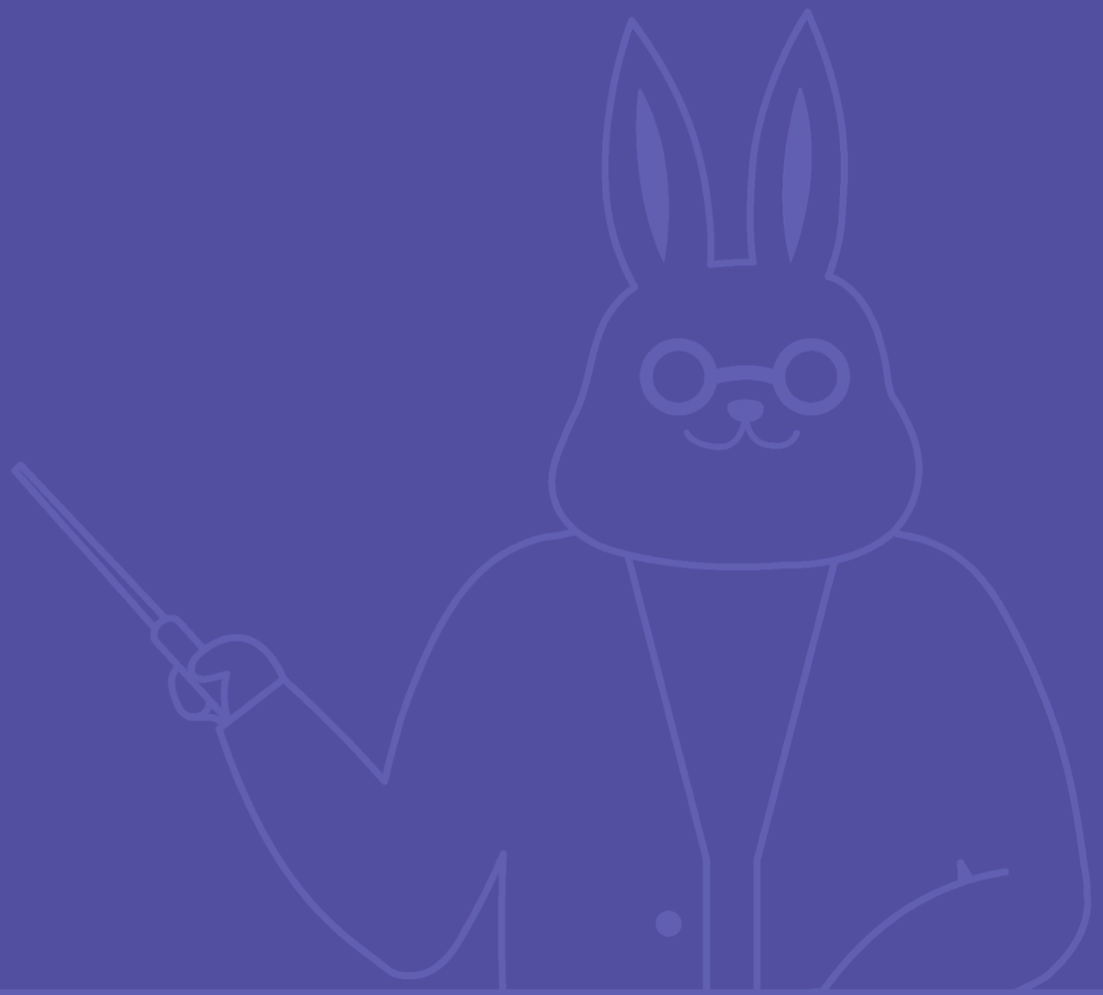
```
e = driver \
    .find_element(By.CLASS_NAME,'small') \
    .find_element(By.TAG_NAME,'p')

print(e.text)
```

실행결과

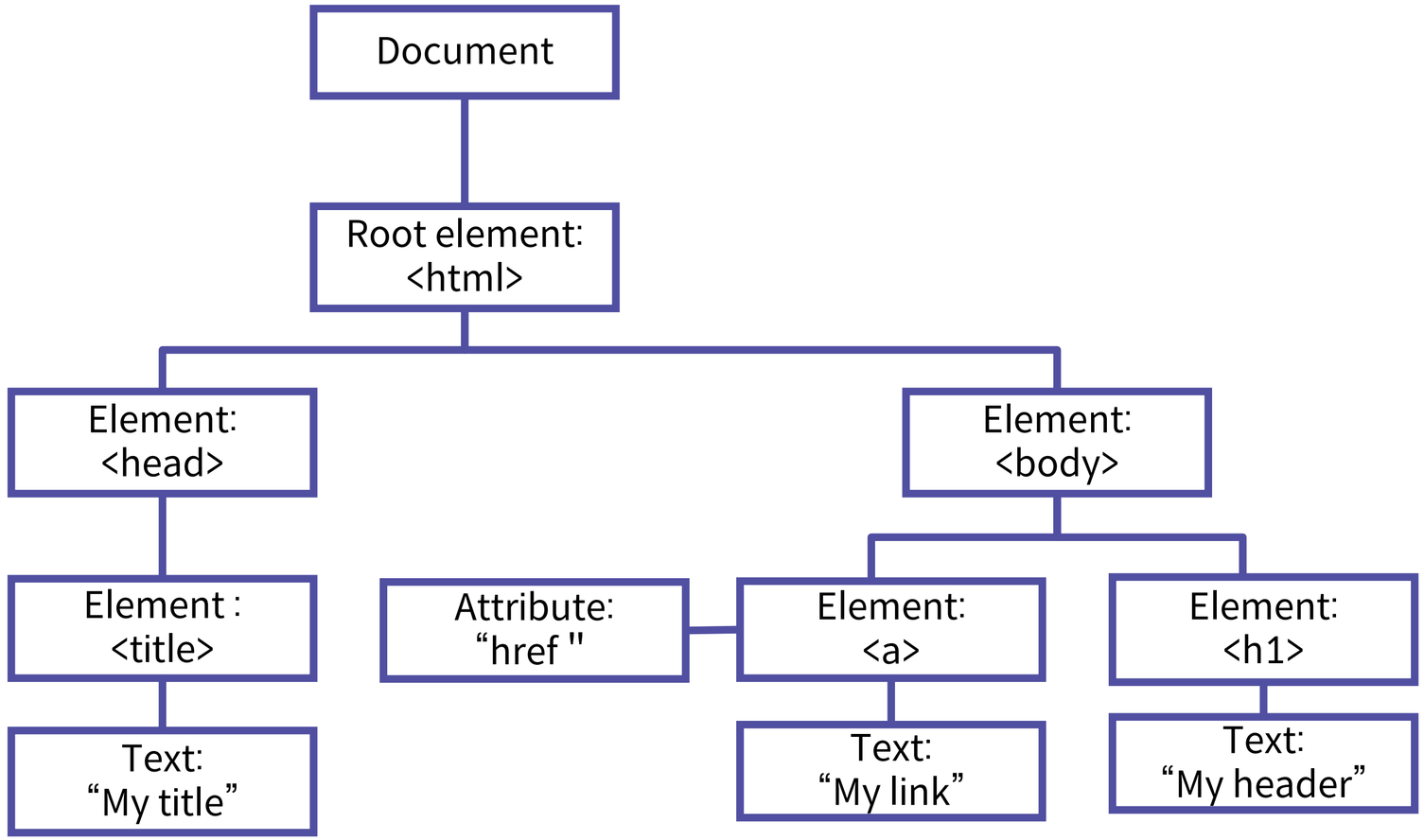
```
(한글) 웹 스크래핑
```

04

요소의 주소, XPath



DOM (Document Object Model)



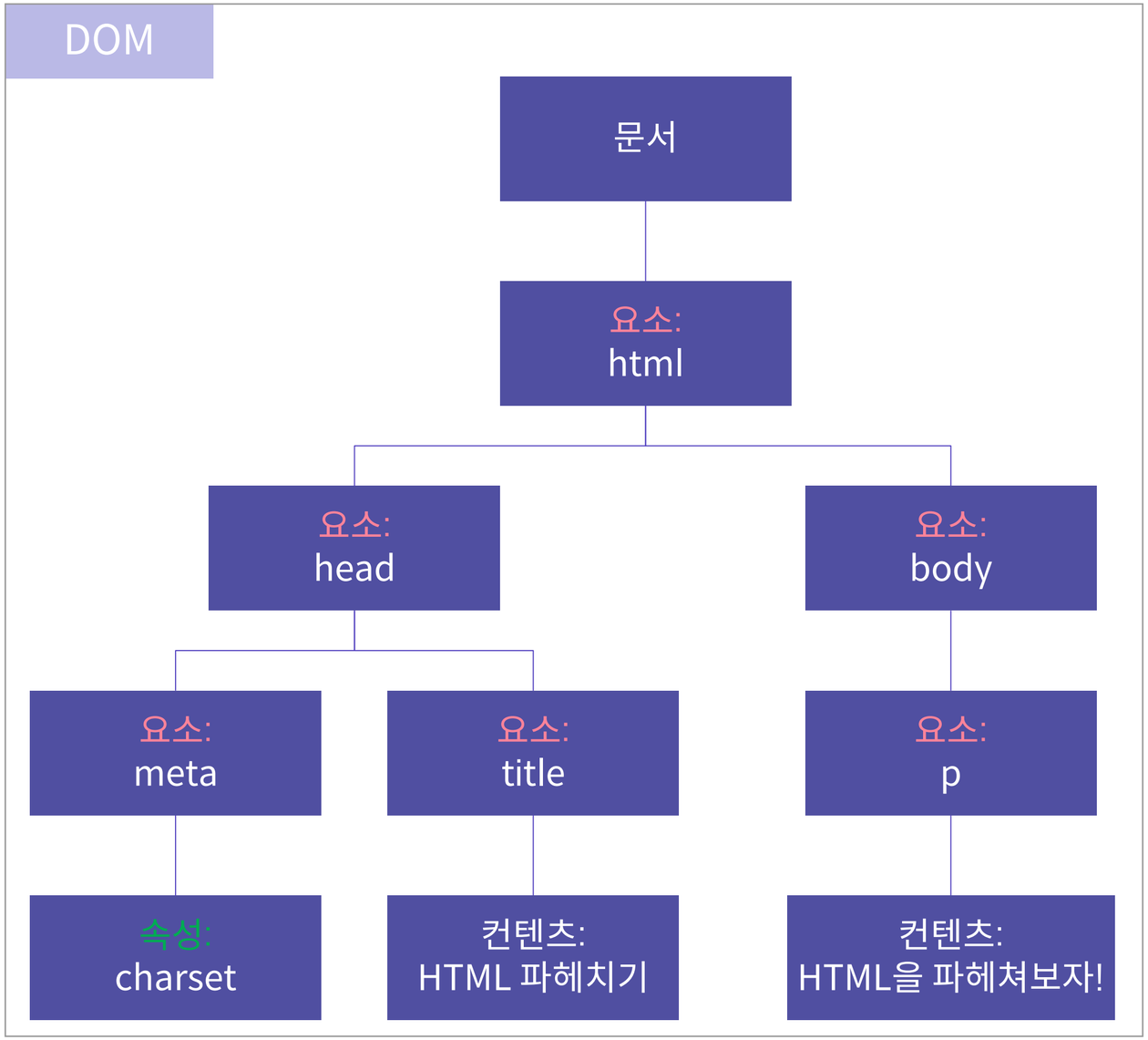
DOM은 문서 내의 모든 요소를 각 **노드** 간의 **계층 구조**로 표현함

DOM 표현

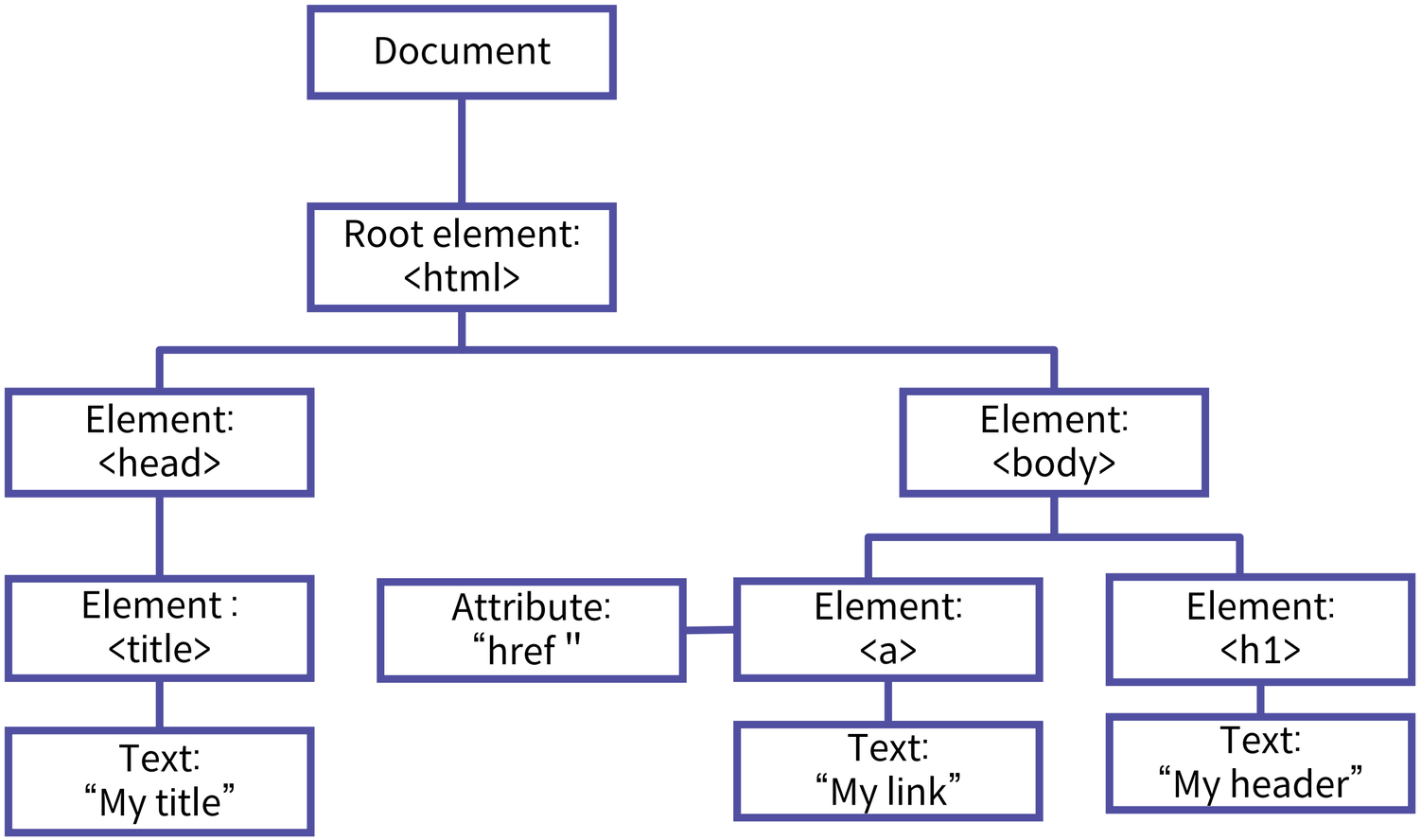
HTML

```
<html>
  <head>
    <meta charset="utf-8">
    <title>HTML 파헤치기</title>
  </head>
  <body>
    <p>HTML을 파헤쳐보자!</p>
  </body>
</html>
```

DOM



XPath란?



XML 문서의 특정 **노드**에 접근하기 위한 **질의어**

XPath 문법

HTML

```
<html>
  <body>
    <h1>Web Scraping</h1>
    <p>원하는 데이터를 수집한다.</p>
    <p>데이터를 가공한다.</p>
    <div>
      <p class="bold">(한글) 웹 스크래핑</p>
    </div>
  </body>
</html>
```

XPath

```
/html/body/p
```

> /
>
> - 현재 위치의 **자식** 노드만 검색

XPath 문법

HTML

```
<html>
  <body>
    <h1>Web Scraping</h1>
    <p>원하는 데이터를 수집한다.</p>
    <p>데이터를 가공한다.</p>
    <div>
      <p class="bold">(한글) 웹 스크래핑</p>
    </div>
  </body>
</html>
```

XPath

```
//p
```

//

- 현재 위치의 **모든 자손** 노드에서 검색

XPath 문법

HTML

```
<html>
  <body>
    <div>
      <h1>Web Scraping</h1>
      <p>원하는 데이터를 수집한다.</p>
      <p>데이터를 가공한다.</p>
      <div>
        <p class="bold">(한글) 웹 스크래핑</p>
      </div>
    </div>
  </body>
</html>
```

XPath

```
//div/p
```

//

- 현재 위치의 **모든 자손** 노드에서 검색

XPath 문법

HTML

```
<html>
  <body>
    <h1>Web Scraping</h1>
    <p>원하는 데이터를 수집한다.</p>
    <p>데이터를 가공한다.</p>
    <div>
      <p class="bold">(한글) 웹 스크래핑</p>
    </div>
  </body>
</html>
```

XPath

```
/html/body/*
```

*

- 와일드 카드
- 경로에 있는 모든 노드를 의미

XPath 문법

HTML

```
<html>
  <body>
    <h1>Web Scraping</h1>
    <p>원하는 데이터를 수집한다.</p>
    <p>데이터를 가공한다.</p>
    <div>
      <p class="bold">(한글) 웹 스크래핑</p>
    </div>
  </body>
</html>
```

XPath

```
/html/body//*
```

*

- 와일드 카드
- 경로에 있는 모든 노드를 의미

XPath 문법

HTML

```
<html>
  <body>
    <h1>Web Scraping</h1>
    <p>원하는 데이터를 수집한다.</p>
    <p>데이터를 가공한다.</p>
    <div>
      <p class="bold">(한글) 웹 스크래핑</p>
    </div>
  </body>
</html>
```

XPath

```
//p
```

[]

- 필터 표현식
- 인덱스, 속성 등을 통해 특정 요소를 검색할 수 있다.

XPath 문법

HTML

```
<html>
  <body>
    <h1>Web Scraping</h1>
    <p>원하는 데이터를 수집한다.</p>
    <p>데이터를 가공한다.</p>
    <div>
      <p class="bold">(한글) 웹 스크래핑</p>
    </div>
  </body>
</html>
```

XPath

```
//p[1]
```

[index]

- 검색된 노드들 중 index에 해당하는 노드 반환
- 여타 언어와 달리 1부터 시작

XPath 문법

HTML

```
<html>
  <body>
    <h1>Web Scraping</h1>
    <p>원하는 데이터를 수집한다.</p>
    <p class="big">데이터를 가공한다.</p>
    <div>
      <p class="bold">(한글) 웹 스크래핑</p>
    </div>
  </body>
</html>
```

XPath

```
//p[@class]
```

[@attr]

- 검색된 노드들 중 해당 속성을 가지고 있는 노드를 모두 반환

XPath 문법

HTML

```
<html>
  <body>
    <h1>Web Scraping</h1>
    <p>원하는 데이터를 수집한다.</p>
    <p class="big">데이터를 가공한다.</p>
    <div>
      <p class="bold">(한글) 웹 스크래핑</p>
    </div>
  </body>
</html>
```

XPath

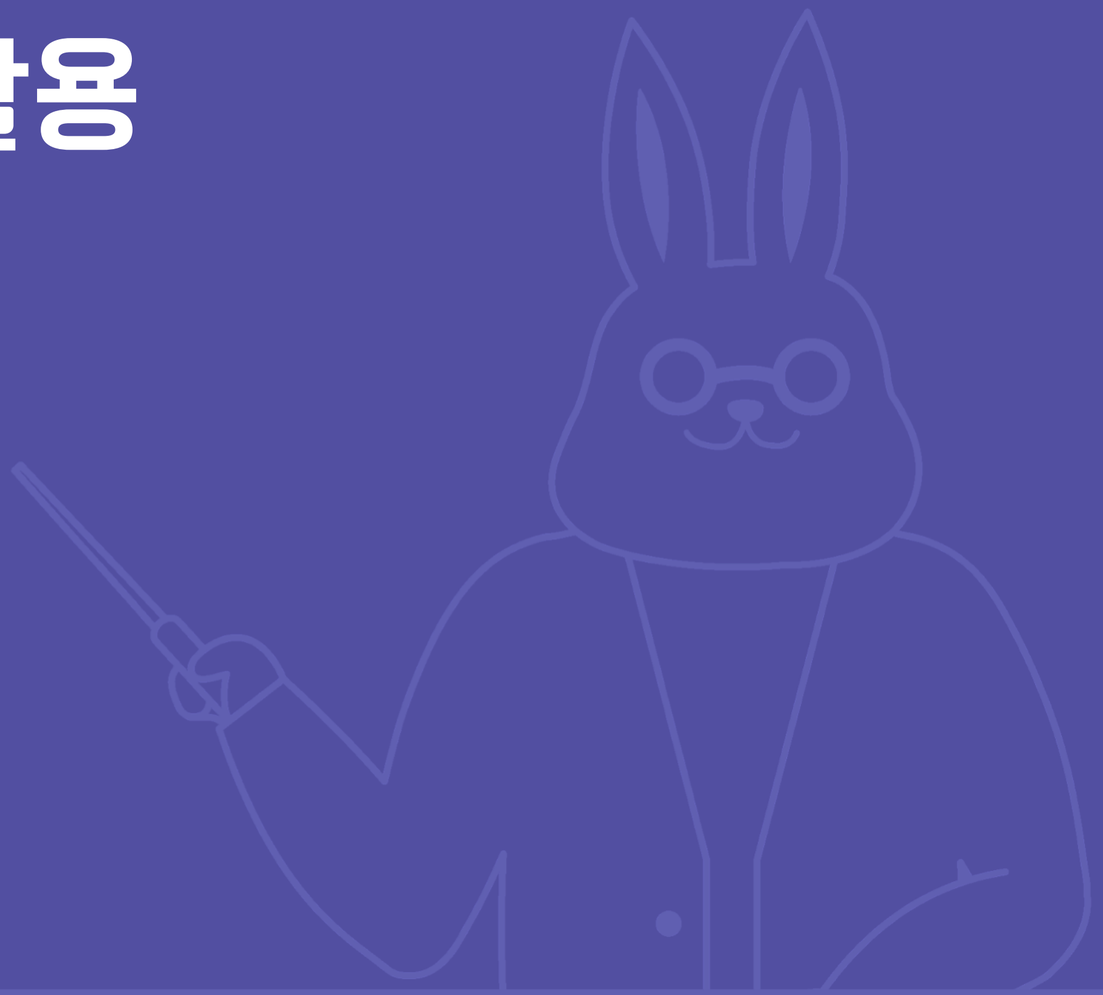
```
//p[@class="bold"]
```

[@attr="value"]

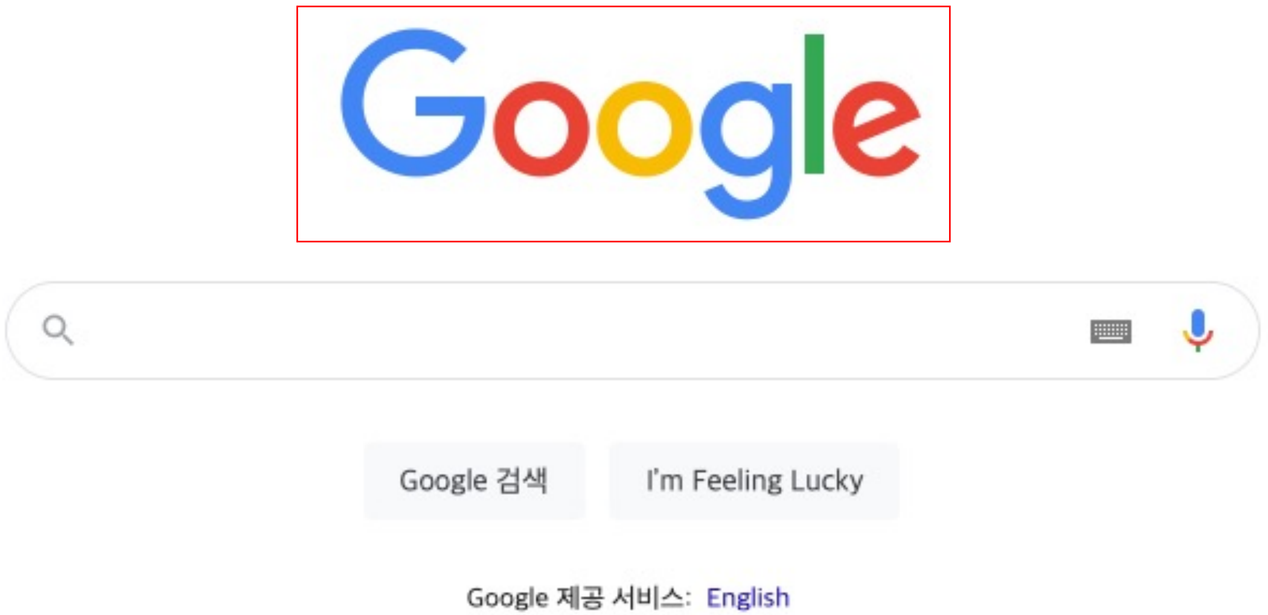
- 검색된 노드들 중 해당 속성과 속성값이 일치하는 노드를 모두 반환
- 속성값이 완벽히 일치할 때만 반환

05

브라우저와 XPath의 활용

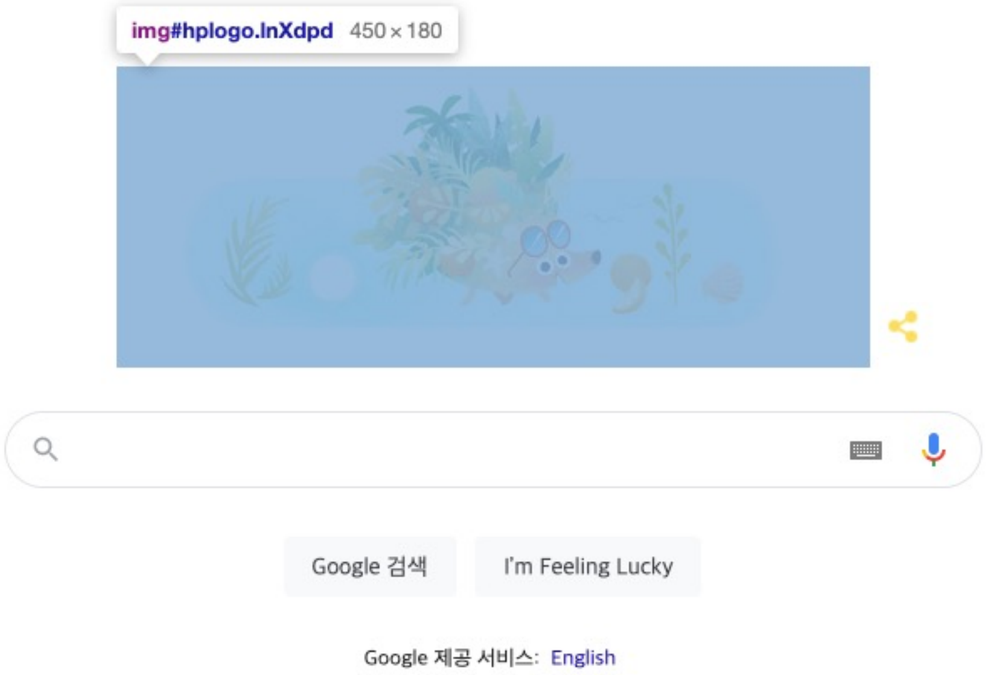


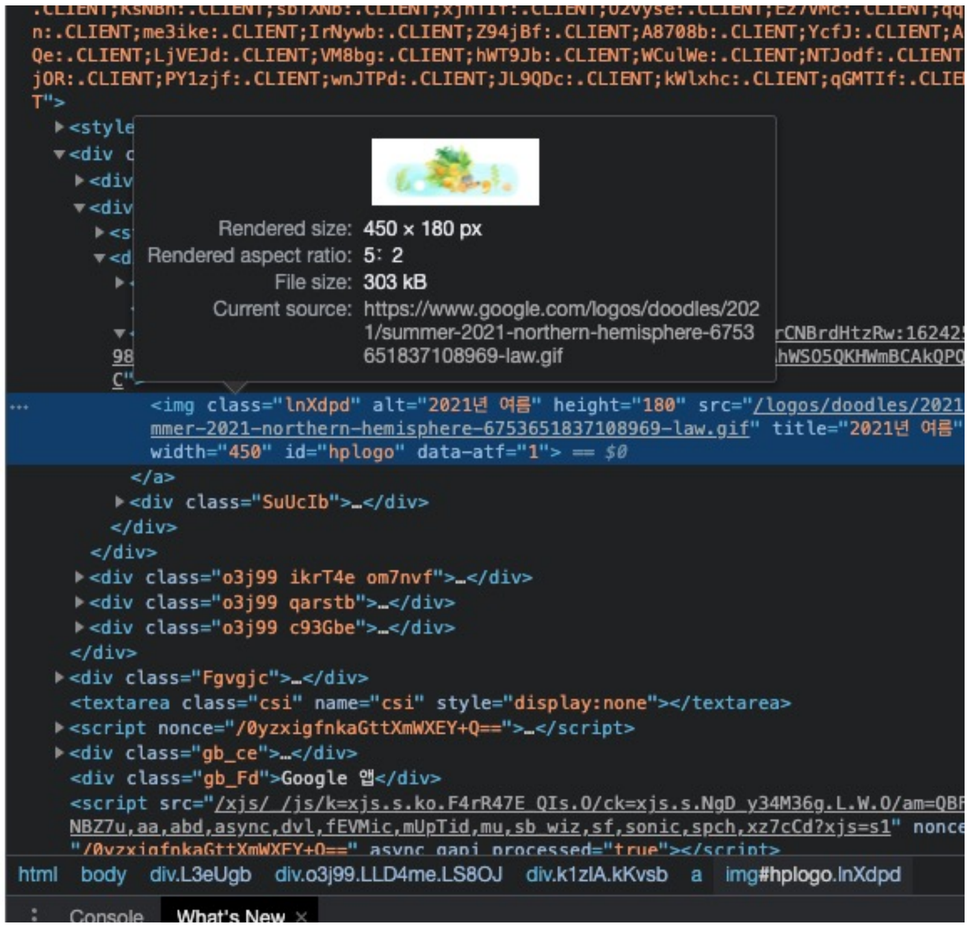
웹 페이지에서 요소 위치를 알아내는 방법은 없나요?



단순한 웹 사이트 내에서도 **요소의 위치**를 파악하기 쉽지 않다.

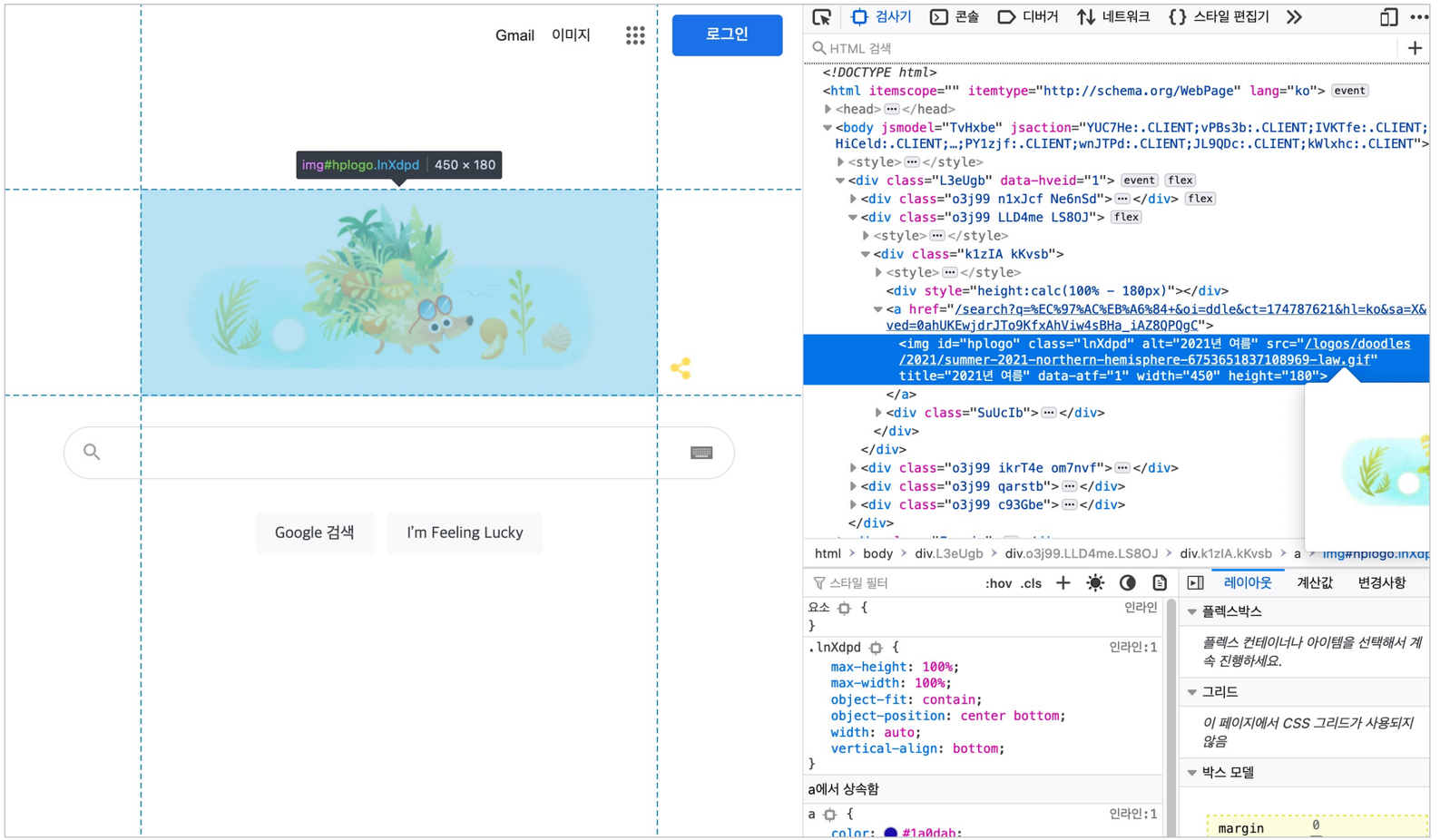
브라우저의 활용 - Chrome





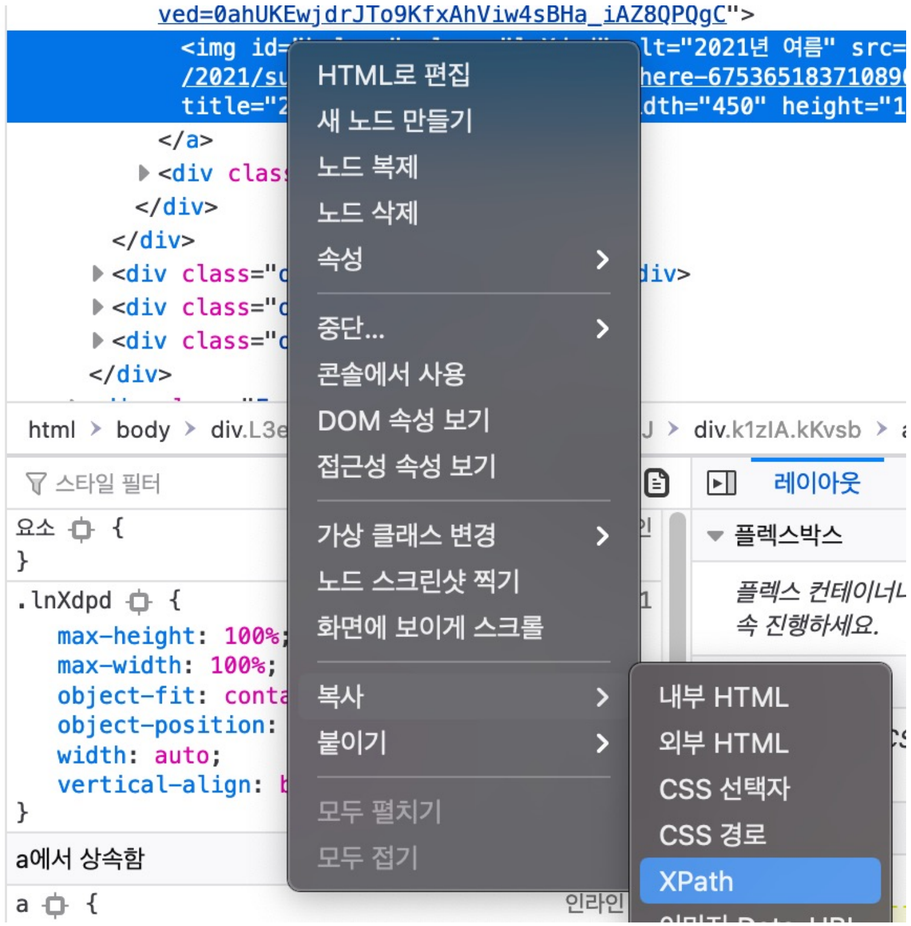
찾고싶은 요소 우클릭 – 검사

브라우저의 활용 - Firefox



찾고싶은 요소 우클릭 – 검사

웹 페이지 내 요소의 XPath 알아내기

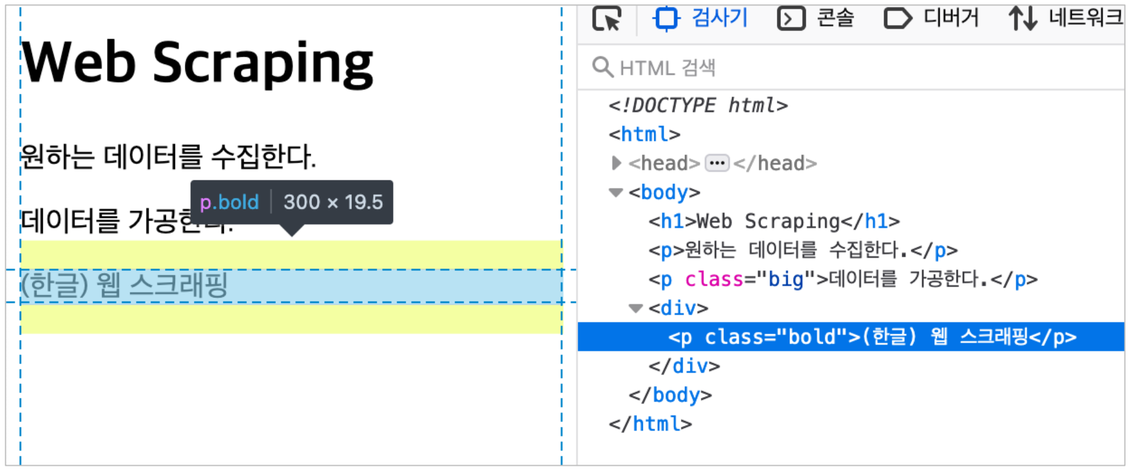


HTML 문서에서 요소를 우클릭 함으로써 XPath도 복사할 수 있음

XPath로 요소 찾기

HTML

```
<html>
<body>
  <h1>Web Scraping</h1>
  <p>원하는 데이터를 수집한다.</p>
  <p class="big">데이터를 가공한다.</p>
  <div>
    <p class="bold">(한글) 웹 스크래핑</p>
  </div>
</body>
</html>
```



> 브라우저를 통해 **복사**해온 **XPath**
>
> -> /html/body/div/p

XPath로 요소 찾기

HTML

```
<html>
<body>
  <h1>Web Scraping</h1>
  <p>원하는 데이터를 수집한다.</p>
  <p class="big">데이터를 가공한다.</p>
  <div>
    <p class="bold">(한글) 웹 스크래핑</p>
  </div>
</body>
</html>
```

Python

```
xpath = '/html/body/div/p'  # 복사해온 xpath

e = driver.find_element(By.XPATH,xpath)
print(e.text)
```

실행결과

```
(한글) 웹 스크래핑
```

XPath로 요소 찾기

HTML

```
<html>
<body>
  <h1>Web Scraping</h1>
  <p>원하는 데이터를 수집한다.</p>
  <p class="big">데이터를 가공한다.</p>
  <div>
    <p class="bold">(한글) 웹 스크래핑</p>
  </div>
</body>
</html>
```

Python

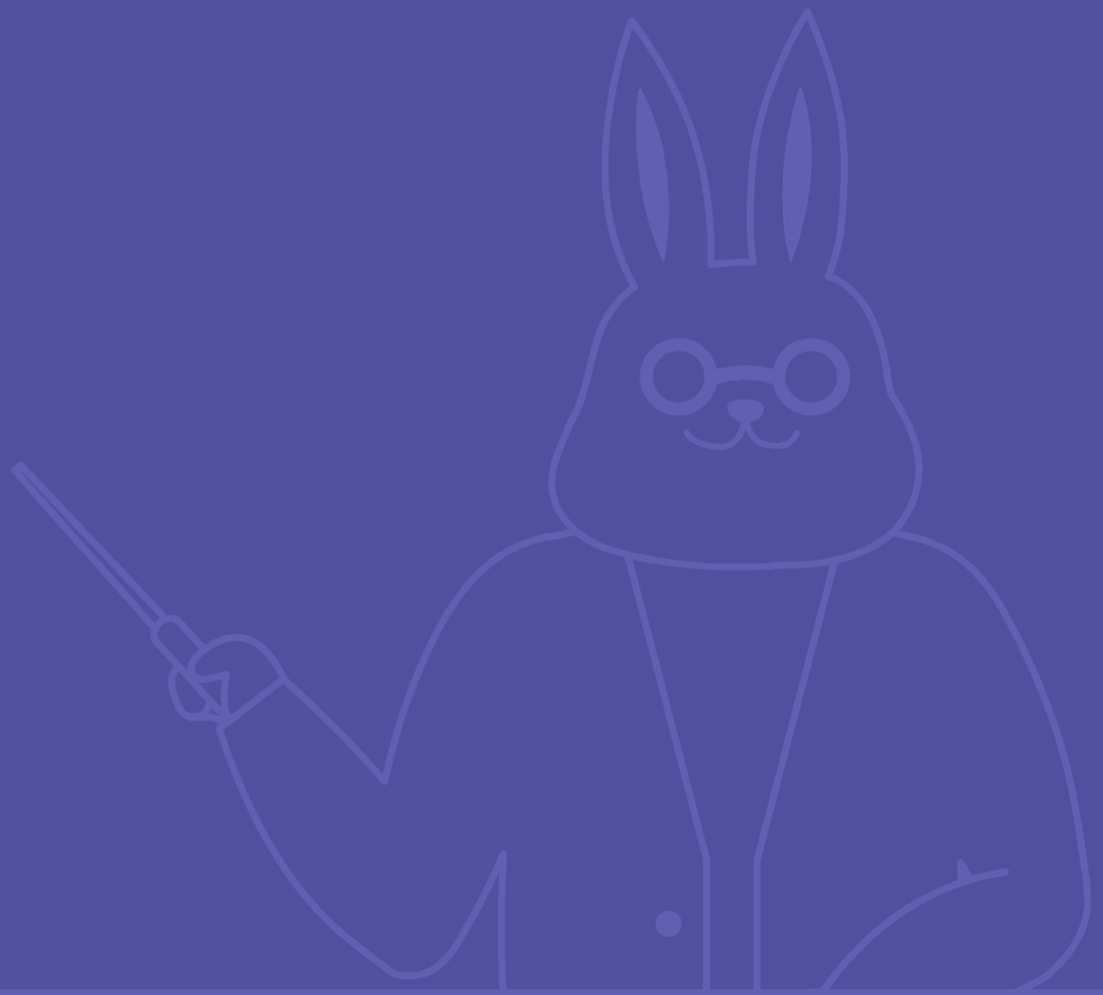
```
xpath = '//p'

e = driver.find_elements(By.XPATH,xpath)
print(e.text)
```

실행결과

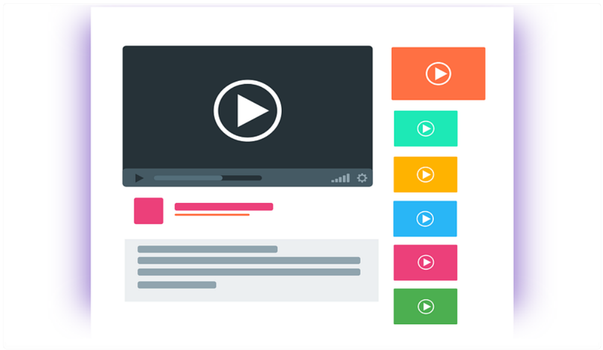
```
원하는 데이터를 수집한다.
데이터를 가공한다.
(한글) 웹 스크래핑
```

06

맺으며

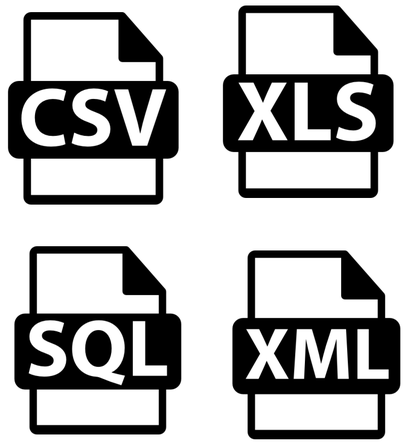


웹 스크래핑과 셀레니움



Website Pages,
Unstructured Data

Web Scraping/
Data Extraction



Structured Data

웹 사이트에서 **원하는** 데이터를 **수집**하고, **가공**하는 행위

셀레니움 활용

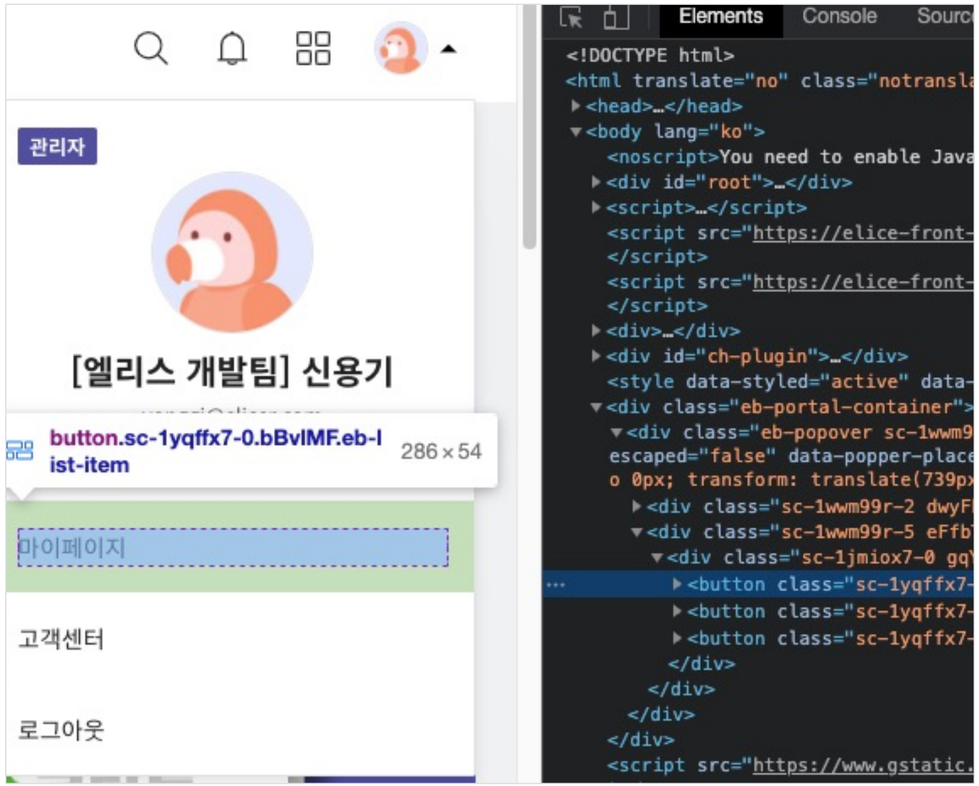
Python

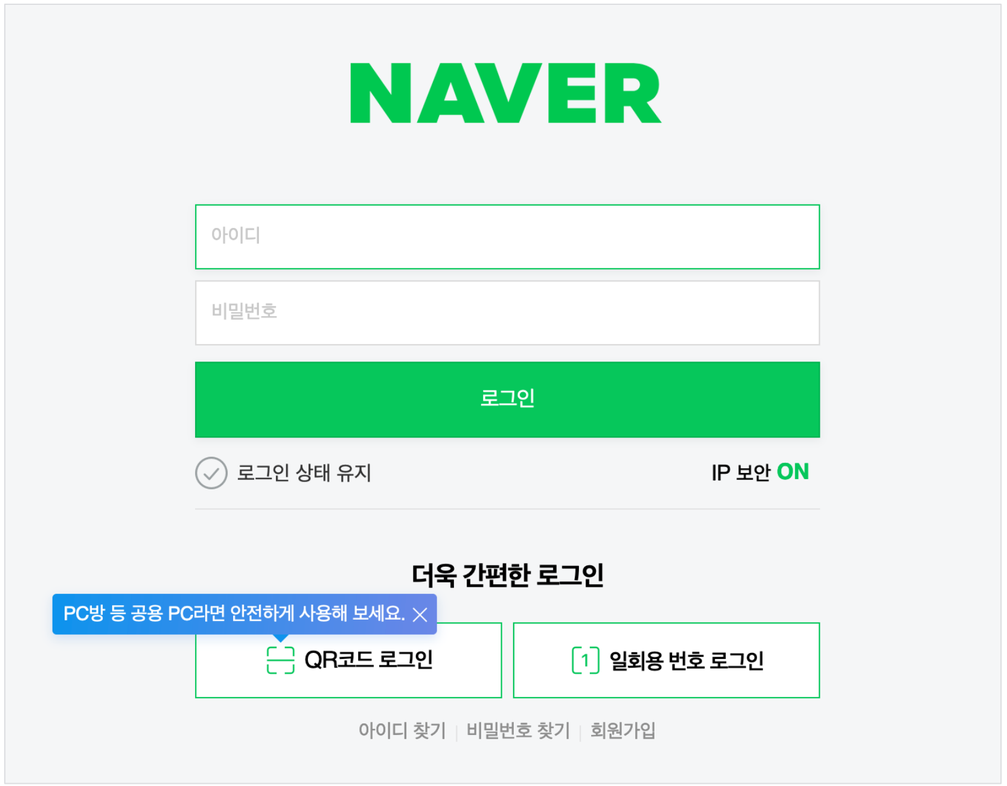
```
from selenium import webdriver

driver = webdriver.Firefox()
driver.get('https://news.naver.com')
...
driver.close()
```

Python

```
driver.find_element(By.TAG_NAME,‘tag_name’)
driver.find_element(By.CLASS_NAME,‘class_name’)
driver.find_element(By.ID,‘id’)
driver.find_element(By.XPATH,‘xpath’)

driver.find_elements(By.TAG_NAME,‘tag_name’)
driver.find_elements(By.CLASS_NAME,‘class_name’)
driver.find_elements(By.ID,‘id’)
driver.find_elements(By.XPATH,‘xpath’)
```

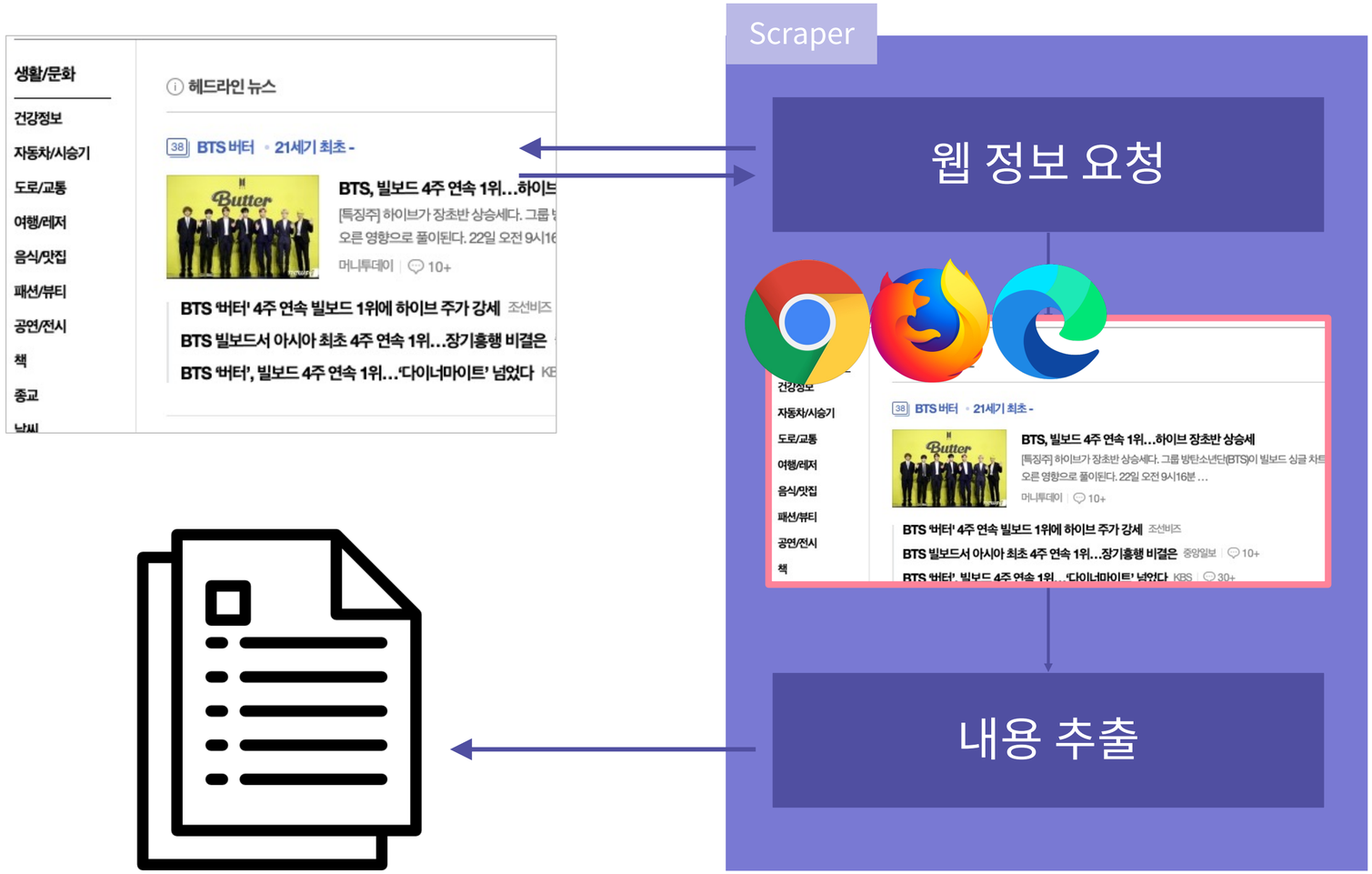
다음 시간 예고





드롭다운 요소, 로그인이 필요한 페이지 등 **제어가 필요한** 웹 페이지

다음 시간 예고



브라우저를 직접 **제어**하는 방법을 배워봅시다!

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