**클라이언트 & 서버**

클라이언트 역할: 서버에 자료를 요청하는 역할

서버의 역할: 클라이언트가 요청한 자료를 회신하는 역할

https://www.naver.com/

http 혹은 https : 프로토콜 => 통신규약

> 앞으로 웹 서버에서 클라이언트가 특정 데이터를 요청한다면,

서로 약속한 데이터 형식을 가지고 요청할 때만 값을 출력해주자!

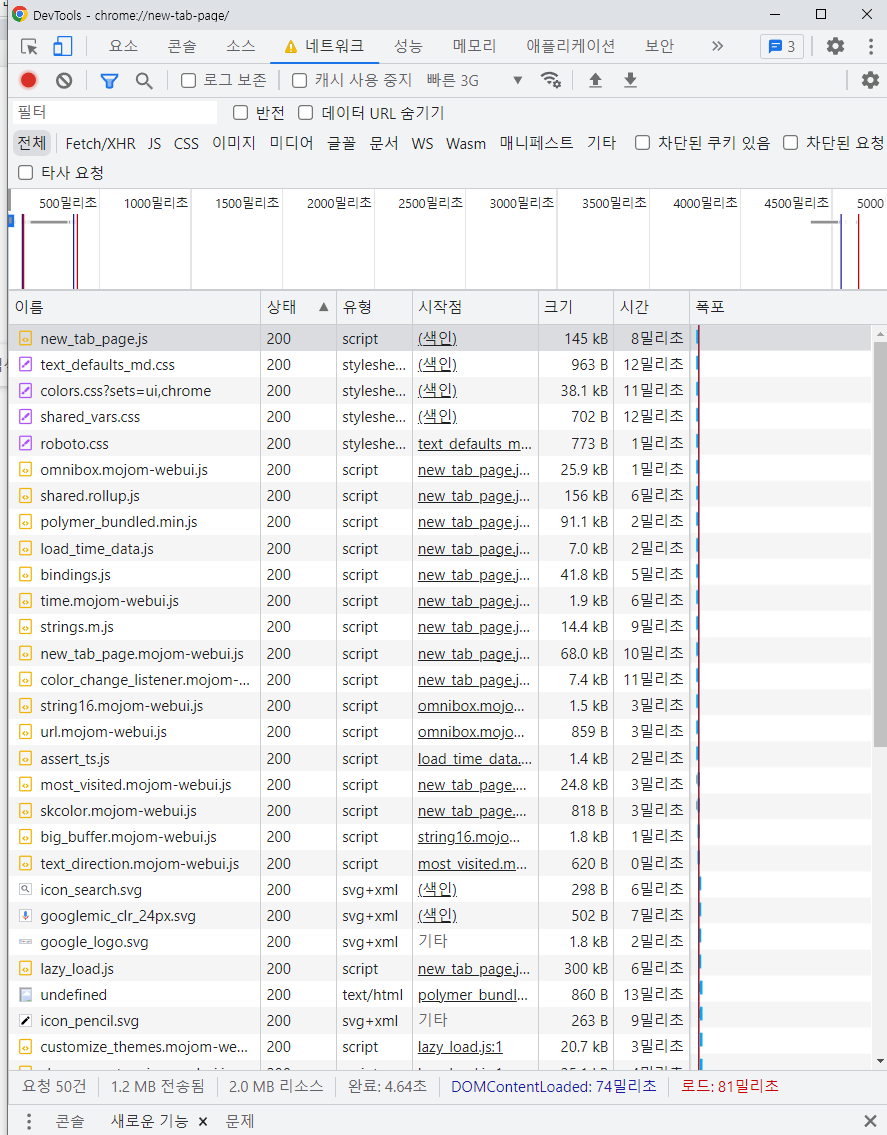
HTML: 하이퍼 텍스트 마크업 랭귀지

HTTP : 하이퍼 텍스트 트랜스퍼 프로토콜

HTTPS : 하이퍼 텍스트 트랜스퍼 프로토콜 (보안 서비스가 구축되어있다.)

거의 모든 홈페이지는 https로 되어있는데, http로 되어있는게 있는데, 파일질라를 활용해서 무료 호스팅 서비스에서 도메인 주소!! 유심히 보면 http로 되어있다.

구글에서 f12 누르고 네트워크를 눌러서 보기



<https://www.google.com/search?q=%EC%9E%90%EB%B0%94%EC%8A%A4%ED%81%AC%EB%A6%BD%ED%8A%B8&source=lmns&bih=1178&biw=925&prmd=ivbn&hl=ko&sa=X&ved=2ahUKEwjvufGH-8b-AhX-plYBHRYoDRwQ_AUoAHoECAAQBA>

get방식 : 서버에 자료를 요청할 때, 사이트 주소 뒤에 자료명을 붙여서 보내는 방식

* 요청하고자하는 자료가 공공 데이터이거나 비보안 문서 혹은 정보검색용데이터!

POST 방식 : 서버에 자료를 요청할 때, 요청 문자 혹은 내용이 도메인 주소에 나타나지 않도록 body 본문 안에 담아서 전송하는 방식

* 예를 들어 로그인 이런 거…..!

404 Page Not Found

2\*\* : 2로 시작한다는 건 성공 메시지!

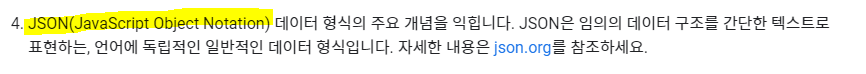
4\*\* : 4로 시작한다는 건 서버로부터 데이터를 받아오는 데 실패했다는 메시지

* 문서를 찾을 수 없을 때 (\* 도메인 오타)
* 401 : 문서에 접속할 권한이 없을 때 (폐쇄몰, vip쇼핑몰 이런 거!)

5\*\* : 서버 자체에 오류가 났을 때

* 500 : 서버다운 (\* 접속량이 급작스럽게 늘었을 때)
* 503 : 서버 자체적으로 데이터를 차단 시킴 (\* 의도적인 차단)

**자바스크립트를 활용해서 외부사이트 데이터를 가져 올 계획 !!!!!!!!!!!!**



**JSON & XML**

XML 데이터 자료 형식은 JSON

JSON = Javascript Object Notation : 자바스크립트 객체 표기법

제이슨은 자바, C, 파이썬, C++ 등등 에서 모두 사용할 수 있다.

1. 용량이 XML 대비 상대적으로 작다
2. 사용할 수 있는 범위가 넓다. (자바, C, 파이썬, C++등등)

\* 사용법 > 키값에 “” 을 붙여줘야한다

{

“name” : “홍길동”

“major” : “science”

“grade” : 2

}

\* 객체와 제이슨의 차이점은?

* 키 값을 표현할 때 “” 여부
* 프로퍼티 & 메서드(함수) vs 제이슨은 메서드 및 함수를 넣을 수 없다

\* 유의사항

- 제이슨에서 키 값을 정의할 때, 무조건 큰따옴표!!!! 작은따옴표 절대 안됨

- 키 값을 정의할 때 원칙적으로 띄어쓰기 가능하나 실무에서는 아무도 띄어쓰기 하지 않는다. 언더바사용이 정설!

- 제이슨에서 value값으로 문자가 온다면, 당연히 큰따옴표

- 제이슨에서 value값에는 또다른 문자열 혹은 배열을 가져올 수 있다.

{

“name” : “홍길동”,

“major” : “science”,

“grade” : 2,

“course” : [“html”, “css”, “python”], // 배열 객체를 넣을 수 있다

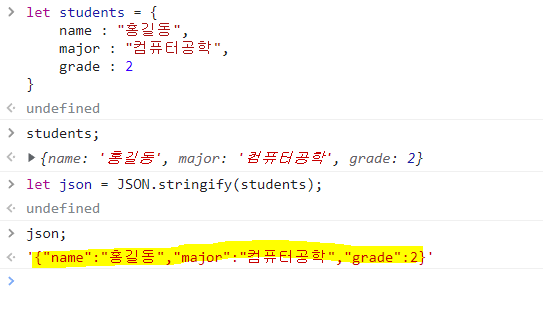
“basic” : {

// 이런식으로도 할 수 있음!  
}

}

**그럼 객체에서 json으로 바꿔보자!!!**

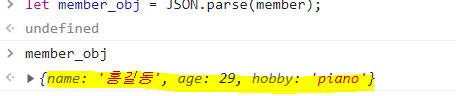
**JSON.stringify() : 객체 => JSON**



반대로, JSON에서 객체로 바꿔보자

**JSON.parse() : JSON => 객체**





**AJAX**

* **비동기 자바스크립트와 엑스엠엘**
* **웹페이지가 새롭게 로딩되지 않고도 필요한 웹 문서를 가져와서 보여주는 것을 AJAX라고 한다. > 자바스크립트 비동기데이터 처리 방식!**

비동기 <-> 동기 : 서버에 자료를 요청하고, 요청한 자료가 정상적으로 수신이 되어야지만, 페이지 전환 및 이동!

웹클라이언트 & 웹 브라우저 & 웹 서버

XMLHttpRequest객체 = XHR객체라고 부른다!

* 프로퍼티 존재
* 1. redyState: 객체에서 서버로 자료 요청을 했는지 자료 도착 여부를 알 수 있는 프로퍼티이다.

>> 0 일 때 > 아직 서버한테 자료요청을 안한 상태

>> 1 일 때 > 서버한테 자료를 성공적으로 요청

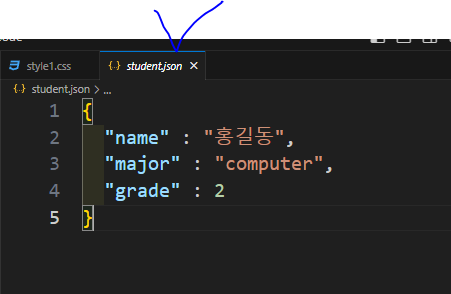
>> 2 일 때 > 서버 요청에 대한 응답이 온 상태

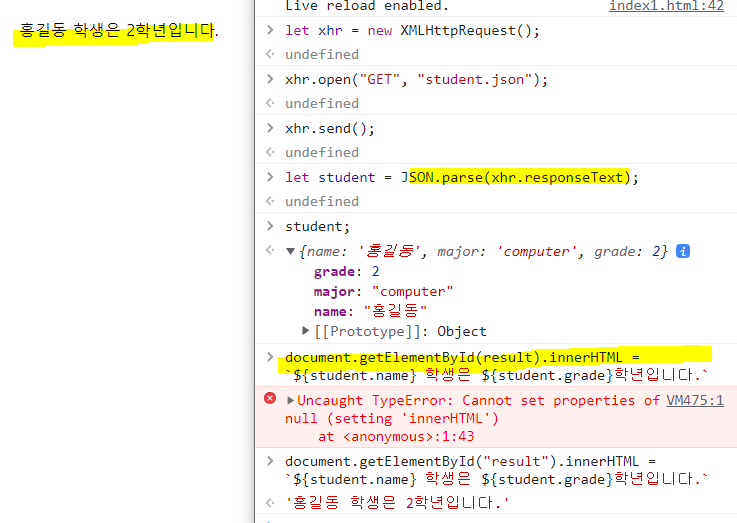
>> 3 일 때 > 서버에서 자료가 로딩중인 상태

>> 4. 일 때 > 자료 전송이 끝나고 클라이언트가 자료를 사용할 수 있는 상태

* Status : 서버와의 통신 상태
* responseText : 서버에 요청한 응답이 문자열로 저장된 상태
* 메서드 존재
* 1. Open() : 어떤 방식으로 사용할지, 어떤 자료가 필요한지, 비동기 처리 여부를 정의하는 메서드이다.
* 2. Sned() : 앞에서 설정한 open() 메서드를 서버로 보내는 메서드이다.

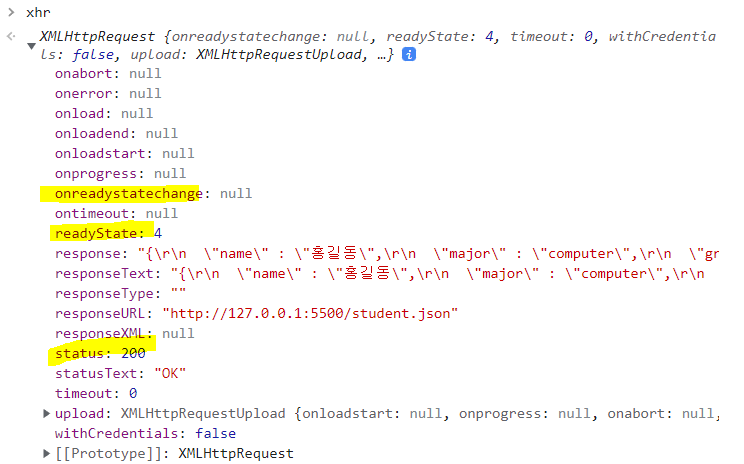
vs코드에서 제이슨 파일을 만들어 제이슨 생성 후 콘솔창에서 제이슨을 객체로 바꿔준 후 #result 공간에 출력해 보기!



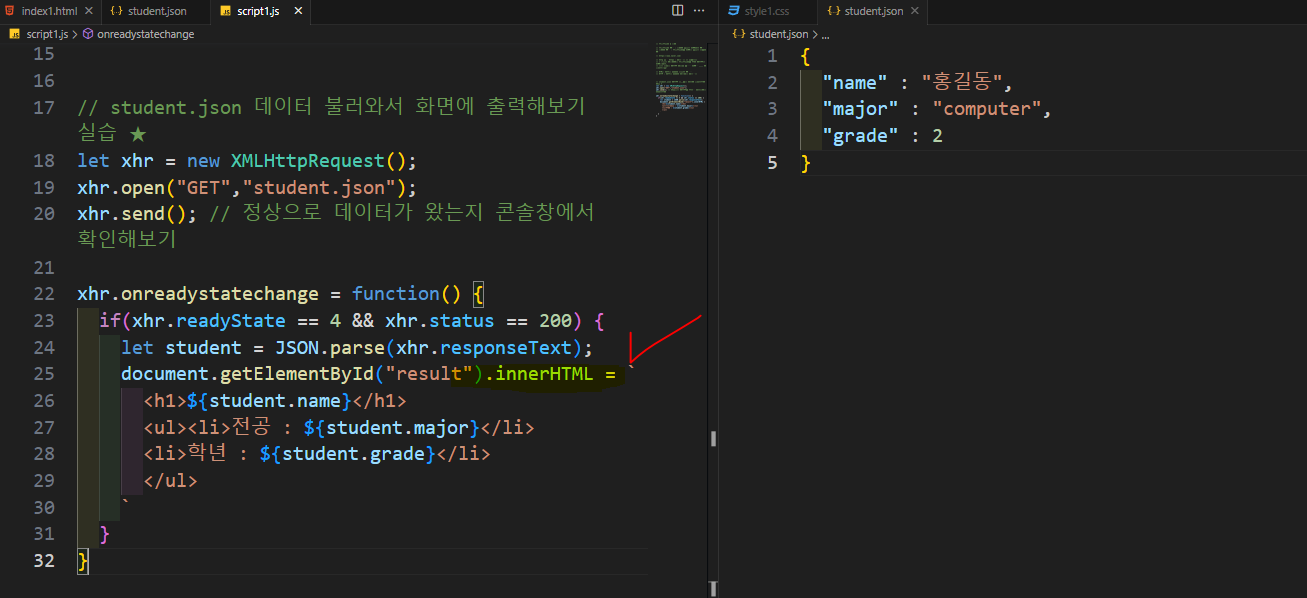


위에서 open이랑 send는 같이 다니는 친구들이다

자료를 보내는 메서드는 2가지가있다. 첫번째는 GET / 두번째는 POST



Onreadystatechange = readState가 정상적으로 돌아가는지 확인하는 것임!



student.json 데이터 불러와서 화면에 출력해보기 실습 ★

[

  {

    "name" : "홍길동",

    "major" : "컴퓨터공학과",

    "grade" : 2

  },

  {

    "name" : "수지",

    "major" : "건축학과",

    "grade" : 4

  },

  {

    "name" : "아이유",

    "major" : "실용음악학과",

    "grade" : 3

  }

]

// 데이터가 여러개일 경우 undefined가 나오므로

// 별도의 함수를 떼어서 반복문을 돌려 실제로

// 데이터들 가운데서 한명 한명씩 ul태그를 활용해서

// 별도로 분리를 시키면 된다.

let xhr = new XMLHttpRequest();

xhr.open("GET","student.json");

xhr.send(); // 정상으로 데이터가 왔는지 콘솔창에서 확인해보기

xhr.onreadystatechange = function() {

  if(xhr.readyState == 4 && xhr.status == 200) {

    let students = JSON.parse(xhr.responseText);

    renderHTML(students);

    // document.getElementById("result").innerHTML = `

    //   <h1>${student.name}</h1>

    //   <ul><li>전공 : ${student.major}</li>

    //   <li>학년 : ${student.grade}</li>

    //   </ul>

    // `

  }

};

function renderHTML(students) {

  let output = "";

  for (let student of students) {

    output += `

    <h2>${student.name}</h2>

    <ul>

    <li>전공 : ${student.major}</li>

    <li>학년 : ${student.grade}</li>

    </ui>

    `

  }

  document.getElementById("result").innerHTML = output;

}

예외처리!!! = 오류 처리

: 프로그램에서 문제가 발생했을 때를 대비해서 작업해 놓은 코드이다.

Try문 / catch문 / throw문

- try문 : 실제 오류가 났을 때, 실행 할 명령문을 작성하는 곳

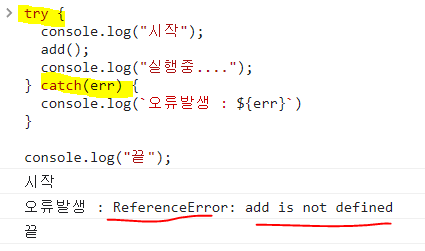
- catch문 : 발생한 오류의 이름 및 설명들이 들어 가 있는 곳

- throw : 개발자가 직접 오류 메시지를 정의할 수 있는 곳

- console.err() : 오류 메시지를 표기할 때 사용할 수 있다

**\* 사용자로부터 특정 숫자를 받아서 어떤 프로그램을 실행 시킨다면**

- 사용자가 숫자를 입력하지 않았을 때 오류.



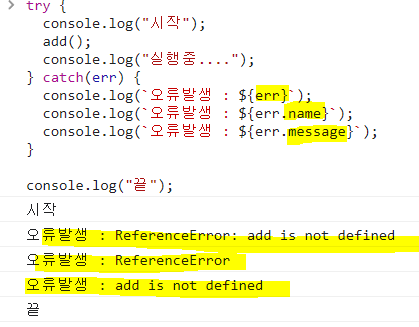
catch는 오류가 나는걸 찾으라는 함수!

발생이 되고 나서 넘어가서 마지막까지 출력할 건 출력할 수 있게 만들어 줌

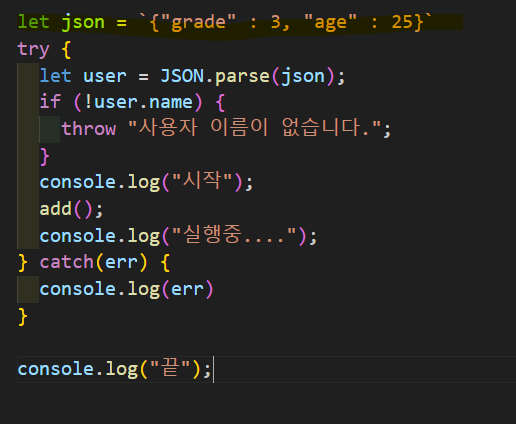
try이랑 catch는 항상 쌍으로 움직임!

예외처리를 처리할 수 있도록 도와준다.

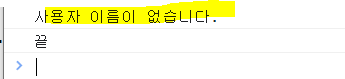
에러를 좀 더 강조하기 위해서 방법이 있다!



경고문구를 나만의 문구로 만들고 싶을 땐?

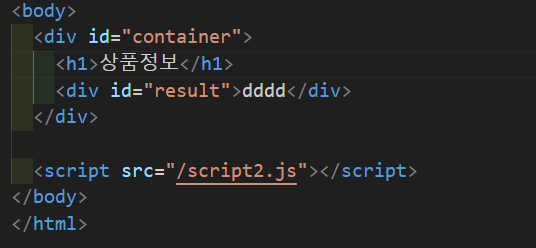


결과



실습실습

[HTML]



[JS]

const result = document.querySelector("#result");

const url = "https://reqres.in/api/product/10"

// 일단 json 데이터를 저장해둘 공간을 정의하자!

let xhr = new XMLHttpRequest();

xhr.open("GET", url, true);

// 위에 true는 비동기 방식으로 불러오는건지 확인하는 것

xhr.send();

xhr.onreadystatechange = function() {

  if (xhr.readyState == 4 && xhr.status == 200) {

    let product = JSON.parse(xhr.responseText);

    console.log(product);

    result.innerHTML = `

    <p>상품명 : ${product.data.name}</p>

    <p>컬러 : ${product.data.color}</p>

    <p>생산년도 : ${product.data.year}년도식</p>

    <p>생산년도 : ${product.support.url}년도식</p>

    `;

    // json 안에 json이 있다!

  }

}

url 잘 불러왔는지 확인하기!



자바스크립트로 인풋태그에 숫자가 10보다 작은 숫자만 아이디가 result인 공간에

출력될 수 있도록 해줘