|  |  |  |
| --- | --- | --- |
| 목차 | | |
| Chapter 1 | 크롤링의 이해 | 1 |
| Chapter 2 | 자바스크립트 문법 | 5 |
| Chapter 3 | puppeteer 소개 | 26 |
| Chapter 4 | 프로젝트 | 32 |
| Chapter 5 | 오류 처리 | 46 |

|  |  |
| --- | --- |
| Chapter 1 | 크롤링의 이해 |

**1. 크롤링이란?**

크롤링(Crawling)은 웹 콘텐츠를 검색하고 해당 콘텐츠(이미지, 텍스트 등)를 추출하는 행위이다.

크롤링은 인터넷 서비스의 핵심 기능 중 하나인 “검색서비스”에서 쓰이는 대표적인 기술이며, 전자상거래상에서의 가격 비교, 정보보호 서비스 등 다양한 분야에서 활용되고 있다.

**2. 크롤링 이야기**

기술적 측면

크롤링은 기본적으로 웹서비스와 같이 개방성을 전제로 한 서비스상에 존재하는 콘텐츠를 통신 프로토콜을 활용하여 수집하는 방식이므로, 기술적으로는 정상적인 콘텐츠 요청과 특별히 다르지 않다

보안 측면

인터넷의 서비스 자체가 태생적으로 개방성을 전제로 하므로 접근제어의 책임이 제공자에게 달려있다.

따라서 공개된 자료이더라도, 특정한 이용자만이 접근하기를 원한다면 서비스 제공자가 이에 대한 기술적 보호조치가 필요하다.

윤리적 측면

각 사이트의 페이지 루트 도메인에서 /robots.txt를 통해 규칙이 정해져 있다.

>> <https://www.google.com/robots.txt>

/robots.txt

Robots.txt는 웹사이트에서 웹 크롤러 같은 로봇들의 접근을 제어하기 위한 규약이다.

물론 권고 정도의 접근 방지 설정이라 파일에 접근할 수 있다.

하지만 상업적으로 사용하는 행위에는 별개의 문제이다. 조심하자.

규칙

User-agent : 규칙이 적용되는 크롤러를 식별한다.

Allow : 크롤링할 수 있는 URL 경로이다.

Disallow : 크롤링할 수 없는 URL 경로이다.

|  |
| --- |
| Youtube의 robots.txt 내용 중 일부이다.  Disallow: /login |

**3. 이런 순서로 배워요**

Javascript의 문법은 다음과 같은 순서로 배운다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Javascript란? | 입출력문 | 자료형 | 조건문 | 반복문 |
| 함수 | 예외처리 | 비동기 |  |  |

**4. 웹 크롤링 과정**

웹 크롤러 -> 웹사이트 방문 -> 데이터 인덱싱 -> 데이터베이스 저장

**5. 비주얼 스튜디오 코드 실행**

[1] 프로젝트 폴더 만들기 : 프로젝트 폴더는 프로그램과 관련된 파일이 위치하는 폴더이다. (로컬디스크 (D:) : 우클릭 – [새로 만들기] – [폴더])

>> 폴더 이름을 “ssf”로 만들자.

[2] 파일 만들기 : 프로그램 명령어들을 직접 입력할 파일을 만든다.

탐색기 아래의 폴더 이름에 마우스를 가져다 대면 4개의 아이콘이 뜬다. 그중 왼쪽에서 첫 번째 아이콘을 클릭하면 파일의 이름을 정할 수 있다.

>> index.js 파일을 만들고 다음 코드를 입력해 보자.

console.log(“SSF”);

[3] 실행하기 :

가. [실행] – [디버깅 시작]을 선택하여 프로그램을 실행한다.

나. [ctrl+`] – 터미널에 node [파일이름] – 엔터로 프로그램을 실행한다.

예시 : node index.js

터미널을 이용하는 방식을 주로 사용하자.

>> 완성된 파일을 실행해보자.

|  |
| --- |
| 많이 사용하는 단축키  저장 : ctrl+s  터미널 : ctrl+` ( ` 백틱은 1 왼쪽에 있는 ~키이다.) |

|  |
| --- |
| Javascript 참고 사이트  <https://www.w3schools.com/js/default.asp>  <https://ko.javascript.info/> |

|  |  |
| --- | --- |
| Chapter 2 | 자바스크립트 문법 |

**6. Javascript란**

자바스크립트는 ‘웹페이지에 생동감을 불어넣기 위해’ 만들어진 프로그래밍 언어이다.

자바스크립트로 작성한 프로그램을 스크립트(script)라고 부른다.

자바스크립트는 브라우저에서만 쓸 목적으로 고안된 언어지만, 지금은 다양한 환경에서 쓰이고 있다.

**7. Javascript의 강점**

웹을 구성하는 HTML/CSS와 완전히 통합할 수 있다.

모든 주요 브라우저에서 지원하고, 기본 언어로 사용된다.

**8. 개발자 콘솔**

여러분이 로봇이 아닌 사람이라면 분명 에러를 만들 것이다.

그러나 브라우저는 스크립트에 문제가 있어서 에러가 발생해도 이를 사용자에게 직접 보여주지 않는다.

브라우저엔 ‘개발자 도구’라는 것이 내장되어 있다. 이 도구를 이용하면 에러를 확인할 수 있다. 스크립트에 대한 쓸만한 정보도 얻을 수 있다.

개발자 도구를 열어 에러를 확인하고, 살펴보자.

|  |
| --- |
| **Chrome**  F12를 눌러보자. Mac 사용자라면 Cmd + Opt + J 를 누르면 된다.  개발자 도구가 보일 것이다. 개발자 도구를 처음 열어봤다면 Console 패널이 기본으로 보인다.    빨간색 에러 메시지가 보인다. ‘lalala’가 정의되지 않았다(not defined)라는 메시지이다.  bug.html:12:5가 보인다. bug.html은 에러가 발생한 파일, 12는 에러가 발생한 줄을 나타낸다.  에러 메시지 아래에 파란색 기호 > 가 있다.  이 기호가 있는 곳엔 자바스크립트 명령어(command)를 입력할 수 있다.  화면 구성은 사용하고 있는 Chrome 버전에 따라 다르다.  버전이 바뀔 때마다 구성이 조금씩 바뀌지만 큰 틀은 바뀌지않는다. |

**9. 출력함수(console.log)**

console.log를 이용하여 문자열을 출력하거나 인수를 서식문자에 맞게 출력할 수 있다.

console.log('2022 ssf AnA');

|  |
| --- |
| 실행결과  2022 ssf AnA |

**10. 변수(variable)와 상수(constant)**

변수는 데이터를 저장할 때 쓰이는 ‘이름이 붙은 저장소’이다.

자바스크립트에서는 let 키워드를 통해 변수를 생성한다.

‘message’ 라는 이름을 가진 변수를 생성(선언)한다.

let message;

할당 연산자 = 를 사용해 변수안에 데이터를 저장한다.

let message;

message = 'Hello World'; //문자열을 저장한다.

변수 선언과 값 할당을 한 줄에 작성할 수도 있다.

let message = 'Hello World'; //변수를 정의하고 값을 할당한다.

한 줄에 여러 변수를 선언하는 것도 가능하다.

let message = 'Hello World', sunrin = 'highschool';

변수명을 이용해 문자열에 접근할 수 있게 된다.

let message = 'Hello World';

console.log(message); // Hello World

|  |
| --- |
| 변수명 작성 규칙  변수명에는 오직 문자와 숫자, 그리고 기호 $와 \_만 들어갈 수 있다.(영문자의 대문자와 소문자는 서로 다른 문자로 인식한다.)  let $ = 1; // '$' 라는 이름의 변수를 선언한다.  let \_ = 2; // '\_' 라는 이름의 변수를 선언한다.  첫 글자는 숫자가 될 수 없다.  let 1a; // 변수명은 숫자로 시작해선 안된다.  예약어(reserved name)는 변수명으로 사용할 수 없다.  let let = 5; // 'let'을 변수명으로 사용할 수 없으므로 에러 |

변화하지 않는 변수를 선언할 땐, let 대신 const를 사용한다.

const birthday = '050506'

이렇게 const로 선언한 변수를 ‘상수(constant)’라고 부른다.

상수는 재할당할 수 없다.

const birthday = '050506'

birthday = '060506' // 상수는 재할당할 수 없으므로 에러

**11. 자료형**

자바스크립트에서 값은 항상 문자열이나 숫자형 같은 특정한 자료형에 속한다.

자바스크립트에는 여덟 가지 기본 자료형이 있다.

자바스크립트에서는 변수가 어떤 순간에 문자열일수도 다른 순간엔 숫자일수도 있다.

let message = 'string';

let = 123

**-숫자형**

let n = 123;

n = 1.23;

숫자형은 정수 및 부동소수점 숫자(floating point number)를 나타낸다.

숫자형과 관련된 연산은 곱셈 \*, 나눗셈 /, 덧셈 +, 뺄셈 – 이 대표적이다

**-BigInt**

BigInt형은 길이에 상관없이 정수를 나타낼 수 있습니다.

let bigInt = 12345678910111213141516171819202122;

**-문자형**

let str = 'ssf';

str = 'sunrin'

자바스크립트에선 문자열(string)을 따옴표로 묶는다.

따옴표는 세 종류가 있다.

가. 큰따옴표: “Hello”

나. 작은따옴표: ‘Hello’

다. 역 따옴표(백틱, backtick): `Hello`

**-불린형**

불린형(논리 타입)은 true와 false 두 가지 값밖에 없는 자료형이다.

불린형은 긍정(yes)이나 부정(no)을 나타내는 값을 저장할 때 사용한다.

true는 긍정, false는 부정을 의미한다.

불린값은 비교 결과를 저장할 때도 사용된다.

let isGreater = 4 > 1;

console.log(isGreater); // true

큰따옴표와 작은따옴표는 ‘기본적인’ 따옴표로, 둘은 차이가 없습니다.

역 따옴표로 변수나 표현식을 감싼 후 ${…}안에 넣어주면, 원하는 변수나 표현식을 중간에 손쉽게 넣을 수 있습니다.

let club = 'AnA';

console.log(`Hello, ${club}`); // Hello, AnA

**-‘null’ 값**

null값은 지금까지 소개한 자료형 중 어느 자료형에도 속하지 않는 값이다.

null은 ‘존재하지 않는(nothing)’ 값, ‘비어있는(empty)’ 값, ‘알 수 없는(unknown)’값을 나타내는데 사용한다.

let age = null;

**-‘undefined’ 값**

undefined 값도 null 값처럼 자신만의 자료형을 형성한다.

undefined는 ‘값이 할당되지 않은 상태’를 나타내는데 사용한다.

let age; // 'undefined'

**-객체형과 심볼형**

객체(object)형은 특수한 자료형이다.

객체형은 다양한 데이터를 담을 수 있다.

키로 구분된 데이터 집합이나 복잡한 개체(entity)를 저장할 수 있다.

객체는 중괄호 {…}를 이용해 만들 수 있다.

중괄호 안에는 ‘키(key): 값(value)’ 쌍으로 구성된 프로퍼티(property)를 여러 개 넣을 수 있다.

키엔 문자형, 값엔 모든 자료형이 허용된다.

빈 객체를 만드는 방법은 두 가지가 있습니다.

let user = new Object(); // '객체 생성자' 문법

let user = {}; // '객체 리터럴' 문법

중괄호 {…}안에는 ‘키:값’ 쌍으로 구성된 프로퍼티가 들어간다.

let club = {

  name: 'AnA', // 키: 'name', 값: 'AnA'

  age: 13 // 키: 'age', 값: 13

};

\*심볼형은 다루지 않고 넘어가도록 하겠습니다.

**12. 조건문**

조건에 따라 다른 행동을 취해야 할 때가 있다.

이럴 땐, if문과 ‘물음표’ 연산자라고도 불리는 조건부 연산자 ? 를 사용하면 된다.

**-‘if’ 문**

if(…)문은 괄호 안에 들어가는 조건을 평가하는 데, 그 결과가 true이면 코드 블록이 실행된다.

let year = 2022

if (year == 2022) {

  console.log('멋지네요!');

}

조건이 true일 때 복수의 문을 실행하고 싶다면 중괄호로 코드 블록을 감싸야 한다.

**-‘else’ 절**

if문엔 else절을 붙일 수 있다.

else 뒤에 이어지는 코드 블록은 조건이 거짓일 때 실행된다.

let year = 2022

if (year == 2022) {

  console.log('멋지네요!');

} else {

  console.log('아쉽네요');

}

**-‘else if’로 복수 조건 처리**

유사하지만 약간씩 차이가 있는 조건 여러 개를 처리해야 할 때가 있다.

이때 else if를 사용할 수 있다.

let year = 2022

if (year == 2022) {

  console.log('멋지네요!');

} else if (year > 2022) {

  console.log('너무 갔네요!');

} else if (year < 2022) {

  console.log('조금 더!');

}

위 예시에서, 자바스크립트는 조건 year == 2022를 먼저 확인한다.

이 조건이 거짓이라면 다음 조건 year > 2022를 확인하고,

이 조건 또한 거짓이라면 year < 2022를 확인한다.

**-조건부 연산자 ‘?’**

조건에 따라 다른 값을 변수에 할당해줘야 할 때가 있다.

“물음표(question mark)연산자”라고도 불리는 “조건부(conditional) 연산자”를 사용하면 if문보다 더 짧고 간결하게 변형할 수 있다.

조건부 연산자는 물음표 ?로 표시한다.

피연산자가 세 개이기 때문에 조건부 연산자를 ‘삼항(ternary) 연산자’라고도 부른다.

let result = condition ? value1 : value2;

평가 대상인 condition이 true라면 value1이, 아니라면 value2가 반환된다.

**13. 반복문**

개발을 하다 보면 여러 동작을 반복해야 하는 경우가 종종 생긴다.

반복문(loop)을 사용하면 동일한 코드를 여러 번 반복할 수 있다.

**-‘while’ 반복문**

while반복문의 문법은 다음과 같다.

while (condition) {

  //코드

}

condition(조건)이 true라면 반복문 본문의 코드가 실행된다.

**-‘for’ 반복문**

for 반복문은 while 반복문보다는 복잡하지만 가장 많이 쓰이는 반복문이다.

for (begin; condition; step) {

  //코드

}

|  |  |  |
| --- | --- | --- |
| begin | i = 0 | 반복문에 진입할 때 단 한 번 실행된다. |
| condition | i < 3 | 반복마다 해당 조건이 확인된다. false라면 반복문을 멈춘다. |
| body | [code] | condition이 true일 동안 계속 실행된다. |
| step | i++ | 각 반복의 body가 실행된 이후 실행된다. |

**-반복문 빠져나오기**

대개는 반복문의 조건이 false가 되면 반복문이 종료된다.

그런데 특별 지시자인 break를 사용하면 언제든 원하는 때에 반복문을 빠져나올 수 있다.

**-다음 반복으로 넘어가기**

continue지시자는 break의 ‘가벼운 버전’이다.

continue는 전체 반복문을 멈추진 않지만, 대신 현재 실행 중인 step을 멈추고 반복문을 다음 step으로 강제로 이동시킨다.

continue는 현재 반복을 종료시키고 다음 반복으로 넘어가고 싶을 때 사용할 수 있다.

**14. 함수**

스크립트를 작성하다 보면 유사한 동작을 하는 코드가 여러 곳에서 필요할 때가 많다.

사용자가 로그인이나 로그아웃을 했을 때 안내 메시지를 보여주는 동작 같은 경우처럼 말이다.

함수를 이용하면 중복 없이 유사한 동작을 하는 코드를 여러 번 호출할 수 있다.

함수 선언(function declaration) 방식을 이용하면 함수를 만들 수 있다.

(함수 선언 방식은 함수 선언문이라고 부르기도 한다.)

function sayMessage() {

  console.log('hello');

}

이처럼 함수를 선언할 수 있다.

function 키워드, 함수 이름, 괄호로 둘러싼 매개변수를 차례로 써주면 함수를 선언할 수 있다.

위 함수에는 매개변수가 없는데, 만약 매개변수가 여러 개 있다면 각 매개변수를 콤마로 구분해 준다. 이어서 함수를 구성하는 코드의 모임인 ‘함수 본문(body)’을 중괄호로 감싸 붙여준다.

새롭게 정의한 함수는 함수 이름 옆에 괄호를 붙여 호출한다.

sayMessage();

함수를 호출하면 함수 본문이 실행된다.

|  |
| --- |
| **매개변수**  매개변수(parameter)를 이용하면 임의의 데이터를 함수 안에 전달할 수 있다.  매개변수는 인수(argument)라고 불리기도 한다.(엄밀히 같진않다.)  아래 예시에서 함수 sayMessage는 매개변수 from과 text를 가진다.  function sayMessage(from, text) {    console.log(from+ ': ' + text);  }  sayMessage('AnA', 'Hello'); |

|  |
| --- |
| **반환값**  함수를 호출했을 때 함수를 호출한 그곳에 특정 값을 반환하게 할 수 있다.  이때 이 특정 값을 반환 값(return value)이라고 부른다.  function sum(a, b) {    return a + b;  }  let result = sum(1, 2);  console.log(result) // 3 |

**15. 예외처리**

아무리 프로그래밍에 능한 사람이더라도 에러가 있는 스크립트를 작성할 수 있다.

원인은 아마도 실수, 예상치 못한 사용자 입력, 혹은 잘못된 서버 응답 등 수천만 가지 이유 때문이다!

에러가 발생하면 스크립트는 ‘죽고’(중시 중단된다), 콘솔에 에러가 출력된다.

그러나 try… catch문법을 사용하면 스크립트가 죽는 것을 방지하고, 에러를 ‘잡아서’(catch) 더 합당한 무언가를 할 수 있게 된다.

**-try…catch**

‘try… catch’문법은 ‘try’와 ‘catch’라는 두 개의 주요 블록으로 구성된다.

try {

  //코드

} catch (err) {

  //에러 핸들링

}

|  |
| --- |
| **try…catch 동작 알고리즘**  1. 먼저 try{…}안의 코드가 실행된다.  2. 에러가 없다면, try 안의 마지막 줄까지 실행되고, catch 블록은 패스  3. 에러가 있다면, try안 코드의 실행이 중단, catch(err) 블록으로 제어 흐름이 넘어간다. 변수 err(아무 이름으로든 사용 가능)는 무슨일이 일어났는지에 대한 설명이 담긴 에러 객체를 포함한다. |

**16. 비동기**

비동기란 동시에 일어나지 않는다는 의미이다.

비동기 방식은, 필요에 따라서 꼭 순서대로 실행되지 않아도 된다.

promise라는 문법을 통해 비동기를 배워보자.

**-promise**

프라미스(promise)는 ‘제작 코드’와 ‘소비 코드’를 연결해 주는 특별한 자바스크립트 객체이다.

|  |
| --- |
| **제작 코드**  원격에서 스크립트를 불러오는 것과 같은 시간이 걸리는 일을한다.  **소비 코드**  ’제작 코드’의 결과를 기다렸다가 이를 소비합니다.  ’프라미스’는 시간이 얼마나 걸리든 상관없이 약속한 결과를 만들어 내는 ‘제작 코드’가 준비되었을 떄, 모든 소비 코드가 결과를 사용할 수 있도록 해준다. |

promise객체는 아래와 같은 문법으로 만들 수 있다.

let promise = new Promise((res, rej) => {

  // executor (제작 코드)

});

new Promise에 전달되는 함수는 executor(실행자, 실행 함수)라고 부른다.

res(value) – 일이 성공적으로 끝난 경우 그 결과를 나타내는 value와 함께 호출

rej(error) – 에러 발생 시 에러 객체를 나타내는 error와 함께 호출

프라미스는 성공 또는 실패만 한다.

executor는 res나 rej중 하나를 반드시 호출해야 한다.

이때 변경된 상태는 더 이상 변하지 않는다.

처리가 끝난 프라미스에 res와 rej를 호출하면 무시된다.

let promise = new Promise((res, rej) => {

  res('완료');

  rej("error"); // 무시

});

executor에 의해 처리가 끝난 일은 결과 혹은 에러만 가질 수 있다.

**-소비자: then, catch, finally**

프라미스 객체는 executor(‘제작 코드’)와 결과나 에러를 받을 소비 함수를 이어주는 역할을 한다. 소비함수는 .then, .catch, .finally 메서드를 사용해 등록된다.

**-then**

.then은 프라미스에서 가장 중요하고 기본이 되는 메서드입니다.

promise.then((res) => {

  console.log(res); //완료

});

.res에는 실행 결과가 들어있다.

**-catch**

.catch는 에러가 발생한 경우를 다룬다.

promise.catch((err) => {

  console.log(err); // 오류 발생!

});

err에는 에러가 들어있다.

**-finally**

.finally는 프라미스가 처리되면 항상 실행된다.

promise.catch((err) => {

  console.log(err); // 오류 발생!

});

finally에는 인수가 없다.

**-async…await**

async와 await라는 특별한 문법을 사용하면 프라미스를 좀 더 편하게 사용할 수 있다.

**-async**

async는 function 앞에 위치한다.

async function func() {

  return 1;

}

function 앞에 async를 붙이면 해당 함수는 항상 프라미스를 반환한다.

async function func() {

  return 1;

}

func().then(console.log) // 1

>>함수를 호출하면 result가 1인 프라미스가 반환된다.

async function func() {

  return Promise.resolve(1);

}

func().then(console.log) // 1

명시적으로 프라미스를 반환하는것도 가능하다.

async가 제공하는 기능은 이뿐만이 아니다.

await는 async함수 안에서만 동작한다.

**-await**

자바스크립트는 await키워드를 만나면 프라미스가 처리될 때까지 기다린다.

다음장에서 예시를 함께 봅시다.

async function func() {

  let promise = new Promise((res, rej) => {

    setTimeout(() => res('완료!'), 1000)

  });

  let result = await promise; // (\*)

  console.log(result);

}

func();

함수를 호출하고, 함수 본문이 실행되는 도중, (\*)로 표시한 줄에서 실행이 잠시 ‘중단’되었다가 프라미스가 처리되면 실행이 재개된다.

|  |
| --- |
| **\*setTimeout(()=> [콜백함수], 시간)**  setTimeout은 특정시간 후에 함수를 실행시킨다.  시간이 1000이라면 1초를 의미한다. |

await는 promise.then보다 좀 더 멋지게 프라미스의 result값을 얻을 수 있도록 해주는 문법이다.

**17. Node.js**

Chrome v8 javascript 엔진으로 빌드된 javascript 런타임이다.

한마디로 javascript를 실행시키는 도구이다.

**18. npm**

Javascript로 크롤링을 진행하기 위해서는 몇가지 패키지가 필요하다.

패키지를 사용하기 위해서 npm(node package manager)을 배워보자.

npm이란

npm은 node에서 사용할 수 있는 패키지 보관소이다.

Chrome - npm검색(<https://www.npmjs.com/>)

>> 1. 터미널에 npm init -y를 입력하면 package.json이라는 파일이 생긴다.

2. 터미널에 npm i [패키지이름]으로 여러가지 패키지들을 설치해보자.

패키지에 대한 기초 설명은 npm사이트(<https://www.npmjs.com/>)에서 찾을 수 있다.

>> 1. 사이트에 접속한다.

2. 검색창에 원하는 패키지의 이름을 검색한다.

|  |  |
| --- | --- |
| Chapter 3 | puppeteer 소개 |

**19. puppeteer**

puppeteer은 chrome 또는 chromium을 제어하는 API를 제공하는 노드 라이브러리이다.

페이지의 스크린샷이나 PDF를 생성하거나 단일 페이지를 크롤링할 수 있다.

설치 npm I puppeteer

모듈 가져오기 const puppeteer = require(‘puppeteer’);

사용해보기

가. 스크린샷 저장하기

const puppeteer = require('puppeteer');

(async () => {

  const browser = await puppeteer.launch();

  const page = await browser.newPage();

  await page.goto('https://sunrint.sen.hs.kr/');

  await page.screenshot({path: 'example.png'});

  await browser.close();

})();

|  |
| --- |
| 스크립트 실행  node [파일명]  >>node index.js |

|  |
| --- |
| 실행 결과  [example.png] |
| 코드 설명  -1행 : 모듈 가져오기  -3~10행 : ( ) 안의 함수 실행  -3행 : async를 이용한 비동기 처리, 익명함수의 시작  -4행 : puppeteer.launch() chromium을 시작한다.launch()에 {headless: false}라는 옵션을 추가하여 브라우저를 시각화할 수 있다.  >> puppeteer.launch({ headless: false })  -5행 : browser.newPage() 새 페이지를 만든다.  -6행 : page.goto([URL]) URL은 탐색할 페이지이다.  -7행 : page.screenshot({path: [경로]}) {path: [경로]}라는 옵션을 추가하여 스크린샷을 저장할 위치를 설정할 수 있다.  -9행 : browser.close() 브라우저를 닫는다. |

나. 크롤링 시작해보기

급식을 크롤링하는 코드이다.

const puppeteer = require('puppeteer');

(async () => {

  const browser = await puppeteer.launch({ headless: false });

  try {

    const page = await browser.newPage();

    await page.goto('https://sunrint.sen.hs.kr/');

    const text = await page.evaluate(() => {

      const score = document.querySelector('.menu');

      if (score) {

        return score.textContent;

      }

    });

    console.log(text.trim());

  } catch (err){

    console.log(err);

  }

  await browser.close();

})()

|  |
| --- |
| 스크립트 실행  node [파일명]  >>node index.js |

|  |
| --- |
| 실행 결과  **보리밥**  **감자옹심이수제비국**  **파채간장불고기**  **칠리만두강정**  **미나리숙주나물**  **배추겉절이** |

|  |
| --- |
| 코드 설명  -1행 : 모듈 가져오기  -3~19행 : ( ) 안의 함수 실행  -3행 : async를 이용한 비동기 처리, 익명함수의 시작  -4행 : puppeteer.launch() chromium을 시작한다.launch()에 {headless: false}라는 옵션을 추가하여 브라우저를 시각화할 수 있다.  >> puppeteer.launch({ headless: false })  -5~17행 : try ~ catch를 이용한 에러 핸들링  -6행 : browser.newPage() 새 페이지를 만든다.  -7행 : page.goto([URL]) URL은 탐색할 페이지이다.  -8행 : page.evaluate() DOM을 사용할 수 있게한다.  -9행 : document.querySelector([CSS 선택자]) 페이지에서 선택자와 일치하는 첫 번째 요소를 반환한다.  \*일치하는 요소가 없으면 null을 반환한다.  -10~12행 : 만약 조건이 참이라면 코드를 실행한다.  -11행 : 8행에서 시작한 함수의 값(score.textContent)을 반환하여 text에 담는다.  -14행 : console.log([값]) 값을 출력한다. trim([값]) 문자열 앞 뒤의 해당 값을 제거한다. 인자를 비워둘 시 공백이 제거된다.  -15행 : 에러가 발생할 시, 프로그램을 바로 종료하지 않고 catch 뒤 중괄호 사이의 코드를 실행한다.  -16행 : 오류를 출력한다.  -18행 : browser.close() 브라우저를 닫는다. |

|  |
| --- |
| 문서 객체 모델(DOM, Document Object Model)  HTML 문서에 접근하기 위한 일종의 인터페이스이다.  문서 내의 모든 요소를 정의하고, 각각의 요소에 접근하는 방법을 제공한다.  >> document.querySelector() |

|  |  |  |
| --- | --- | --- |
| CSS 선택자  지정하려는 요소를 선택하는 데 사용되는 패턴이다.  기초적인 것들을 배워보자.  .[클래스명] 클래스명이 일치하는 모든 요소를 선택한다.  >> .sunrin 이라면 클래스명이 sunrin인 모든 요소를 선택한다.  .[클래스명1].[클래스명2] 클래스명1과 클래스명2가 모두 일치하는 모든 요소를 선택한다.  >> .sunrin.AnA 라면 클래스명이 sunrin과 AnA인 모든 요소를 선택한다.  .[클래스명1] .[클래스명2] 클래스명1을 가진 요소의 자손인 클래스명2를 가진 모든 요소를 선택한다.  >> .sunrin .AnA 라면 클래스명 sunrin의 자손 중 클래스명 AnA를 가진 모든 요소를 선택한다.  #[아이디명] 아이디명이 일치하는 요소를 선택한다.  >> #sunrin 이라면 아이디명이 sunrin인 요소를 선택한다  \* 모든 요소를 선택한다.  [태그] 해당 태그인 모든 요소를 선택한다.  >> p p태그인 모든 요소를 선택한다.  응용  [태그].[클래스명] 해당 태그중 클래스명이 일치하는 모든 요소를 선택한다.  >> p.sunrin p태그이고 클래스명이 sunrin인 모든 요소를 선택한다.  [태그1],[태그2] 태그1인 모든 요소와 태그2인 모든 요소를 선택한다.  >> div,p div태그인 모든 요소와 p태그인 모든 요소를 선택한다.  [태그1] [태그2] 태그1내의 태그2인 모든 요소를 선택한다.  >> div p div태그내의 p태그인 모든 요소를 선택한다.  [태그1]>[태그2] 부모가 태그1인 모든 태그2요소를 선택한다.  >> div>p div태그를 부모로 하는 모든 p태그 요소를 선택한다. | | |
| Chapter 4 | 프로젝트 |

**20. 추가 프로젝트 1: 리그오브레전드 전적분석**

원하는 유저의 최근 20게임의 전적을 가져와서 승률을 분석하는 프로젝트입니다.

\*2022/08/01 오후 3:03을 기준으로하는 프로젝트입니다. 사이트의 업데이트를 통하여 코드가 변경될 수 있습니다.

다음장에서 코드를 함께 봅시다

const puppeteer = require('puppeteer');

const crawler = async (name) => {

  try {

    const browser = await puppeteer.launch({ headless: false });

    const page = await browser.newPage();

    await page.setViewport({

      width: 1920,

      height: 1080,

    });

    await page.goto('https://poro.gg/summoner/kr/'+name);

    let result = [];

    while(result.length < 20) {

      const text = await page.evaluate(() => {

        const Feed = document.querySelector('div.flex.flex-row.items-center > strong > span').textContent;

        return Feed;

      });

      result.push(text);

      await page.evaluate(() => {

        let Feed = document.querySelector('div.flex.flex-row.items-center > strong > span');

        Feed.parentNode.removeChild(Feed);

        Feed = document.querySelector('div.flex.flex-row.items-center > strong > span');

        Feed.parentNode.removeChild(Feed);

      });

      await page.waitForTimeout(1000);

    }

    console.log(result);

    await browser.close();

  } catch (err) {

    console.error(err);

  }

}

crawler('더블은못날아');

|  |
| --- |
| 스크립트 실행  node [파일명]  >>node index.js |

|  |
| --- |
| 실행 결과  **[**  **'승리', '승리', '패배',**  **'승리', '승리', '승리',**  **'패배', '승리', '승리',**  **'패배', '패배', '승리',**  **'승리', '승리', '승리',**  **'승리', '승리', '패배',**  **'패배', '승리'**  **]**  **더블은못날아의 최근 20게임 승률은70%입니다.** |

다음장에서 코드 분석을 함께 봅시다

|  |
| --- |
| -1행 : 모듈 가져오기  -3행 : crawler에 함수저장  -3행 : async를 이용한 비동기 처리, 익명함수의 시작  -4~39행 : try ~ catch를 이용한 에러 핸들링  -5행 : puppeteer.launch() chromium을 시작한다.launch()에 {headless: false}라는 옵션을 추가하여 브라우저를 시각화할 수 있다.  >> puppeteer.launch({ headless: false })  -6행 : browser.newPage() 새 페이지를 만든다.  -7~10행 : page.setViewport() 페이지의 화면크기를 조정한다. width는 길이, height는 높이를 설정한다.  -11행 : page.goto([URL]) URL은 탐색할 페이지이다.  -13행 : 결과를 저장할 변수를 선언한다.  -14~26행 : 저장하고싶은 데이터의 개수를 설정한다. [배열명].length은 배열의 길이를 의미한다.  While([조건]) 해당 조건이 참이라면 반복한다.  -15~18행 : page.evaluate() DOM을 사용할 수 있게한다.  -16행 : document.querySelector([CSS 선택자]).textContent 페이지에서 선택자와 일치하는 첫 번째 요소의 텍스트를 반환한다. \*일치하는 요소가 없으면 null을 반환한다.  -17행 : 15행에서 시작한 함수의 값(Feed)을 반환하여 text에 담는다.  -19행 : result라는 이름을 가진 배열에 text를 넣는다.  -20~25행 : page.evaluate() DOM을 사용할 수 있게한다.  -21, 23행 : document.querySelector([CSS 선택자]) 페이지에서 선택자와 일치하는 첫 번째 요소를 반환한다. \*일치하는 요소가 없으면 null을 반환한다.  -22, 24행 : parentNode 부모노드를 지칭한다. removeChild() 자식노드를 삭제한다.  >> Feed.parentNode.removeChild(Feed) Feed의 부모노드로 이동하여 Feed라는 자식을 삭제한다.  >> Feed를 삭제하는 방법이다.  -27행 : result를 출력한다.  >> result에는 **[**  **'승리', '승리', '패배',**  **'승리', '승리', '승리',**  **'패배', '승리', '승리',**  **'패배', '패배', '승리',**  **'승리', '승리', '승리',**  **'승리', '승리', '패배',**  **'패배', '승리'**  **]**  라는 값이 들어있다.  -28행 : 이긴경기를 저장하기위해 변수를 선언한다.  -29~33행 : [배열].forEach() result라는 이름의 배열의 요소 각각에 대해 실행한다. 첫번째 인자는 값 두번째 인자는 인덱스가 들어간다.  >> 예를들어 r에는 승리 i에는 0이 들어간다.  -30~32행 : if([조건]) 만약 조건이 참이라면 실행한다.  >>if (r == “승리”) 만약 r이 “승리”라면 실행한다.  -31행 : win이라는 변수의 값에 1을 더한다.  -34행 : 승률 계산 식을 만들어 변수에 저장한다.  >>코드에는 20으로 표기했지만 result.length로 바꾸어도 좋을 것 같다.  -35행 : 원하는 값을 출력했다.  >> **더블은못날아의 최근 20게임 승률은70%입니다.** 라는 텍스트를 출력한다.  -36행 : browser.close() 브라우저를 닫는다.  -37행 : 에러가 발생할 시, 프로그램을 바로 종료하지 않고 catch 뒤 중괄호 사이의 코드를 실행한다.  -38행 : 오류를 출력한다.  -44행 : 함수를 실행한다. ‘더블은못날아’라는 닉네임의 유저의 승률을 분석할 것이다.  >> ‘더블은못날아’라는 인자는 11행에서 사용되었다. |

**21. 추가 프로젝트 2: 검색어 책 추천 받기**

생각나는 검색어를 입력해보세요! 책을 추천해줄거예요!

\*2022/08/02 오후 3:27을 기준으로하는 프로젝트입니다. 사이트의 업데이트를 통하여 코드가 변경될 수 있습니다.

다음장에서 코드를 함께 봅시다

const puppeteer = require('puppeteer');

const crawler = async (name) => {

  try {

    const browser = await puppeteer.launch({ headless: false });

    const page = await browser.newPage();

    await page.setViewport({

      width: 1920,

      height: 1080,

    });

    await page.goto('https://www.aladin.co.kr/search/wsearchresult.aspx?SearchTarget=All&SearchWord='+name+'&x=0&y=0');

    let result = [];

    while(result.length < 100) {

      try {

        const text = await page.evaluate(() => {

          const Feed = document.querySelector('.ss\_book\_list .bo3').textContent;

          return Feed;

        });

        result.push(text);

        await page.evaluate(() => {

          const Feed = document.querySelector('.ss\_book\_list .bo3');

          Feed.parentNode.removeChild(Feed);

        });

      } catch (err) {

        break;

      }

    }

    console.log(result);

    await page.goto('http://reading.ssem.or.kr/r/newReading/search/schoolSearchForm.jsp');

    await page.waitForTimeout(1000);

    await page.type('#schoolSearch', '선린인터넷고등학교');

    await page.waitForTimeout(1000);

    await page.click('.material-icons');

    await page.waitForTimeout(1000);

    await page.click('.school\_name>a');

    await page.waitForSelector('#searchCon1');

    let i = 0

    while(i < result.length) {

      await page.waitForSelector('#searchCon1');

      await page.evaluate(() => {

        document.querySelector('#searchCon1').value = "";

      });

      await page.type('#searchCon1', result[i]);

      await page.waitForTimeout(1000);

      await page.click('.book\_search > div.btn\_search\_box > button');

      await page.waitForSelector('.bd\_list.bd\_book\_list.school\_lib');

      let recommend = await page.evaluate(() => {

        if(!document.querySelector('.bd\_list.bd\_book\_list.school\_lib').textContent.includes('검색결과가 없습니다.')) {

          let title = document.querySelector('.bd\_list\_title').textContent;

          title = title.replaceAll('\t', '').replaceAll('\n', '');

          let number = document.querySelector('.bd\_list\_year').textContent;

          number = number.replaceAll('\t', '').replaceAll('\n', '');

          return {title, number};

        }

      });

      if (recommend != undefined) {

        console.log(recommend.title+'(이)라는 책이 도서관에 있어요! 청구기호는 '+recommend.number+'입니다');

        await browser.close();

        process.exit()

      }

      await page.waitForTimeout(1000);

      i++;

    }

    console.log('아쉽지만 책을 찾지못했어요');

    await browser.close();

  } catch (err) {

    console.error(err);

  }

}

crawler('프로그래밍');

|  |
| --- |
| 스크립트 실행  node [파일명]  >>node index.js |

|  |
| --- |
| 실행 결과  **[**  **'Do it! 조코딩의 프로그래밍 입문',**  **'윤성우의 열혈 C 프로그래밍',**  **'러스트 프로그래밍',**  **'윤성우의 열혈 C++ 프로그래밍',**  **'한 권으로 읽는 컴퓨터 구조와 프로그래밍',**  **'흥달쌤의 문풀 프로그래밍 언어',**  **'러스트 프로그래밍',**  **'C포자를 위한 본격 C 언어 프로그래밍',**  **'자바 ORM 표준 JPA 프로그래밍',**  **'솔리디티 프로그래밍 완벽 가이드',**  **'혼자 공부하는 첫 프로그래밍 with 파이썬',**  **'당신의 프로그래밍에 디버깅 더하기 : Visual C++ 디버깅 기초에서 고급까지',**  **'TCP/IP 소켓 프로그래밍',**  **'레트로의 유니티 게임 프로그래밍 에센스',**  **'코어 이더리움 프로그래밍',**  **'Do it! 자바 프로그래밍 입문',**  **'HTML5 웹 프로그래밍 입문',**  **'윤성우의 열혈 TCP/IP 소켓 프로그래밍',**  **'Do it! 안드로이드 앱 프로그래밍',**  **'C# 프로그래밍 : 프로그래밍 기초부터 객체 지향 핵심까지',**  **'타입스크립트 프로그래밍',**  **'Do it! 조코딩의 프로그래밍 입문',**  **'초보 웹 개발자를 위한 스프링5 프로그래밍 입문',**  **'파이썬 객체지향 프로그래밍 4/e'**  **]**  **(윤성우의)열혈 C++ 프로그래밍(이)라는 책이 도서관에 있어요! 청구기호는 청구기호 : 005.13 윤63ㅇ 입니다** |

다음장에서 코드 분석을 함께 봅시다

|  |
| --- |
| -1행 : 모듈 가져오기  -3행 : crawler에 함수저장  -3행 : async를 이용한 비동기 처리, 익명함수의 시작  -4~71행 : try ~ catch를 이용한 에러 핸들링  -5행 : puppeteer.launch() chromium을 시작한다.launch()에 {headless: false}라는 옵션을 추가하여 브라우저를 시각화할 수 있다.  >> puppeteer.launch({ headless: false })  -6행 : browser.newPage() 새 페이지를 만든다.  -7~10행 : page.setViewport() 페이지의 화면크기를 조정한다. width는 길이, height는 높이를 설정한다.  -11행 : page.goto([URL]) URL은 탐색할 페이지이다.  -13행 : 결과를 저장할 변수를 선언한다.  -14~28행 : 저장하고싶은 데이터의 개수를 설정한다. [배열명].length은 배열의 길이를 의미한다.  While([조건]) 해당 조건이 참이라면 반복한다.  -15~27행 : try ~ catch를 이용한 에러 핸들링  -16~19행 : page.evaluate() DOM을 사용할 수 있게한다.  -17행 : document.querySelector([CSS 선택자]).textContent 페이지에서 선택자와 일치하는 첫 번째 요소의 텍스트를 반환한다. \*일치하는 요소가 없으면 null을 반환한다.  -18행 : 16행에서 시작한 함수의 값(Feed)을 반환하여 text에 담는다.  -20행 : result라는 이름을 가진 배열에 text를 넣는다.  -21~24행 : page.evaluate() DOM을 사용할 수 있게한다.  -22행 : document.querySelector([CSS 선택자])  페이지에서 선택자와 일치하는 첫 번째 요소를 반환한다.  \*일치하는 요소가 없으면 null을 반환한다.  -23행 : parentNode 부모노드를 지칭한다. removeChild() 자식노드를 삭제한다.  -25~27행 : 에러가 발생하면 반복문을 종료한다.  -30행 : result를 출력한다.  >> result에는 **[**  **'Do it! 조코딩의 프로그래밍 입문',**  **'윤성우의 열혈 C 프로그래밍',**  **'러스트 프로그래밍',**  **'윤성우의 열혈 C++ 프로그래밍',**  **'한 권으로 읽는 컴퓨터 구조와 프로그래밍',**  **'흥달쌤의 문풀 프로그래밍 언어',**  **'러스트 프로그래밍',**  **'C포자를 위한 본격 C 언어 프로그래밍',**  **'자바 ORM 표준 JPA 프로그래밍',**  **'솔리디티 프로그래밍 완벽 가이드',**  **'혼자 공부하는 첫 프로그래밍 with 파이썬',**  **'당신의 프로그래밍에 디버깅 더하기 : Visual C++ 디버깅 기초에서 고급까지',**  **'TCP/IP 소켓 프로그래밍',**  **'레트로의 유니티 게임 프로그래밍 에센스',**  **'코어 이더리움 프로그래밍',**  **'Do it! 자바 프로그래밍 입문',**  **'HTML5 웹 프로그래밍 입문',**  **'윤성우의 열혈 TCP/IP 소켓 프로그래밍',**  **'Do it! 안드로이드 앱 프로그래밍',**  **'C# 프로그래밍 : 프로그래밍 기초부터 객체 지향 핵심까지',**  **'타입스크립트 프로그래밍',**  **'Do it! 조코딩의 프로그래밍 입문',**  **'초보 웹 개발자를 위한 스프링5 프로그래밍 입문',**  **'파이썬 객체지향 프로그래밍 4/e'**  **]**  라는 값이 들어있다.  -31행 : page.goto([URL]) URL은 탐색할 페이지이다.  -32, 34, 36행 : page.waitForTimeout([시간])  시간동안 페이지를 멈추고 기다린다.  -33행 : page.type([CSS 선택자], [문자열])  페이지에서 선택자에 문자열을 입력한다.  -35, 37행 : page.click([CSS 선택자])  페이지에서 선택자를 클릭한다.  -39행 : page.waitForSelector([CSS 선택자])  CSS 선택자가 탐색될 때까지 페이지를 멈추고 기다린다.  -40~66행 : result라는 이름의 배열의 길이만큼 반복문을 실행한다.  -42행 : page.waitForSelector([CSS 선택자])  CSS 선택자가 탐색될 때까지 페이지를 멈추고 기다린다.  -43~45행 : page.evaluate() DOM을 사용할 수 있게한다.  -44행 : document.querySelector([CSS 선택자]).value  페이지에서 선택자와 일치하는 첫 번째 요소의 value를 반환한다.  -46행 : page.type([CSS 선택자], [문자열])  페이지에서 선택자에 문자열을 입력한다.  -47행 : page.waitForTimeout([시간])  시간동안 페이지를 멈추고 기다린다.  -48행 : page.click([CSS 선택자])  페이지에서 선택자를 클릭한다.  -49행 : page.waitForSelector([CSS 선택자])  CSS 선택자가 탐색될 때까지 페이지를 멈추고 기다린다.  -50~58행 : page.evaluate() DOM을 사용할 수 있게한다.  -51행 : document.querySelector([CSS 선택자]).textContent가 ‘검색결과가 없습니다.’가 아니라면 조건문을 실행한다.  \* ![변수] true -> false, false -> true  -52, 54행 : document.querySelector([CSS 선택자]).textContent 페이지에서 선택자와 일치하는 첫 번째 요소의 텍스트를 반환한다. \*일치하는 요소가 없으면 null을 반환한다.  -53, 55행 : [변수].replaceAll([value1], [value2]) 변수에 들어있는 value1을 모두 value2로 바꾼다.  -56행 : 50행에서 시작한 함수의 값({title, number})을 반환하여 recommend에 담는다.  -59행 : recommend 가 undefined가 아니라면 조건문을 실행한다.  -60행 : 문자열을 출력한다.  문자열에는 **(윤성우의)열혈 C++ 프로그래밍(이)라는 책이 도서관에 있어요! 청구기호는 청구기호 : 005.13 윤63ㅇ 입니다**  라는 값이 들어있다.  -61행 : browser.close() 브라우저를 닫는다.  -62행 : process.exit() 스크립트를 종료시킨다.  -64행 : page.waitForTimeout([시간])  시간동안 페이지를 멈추고 기다린다.  -67행 : 문자열을 출력한다. 해당 코드는 반복문이 종료할 때까지 원하는 값(추천할 책)을 찾지 못한다면 출력한다.  -68행 : browser.close() 브라우저를 닫는다.  74행 : 함수를 실행한다. ‘프로그래밍’라는 검색어로 책을 추천한다. |

|  |  |
| --- | --- |
| Chapter 5 | 오류 처리 |

Npm 설치시 오류

1. Npm ERR! Code ENOSELF

설치하려는 라이브러리와 프로젝트의 이름이 동일하기 때문이다.

[해결 방법]

1. 폴더명 변경
2. Package.json파일의 name변경

2. WARN npm does not support Node.js vXX.XX.X

버전이 안맞기 때문이다.

가. Nodejs 재설치

나. npm install -g npm@latest으로 npm 업데이트하기

코드 오류

1. UnhandledPromiseRejectionWarning: ReferenceError: XXXX is not defined

혹은 ReferenceError: XXXX is not defined

대부분은 오타이다.

2. net::ERR\_NAME\_NOT\_RESOLVED at XXXX

웹페이지에 접속할 수 없기 때문이다.

URL에 오타가 있는지 확인해라.