

12. HTTP 통신과 JSON

HTTP/HTTPS

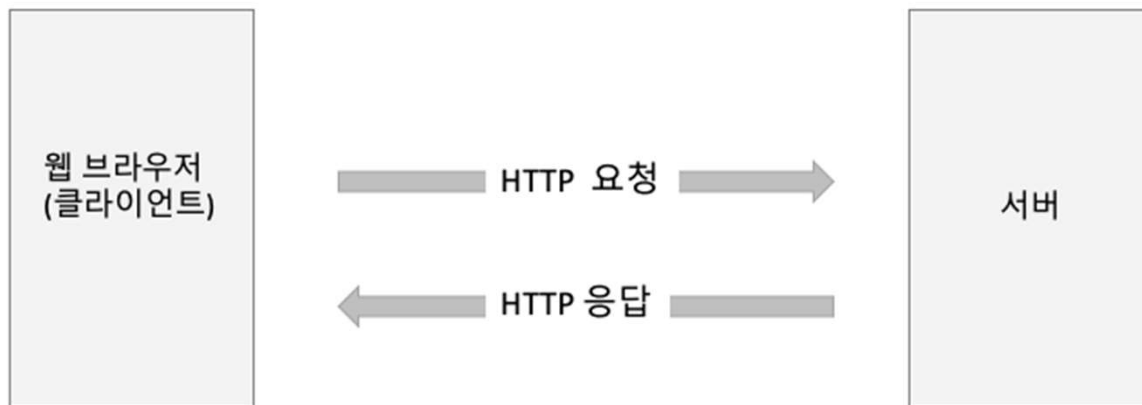
HTTP란

클라이언트와 서버 간에 자료를 주고받으려면 미리 약속된 규칙이 필요하다.

이것을 프로토콜^{protocol}이라고 하고, 웹에서는 HTTP ^{HyperText Transfer Protocol}라는 프로토콜을 사용한다.

클라이언트에서 서버로 자료 요청하는 것은 HTTP 요청^{HTTP request},

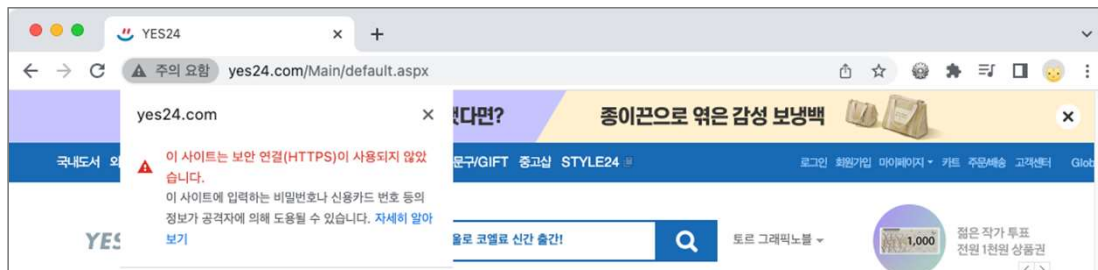
서버에서 응답해서 클라이언트로 자료를 보내는 것은 HTTP 응답^{HTTP response}이라고 한다.



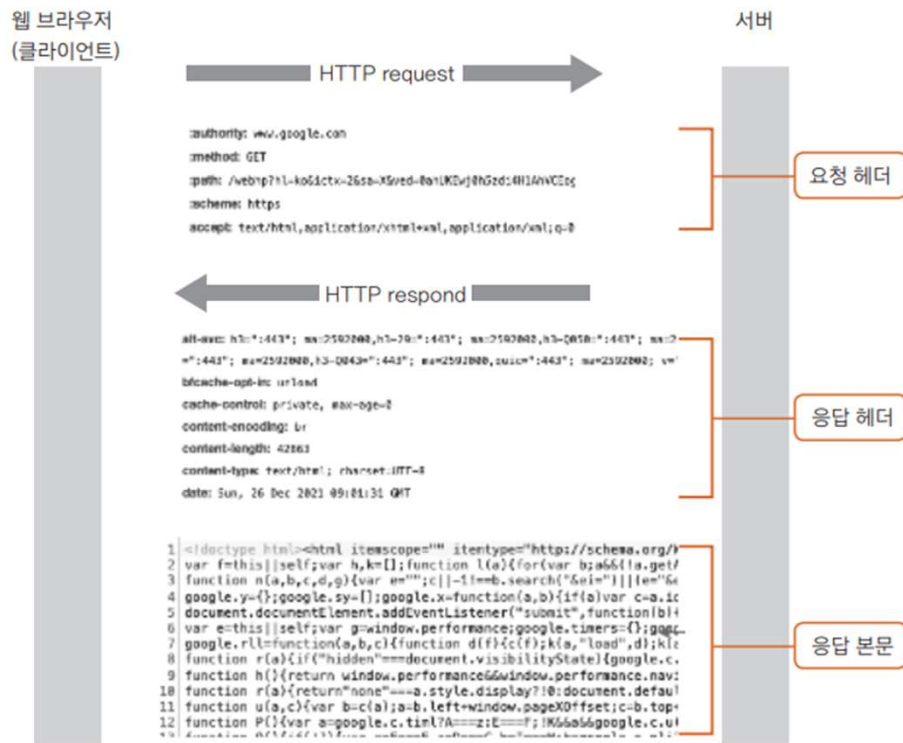
(예) 웹 브라우저에서 구글 검색 사이트를 찾아가려면 'https://www.google.com'을 입력한다
사이트 주소의 맨 앞에 붙는 http 또는 https가 현재 문서의 프로토콜



최근 크롬 웹 브라우저에서는 보안을 위해 https 프로토콜을 사용할 것을 권장하고 있고
아직 http를 사용하는 사이트에서는 '주의 요함'이라는 경고 메시지가 표시된다.



요청 헤더와 응답 헤더

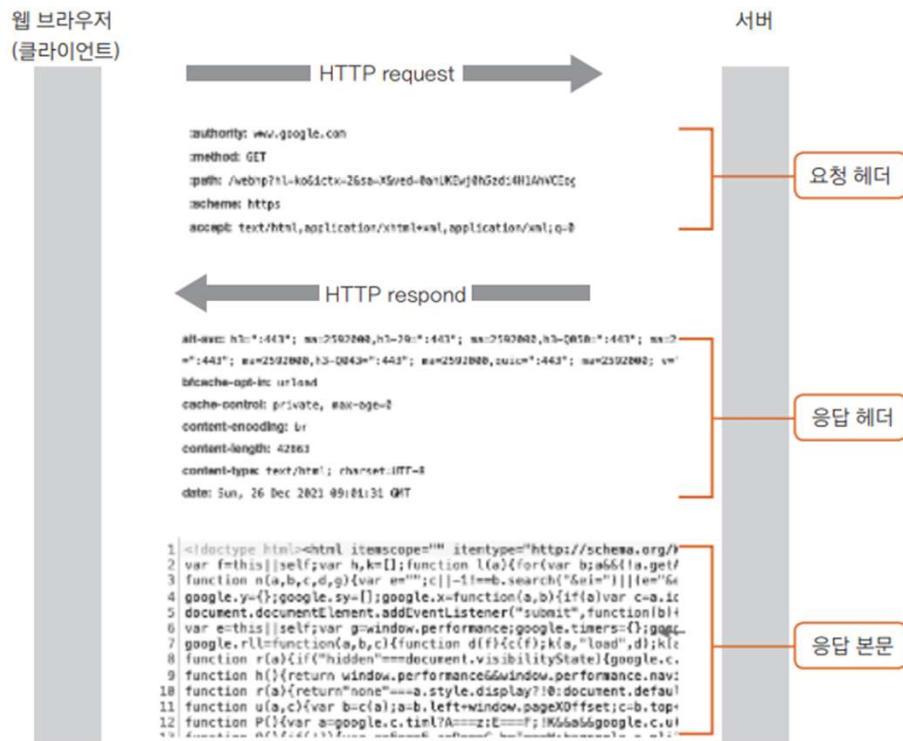


클라이언트에서 사이트 주소를 입력하고 [Enter]를 누를 때 사이트 주소뿐만 아니라 사용 중인 시스템 정보와 웹 브라우저 정보, 사용한 언어 등 다른 정보까지 함께 전송된다. → 서버로 요청할 때 보내는 헤더를 '요청 헤더(request header)'라고 한다.

서버에서 입력한 사이트를 찾아서 클라이언트로 보낼 때

- 응답 메시지를 보내는 시간, 메시지를 클라이언트에 어떻게 표시할지 등의 정보는 '응답 헤더(response header)'에 담기고
- 이미지나 텍스트 같은 실제 사이트 내용은 '응답 본문(response body)'에 담겨서 전달된다.

크롬에서 네트워크 확인하기



클라이언트에서 사이트 주소를 입력하고 [Enter]를 누를 때 사이트 주소뿐만 아니라 사용 중인 시스템 정보와 웹 브라우저 정보, 사용한 언어 등 다른 정보까지 함께 전송된다. → 서버로 요청할 때 보내는 헤더를 '요청 헤더(request header)'라고 한다.

서버에서 입력한 사이트를 찾아서 클라이언트로 보낼 때

- 응답 메시지를 보내는 시간, 메시지를 클라이언트에 어떻게 표시할지 등의 정보는 '응답 헤더(response header)'에 담기고
- 이미지나 텍스트 같은 실제 사이트 내용은 '응답 본문(response body)'에 담겨서 전달된다.

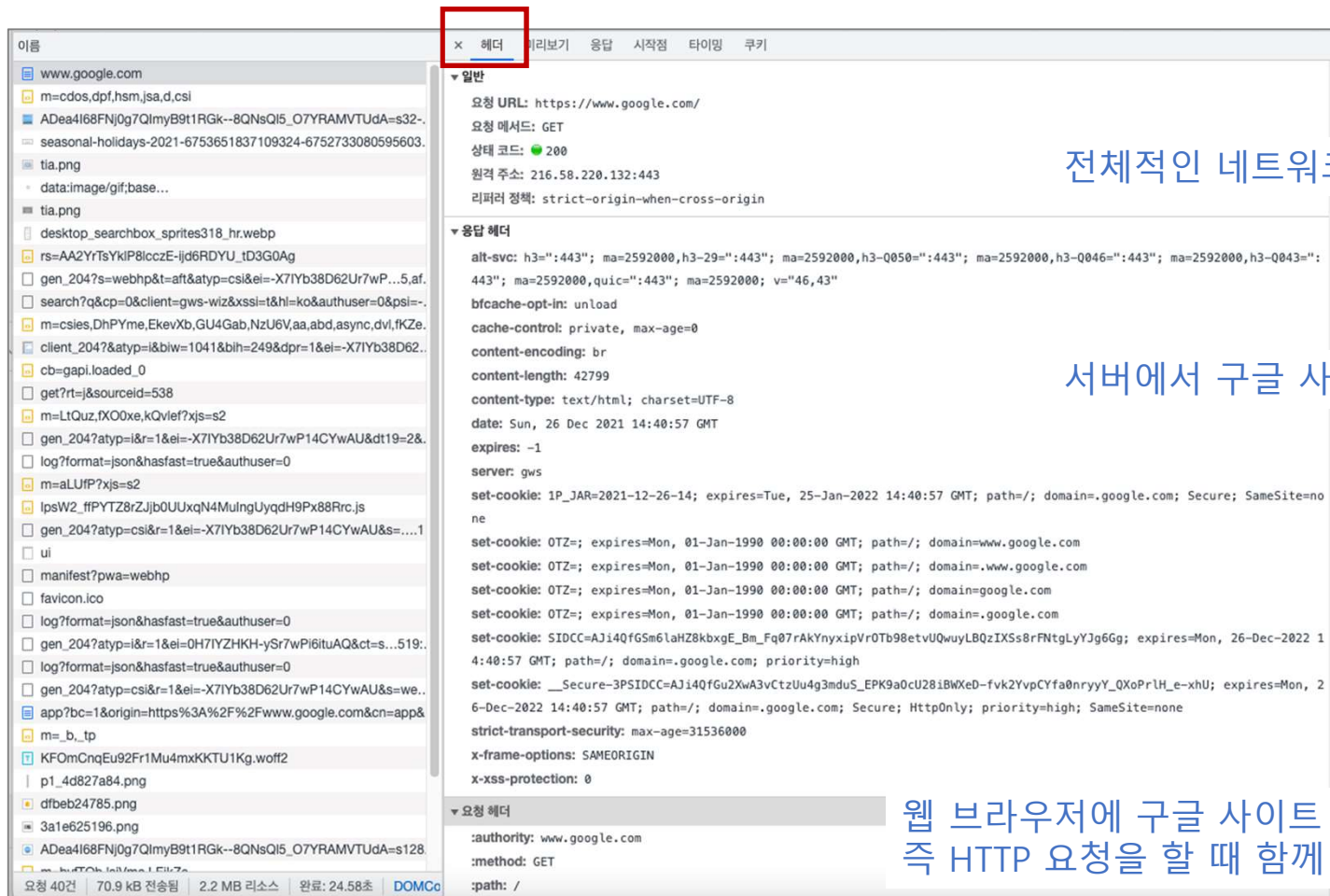
12. HTTP 통신과 JSON

1. 웹 브라우저에서 구글 사이트(www.google.com)로 접속해서 웹 개발자 도구 창을 열고 [네트워크] 탭 클릭
2. [F5]를 누르거나 [새로 고침] 아이콘을 클릭해서 현재 사이트를 다시 불러온다.
3. 클라이언트에서 구글 사이트를 보여 달라고 했기 때문에 구글 사이트에서 사용한 텍스트와 아이콘, 이미지 등 여러 요소들을 서버에서 다운로드한다

이름	상태	유형	시작점	크기	시간	폭포
lpsW2_ffPYTZ8rZJb0UUXqN4MuIngUyqdh9Px88...	200	script	rs=AA2YrTsYkiP8lcczE-ij...	(디스크 캐시)	2밀리초	
gen_204?atyp=csi&r=1&ei=-X7IYb38D62Ur7wP14...	204	ping	m=cdo5.dpf.hsm.jsa.d.c...	20 B	92밀리초	
ui	204	text/html	m=csies.DhPYme.EkevX...	0 B	151밀리초	
manifest?pwa=webhp	200	manifest	기타	(디스크 캐시)	2밀리초	
favicon.ico	200	x-icon	기타	(디스크 캐시)	5밀리초	
log?format=json&hasfast=true&authuser=0	200	xhr	rs=AA2YrTsYkiP8lcczE-ij...	155 B	90밀리초	
gen_204?atyp=i&r=1&ei=0H7IYZHKH-ySr7wPi6it...	204	ping	m=cdo5.dpf.hsm.jsa.d.c...	20 B	109밀리초	
log?format=json&hasfast=true&authuser=0	200	xhr	rs=AA2YrTsYkiP8lcczE-ij...	155 B	120밀리초	
gen_204?atyp=csi&r=1&ei=-X7IYb38D62Ur7wP14...	204	ping	m=cdo5.dpf.hsm.jsa.d.c...	20 B	93밀리초	
app?bc=1&origin=https%3A%2F%2Fwww.googl...	200	document	rs=AA2YrTsYkiP8lcczE-ij...	14.1 kB	229밀리초	
m=_b_..tp	200	script	app?bc=1&origin=https...	(디스크 캐시)	26밀리초	
KFOmCnqEu92Fr1Mu4mxKKTU1Kg.woff2	200	font	app?bc=1&origin=https...	(메모리 캐시)	0밀리초	
p1_4d827a84.png	200	png	app?bc=1&origin=https...	(메모리 캐시)	0밀리초	
dfbeb24785.png	200	png	app?bc=1&origin=https...	(메모리 캐시)	0밀리초	
3a1e625196.png	200	png	app?bc=1&origin=https...	(메모리 캐시)	0밀리초	
ADea4I68FNj0g7QlmyB9t1RGk--8QNsQI5_O7YR...	200	png	app?bc=1&origin=https...	(메모리 캐시)	0밀리초	
m=byfTOb,lsjVmc,LEikZe	200	script	m=_b_..tp:666	(디스크 캐시)	1밀리초	
m=n73qwf,ws9Tic,iZT63,e5qFLc,GkRiKb,UUJqVe...	200	script	m=_b_..tp:666	(디스크 캐시)	3밀리초	
m=Wt6vjf,hhhU8,FCpbqb,WhJNk	200	script	m=_b_..tp:666	(디스크 캐시)	1밀리초	
log?format=json&hasfast=true&authuser=0	200	xhr	m=_b_..tp:640	155 B	89밀리초	

요청 39건 | 70.9 kB 전송됨 | 2.2 MB 리소스 | 완료: 5.21초 | DOMContentLoaded: 496밀리초 | 로드: 901밀리초

왼쪽의 이름 목록에서 맨 위에 있는 `www.google.com`을 클릭하면 오른쪽에 창이 열리면서 `www.google.com` 문서에서 무엇을 주고받았는지 나타난다.

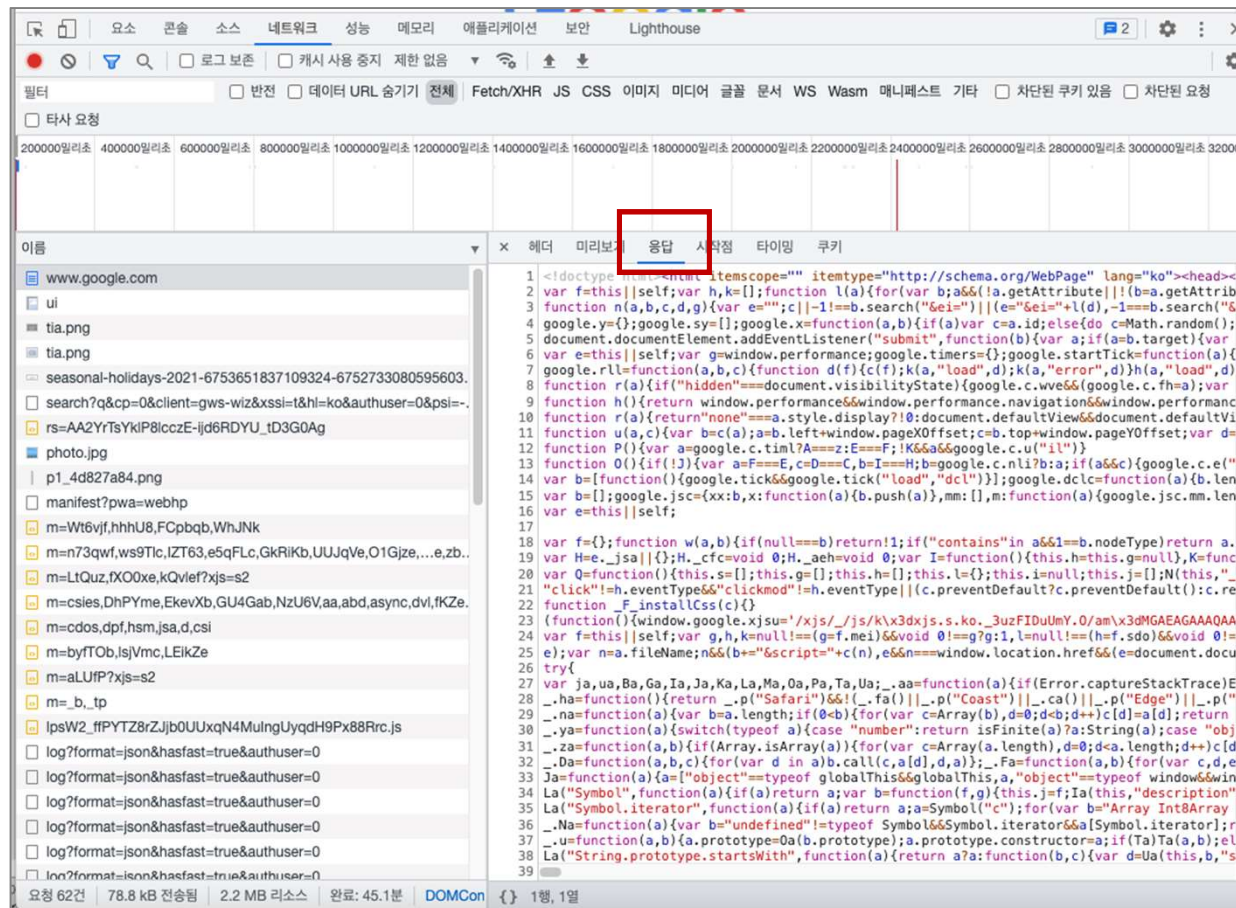


전체적인 네트워크 상태를 요약한 것

서버에서 구글 사이트 정보를 보내면서 함께 보내온 것

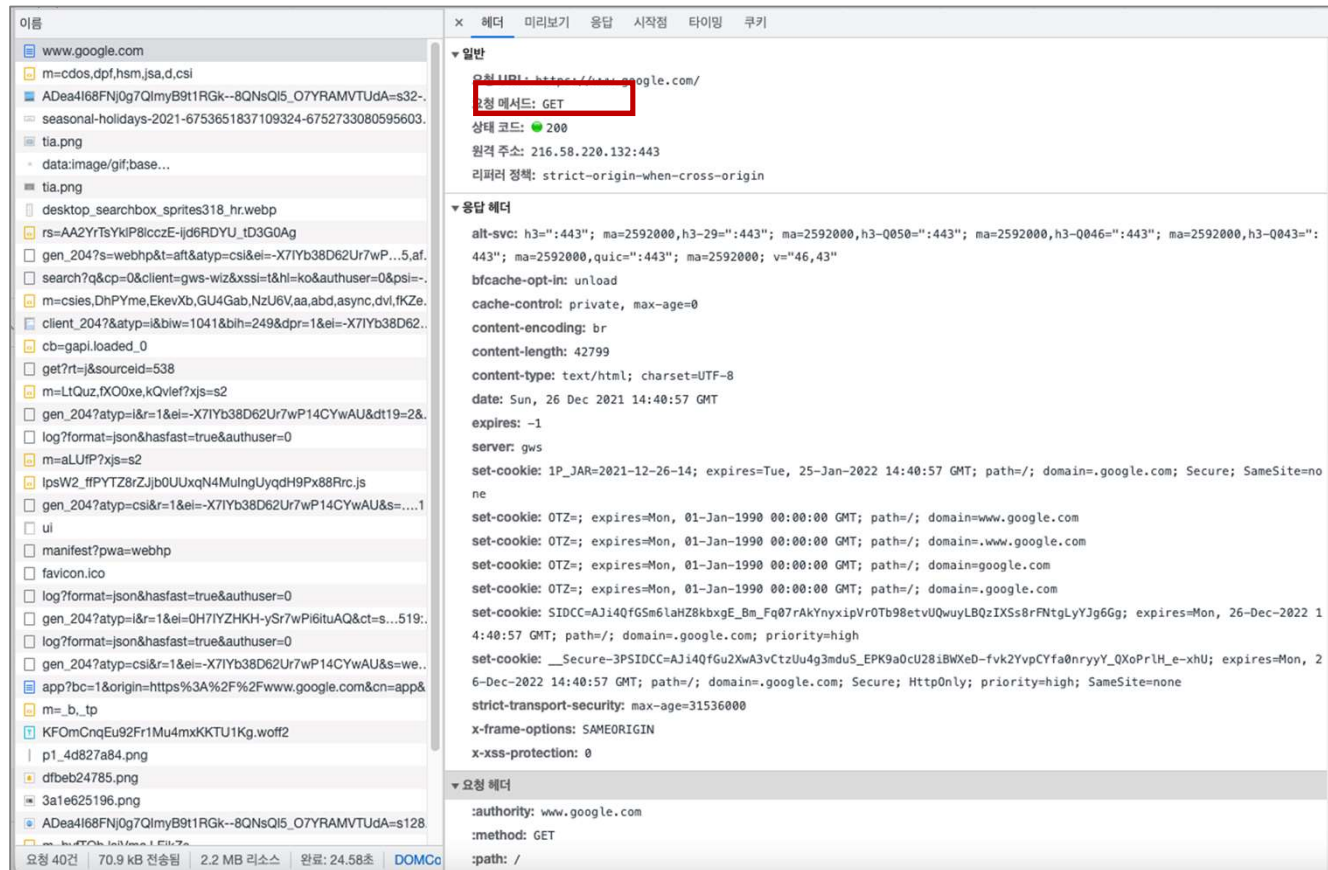
웹 브라우저에 구글 사이트 주소를 입력해서 서버로 보낼 때, 즉 HTTP 요청을 할 때 함께 넘겨진 정보

[응답] 탭을 클릭하면 서버에서 클라이언트로 응답 헤더와 함께 넘어온 실제 내용이 나타난다.
우리가 알고 있는 HTML 문서 형태인데, 이 내용이 웹 브라우저에 표시된다.



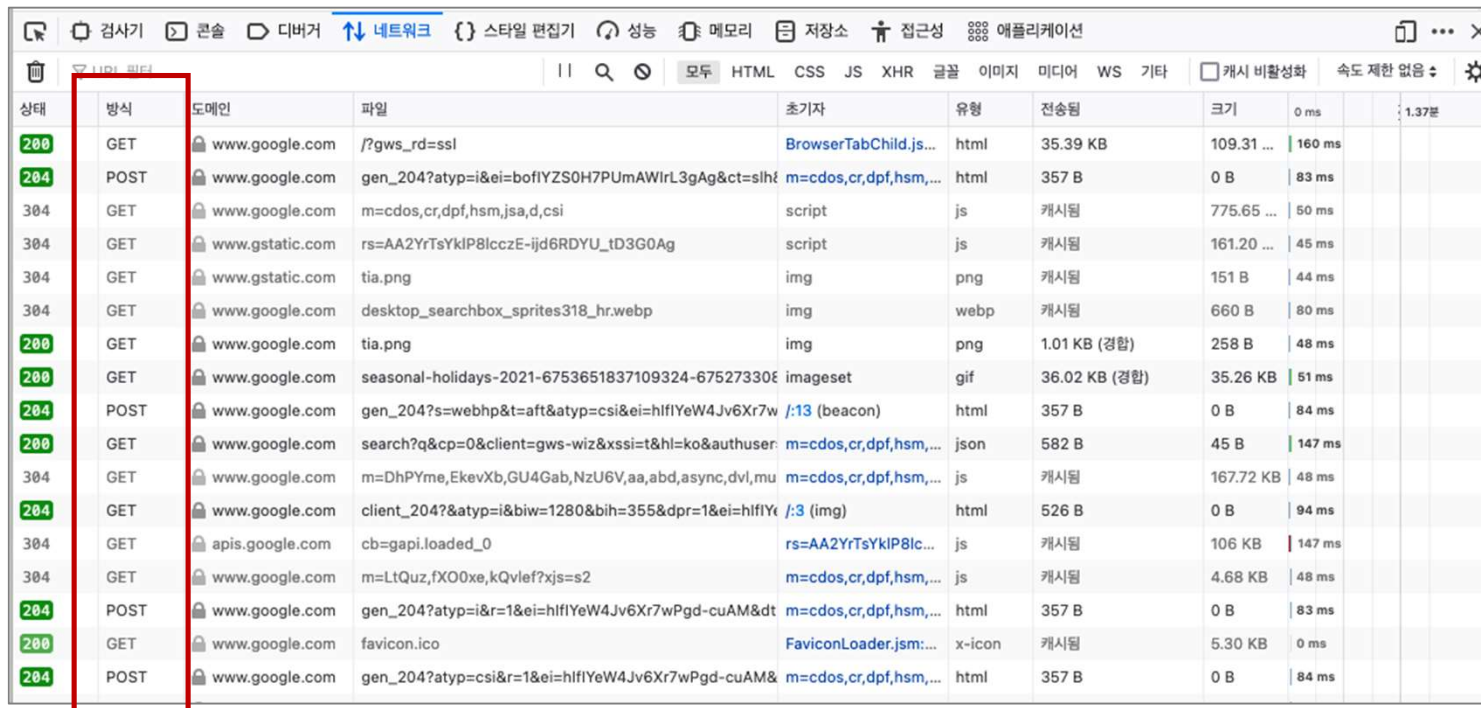
요청 방식, GET과 POST

요청 헤더에 있는 여러 정보 중에서 주의해서 볼 것은 요청 방식



요청 방식, GET과 POST

파이어폭스 웹 브라우저의 네트워크 창에는 GET이나 POST 같은 요청 방식이 함께 표시된다.

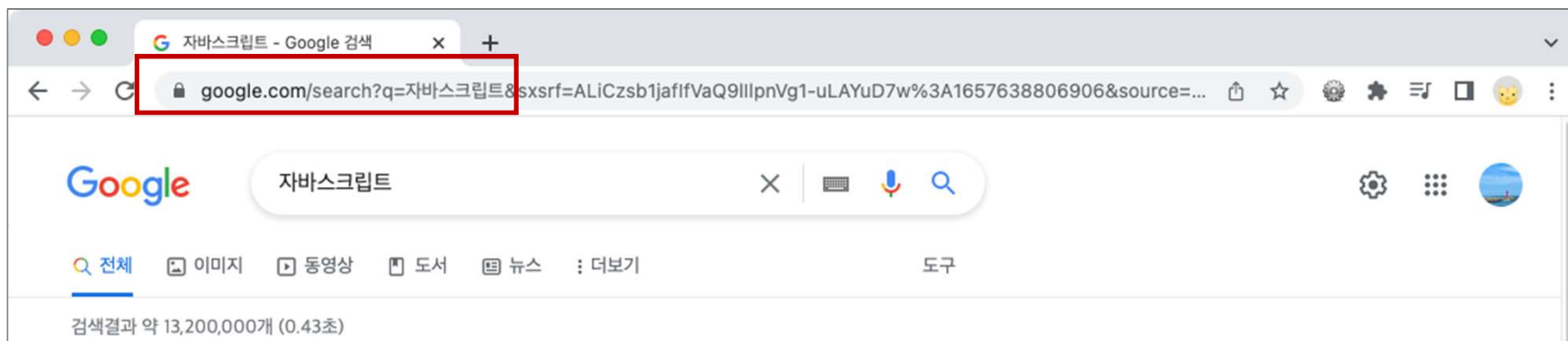


상태	방식	도메인	파일	초기자	유형	전송됨	크기	0 ms	1.37분
200	GET	www.google.com	/?gws_rd=ssl	BrowserTabChild.js...	html	35.39 KB	109.31 ...	160 ms	
204	POST	www.google.com	gen_204?atyp=i&ei=bofIYZS0H7PUMAWlrL3gAg&ct=slh&	m=cDos,cr,dpf,hsm,...	html	357 B	0 B	83 ms	
304	GET	www.google.com	m=cDos,cr,dpf,hsm,jsa,d,csi	script	js	캐시됨	775.65 ...	50 ms	
304	GET	www.gstatic.com	rs=AA2YrTsYklP8lcczE-ijD6RDYU_tD3G0Ag	script	js	캐시됨	161.20 ...	45 ms	
304	GET	www.gstatic.com	tia.png	img	png	캐시됨	151 B	44 ms	
304	GET	www.google.com	desktop_searchbox_sprites318_hr.webp	img	webp	캐시됨	660 B	80 ms	
200	GET	www.google.com	tia.png	img	png	1.01 KB (경합)	258 B	48 ms	
200	GET	www.google.com	seasonal-holidays-2021-6753651837109324-67527330E	imageset	gif	36.02 KB (경합)	35.26 KB	51 ms	
204	POST	www.google.com	gen_204?s=webhp&t=aft&atyp=csi&ei=hlfYeW4Jv6Xr7w	:/:13 (beacon)	html	357 B	0 B	84 ms	
200	GET	www.google.com	search?q&cp=0&client=gws-wiz&xssi=t&hl=ko&authuser:	m=cDos,cr,dpf,hsm,...	json	582 B	45 B	147 ms	
304	GET	www.google.com	m=DhPYme,EkevXb,GU4Gab,NzU6V,aa,abd,async,dvl,mu	m=cDos,cr,dpf,hsm,...	js	캐시됨	167.72 KB	48 ms	
204	GET	www.google.com	client_204?atyp=i&biw=1280&bih=355&dpr=1&ei=hlfYe	:/:3 (img)	html	526 B	0 B	94 ms	
304	GET	apis.google.com	cb=gapi.loaded_0	rs=AA2YrTsYklP8lcczE-ijD6RDYU_tD3G0Ag	js	캐시됨	106 KB	147 ms	
304	GET	www.google.com	m=LtQuz,fXO0xe,kQvlef?xjs=s2	m=cDos,cr,dpf,hsm,...	js	캐시됨	4.68 KB	48 ms	
204	POST	www.google.com	gen_204?atyp=i&r=1&ei=hlfYeW4Jv6Xr7wPgD-cuAM&dt	m=cDos,cr,dpf,hsm,...	html	357 B	0 B	83 ms	
200	GET	www.google.com	favicon.ico	FaviconLoader.jsm:...	x-icon	캐시됨	5.30 KB	0 ms	
204	POST	www.google.com	gen_204?atyp=csi&r=1&ei=hlfYeW4Jv6Xr7wPgD-cuAM&	m=cDos,cr,dpf,hsm,...	html	357 B	0 B	84 ms	

GET 방식

- 서버에 자료를 요청할 때, 사이트 주소 뒤에 자료를 붙여서 보내는 방식
- GET을 사용하면 웹 브라우저의 주소 표시줄에 요청 메시지가 함께 표시되고 따로 요청 본문은 사용하지 않는다.
- 서버로 사이트 주소를 보내면서 요청 자료도 함께 공개되기 때문에, 이렇게 요청 자료가 무엇인지 공개 되더라도 문제가 없을 경우 사용하는 방식.

예) 구글 사이트에서 '자바스크립트'를 검색한다면 웹 브라우저에서 서버로 보내는 요청 헤더에 GET 메서드를 사용한다.

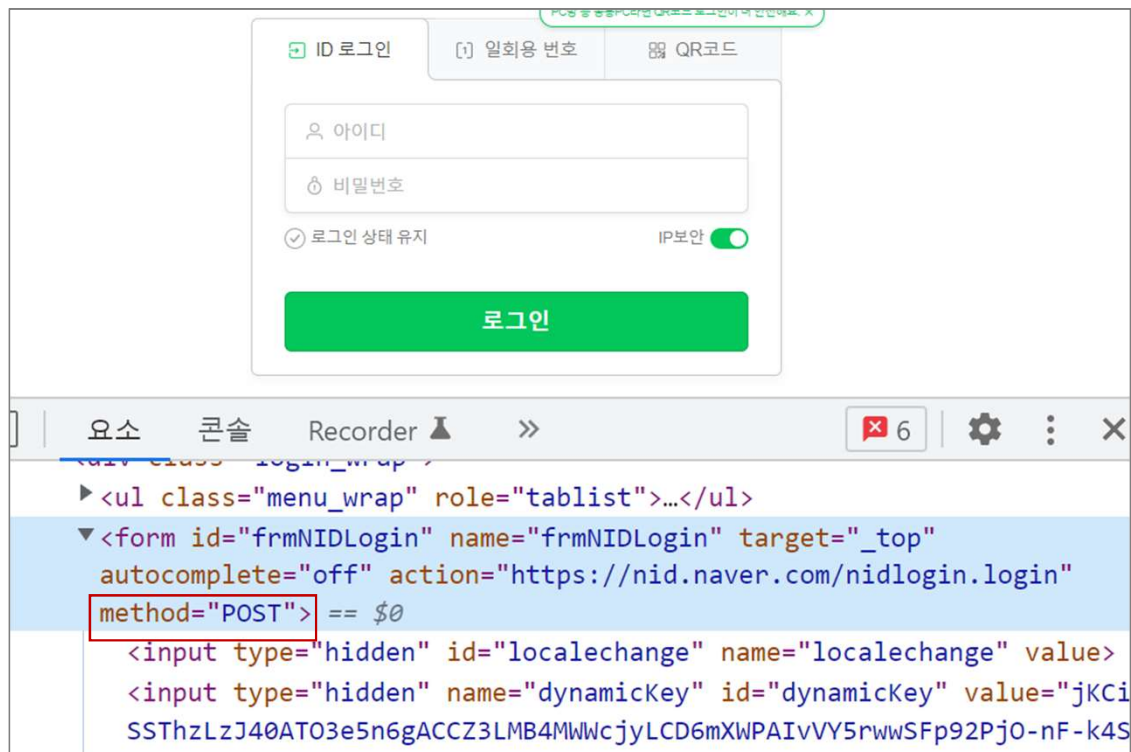


POST 방식

- POST를 사용하면 요청 내용이 겉으로 드러나지 않고 요청 본문(request body)에 따로 담아서 보낸다.

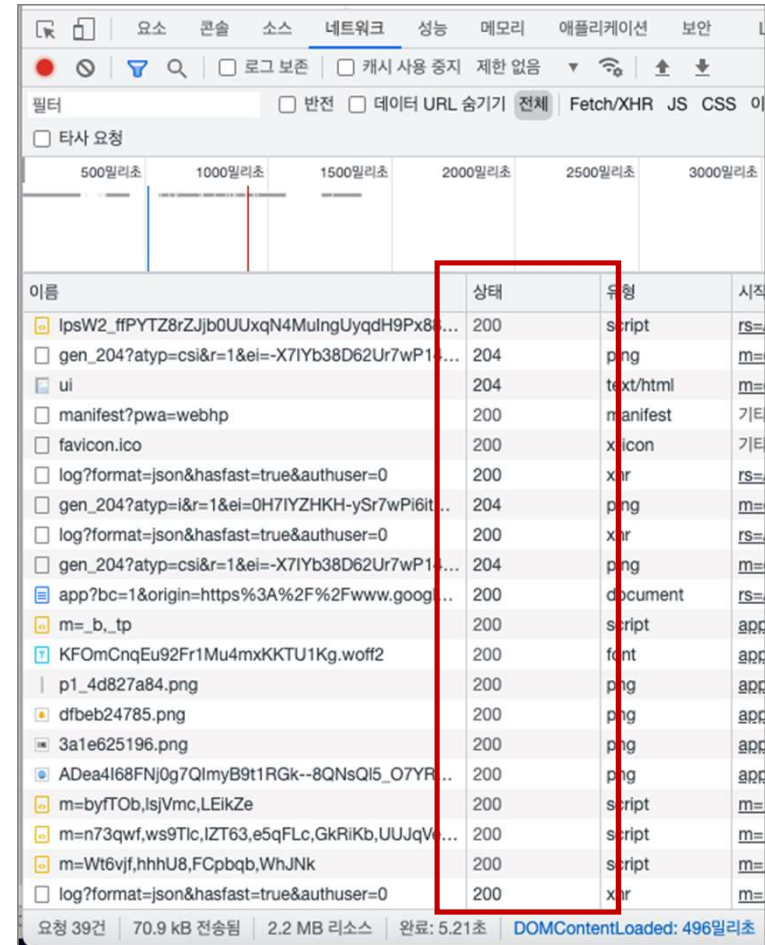
예) 로그인 창에 아이디와 비밀번호를 입력한 후 [로그인] 버튼을 클릭하면

사용자가 입력한 아이디나 비밀번호는 네트워크 외부에서 알아볼 수 없도록 요청 본문에 담아서 서버로 넘겨진다.



응답 상태

- 클라이언트의 요청을 받은 서버가 필요한 작업을 처리하고 그 결과를 클라이언트로 보낼 때
- 서버로 요청한 것이 성공적으로 처리되었는지, 또는 요청한 파일이 없어서 실패했는지 등을 응답 상태를 '상태' 칼럼에 숫자로 표시한다.
- 서버에서 자료를 받아 프로그래밍할 때는 응답 상태를 확인한 후 진행한다.



이름	상태	유형	시작
lpsW2_ffPYTZ8rZJjb0UUxqN4MulngUyqdH9Px8...	200	script	rs=
gen_204?atyp=csi&r=1&ei=-X7IYb38D62Ur7wP1...	204	png	m=
ui	204	text/html	m=
manifest?pwa=webhp	200	manifest	기타
favicon.ico	200	x/icon	기타
log?format=json&hasfast=true&authuser=0	200	x/hr	rs=
gen_204?atyp=i&r=1&ei=0H7IYZHKKH-ySr7wPi6it...	204	png	m=
log?format=json&hasfast=true&authuser=0	200	x/hr	rs=
gen_204?atyp=csi&r=1&ei=-X7IYb38D62Ur7wP1...	204	png	m=
app?bc=1&origin=https%3A%2F%2Fwww.goog...	200	document	rs=
m=_b_tp	200	script	app
KFOmCnqEu92Fr1Mu4mxKKTU1Kg.woff2	200	font	app
p1_4d827a84.png	200	png	app
dfbeb24785.png	200	png	app
3a1e625196.png	200	png	app
ADea4i68FNj0g7QlmyB9t1RGk--8QNsQI5_O7YR...	200	png	app
m=byfTOb,IsjVmc,LEikZe	200	script	m=
m=n73qwf,ws9Tlc,lZT63,e5qFLc,GkRiKb,UUJqV...	200	script	m=
m=Wt6vjf,hhhU8,FCpbqb,WhJNk	200	script	m=
log?format=json&hasfast=true&authuser=0	200	x/hr	m=

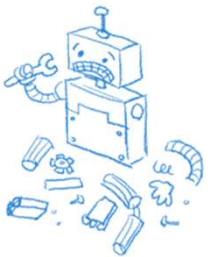
요청 39건 | 70.9 kB 전송됨 | 2.2 MB 리소스 | 완료: 5.21초 | DOMContentLoaded: 496밀리초

상태	메시지	기능
2XX	자료 요청을 수락했거나 자료 전송이 성공적으로 끝났습니다	
200	OK	서버에서 클라이언트로 성공적으로 전송했습니다.
202	Accepted	서버에서 클라이언트 요청을 수락했습니다.
4XX	클라이언트에서 주소를 잘못 입력했거나 요청이 잘못되었습니다.	
400	Bad Request	요청을 실패했습니다.
401	Unauthorized	권한이 없어 거절되었습니다. 인증 가능합니다.
403	Forbidden	권한이 없어 거절되었습니다. 인증을 시도해도 계속 거절됩니다.
404	Not Found	문서를 찾을 수 없습니다.
408	Request Timeout	요청 시간이 초과되었습니다.
5XX	서버 측의 오류로 처리할 수 없습니다	
500	Internal Server Error	서버 내부에 오류가 발생했습니다.
503	Service Unavailable	요청한 서비스를 이용할 수 없습니다.



404. That's an error.

The requested URL /rustybrick was not found on this server. That's all we know.



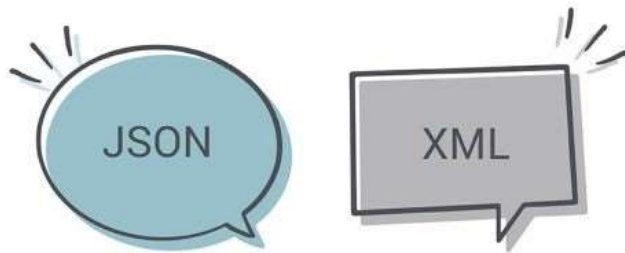
JSON

JavaScript Object Notation

데이터 교환 방식

서버와 클라이언트 간에 자료를 주고받기 위해 양쪽 모두 이해할 수 있는 형식을 사용해야 한다.

- 1) XML은 컴퓨터에서 처리하는 모든 문서의 표준 형식이기 때문에 웹에서 사용 가능
- 2) 최근에는 JSON이라는 형식을 더 많이 사용한다



JSON의 특징

- 텍스트로만 구성되었기 때문에 서버와 클라이언트 사이에 주고 받을 때 전송 속도가 아주 빠르다.
- JSON은 프로그래밍 언어나 플랫폼에 대해 독립적이기 때문에 C++이나 자바, 자바스크립트, 파이썬 등 많은 언어에서 사용할 수 있다.
- 자바스크립트 사용자라면 누구나 알고 있는 표기법을 사용하기 때문에 읽기도 쉽고 필요에 따라 자바스크립트 객체로 변환하기도 쉽다.

유튜브에서 프로그램에 필요한
자료를 넘겨줄 때 사용하는
JSON의 사용 예

The screenshot shows the YouTube Data API documentation page. The left sidebar contains navigation links under 'Overview', 'Guides and Tutorials', and 'Implementation and Migration Guide'. The main content area displays a sample API response in JSON format for a video search query.

설명: 이 예에서는 video 리소스를 검색하며 API 응답에 포함되어야 하는 몇 가지 리소스 부분을 식별합니다.

API 응답:

```
{
  "kind": "youtube#videoListResponse",
  "etag": "\"UCBpFjp2h75_b92t44sqraUcyu0/sDA1sG9NGKfr6v5A1PZKSEZdtqA\"",
  "videos": [
    {
      "id": "71CDEYXw3mM",
      "kind": "youtube#video",
      "etag": "\"UCBpFjp2h75_b92t44sqraUcyu0/iYynQR8AtacsFUwWmrVaw4Smb_Q\"",
      "snippet": {
        "publishedAt": "2012-06-20T22:45:24.000Z",
        "channelId": "UC_x5XG10V2P6uZZ5FSM9Ttw",
        "title": "Google I/O 101: Q&A On Using Google APIs",
        "description": "Antonio Fuentes speaks to us and takes questions on working with Google APIs",
        "thumbnails": {
          "default": {
            "url": "https://i.ytimg.com/vi/71CDEYXw3mM/default.jpg"
          },
          "medium": {
            "url": "https://i.ytimg.com/vi/71CDEYXw3mM/mqdefault.jpg"
          },
          "high": {
            "url": "https://i.ytimg.com/vi/71CDEYXw3mM/hqdefault.jpg"
          }
        }
      },
      "categoryId": "28"
    }
  ]
}
```

JSON의 형식

```
{
  "이름" : 값,
  .....
}
```

- 중괄호 { } 사이에 '이름'과 '값' 으로 구성된다.
- JSON에서는 '이름' 부분에 반드시 큰따옴표를 붙이는 것이 큰 차이점

예) '도레미'라는 학생의 수업 신청 정보

객체

```
{
  name : "도레미",
  major : "컴퓨터 공학",
  grade : 2
}
```

JSON

```
{
  "name" : "도레미",
  "major" : "컴퓨터 공학",
  "grade" : 2
}
```

JSON 문자열

'{ "name" : "도레미", "major" : "컴퓨터 공학", "grade" : 2 }'

JSON의 '이름'

반드시 큰따옴표(" ")로 묶어야 한다.

'이름'에 작은따옴표를 사용하거나 큰따옴표가 없는 이름은 사용할 수 없습니다.

맞게 사용한 예

```
{ "name" : "도레미" }
```

잘못 사용한 예

```
{ 'name' : "도레미" }
```

```
{ name : "도레미" }
```

- JSON 이름에는 공백^{space}이나 하이픈(-), 언더바(_)를 함께 사용할 수 있다.

문법적으로 다음 형식도 가능

```
{ "full name" : "도레미" }  
{ "full-name" : "도레미" }
```

- 하지만 이름에 공백이나 하이픈이 있을 경우 프로그램을 통해 그 이름에 접근할 때 쉽지 않기 때문에 둘 이상의 단어로 된 이름을 사용한다면 언더스코어(_)를 사용하는 것이 좋다.

```
{ "full_name" : "도레미" }
```

JSON의 '값'

객체에서는 '값' 부분에 함수(메서드)도 사용할 수 있지만

JSON의 '값'에는 숫자, 문자열, boolean, null, 배열만 사용할 수 있고, **함수는 사용할 수 없다.**

- **숫자형**: JSON에서는 정수와 실수 모두 사용할 수 있지만, 8진수나 16진수를 사용한 표기법은 지원하지 않는다.
- **문자열**: JSON 문자열은 항상 큰따옴표(" ")로 묶어야 한다.
- **논릿값과 null**: true/false 값을 가지는 논리형을 사용할 수도 있고 null 유형도 사용할 수 있다.
- **문자열, 배열**: 문자열이나 배열을 값으로 사용할 수 있다. JSON 문자열 안에 또다른 JSON 문자열을 넣을 수도 있다. JSON에서 배열을 사용할 때에도 일반 배열과 마찬가지로 대괄호([])를 사용한다.

JSON 문자열

JSON 문자열 안에 또 다른 JSON 문자열을 지정할 수 있다.

예) 신청한 과목의 이름과 주당 시간을 새로운 JSON 문자열로 사용

```
{  
  "name" : "도레미",  
  "major" : "컴퓨터 공학",  
  "grade" : 2,  
  "course" : {  
    "title" : "웹 기초",  
    "timePerWeek" : 3  
  }  
}
```

여러 개의 JSON 문자열을 배열 형태로 저장할 수 있다.

(예) <https://jsonplaceholder.typicode.com/users>



```
[
  {
    "id": 1,
    "name": "Leanne Graham",
    "username": "Bret",
    "email": "Sincere@april.biz",
    "address": {
      "street": "Kulas Light",
      "suite": "Apt. 556",
      "city": "Gwenborough",
      "zipcode": "92998-3874",
      "geo": {
        "lat": "-37.3159",
        "lng": "81.1496"
      }
    },
    "phone": "1-770-736-8031 x56442",
    "website": "hildegard.org",
    "company": {
      "name": "Romaguera-Crona",
      "catchPhrase": "Multi-layered client-server neural-net",
      "bs": "harness real-time e-markets"
    }
  },
  {
    "id": 2,
    "name": "Ervin Howell",
    "username": "Antonette",
    "email": "Shanna@melissa.tv",
    "address": {
      "street": "Victor Plains",
      "suite": "Suite 879",
      "city": "Wisokyburgh",
      "zipcode": "90566-7771",
      "geo": {
        "lat": "-43.9509",
```

- 전체가 하나의 배열로 묶여 있고 그 안에 여러 사용자 정보가 들어 있다.
- 각 사용자 정보에서 'address'나 'company' 이름 부분을 보면 주소와 회사 정보가 또 다른 JSON 문자열로 구성되어 있다.
- 하나의 JSON 문자열 안에 얼마든지 많은 정보를 저장할 수 있다.

객체를 JSON 형식으로 변환하기

클라이언트에서 정보를 처리할 때는 객체를 사용한다.

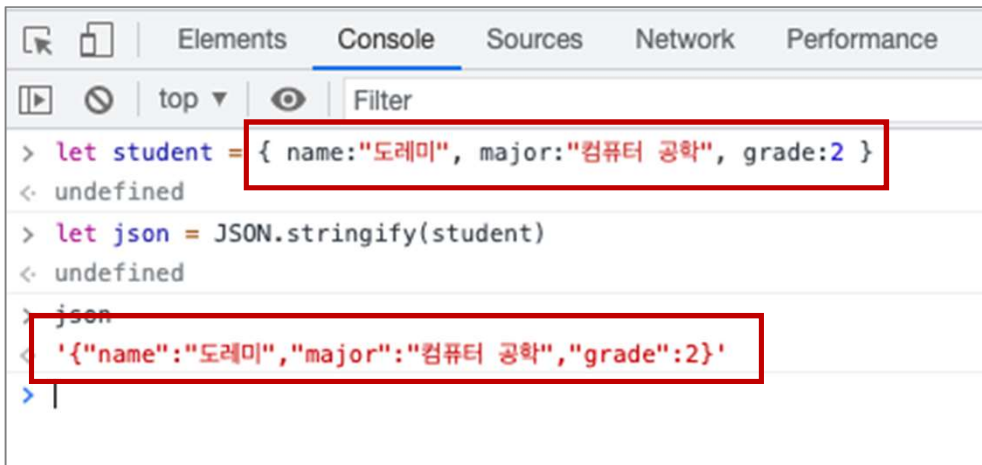
객체를 JSON 형식으로 저장하거나, JSON 형식을 요구하는 서버로 넘기려면

→ 객체를 JSON 형식으로 변환해야 한다. 직렬화(stringify)라고 한다.

JSON.stringify(객체)

```
let student = {name:"도레미", major:"컴퓨터 공학", grade:2}
```

```
let json = JSON.stringify(student)
```



student 객체와 json 문자열은 내용이 같지만,
네트워크에서 자료를 주고받을 때에는
가벼운 JSON 형식으로 바뀌어서 사용한다.

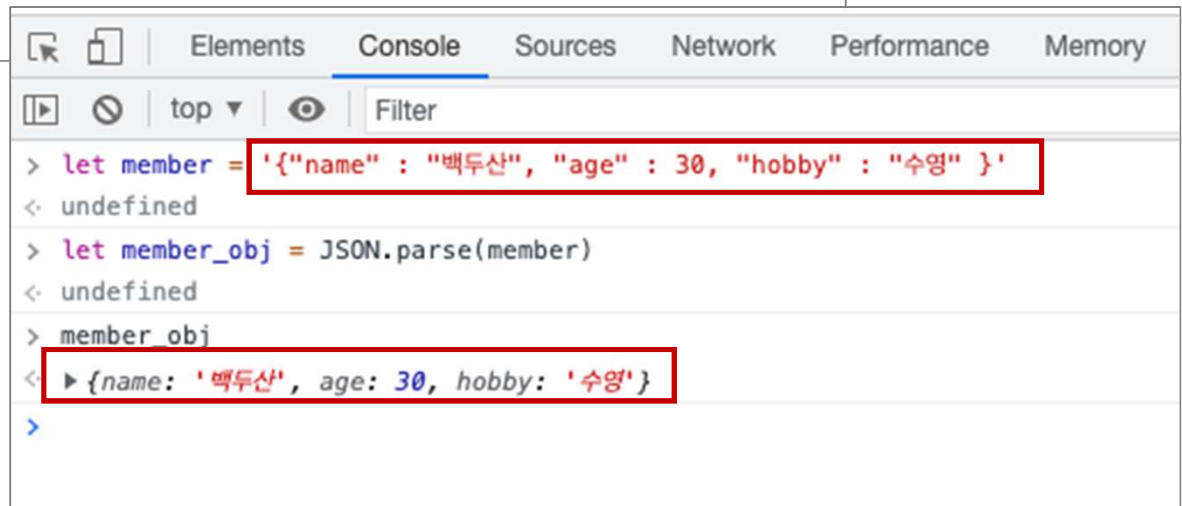
JSON 을 객체로 변환하기

서버에서 가져온 JSON 자료를 사용하려면 객체 형태로 변환해야 한다 → 이것을 파싱(parsing)이라고 한다.

```
JSON.parse(JSON 문자열)
```

서버에서 JSON 문자열을 가져오는 방법은 배우지 않았기 때문에 일단 서버에서 자료를 가져와서 member 변수에 저장했다고 가정한다.

```
let member = '{"name" : "백두산", "age" : 30, "hobby" : "swimming" }'  
let member_obj = JSON.parse(member)
```



서버에서 자료 가져오기

일반적인 서버와 클라이언트의 통신

- 1) 웹 브라우저 화면에 'www.daum.net'을 입력하고 [Enter]를 누르면 인터넷 회선을 통해 서버 컴퓨터로 접속한다.
- 2) 서버 컴퓨터에서 해당 페이지를 찾아낸 후 내용을 다운로드해서 웹 브라우저 화면에 보여준다.
- 3) 메뉴 중에서 '게임'을 클릭하면 현재 화면이 완전히 사라지고 게임과 관련된 페이지로 이동한다.

메뉴나 링크를 클릭하면 현재 페이지를 완전히 지우고 새로운 화면을 가져와 보여주는 방식

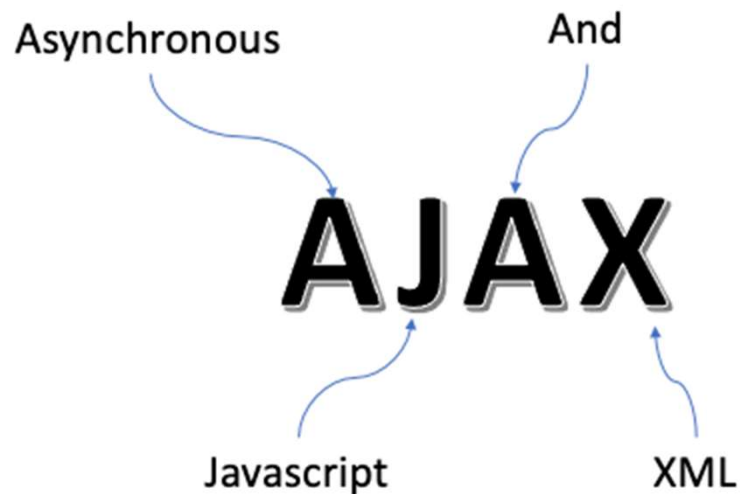
비동기적으로 통신한다면

페이스북이나 트위터 같은 SNS 사이트를 사용할 때에도 화면을 스크롤하면 사이트 전체가 새로 로딩되는 것이 아니라 기존 내용은 그대로 둔 상태에서 다음 내용만 가져와서 보여준다..

이렇게 웹 문서 전체를 다시 불러오지 않고 일부분만 가져와 실행할 수 있는 것은
AJAX(Asynchronous Javascript And XML)기능 때문

AJAX

서버와의 비동기 통신을 위한 방법



- AJAX란, 서버에 요청하는 것과 서버의 응답이 한꺼번에 일어나지 않는 것을 말합니다. (요청한 후 응답을 기다리는 동안 다른 요청을 할 수 있습니다.)
- 비동기적인 통신을 위해 서버와 클라이언트 사이에 주고받은 통신 방법이 **XMLHttpRequest**입니다.
- ES6 이후에는 **fetch**를 사용하고 있습니다.

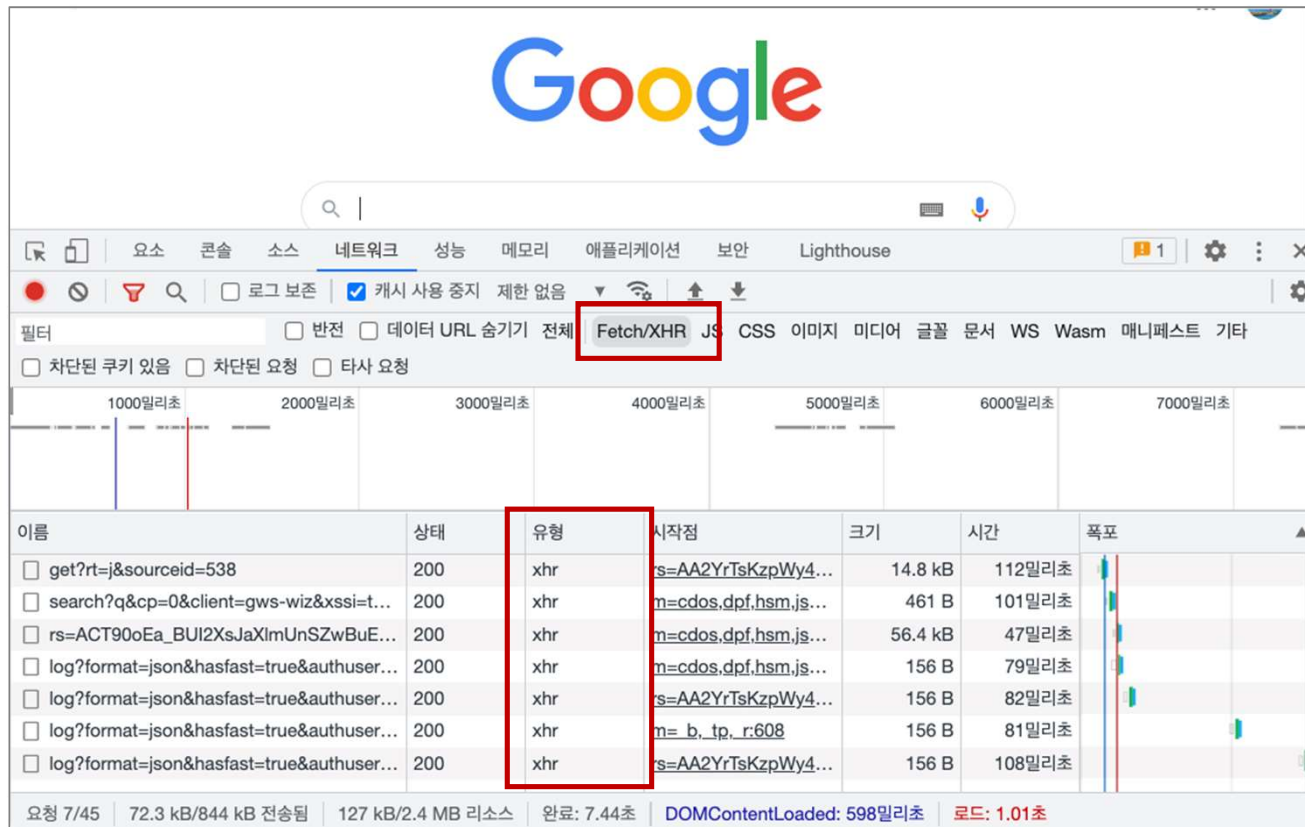
XMLHttpRequest 객체

- 웹 브라우저에서 서버로 데이터를 요청하고 서버에서 자료를 받아올 때는 HTTP 통신이 가능한 XMLHttpRequest 객체를 사용한다.
- XMLHttpRequest 객체의 프로퍼티와 메서드를 사용해서 자료를 주고받거나 상태를 체크한다.
- 웹 페이지 전체가 아니라 필요한 부분만 자료만 가져올 수 있다.

XMLHttpRequest

'XML'이라는 자료를 'HTTP' 프로토콜을 사용해서 'Request(요청)'한다

12. HTTP 통신과 JSON



XMLHttpRequest 객체 만들기

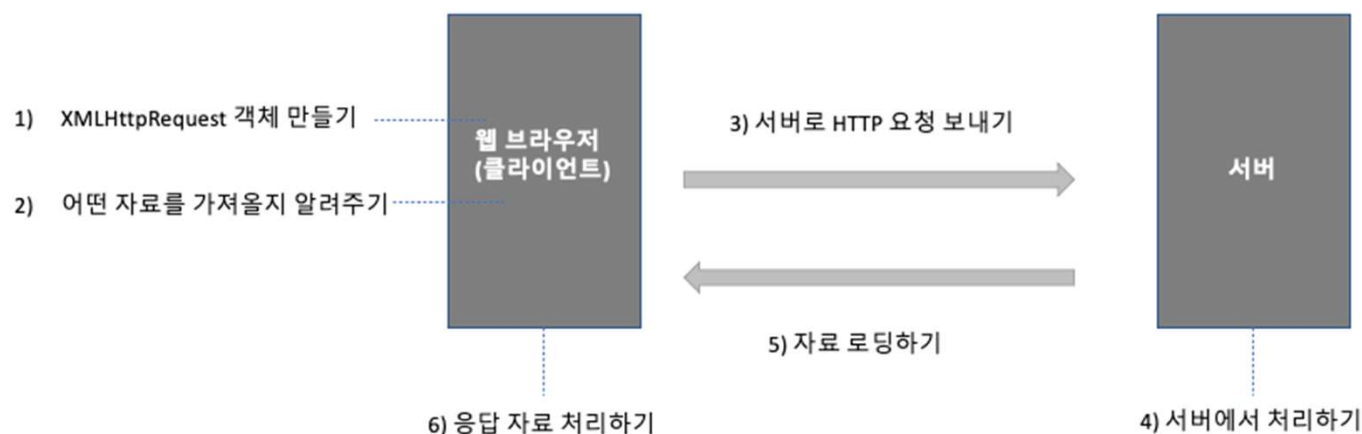
new 예약어를 사용해서 XMLHttpRequest 객체의 인스턴스를 만든다.

인스턴스는 xhr이라는 이름을 많이 사용한다.

```
new XMLHttpRequest()
```

```
let xhr = new XMLHttpRequest()
```

XMLHttpRequest 객체를 만들면 서버로 자료를 요청하고 자료를 받아올 수 있다.



open() – 어떤 자료를 가져올지 지정

서버로 자료를 요청할 때 어떤 방식을 사용할지, 어떤 자료가 필요한지, 그리고 비동기 처리 여부를 지정한다.

```
open(방식, 자료 위치, 비동기 여부)
```

- 방식: HTTP 요청 방식을 지정한다. GET이나 POST 중 하나이고 대문자로 사용해야 한다.
- 자료 위치: 요청할 서버의 URL을 지정한다.
- 비동기 여부: 비동기 요청인지, 동기 요청인지의 여부를 판단하는 항목.

true - 비동기, false - 동기

기본적으로 비동기 처리하므로 따로 지정하지 않으면 비동기로 처리한다

send() – 서버로 요청 전송

사용자 요청을 서버로 보내는 메서드

```
send(내용)
```

- send() 괄호 안에 들어가는 매개변수는 옵션다.
- POST 방식을 사용할 경우에는 서버로 넘길 내용을 매개변수로 넘겨주고, GET 방식을 사용할 경우에는 null로 넘기거나 빈 상태로 남겨 둔다.

(예) GET 방식을 이용해 test.txt 파일에 비동기 방식으로 연결하려면

```
xhr.open("GET", "test.txt", true);  
xhr.send();
```

JSON 가져오기 연습

- JSON 자료는 기본적으로 서버에 저장되어 있다.
- 연습을 위해 사용자 컴퓨터를 서버로 만들어 주는 VS Code의 '라이브 서버' 를 사용한다.

12 폴더에는 student.json 파일이 미리 만들어져 있다.

```
{  
  "name" : "도레미",  
  "major" : "컴퓨터 공학",  
  "grade" : 2  
}
```

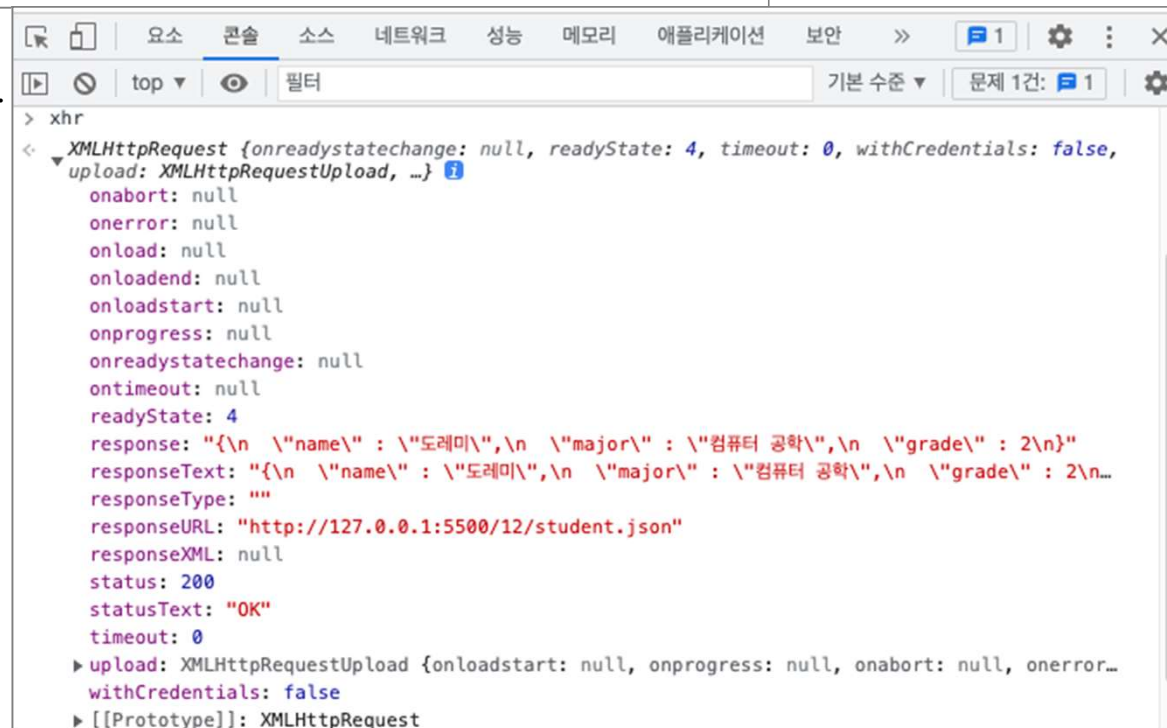
반드시 VS Code에서 라이브 서버를 사용해서 문서를 열어야 한다.

- 1) VS Code에서 26@student.html 파일을 열고 '라이브 서버' 를 사용해 브라우저에 표시한다.
- 2) 콘솔 창을 열고 서버에 있는 student.json 파일을 가져오는 소스를 입력한다

```
let xhr = new XMLHttpRequest();
xhr.open("GET", "student.json");
xhr.send();
```

- 3) json 자료 확인하기.

xhr



```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <div id="demo">
    <p>Ajax Test</p>
    <button type="button" onclick="loadDoc()">Json 로드</button>
  </div>

  <script>
    function loadDoc() {
      const xhttp = new XMLHttpRequest();
      xhttp.onload = function() {
        document.getElementById("demo").innerHTML = this.responseText;
      }
      xhttp.open("GET", "student.json");
      xhttp.send();
    }
  </script>
</body>
</html>
```

readyState 프로퍼티

readyState 프로퍼티는 XMLHttpRequest 객체의 현재 상태를 나타낸다.

객체에서 서버로 자료를 요청했는지, 자료가 도착했는지, 사용할 준비가 되었는지 등을 알 수 있다.

상태	기능
0	아직 아무 요청도 하지 않은 상태입니다.
1	서버로 자료를 요청하고 성공한 상태입니다.
2	서버 요청에 대한 응답으로 헤더가 도착한 상태입니다.
3	서버에서 자료들이 로딩 중인 상태입니다.
4	자료 처리가 끝나서 프로그램에서 사용할 수 있는 상태입니다.

0 → 1 → 2 → 3 → 4 →
0 → 1 ...처럼 순서대로
반복한다

```

ontimeout: null
readyState: 4
response: "{\n  \"name\" : \"도레미\", \n  \"major\" : \"컴퓨터 공학\", \n  \"grade\" : 2\n}"
responseText: "{\n  \"name\" : \"도레미\", \n  \"major\" : \"컴퓨터 공학\", \n  \"grade\" : 2\n..."
responseType: ""
responseURL: "http://127.0.0.1:5500/12/student.json"
responseXML: null
status: 200
statusText: "OK"

```

state, statusText 프로퍼티

status 프로퍼티는 HTTP 상태 코드를 나타내고

statusText 프로퍼티는 상태에 대한 설명 메시지를 알려준다.

상태	메시지	기능
200	OK	서버에서 클라이언트로 성공적으로 전송했습니다.
202	Accepted	서버에서 클라이언트 요청을 수락했습니다.
400	Bad Request	요청을 실패했습니다.
401	Unauthorized	권한이 없어 거절되었습니다. 인증 가능합니다.
403	Forbidden	권한이 없어 거절되었습니다. 인증을 시도해도 계속 거절됩니다.
404	Not Found	문서를 찾을 수 없습니다.
408	Request Timeout	요청 시간이 초과되었습니다.
500	Internal Server Error	서버 내부에 오류가 발생했습니다.
503	Service Unavailable	요청한 서비스를 이용할 수 없습니다.

readyState와 state를 어디에 쓰나

readyState 값이 바뀔 때마다 readystatechange 이벤트가 발생한다.

(예) 요청이 성공적으로 끝났을 때(즉, readyState 값이 4일 때) 실행할 명령은

```
xhr.onreadystatechange = function() {  
    if (xhr.readyState == 4) {    // 요청이 성공했다면  
        .....  
    }  
}
```

- readystatechange 이벤트가 발생했을 때 실행할 함수를 연결한 후
- 함수 안에서 readyState 값이 4일 경우에 명령을 처리합니다

readyState 값은 요청이 성공했는지를 알려주기 때문에 만약 서버에 없는 파일을 요청하더라도 readyState 값은 4.
요청에 성공하고 서버에서 필요한 파일을 가져왔는지 체크하려면
readyState 프로퍼티 값이 4이면서 state 프로퍼티 값이 200일 경우

```
xhr.onreadystatechange = function() {  
    if (xhr.readyState === 4 && xhr.status === 200) { // 자료가 있고 가져오는 데 성공했다면  
        .....  
    }  
}
```


response,.responseText 프로퍼티

- response : 요청에 대한 응답
- responseText : 요청에 대한 응답이 문자열 형태로 저장된다. 이 값을 프로그래밍에 사용.
- responseType : 응답 데이터의 종류
- responseURL : 응답을 보낸 URL
- responseXML : HTML이나 XML 같은 형식의 데이터를 받아올 때 사용

콘솔 창에 다음과 같이 입력하면 어떤 값을 가져왔는지 알 수 있다.

```
xhr.responseText
```

앞의 실습(student.json)에 이어서 연습해 보기

4) 가져온 값 확인하기

```
xhr.responseText
```

5) JSON 문자열을 객체로 바꾸기

6) 12Wstudent.html 문서에 #result 영역을 미리 만들어 두었으므로 #result 영역에 가져온 값을 표시하자.

```
let students = JSON.parse(xhr.responseText);  
  
document.getElementById("result").innerHTML = `${students.name} 학생은 ${students.grade}학년입니다.`
```

도레미 학생은 2학년입니다.



```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>JSON</title>
  <link rel="stylesheet" href="css/student.css">
</head>
<body>
  <div id="container">
    <h1>수강생 명단</h1>

    <div id="result"></div>
  </div>

</body>
</html>
```

```
#container {
  width: 400px;
  margin: 20px auto;
}
h1 {
  font-size: 1.5em;
  text-align: center;
}
h2 {
  font-size: 1.2em;
}
#result {
  width: 100%;
  margin: 20px 10px;
  padding: 10px;
}
```

student.css

[실습] JSON 자료를 가져와 표시하기 1

12Wstudent.html과 12WjsWstudent.js에서 연습하기

```
let xhr = new XMLHttpRequest();
xhr.open("GET", "student.json");
xhr.send();

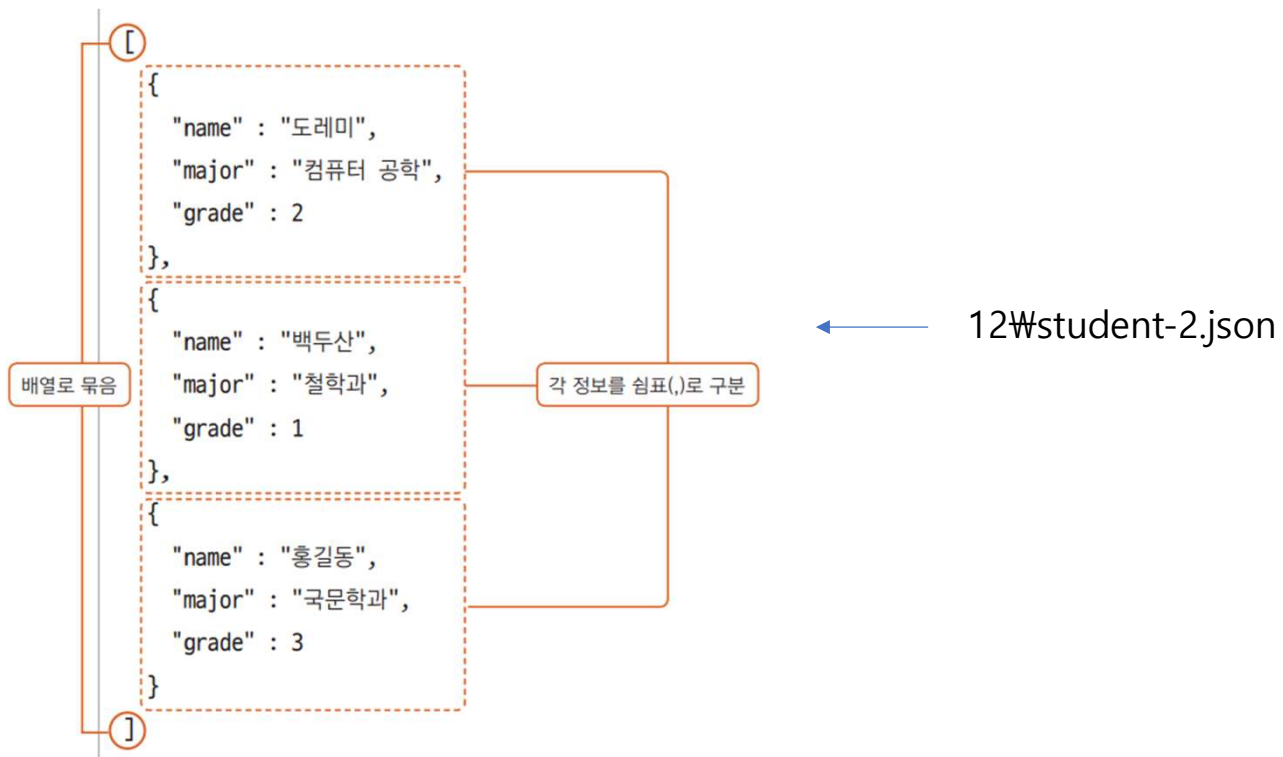
xhr.onreadystatechange = function () {
  if (xhr.readyState === 4 && xhr.status === 200) {
    let student = JSON.parse(xhr.responseText);
    document.getElementById("result").innerHTML = `
      <h1>${student.name}</h1>
      <ul>
        <li>전공: ${student.major}</li>
        <li>학년: ${student.grade}</li>
      </ul>
    `;
  }
};
```

도레미

- 전공: 컴퓨터 공학
- 학년: 2

[실습] JSON 자료를 가져와 표시하기 1

12Wstudent-2.html과 12WjsWstudent-2.js에서 연습하기



[실습] JSON 자료를 가져와 표시하기

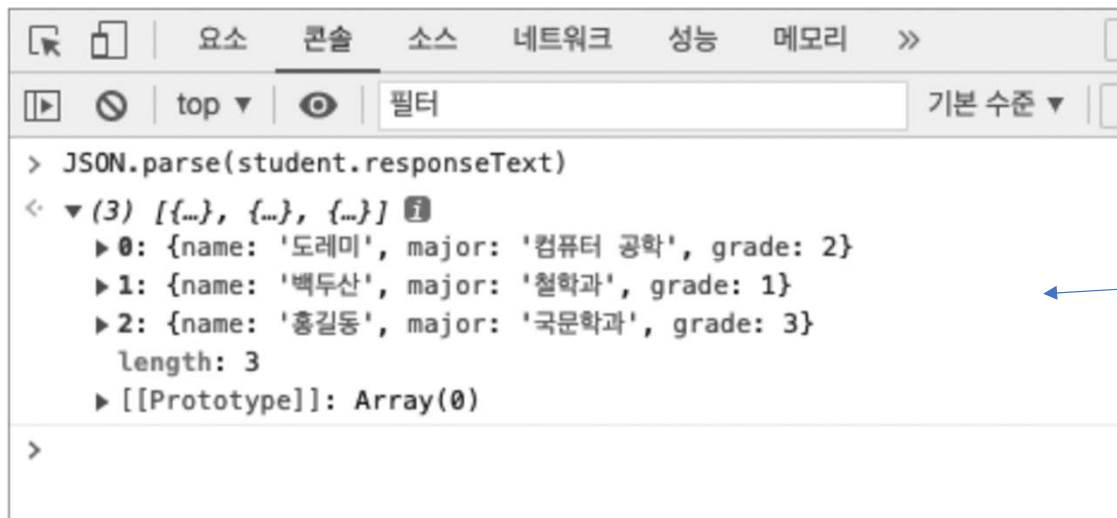
```
let xhr = new XMLHttpRequest();
xhr.open("GET", "student-2.json");
xhr.send();

xhr.onreadystatechange = function () {
  if (xhr.readyState === 4 && xhr.status === 200) {
    let students = JSON.parse(xhr.responseText);
    renderHTML(students);
  }
};
```

json 문자열이 여러 개인 정보를 가져오면 결국값에는 어떻게 정보가 저장될까?

콘솔 창에서 확인해 보자

```
JSON.parse(xhr.responseText)
```



students는 배열로 저장되어 있고
배열에 있는 객체에 순서대로 접근해서
내용을 가지고 오면 됩니다.

```
let xhr = new XMLHttpRequest();
xhr.open("GET", "student-2.json");
xhr.send();

xhr.onreadystatechange = function () {
    .....
}

function renderHTML(contents) {
    let htmlString = "";
    for (let content of contents) {
        htmlString += `
        <h2>${content.name}</h2>
        <ul>
            <li>전공: ${content.major}</li>
            <li>학년: ${content.grade}</li>
        </ul>
        <hr>
        `;
    }
    document.getElementById("result").innerHTML = htmlString;
}
```


student-2.html

```

<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>JSON</title>
  <link rel="stylesheet" href="css/student.css">
</head>
<body>
  <div id="container">
    <h1>수강생 명단</h1>
    <button type="button" onclick="reqJson()">Json 로드</button>
    <div id="result"></div>
  </div>

  <script>

    </script>
</body>
</html>

```

student-2.js

```

let xhr = new XMLHttpRequest();
function reqJson() {

  xhr.open("GET", "student-2.json");
  xhr.send();
}

xhr.onreadystatechange = function () {
  if (xhr.readyState === 4 && xhr.status === 200) {
    let students = JSON.parse(xhr.responseText);
    renderHTML(students);
  }
};
function renderHTML(contents) {
  let output = "";
  for (let content of contents) {
    output += `
    <h2>${content.name}</h2>
    <ul>
      <li>전공 : ${content.major}</li>
      <li>학년 : ${content.grade}</li>
    </ul>
    <hr>
  `;
  }
  document.getElementById("result").innerHTML = output;
}

```

수강생 명단

도레미

- 전공: 컴퓨터 공학
 - 학년: 2
-

백두산

- 전공: 철학과
 - 학년: 1
-

홍길동

- 전공: 국문학과
 - 학년: 3
-

예외 처리하기

예외 처리란

프로그램에서 문제가 발생하면 프로그램은 실행을 멈추기 때문에 소스를 작성할 때부터 발생할 만한 문제를 미리 고려하고 대비하는 것. (예외 처리, 에러 핸들링, 오류 처리 등으로 부른다)

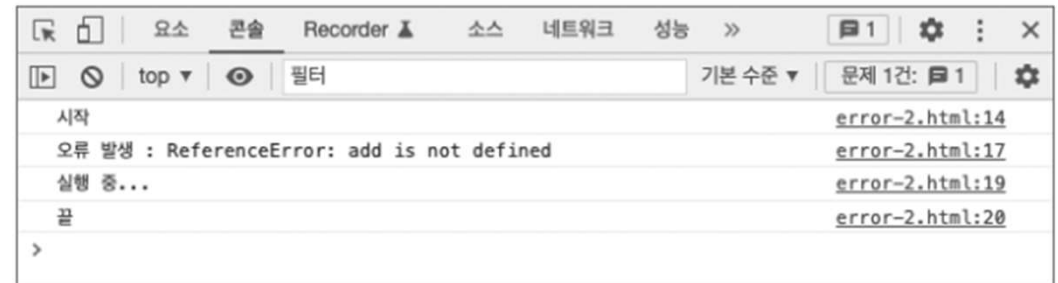
예외(exception)

- 문법적인 실수로 발생하는 오류
 - 문법 오류 뒤에 오는 소스는 실행되지 않아서 프로그램이 중단되기도 한다.
- 프로그램의 작성 의도와 다르게 사용했을 때 (예, 프롬프트 창에서 '취소' 버튼 클릭했을 때)
 - 이런 상황을 미리 예상해서 처리해 준다

try, catch, finally

```
기본형 try {
    // 실행할 코드
} catch (error) {
    // try 블록에서 예외가 발생했을 때 실행할 코드입니다.
} finally {
    // try 블록 이후에 실행할 코드. 예외와 상관없이 실행됩니다.
}
```

```
<script>
try {
  console.log("시작");
  add();
  console.log("실행 중...");
} catch(err) {
  console.log(`오류 발생 : ${err}`);
}
console.log("끝");
</script>
```



예외 처리를 사용해서 오류가 발생해도 멈추지 않는 프로그램

console.error()

콘솔 창에 오류를 표시한다

오류가 발생하면 빨간색으로 표시되기 때문에 일반 메시지와 구별된다

```
<script>
  try {
    console.log("시작");
    add();
    console.log("실행 중...");
  } catch(err) {
    console.log(`오류 발생 : ${err}`);
  }
  console.log("끝");
</script>
```

error 객체는 error.name과 error.message로 구성된다



```
<script>
  try {
    ...
  } catch(err) {
    console.error(`오류 발생 : ${err}`);
    console.error(`오류 발생 : ${err.name}`);
    console.error(`오류 발생 : ${err.message}`);
  }
  ...
</script>
```

throw()

사용자가 직접 예외를 만들고 오류 메시지를 지정한다.

기본형 throw 메시지

기본형 throw new Error(메시지)

(예) JSON 자료에 '이름' 정보가 없는데 '이름' 정보를 가져오려고 했을 때 오류 메시지를 표시하고 싶다면

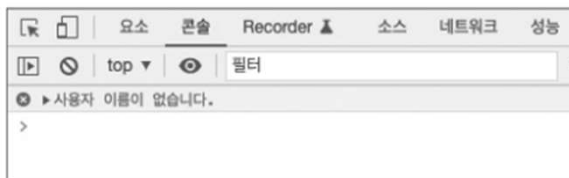
```
<script>
let json = '{"grade": 3, "age": 25}';

try {
  let user = JSON.parse(json);
  if (!user.name) {
    throw "사용자 이름이 없습니다.";
  }
} catch (err) {
  console.error(err);
}
</script>
```

또는

```
<script>
let json = '{"grade": 3, "age": 25}';

try {
  let user = JSON.parse(json);
  if (!user.name) {
    throw new Error("사용자 이름이 없습니다.");
  }
} catch (err) {
  console.error(err);
}
</script>
```



[실습] 서버에서 자료 가져오기

- 'jsonplaceholder.typicode.com'에 접속한 후
- [/users] 클릭

열 명의 사용자 정보가 들어 있는 JSON 자료
id와 name, username 등 여러 속성이 있다.

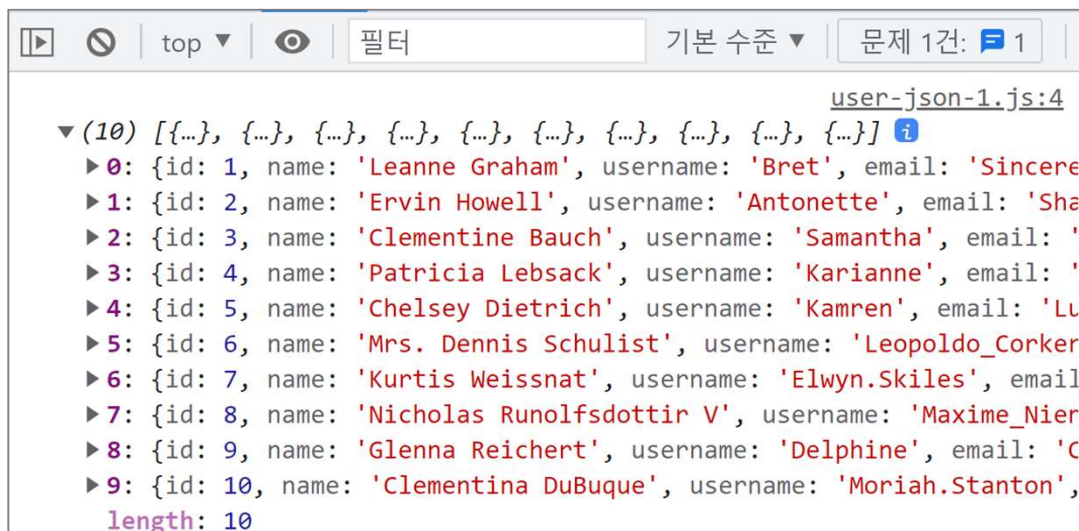


```
[{"id": 1, "name": "Leanne Graham", "username": "Bret", "email": "Sincere@april.biz", "address": {"street": "Kulas Light", "suite": "Apt. 556", "city": "Gwenborough", "zipcode": "92998-3874", "geo": {"lat": "-37.3159", "lng": "81.1496"}}, "phone": "1-770-736-8031 x56442", "website": "hildegard.org", "company": {"name": "Romaguera-Crona", "catchPhrase": "Multi-layered client-server neural-net", "bs": "harness real-time e-markets"}}, {"id": 2, "name": "Ervin Howell", "username": "Antonette", "email": "Shanna@melissa.tv", "address": {"street": "Victor Plains", "suite": "Suite 879", "city": "Wisokyburgh", "zipcode": "90566-7771", "geo": {"lat": "-43.9509", "lng": "-34.4618"}}, "phone": "010-692-6593 x09125", "website": "anastasia.net", "company": {"name": "Kovach-Walker", "catchPhrase": "Profoundly functional human-centric neural-net", "bs": "revolutionize end-to-end e-markets"}
```


fetch를 사용해 자료 가져오기

```
const url = 'https://jsonplaceholder.typicode.com/users';
fetch(url)
.then(response => response.json())
.then(users => console.log(users));
```

콘솔 창에서 확인하기



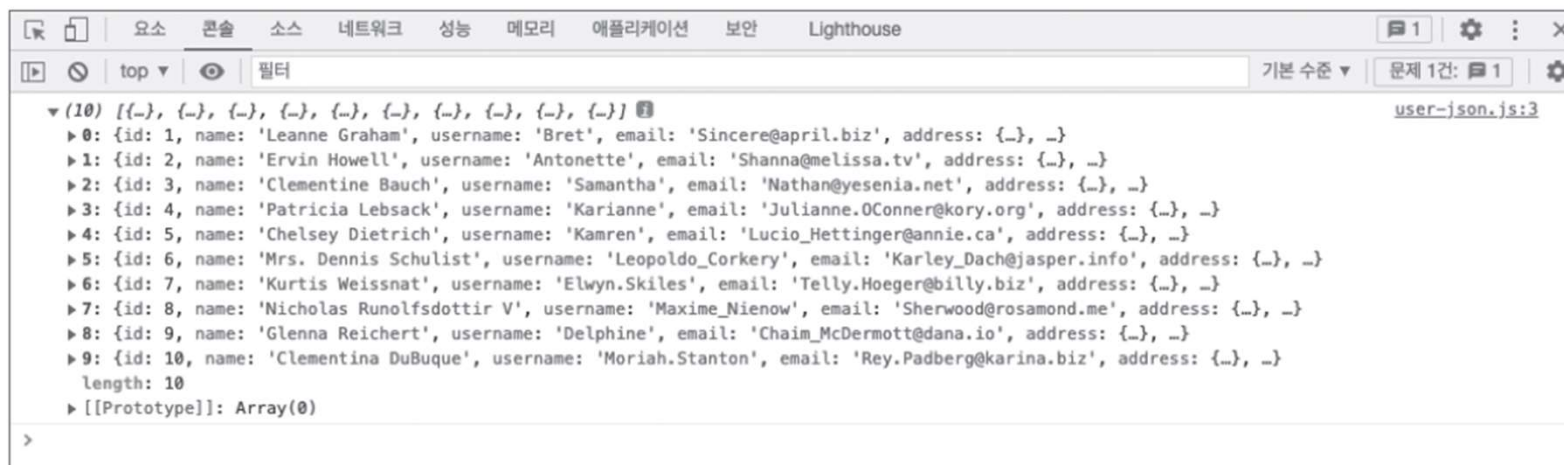
async와 await 사용하기

12. HTTP 통신과 JSON

```
const url = 'https://jsonplaceholder.typicode.com/users';
fetch(url)
  .then(response => response.json())
  .then(users =>
    async function init() {
      const response = await fetch(url);
      const users = await response.json();
      console.log(users);
    }
  );

init();
```

← fetch() 구문 삭제하기



가져온 자료를 웹 브라우저 창에 표시하기

12. HTTP 통신과 JSON

```
async function init() {  
    .....  
}  
  
function display(users) {  
    const result = document.querySelector("#result");  
    let string = "";  
    users.forEach(user => {  
        string += `<ul> <li>${user.name}</li>  
                <li>${user.username}</li>  
                <li>${user.email}</li> </ul>`;   
    });  
    result.innerHTML = string;  
}  
  
init();
```

회원 정보

- Leanne Graham
- Bret
- Sincere@april.biz
- Ervin Howell
- Antonette
- Shanna@melissa.tv
- Clementine Bauch
- Samantha
- Nathan@yesenia.net
- Patricia Lebsack
- Karianne
- Julianne.OConner@kory.org
- Chelsey Dietrich
- Kamren
- Lucio_Hettinger@annie.ca
- Mrs. Dennis Schulist
- Leopoldo_Corkery
- Karley_Dach@jasper.info
- Kurtis Weissnat
- Elwyn.Skiles
- Telly.Hoeger@billy.biz
- Nicholas Runolfsdottir V
- Maxime_Nienow
- Sherwood@rosamond.me

표 형태로 표시하려면?

회원 정보

이름	Leanne Graham
아이디	Bret
이메일	Sincere@april.biz

이름	Ervin Howell
아이디	Antonette
이메일	Shanna@melissa.tv

이름	Clementine Bauch
아이디	Samantha
이메일	Nathan@yesenia.net

이름	Patricia Lebsack
아이디	Karianne
이메일	Julianne.OConner@kory.org

이름	Chelsey Dietrich
아이디	Kamren
이메일	Lucio_Hettinger@annie.ca

이름	Mrs. Dennis Schulist
아이디	Leopoldo_Corkery
이메일	Karley_Dach@jasper.info

표 형태로 표시하려면?

1) 표 관련 태그를 사용해 화면에 표시

```
.....  
function display(users) {  
  const result = document.querySelector("#result");  
  let string = "";  
  users.forEach(user => {  
    string += `<table><tr><th>이름</th><td>${user.name}</td></tr>  
              <tr><th>아이디</th><td>${user.username}</td></tr>  
              <tr><th>이메일</th><td>${user.email}</td></tr></table>`;  
  });  
  result.innerHTML = string;  
}
```

표 형태로 표시하려면?

2) 표 관련 스타일 추가 (여기에서는 html 문서 안에)

```
<style>
  table {
    display:inline-block;
    width:300px;
    margin:10px;
  }
  table, td, th {
    border:1px solid #ccc;
    border-collapse:collapse;
  }
```

```
  th {
    width:80px;
  }
  td {
    width:210px;
    padding:10px 20px;
  }
</style>
```