

MEAN 스택 활용 웹 개발

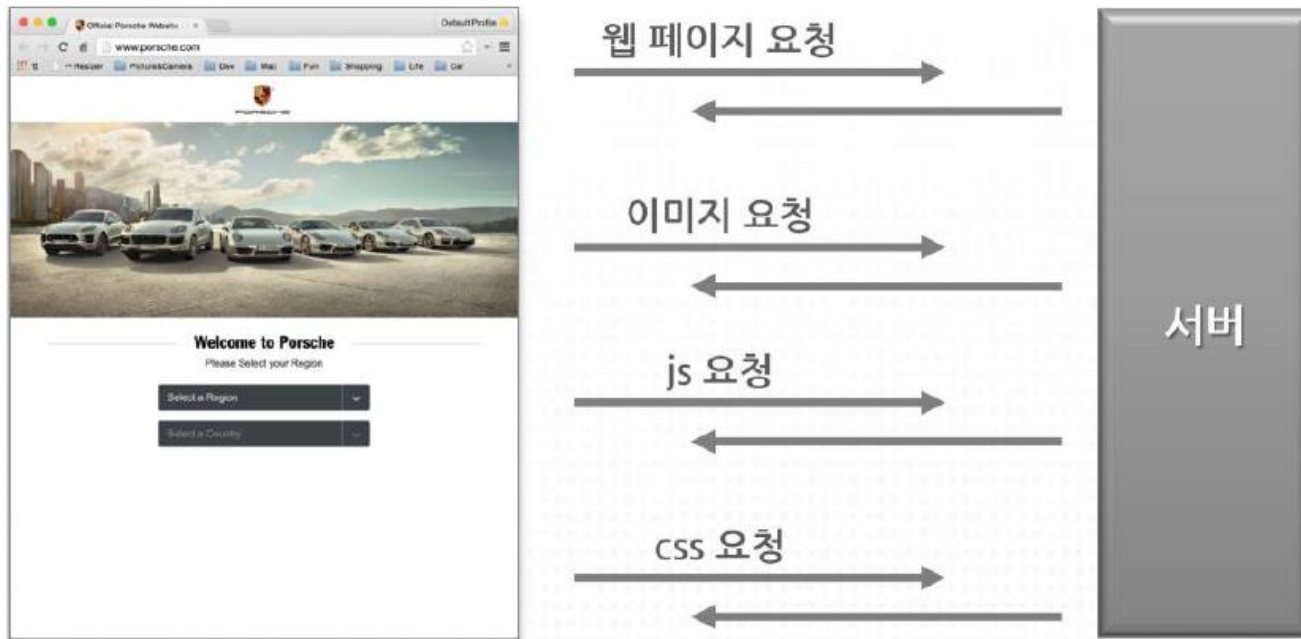
Node.js HTTP 통신 1



2016년도 2학기 2학년 방과후학교

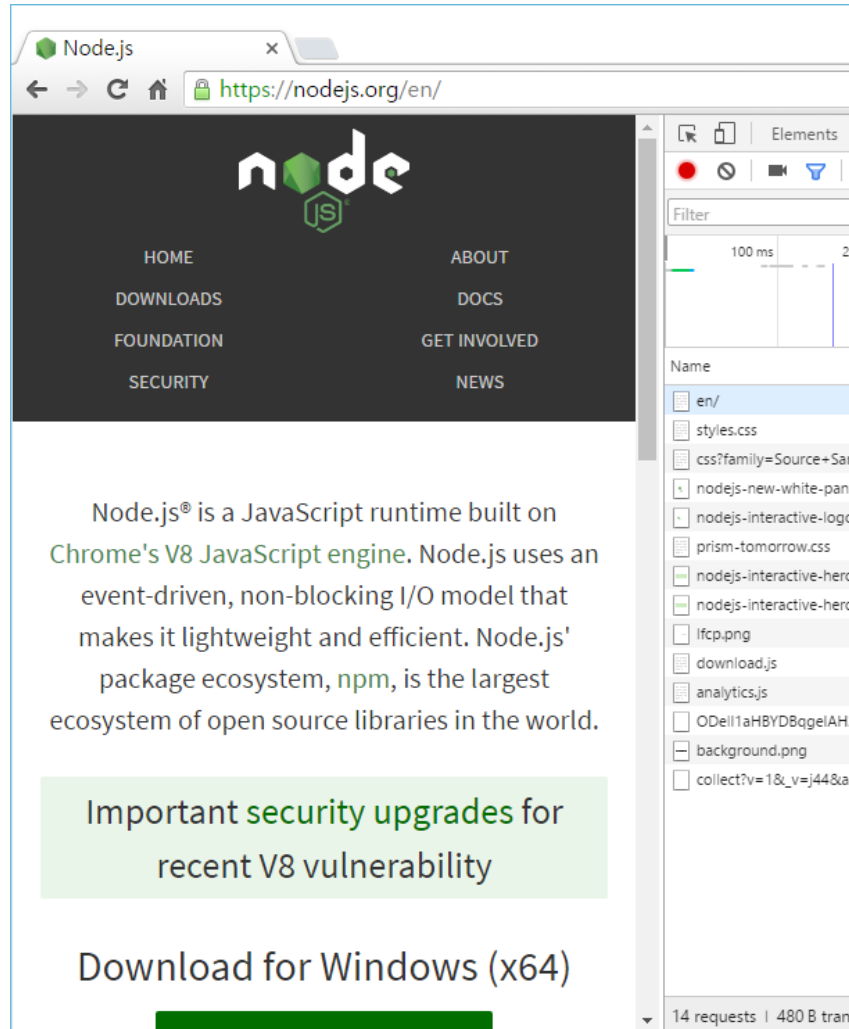
HTTP 통신 개요

- HTTP 통신은 요청(Request)과 응답(Response)으로 이루어진다.
- 클라이언트인 웹 브라우저에서 사용자가 주소를 입력하여 서버에 콘텐츠를 요청하는 요청 메시지를 보낸다.
- 서버는 요청 메시지를 분석하고, 사용자가 원하는 콘텐츠를 응답 메시지에 담아서 클라이언트에게 보낸다.



요청 응답 메시지 확인

■ 웹 브라우저의 개발자 도구 이용




The screenshot shows the Node.js website in a web browser. The browser's address bar displays `https://nodejs.org/en/`. The website features the Node.js logo and a navigation menu with links: HOME, ABOUT, DOWNLOADS, DOCS, FOUNDATION, GET INVOLVED, SECURITY, and NEWS. Below the menu, a paragraph describes Node.js as a JavaScript runtime built on Chrome's V8 JavaScript engine, highlighting its event-driven, non-blocking I/O model and the npm package ecosystem. A green box contains the text "Important security upgrades for recent V8 vulnerability". At the bottom, there is a link for "Download for Windows (x64)". The browser's developer tools are open on the right side, showing the "Elements" panel with a list of page resources and the "Network" panel at the bottom.

Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js' package ecosystem, npm, is the largest ecosystem of open source libraries in the world.

Important security upgrades for recent V8 vulnerability

Download for Windows (x64)



The screenshot shows the browser's developer tools with the "Network" tab selected. It displays the details of a request to `https://nodejs.org/en/`. The "General" section shows the request method as GET, status code as 304 (Not Modified), and remote address as 104.20.22.46:443. The "Response Headers" section lists various headers including Cache-Control, CF-Cache-Status, CF-RAY, Connection, Date, ETag, Expires, Last-Modified, Server, and Vary. The "Request Headers" section lists headers such as Accept, Accept-Encoding, Accept-Language, Cache-Control, Connection, Cookie, Host, If-Modified-Since, Upgrade-Insecure-Requests, and User-Agent. The bottom of the network panel shows a summary of the request: 14 requests, 480 B transferred, Finish: 254 ms, DOMContentLoaded: 145 ms, Load: 259 ms.

Headers Preview Response Cookies Timing

▼ General

Request URL: `https://nodejs.org/en/`
Request Method: GET
Status Code: 304 Not Modified
Remote Address: 104.20.22.46:443

▼ Response Headers view source

Cache-Control: public, max-age=14400
CF-Cache-Status: HIT
CF-RAY: 2c9ce8d3d0bb3a4e-ICN
Connection: keep-alive
Date: Fri, 29 Jul 2016 01:48:35 GMT
ETag: "57938051-bb7"
Expires: Fri, 29 Jul 2016 05:48:35 GMT
Last-Modified: Sat, 23 Jul 2016 14:33:53 GMT
Server: cloudflare-nginx
Vary: Accept-Encoding

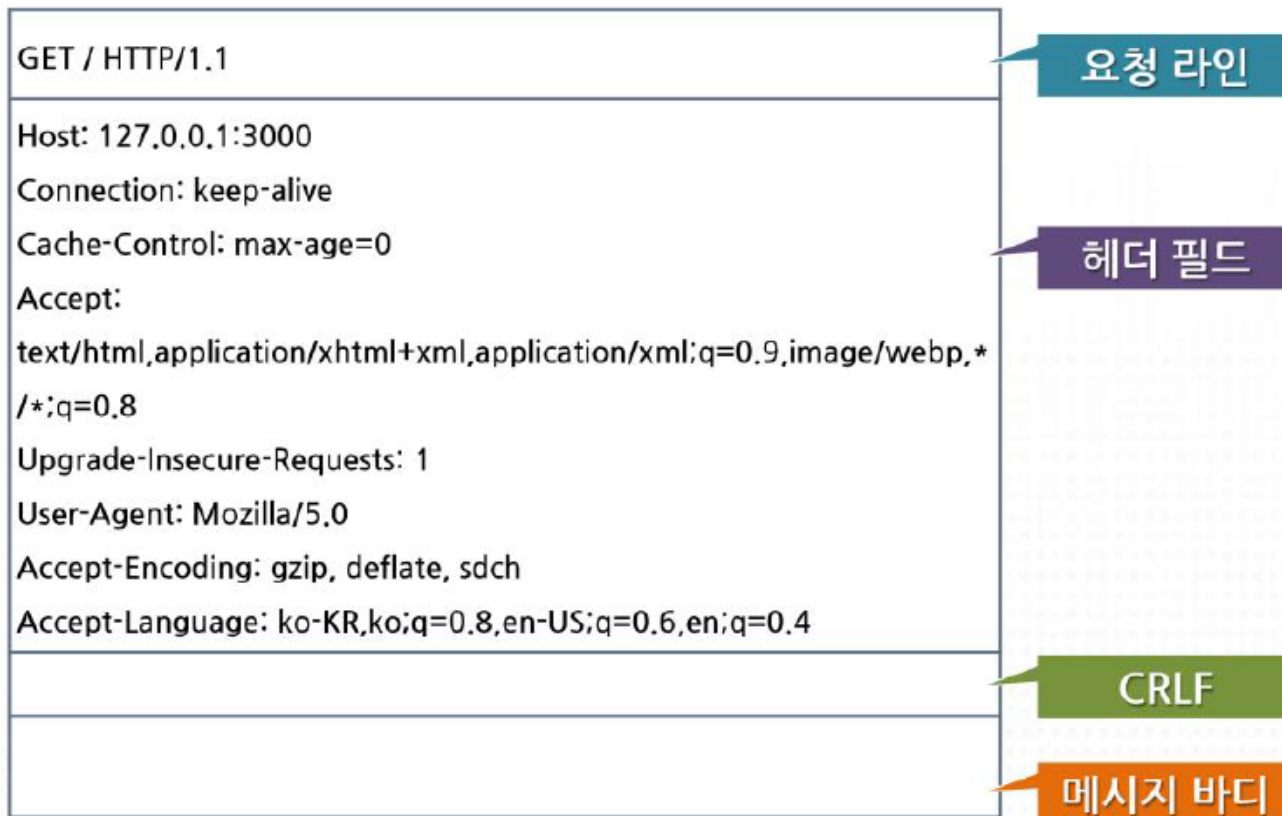
▼ Request Headers view source

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Encoding: gzip, deflate, sdch, br
Accept-Language: ko-KR,ko;q=0.8,en-US;q=0.6,en;q=0.4
Cache-Control: max-age=0
Connection: keep-alive
Cookie: __cfduid=d1e8eb699556d6ed19074e89400a64e141468223736; _ga=GA1.2.484900426.1468223738; _gat=1
Host: nodejs.org
If-Modified-Since: Sat, 23 Jul 2016 14:33:53 GMT
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/52.0.2743.82 Safari/537.36

14 requests | 480 B transferred | Finish: 254 ms | DOMContentLoaded: 145 ms | Load: 259 ms

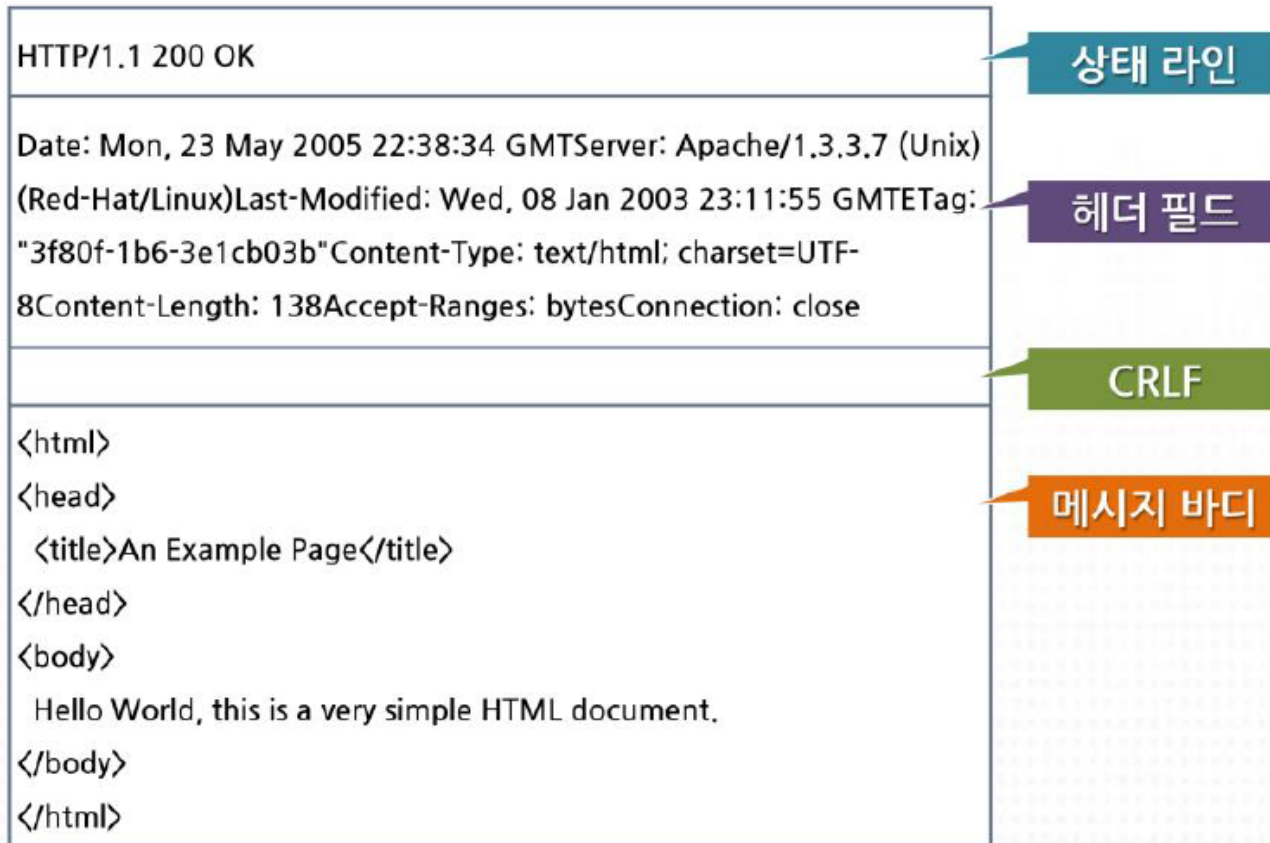
HTTP 요청 메시지 구조

- 요청 메시지 : 요청 라인, 요청 헤더, CRLF, 요청 바디
- 요청 라인 : 요청 메소드 / HTTP 버전



HTTP 응답 메시지 구조

- 응답 메시지 : 응답(상태) 라인, 응답 헤더, CRLF, 응답 바디
- 응답 라인 : HTTP 버전 / 상태코드 / 상태 메시지



HTTP 메소드 종류

- HTTP 요청 메소드는 리소스를 다루는 행위를 동사 형태로 표현한다.
- HTTP 메소드 종류
 - ✓ GET : 일반적으로 웹 브라우저에서 콘텐츠를 요청하는 메소드
 - ✓ POST : 웹 브라우저에서 글을 쓰거나 사진을 올리는 요청 시 사용
전송하는 데이터를 서버에서 처리할 것을 요청하는 메소드
 - ✓ PUT : 전송하는 데이터를 이용해서 저장하거나 수정하는 것을 요청하는
메소드
 - ✓ DELETE : 삭제를 요청하는 메소드

HTTP 헤더 필드

- HTTP 헤더 필드는 키 : 값 방식으로 작성된다.

▼ Request Headers [view source](#)

```
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/
Accept-Encoding: gzip, deflate, sdch
Accept-Language: ko-KR,ko;q=0.8,en-US;q=0.6,en;q=0.4
Connection: keep-alive
Cookie: __cfduid=d5ca8d26a8e5c08f263134356cf0657a31447650175; _ga=GA1
49
Host: dimigo.in
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64;
hrome/52.0.2743.82 Safari/537.36
```

▼ Response Headers [view source](#)

```
Cache-Control: pre-check=0, post-check=0, max-age=0
CF-RAY: 2c9cf2bafc0809d6-ICN
Connection: keep-alive
Content-Encoding: gzip
Content-Type: text/html; charset=utf-8
Date: Fri, 29 Jul 2016 01:55:21 GMT
Expires: 0
Last-Modified: Fri, 29 Jul 2016 01:55:21 GMT
P3P: CP="ALL CURa ADMa DEVa TAIa OUR BUS IND PHY ONL UNI PUR FIN COM NAV IN
POL HEA PRE LOC OTC"
Pragma: no-cache
Server: cloudflare-nginx
Set-Cookie: 2a0d2363701f23f8a75028924a3af643=MTQxLjEwMS44NS4xNDE%3D; expires
016 01:55:21 GMT; Max-Age=86400; path=/; domain=.dimigo.in
Set-Cookie: PHPSESSID=07bt4c53tm1o5rb4bv6istqg06; path=/; domain=.dimigo.in
Transfer-Encoding: chunked
X-Powered-By: PHP/5.6.18
```

요청 정보 전달 방식 (1/4)

- URL을 이용한 요청 정보 전달
- 사용 메소드 : GET, TRACE 메소드
- 특징 : 메시지 바디를 사용하지 않고, URL 경로와 쿼리 스트링을 사용한다.
예) `http://idols.com/q?group=bigbang&member=GD`
- url 모듈을 사용하면 쿼리 스트링 정보를 쉽게 얻어올 수 있다.

요청 정보 전달 방식 (2/4)

- URLEncoded 방식을 이용한 요청 정보 전달
- 사용 메소드 : POST, PUT 메소드
- 특징 : 메시지 바디를 사용하여 요청 정보를 전달한다.
- 메시지 헤더 (Content-Type) : application/x-www-form-urlencoded
- urlencoded 방식은 이름=값 형태로 데이터를 바디에 작성한다.
- querystring 모듈을 사용하여 바디 데이터 정보를 쉽게 얻어올 수 있다.
예) Content-Type : application/x-www-form-urlencoded
group=bigbang&member=GD

요청 정보 전달 방식 (3/4)

- JSON/XML을 이용한 요청 정보 전달
- 메시지 헤더 (Content-Type) : application/json, application/xml
- 특징 : RESTFUL API에서 많이 사용한다.

예) Content-Type : application/json

```
{  
    "group" : "bigbang",  
    "member" : ["GD", "TOP", "태양", "대성", "승리"]  
}
```

요청 정보 전달 방식 (4/4)

- 멀티 파트(Multipart)를 이용한 요청 정보 전달
 - ✓ 네트워크를 이용해서 바이너리 데이터를 전송할 때 주로 사용한다.
 - ✓ 하나의 메시지 바디에 여러 개의 데이터를 파트로 구분해서 작성한다.
- 메시지 헤더 (Content-Type) : multipart/form-data; boundary=xxxyyyzzz
- 특징 : 메시지에는 각 파트를 구별하기 위한 구분자를 정의한다.

```
--xxxyyyzzz
```

```
Content-Type: text/plain
```

```
This is the body of the message.
```

```
--xxxyyyzzz
```

```
Content-Type: application/octet-stream
```

```
Content-Transfer-Encoding: base64
```

```
PGh0bWw+CiAgPGhlYWQ+CiAgPC9oZWFrPgogIDxib2R5PgogICAgPHA+VGhpcyBpcyB0aGUg  
Ym9keSBvZiB0aGUgbWVzc2FnZS48L3A+CiAgPC9ib2R5Pgo8L2h0bWw+Cg==
```

```
--xxxyyyzzz-
```

상태 코드 종류

- 응답 코드와 응답 메시지

구분	응답 코드	응답 메시지
1xx : Information	100	Continue
2xx : Success	200	OK
	201	Created
	202	Accepted
3xx : Redirection	301	Moved Permanently
	302	Found
4xx : Client Error	400	Bad Request
	401	Unauthorized
	403	Forbidden
	404	Not Found
5xx : Server Error	500	Internal Server Error

응답 메시지

- 응답 메시지의 Content-Type : 대분류/소분류
 - ✓ text/plain, text/html
 - ✓ application/xml, application/json
 - ✓ image/png, image/jpeg
 - ✓ audio/mp3, video/mp4
- 메시지의 바디에 담긴 내용(이미지)과 Content-Type(text/html)이 다르면 정상적으로 메시지를 해석할 수 없다.

HTTP 모듈

- HTTP 모듈은 HTTP 통신을 다루는 기능을 제공하는 기본모듈이다.
- `var http = require('http');`
- http 모듈은 HTTP 서버를 생성하거나 HTTP클라이언트를 작성할 수 있는 기능을 제공한다. 서버나 클라이언트 모두 HTTP 요청 메시지와 응답 메시지를 다뤄야 한다.

HTTP 서버

- HTTP 서버 생성
 - ✓ `var server = http.createServer([requestListener]);`
 - ✓ `requestListener` : 자동으로 추가되는 request 이벤트의 핸들러 함수
- `http.Server`의 주요 이벤트
 - ✓ `request` : 클라이언트의 요청 메시지 도착
 - ✓ `connection` : 소켓 연결 이벤트
 - ✓ `close` : 서버 연결 종료, 추가 요청 받지 않음
- Class: `http.Server`
 - Event: 'checkContinue'
 - Event: 'clientError'
 - Event: 'close'
 - Event: 'connect'
 - Event: 'connection'
 - Event: 'request'
 - Event: 'upgrade'
 - `server.close([callback])`
 - `server.listen(handle[, callback])`
 - `server.listen(path[, callback])`
 - `server.listen(port[, hostname][, backlog][, callback])`
 - `server.maxHeadersCount`
 - `server.setTimeout(msecs, callback)`
 - `server.timeout`

HTTP 서버

- HTTP 서버 주요 메소드

- ✓ `listen()` : 서버의 네트워크 포트와 결합해서, 해당 포트에 전달되는 요청에 반응한다.
- ✓ `close()` : 추가적인 클라이언트 요청을 더 이상 받지 않는다
- ✓ `setTimeout()` : 요청을 처리하는 제한 시간을 설정한다.

- Class: `http.Server`
 - Event: 'checkContinue'
 - Event: 'clientError'
 - Event: 'close'
 - Event: 'connect'
 - Event: 'connection'
 - Event: 'request'
 - Event: 'upgrade'
 - `server.close([callback])`
 - `server.listen(handle[, callback])`
 - `server.listen(path[, callback])`
 - `server.listen(port[, hostname][, backlog][, callback])`
 - `server.maxHeadersCount`
 - `server.setTimeout(msecs, callback)`
 - `server.timeout`

HTTP 서버 동작

- 서버 객체 생성 후 listen 함수를 이용하여 클라이언트 접속을 대기한다.
- 서버에서 사용할 포트, 즉 클라이언트가 접속할 포트를 listen 함수의 파라미터로 입력한다.

```
var http = require('http');  
// 서버 객체 생성  
var server = http.createServer(function(req, res) {  
  res.end('<h1>Hello World</h1>');  
});  
// 특정 포트 listen  
server.listen(3000);
```

HTTP Request 처리

- 클라이언트가 요청 메시지를 보내면 request 이벤트가 발생하고 request 이벤트 리스너 함수가 동작한다.
- req라는 이름의 요청 객체의 타입은 IncomingMessage 이다.
 - ✓ req.url : 요청 URL
 - ✓ req.method : 요청 메소드
 - ✓ req.headers : 요청 메시지 헤더

HTTP Request 처리

- url 모듈을 이용하여 요청 URL 경로와 쿼리 문자열 분석하기

```
var url = require('url');  
var parsed = url.parse(req.url, true);  
var query = parsed.query; // true로 설정한 경우  
  
// http://127.0.0.1:3000/idol?group=bigbang&member=GD  
console.log(query); // { group: 'bigbang', member: 'GD' }
```

HTTP Response 처리

- 메시지 상태

- ✓ `res.statusCode` : 응답 코드
- ✓ `res.statusMessage` : 응답 메시지

- 메시지 헤더

- ✓ `res.writeHead(statusCode[, statusMessage][, headers])`
- ✓ `res.removeHeader(name)`
- ✓ `res.getHeader(name)`
- ✓ `res.setHeader(name, value)`

- 메시지 바디

- ✓ `res.write(chunk[, encoding][, callback])`
- ✓ `res.end([data][, encoding][, callback])`

[실습] HTTP 통신 기본

- URL 모듈을 이용해서 쿼리스트링으로 넘어온 start, end 값의 합을 출력하도록 해보자!! (<http://127.0.0.1:3000/cal?start=1&end=10>)
- 둘 중 하나라도 입력값이 없거나, start가 end보다 크면 에러 처리
- 파일명 : 14_httpWhttpRequest2.js

