웹 크롤링과 정적 데이터 수집

made by Jay Hong

Check

웹 크롤링과 정적 데이터 수집

1. 웹 크롤링이란?

2. 정적 데이터 수집 방법

1. 웹 크롤링이란?

1. 웹 크롤링의 정의
2. 웹의 구조
3. HTTP
4. HTML
5. 인코딩
6. 실습

1) 크롤링의 정의

- ✓ 웹 컨텐츠를 수집하기 위해
  자동으로 웹사이트를 방문하여 컨텐츠를 수집하는
  프로세스 혹은 프로그램

1) 웹 구조의 이해

- ✓ 웹 주소의 입력
  - ✓ 해당 주소에 해당하는 문서를 호출하는 행위
- ✓ HTTP : hyptertext transfer protocol
  - ✓ 웹에서 사용하는 통신 규약
- ✓ 웹 페이지
  - ✓ 웹상의 문서로 텍스트, 그림, 소리, 동영상 등을 표현하여 대부분 HTML로 구성
- ✓ CSS
  - ✓ HTML에서 서식을 지정할 때 사용하는 형식
- ✓ JAVA SCRIPT
  - ✓ HTML에서 여러가지 동적인 기능을 부여하는 프로그래밍 언어

2) URL의 구조

- ✓ http://www.domain.com:1234/path/to/resource?a=b&x=y
  - ✓ http : protocol
  - ✓ www.domain.com : host
  - ✓ 1234 : port
  - ✓ path/to/resource : 원하는 resource가 있는 경로
  - ✓ ?a=b&x=y : query : 원하는 정보를 불러오는 방법
  - ✓ Example)
    https://news.naver.com/main/ranking/popularDay.naver?mid=etc&sid1=111
    보안 프로토콜에서, 네이버 뉴스 호스트로 들어가
    오늘자 ranking이 담긴 경로로 이동하여, 1번 페이지를 보겠다.

1) HTTP 구조

- ✓ 웹에서 HTML 문서 등을 주고 받을때 사용하는 프로토콜
- ✓ 클라이언트는 request를 서버에 보냄
- ✓ 서버는 request에 응답해 필요한 정보를 사용자에게 전달 (response)
- ✓ request : 서버에 보내는 요청
  - ✓ Get : 웹페이지의 자료를 요청
  - ✓ post : 관련 데이터도 함께 보내 요청 (보안 필요시 key를 함께 보내는 등)

2) HTTP 응답

- ✓ HTTP 상태 코드
  - ✓ 2XX : 성공
  - ✓ 3XX : 자료가 존재하지만 위치가 바뀌어 , 자동으로 페이지를 전환해 요청
  - ✓ 4XX : 클라이언트 오류
    - ✓ 400 : 잘못된 요청
    - ✓ 404 : 잘못된 주소
    - ✓ 405 : get, post 등 통신 방식 오류
  - ✓ 5xx : 서버 오류
    - ✓ 500 : 서버 내부 에러
    - ✓ 503 : 서버 폭주 등

3) User Agent

- ✓ User Agent String
  - ✓ 브라우저의 종류를 나타내는 표현

- ✓ 403 에러
  - ✓ 일부 사이트는 크롤링을 제한하기 위해 비정상적 웹브라우저의 접속 차단

  - ✓ 이 경우, 정상적 웹브라우저인 척 크롤링 가능

1) HTML 구성요소

- ✓ HTML
  - ✓ Hyper Text Markup Langugae

- ✓ 트리 형태의 구조
  - ✓ <html>로 시작하여 </html>로 종료
  - ✓ 그 아래에 <head> , <body> 등 다양한 태크가 존재

- ✓ HTML의 구성
  - ✓ 모든 노드는 TAG로 감싸져 있음
  - ✓ <> 안에 태그 이름 기입
  - ✓ <tagname> 내용 </tagname> 형식으로 작성
  - ✓ 노드는 내용과 별개로 속성을 가짐
  - ✓ 문단
    - ✓ p는 문단을 의미하는 html tag
    - ✓ align이 여기서 “속성 " 을 의미, align=center 라면 가운데 정렬이라는 뜻

2) 자주 사용하는 TAG

- ✓ 제목
  - ✓ <h1> <h2> …
- ✓ 문단
  - ✓ <p>
- ✓ 링크
  - ✓ <a href = ‘링크’>
- ✓ 이미지
  - ✓ <img scr = ‘파일명’ width=‘넓이’ height = ‘높이’>
- ✓ 리스트
  - ✓
     ~~
     ~~

2) 자주 사용하는 TAG

- ✓ 구역
  - ✓ 문서에서 일부분을 하나의 구역으로 지정
  - ✓

- ✓ 테이블
  - ✓ <table>
  - ✓ <tr> : 각 행 구분
  - ✓ <th> : 각 열의 제목
  - ✓ <td> : 각 칸의 내용

- ✓ 텍스트 서식
  - ✓ <b> 굵게
  - ✓ <i> 기울임
  - ✓ 중요한 텍스트

2) 자주 사용하는 TAG

- ✓ alt : 이미지가 출력되지 않을 때 출력한 텍스트
- ✓ src : 이미지 주소나 파일 명
- ✓ href : URL 링크 주소
- ✓ id : element에 아이디 부여
- ✓ class : 같은 유형의 element

1) 인코딩의 의미

- ✓ 숫자를 인식하는 단위 : 1Bit
- ✓ 문자를 인식하는 단위 : 1Byte = 8Bit
- ✓ 컴퓨터는 문자를 숫자로 바꿔 기억하는 특성
- ✓ 인코딩
  - ✓ 컴퓨터가 문자를 처리하기 위해 이진수로 변환되는 표준 규칙
  - ✓ 띄어쓰기 개념이 없어 어디서 끊어야 하는지 모호해지는 문제 발생
  - ✓ 해결하기 위해 1Byte를 한 글자로 인식해 255문자로 표현
  - ✓ 숫자가 길어짐에 따라 문자를 쓸 때는 16진수 사용 (0x…)

2) 국가별 인코딩 방법

- ✓ 영어
  - ✓ ASCII 인코딩
- ✓ ISO8859
  - ✓ 라틴어 등 알파벳 이외의 문자를 ASCII 빈칸에 할당
- ✓ 한글
  - ✓ 128Byte로 표현이 불가능 (11,172개 글자)
  - ✓ 완성형으로 euc-kr 사용
  - ✓ 윈도우 독자적 cp949 사용
- ✓ Unicode : 국제 표준
  - ✓ utf-8, utf-16

실습

✓ https://news.naver.com/ 에 접속합시다.

1) 네이버 뉴스에서 f12를 눌러 HTML을 살펴봅시다.

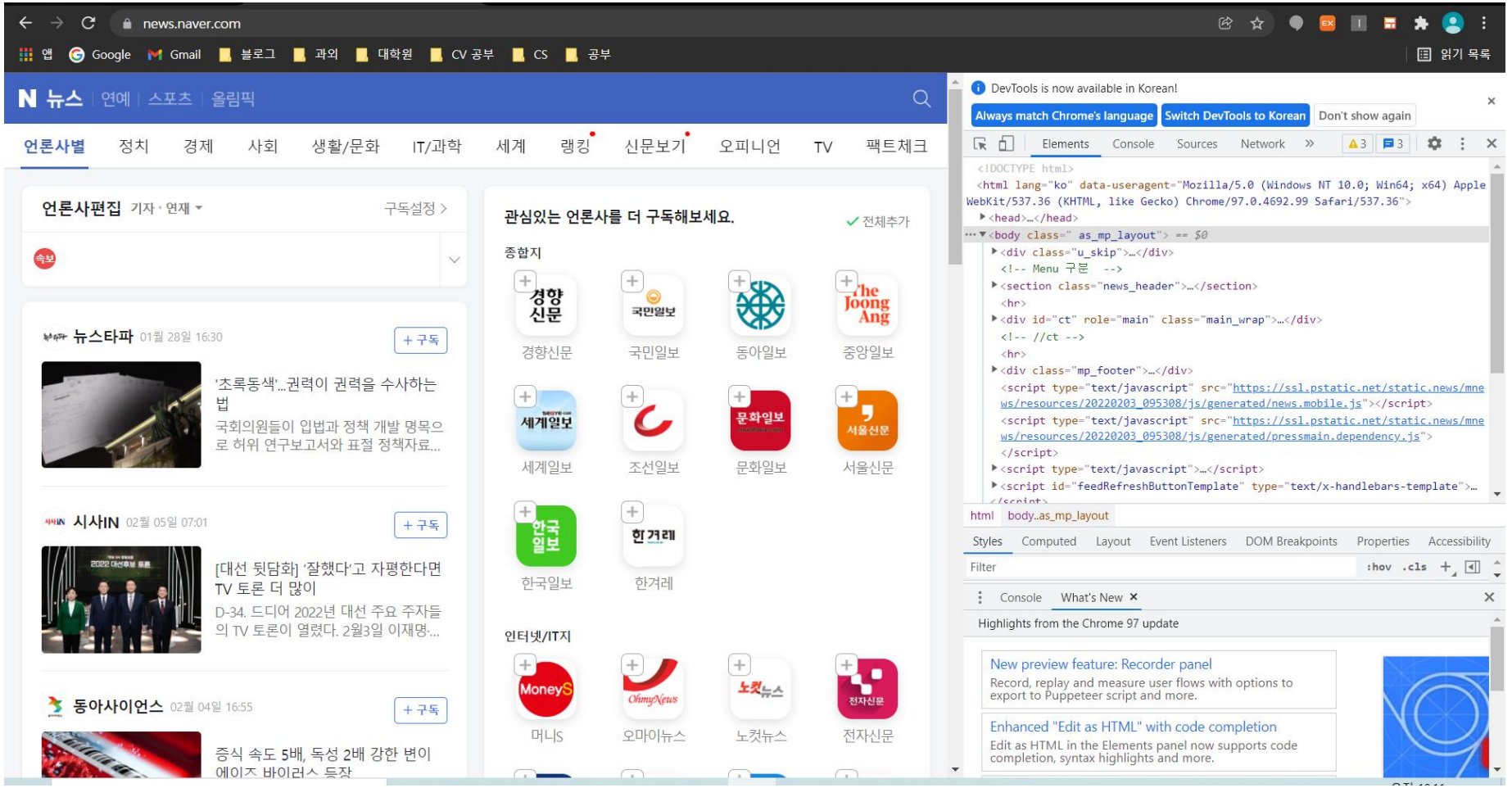
2) 상단 바에서 “IT/과학“을 우클릭 해 “검사”버튼을 눌러봅시다.

실습

✓ https://news.naver.com/ 에 접속합시다.

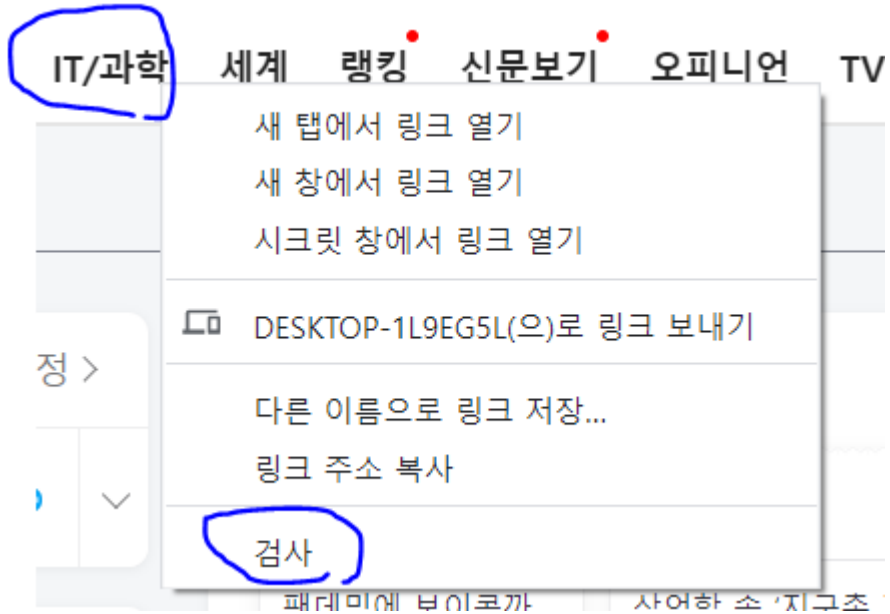
실습

✓ 네이버 뉴스에서 f12를 눌러 HTML을 살펴봅시다.



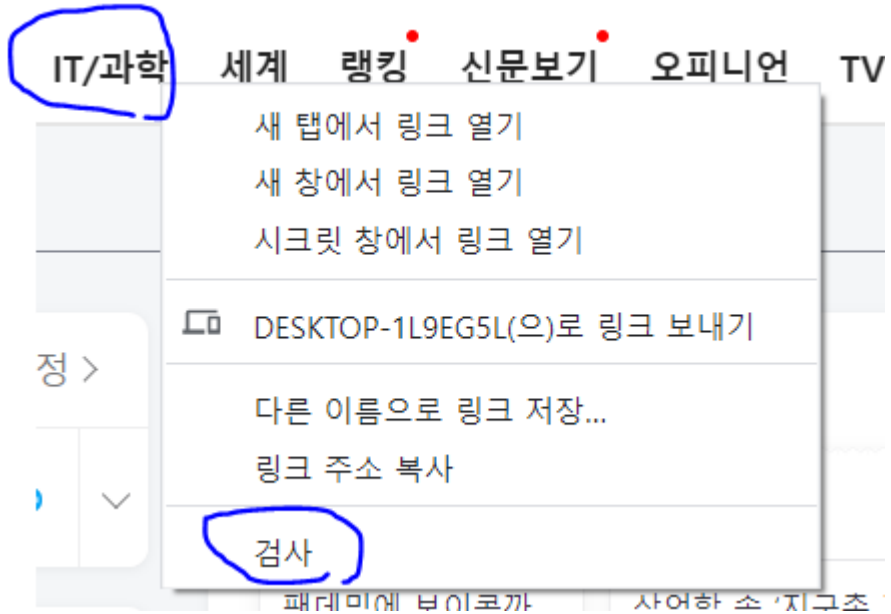
실습

✓ 상단 바에서 IT/과학을 우클릭 후 “검사“ 버튼을 눌러봅시다.



실습

✓ 상단 바에서 IT/과학을 우클릭 후 “검사” 버튼을 눌러보고, 결과를 해석해봅시다.




```
▶ <li class="Nlist_item is_active">…</li>
▶ <li class="Nlist_item">…</li>
▶ <li class="Nlist_item">…</li>
▶ <li class="Nlist_item">…</li>
▶ <li class="Nlist_item">…</li>
▼ <li class="Nlist_item">
  ▼ <a href="https://news.naver.com/main/main.naver?mode=LSD&
  mid=shm&sid1=105" class="Nitem_link" role="menuitem" aria-
  selected="false" onclick="nclk(event,'lnb.sci','','');">
      <span class="Nitem_link_menu">IT/과학</span> == $0
    </a>
  </li>
▶ <li class="Nlist_item">…</li>
▶ <li class="Nlist_item _isNew is_new">…</li>
▶ <li class="Nlist_item _isNew is_new">…</li>
▶ <li class="Nlist_item">…</li>
▶ <li class="Nlist_item">…</li>
▶ <li class="Nlist_item">…</li>
```

실습

- ✓ 그 외에 원하는 버튼을 클릭 -> 검사를 통해 html의 구조를 파악해봅시다.

2. 정적 데이터 수집

1. 정적, 동적 페이지
2. 정적 데이터 수집 Process
3. 실습

1) 정적 웹 페이지

- ✓ 서버에 미리 저장된 파일이 그대로 전달되는 웹 페이지
- ✓ 사용자의 요청에 따라 저장된 값을 고정된 상태로 그대로 전달
- ✓ 이러한 페이지들은 주로 host뒤의 path에 차이를 두는 경우가 많음

Example) 네이버 IT/과학 페이지 번호를 옮겨보기

1페이지
https://news.naver.com/main/main.naver?mode=LSD&mid=shm&sid1=105#&date=%2000:00:00&page=1

2페이지
https://news.naver.com/main/main.naver?mode=LSD&mid=shm&sid1=105#&date=%2000:00:00&page=2

2) 동적 웹 페이지

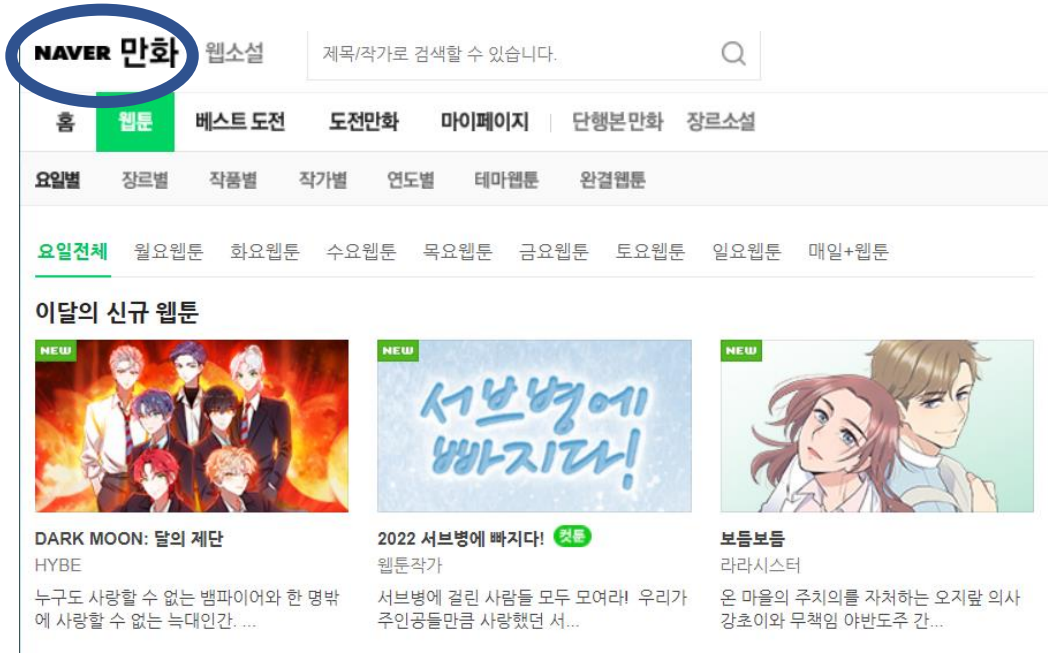
- ✓ 서버에 있는 데이터들을 Script에 의해 가공처리한 후 생성되어 전달되는 웹페이지
  - ✓ 클릭 -> 서버내부 처리 -> 페이지 가공 -> 유저에게 전달
- ✓ 사용자는 상황, 시간, 요청 등에 따라 달라지는 웹페이지를 보게 됨
- ✓ 이러한 웹페이지는 페이지 내부에서 클릭해도 url이 바뀌지 않음
- ✓ F12를 누른 상태에서 보면 반짝거리며 실시간으로 수정되는게 보임

Example) https://data.kma.go.kr/data/grnd/selectAsosRltmList.do?pgmNo=36

- ✓ 해당 페이지에서 기간을 수정해보자.

1) 과정

- ✓ 정적 웹페이지의 자료를 Get 혹은 Post로 받는다.
- ✓ 이 자료는 HTML로 구성되어 있기 때문에, 해석을 위해 'PARSING' 이라는 단계를 거친다.
- ✓ CSS를 이용해 원하는 부분을 선택해서 뽑아낸다.
- ✓ 예시 목표 : 네이버 웹툰의 제목 가져오기!



과정 1: 정적 웹페이지의 자료를 Get 혹은 Post로 받는다.

- ✓ Request Library 사용
- ✓ 파이썬에 내장되어있는 requests library를 사용하면 특정 url의 데이터를 받을 수 있다.

```
import urllib
```

```
url = 'https://comic.naver.com/index'

res = requests.get(url)
res
```

```
<Response [200]>
```

```
res.encoding
```

```
'UTF-8'
```

- ✓ Response [200]은 정상적으로 데이터를 수집했음을 의미한다.
- ✓ UTF-8을 이용해서 encoding 되어있음을 확인할 수 있다.

과정 2 : Parsing

✓ Parsing의 뜻

어떤 페이지에서 내가 원하는 데이터를
특정 패턴이나 순서로 추출해 가공하는 과정

현재 오른쪽 이미지를 보면,
HTML코드를 그대로 불러왔기 때문에
패턴이나 순서가 사라져 있음을 알 수 있다.

res.text

'₩r₩n₩r₩n₩r₩n ₩r₩n₩r₩n₩r₩n₩r₩n₩r₩n₩r₩n₩r₩n₩r₩n<!DOCTYPE html>₩r₩n<html lang
="ko">₩r₩n<head>₩r₩n₩t₩r₩n₩t₩t₩r₩n₩t₩t₩r₩n₩t₩t₩r₩n₩t₩t₩r₩n₩t₩t<meta http-equi
v="X-UA-Compatible" content="IE=edge,chrome=1">₩r₩n₩t₩t₩r₩n₩t<meta http-equiv
="Content-type" content="text/html; charset=UTF-8">₩r₩n₩t<title>네이버 만화</
title>₩r₩n₩t₩r₩n₩r₩n₩r₩n₩r₩n₩r₩n₩r₩n₩r₩n₩r₩n₩r₩n₩r₩n₩r₩n₩r₩n₩r₩n₩r₩n₩r₩n₩r₩n
₩r₩n₩r₩n₩r₩n ₩r₩n ₩r₩n ₩r₩n ₩r₩n ₩r₩n ₩r₩n ₩r₩n ₩r₩n
₩r₩n ₩r₩n ₩r₩n <meta property="og:title" content="네이버 웹툰" >
₩r₩n <meta property="og:image" content="https://ssl.pstatic.net/stati
c/comic/images/og_tag_v2.png" >₩r₩n <meta property="og:description" co
ntent="매일매일 새로운 재미, 네이버 웹툰.">₩r₩n ₩r₩n₩r₩n₩r₩n<meta property

이를 원본 HTML처럼 복원하고,
우리가 원하는 부분을 추출하기 위해 parsing이라는 단계를 거친다.

과정 2 : Parsing

- ✓ Parsing을 지원하는 Library
  - ✓ Beautiful Soup => 채택
    - ✓ 장점 : 깨진 HTML을 잘 인식
    - ✓ 단점 : 기능이 적고 느림
  - ✓ lxml
    - ✓ 장점 : 기능이 많고 빠름
    - ✓ 단점 : HTML 인식이 어려움
- ✓ BeautifulSoup 설치

```
pip install --user beautifulsoup4
```

- ✓ Beautiful Soup 불러오기

```
from bs4 import BeautifulSoup
```

과정 2 : Parsing

✓ Parsing 적용하기

이제 어떤 tag에
어떤 content가 담겨있는지
쉽게 파악할 수 있다.

```
from bs4 import BeautifulSoup
```

```
soup = BeautifulSoup(res.text, 'html.parser')
```

```
soup
```

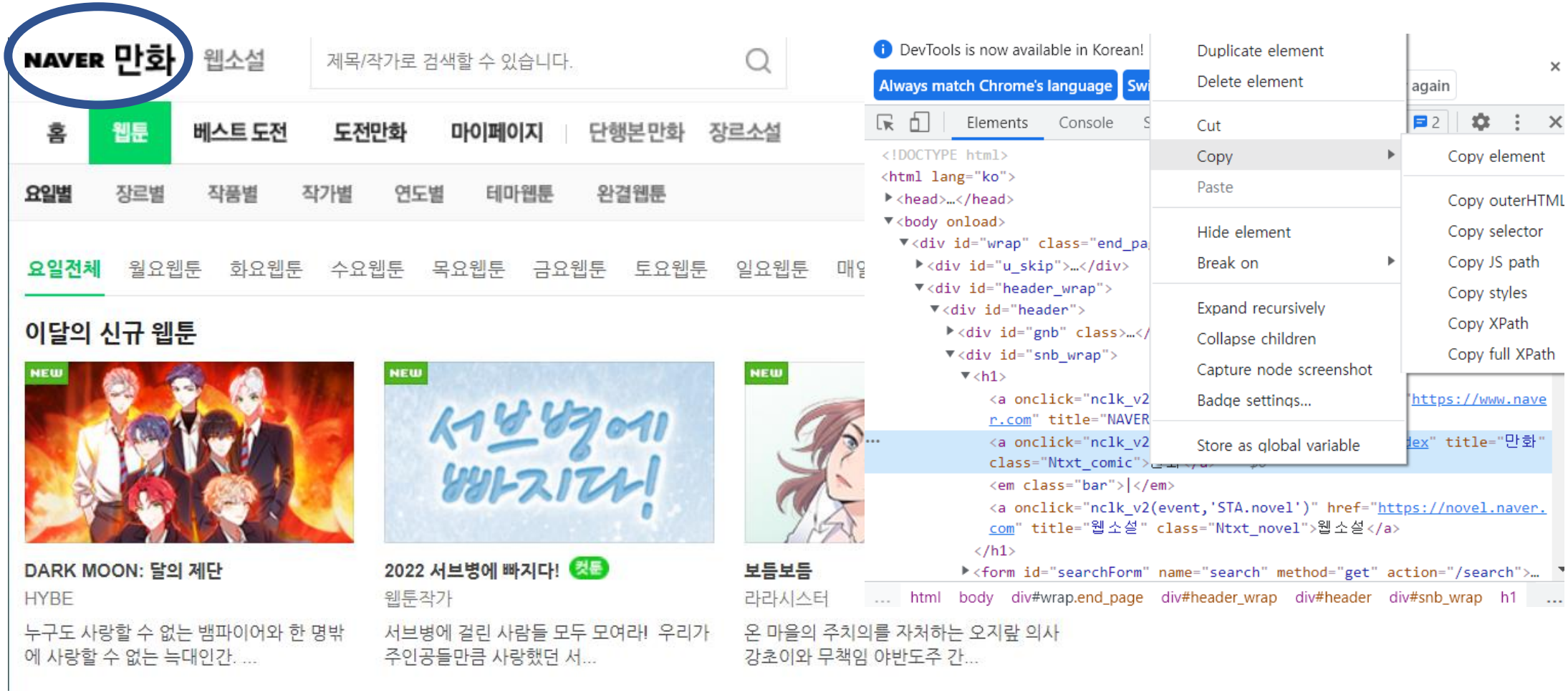
```
<!DOCTYPE html>

<html lang="ko">
<head>
<meta content="IE=edge,chrome=1" http-equiv="X-UA-Compatible"/>
<meta content="text/html; charset=utf-8" http-equiv="Content-type"/>
<title>네이버 만화</title>
<meta content="네이버 웹툰" property="og:title"/>
<meta content="https://ssl.pstatic.net/static/comic/images/og_tag_v2.png" prope
rty="og:image"/>
<meta content="매일매일 새로운 재미, 네이버 웹툰." property="og:description"/>
```

과정 3 : CSS를 이용해 원하는 부분을 선택해서 뽑아낸다.

- ✓ 마지막으로, 정리한 HTML 구조에서 원하는 부분을 추출해야 한다.
- ✓ 특정 부분을 추출하는 방법은 2가지를 많이 사용한다.
- ✓ CSS 선택자
  - ✓ HTML 문서의 서식을 지정하기 위한 선택자로, 짧고 간단
  - ✓ 대부분의 경우에 충분히 가능
- ✓ XPATH
  - ✓ 복잡한 조건으로 노드를 선택할 때 사용
  - ✓ lxml은 이를 xpath를 사용해서 함

과정 3 : CSS를 이용해 원하는 부분을 선택해서 뽑아낸다.

- ✓ 동그라미 부분 우클릭 -> 검사 -> 색칠된 부분 우클릭 -> copy -> copy selector



과정 3 : CSS를 이용해 원하는 부분을 선택해서 뽑아낸다.

- ✓ #snb_wrap > h1 > a.Ntxt_comic
- ✓ 해당 tag에 “만화”라는 글자가 담겨있다는 뜻이다.
- ✓ 따라서 soup를 이용해 해당 부분을 select하면 list로 담아서 준다.
- ✓ 해당 list에서 원하는 부분을 가져와 .get_text()를 사용하면 내부의 content를 반환한다.

```
selector = '#snb_wrap > h1 > a.Ntxt_comic'
soup.select(selector)
```

```
[<a class="Ntxt_comic" href="/index" onclick="nclk_v2(event,'STA.comic')" title="만화">만화</a>]
```

```
soup.select('#snb_wrap > h1 > a.Ntxt_comic')[0].get_text()
```

```
'만화'
```

네이버에 “AI”를 검색한 후, 뉴스 탭에서 뉴스 제목 100개를 수집해봅시다.