

# 스타트업 개발자와 함께 공부하는 Node.js

00. 제목

강의 내용은 강사가 별도로 명시하지 않는 한 비공개로 간주합니다.  
녹음이나 사진 촬영을 허락하지 않으며 콘텐츠를 블로그, SNS 등에 게시하거나 공개적으로 공유하지 마세요.

콘텐츠 공유 가능 여부에 대해 궁금한 점이 있는 경우 강사에게 문의하시기 바랍니다.



# 목차

1. HTTP
2. RESTful API
3. curl
4. postman
5. 공지사항 API로 변경



# HTTP

# HTTP

## 개요

- 하이퍼텍스트 전송 프로토콜(Hypertext Transfer Protocol)
- 웹에서 데이터를 주고 받는 데 사용
- 요청 - 응답 방식
- TCP / IP 기반

# HTTP

## 개요

### □ 주요 특징

1. 비연결성 (Connectionless)
2. 무상태성 (Stateless)

### □ HTTP 요청 메서드

1. GET
2. POST
3. PUT
4. DELETE
5. PATCH

# HTTP

## 개요

### □ HTTP 상태 코드

1. 1xx (Informational)
2. 2xx (Success)
3. 3xx (Redirection)
4. 4xx (Client Error)
5. 5xx (Server Error)

### □ HTTP 헤더

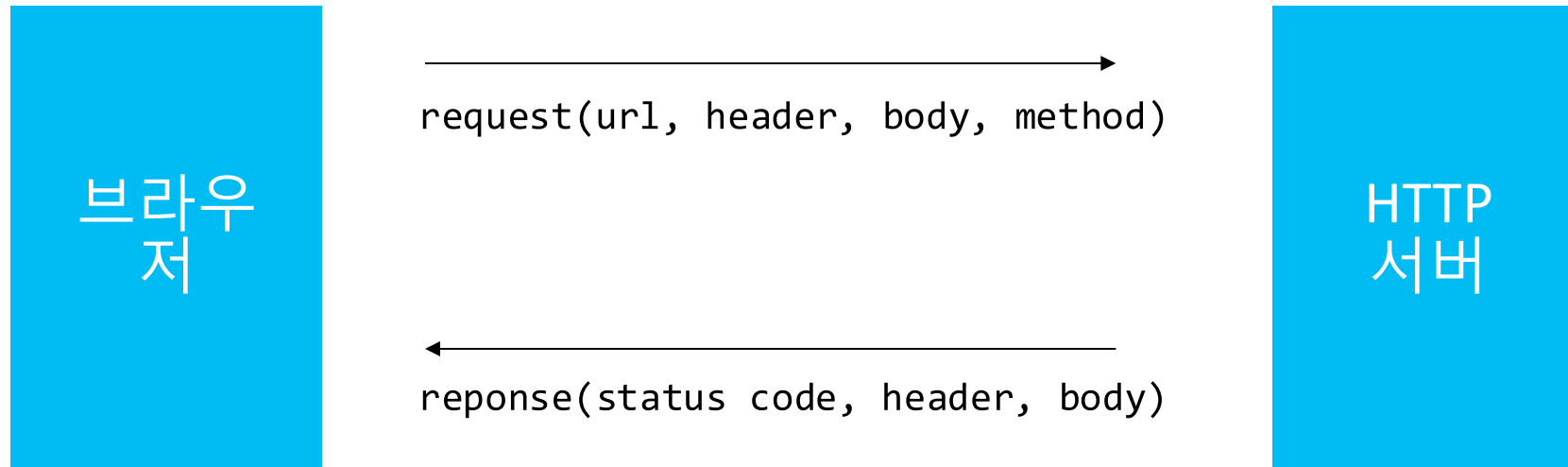
1. 요청 헤더
2. 응답 헤더

### □ 주요 헤더 정보

1. Content-Type
2. Authorization
3. Cookie

# HTTP

## 동작방식





# STful API

# RESTful API

## 개요

- REST(Representational State Transfer) 아키텍처
- 원칙

HTTP URL을 통해 자원을 명시하고  
HTTP 메서드를 통해 해당 자원에 대한  
CRUD Operation을 적용

# RESTful API

## 설계 규칙

- ❑ 슬래시(/) 구분자는 계층 관계를 나타냄
- ❑ CRUD에는 동사를 사용하지 않음
- ❑ 적절한 HTTP 메소드 사용
- ❑ 소문자 사용
- ❑ URI 에 확장자 사용 안함
- ❑ 명사에는 단수형보다 복수형 사용

# RESTful API

## 특징

### □ 자원 기반

1. 웹 서비스를 자원(Resource)으로 모델링
2. 자원은 URI를 통해 고유하게 식별

**`https://api.github.com/users/123`**

# RESTful API

## 특징

### □ HTTP 메서드 사용

- **GET: 자원의 상태나 데이터 조회**
- **POST: 새로운 자원 생성**
- **PUT: 기존 자원 업데이트**
- **DELETE: 자원 삭제**
- **PATCH: 자원 일부 업데이트**

# RESTful API

## 특징

### □ 표현(Representation)

1. 자원은 JSON, XML 등 다양한 형식으로 표현
2. 클라이언트는 서버에 자원 형식 요청 하고, 서버는 적절한 자원으로 전달

```
{  
  "id": 123,  
  "name": "John Doe",  
  "email": "john.doe@example.com"  
}
```

# RESTful API

예시

GET /users

GET /users/123

POST /users

PUT /users/123

DELETE /users/123

curl

```
function todoitem(data) {
  var self = this
  data = data || {}
  self.persistedProperties = {
    id: data.id,
    title: data.title,
    completed: data.completed
  }
  self.observe()
}

function todoitem(data) {
  var self = this
  data = data || {}
  self.persistedProperties = {
    id: data.id,
    title: data.title,
    completed: data.completed
  }
  self.observe()
}

// Sea - persisted properties function
function todoitem(data) {
  var self = this
  data = data || {}
  self.persistedProperties = {
    id: data.id,
    title: data.title,
    completed: data.completed
  }
  self.observe()
}
```



# curl

## 개요

□ 명령줄(Command line)에서 HTTP 요청을 보내고 서버 응답을 확인하는 도구

1. 설치 : 아래 주소에서 프로그램 다운로드

**`https://curl.se/windows/`**

2. 환경변수 path 에 설치 경로 등록
3. 파워 쉘 또는 도스 창을 열고 아래 명령어로 설치 확인

**`curl --version`**

# curl

## 기본 문법

- ❑ curl [옵션] [URL]
- ❑ curl <http://github.com>
- ❑ curl -X POST "param=value" <http://github.com>
- ❑ curl -X POST -H "Content-Type: application/json" -d '{"k":"v"}
- ❑ curl -X PUT -d "param=value" <http://github.com>
- ❑ curl -X DELETE http://github.com/resource/1
- ❑ curl -H "Authorization: Bearer <token>" <http://github.com/protected>

# curl

## 연습

```
// 06/ch06_01.curl
```

```
1 curl http://jsonplaceholder.typicode.com/posts
2
3 curl -X POST -H "Content-Type: application/json" -d '{"title":"foo","body":"bar","userId":1}' \
4 http://jsonplaceholder.typicode.com/posts
5
6 curl -X PUT -H "Content-Type: application/json" -d '{"id":1,"title":"foo","body":"bar","userId":1}' \
7 http://jsonplaceholder.typicode.com/posts/1
8
9 curl -X DELETE http://jsonplaceholder.typicode.com/posts/1
10
11 curl -H "Authorization: Bearer <token>" http://jsonplaceholder.typicode.com/protected
12
```

# Postman



# Postman

## 개요

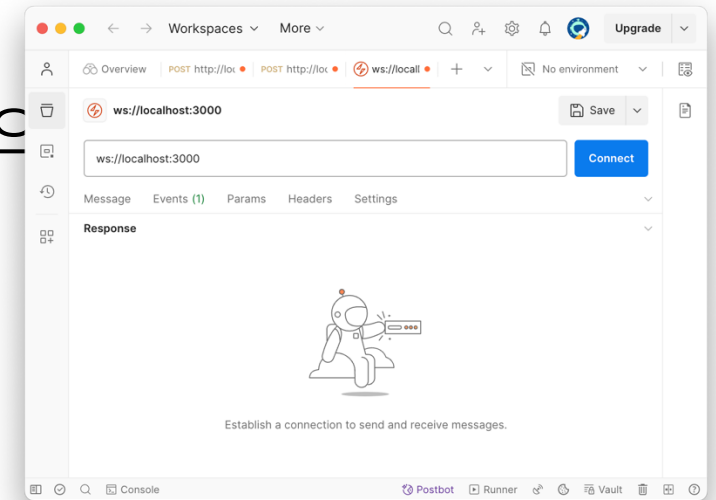
□ GUI 환경에서 HTTP 요청을 보내고 서버 응답을 확인하는 도구

1. 설치 : 아래 주소에서 프로그램 다운로드

**<https://www.postman.com/downloads/>**

2. 다운로드 받은 파일을 설치

3. 동작화면 및 로그인 화면에서 계정 생성 또는 구글 계정의



# Postman

## 연습

❑ 연습 URL : <http://jsonplaceholder.typicode.com/posts>

❑ GET

❑ POST

❑ PUT

❑ DELETE

❑ Authorization header



# 공지사항 게시판

## API

# 공지사항 게시판 API

## 프로젝트 생성

```
npm init -y  
npm install express nodemon
```

1. [C]-[nodejs]-[project]-[06]-[ch06\_02]  
디렉토리 생성
2. 왼쪽 코드 입력 하여 프로젝트 생성



# 공지사항 게시판 API

## 목록 가져오기

```
1 const express = require('express');
2 const fs = require('fs');
3
4 var app = express();
5
6 app.use(express.json());
7
8 app.get('/list', (req, res) => {
9     const result = fs.readFileSync('test.json', 'utf-8');
10    const data = JSON.parse(result);
11
12    res.json(data);
13 });
```

# 공지사항 게시판 API

## 공지사항 상세 가져오기

```
15 app.get('/detail/:id', (req, res) => {  
16   const id = req.params.id;  
17  
18   const result = fs.readFileSync('test.json', 'utf-8');  
19   const data = JSON.parse(result);  
20   let detail = {};  
21  
22   data['result'].forEach((item) => {  
23     if(item['id'] == id) {  
24       detail = item;  
25     }  
26   });  
27   res.json({detail:detail})  
28 });
```

# 공지사항 게시판 API

## 공지사항 쓰기

```
31 app.post('/write', (req, res) => {
32   console.log(req.body);
33   const result = fs.readFileSync('test.json', 'utf-8');
34   let data = JSON.parse(result);
35
36   const last = data['result'].slice(-1);
37   const last_id = last[0]['id'] + 1;
38   const write_date = moment().format('YYYY-MM-DD');
39   data['result'].push({
40     'id': last_id, 'title': req.body.title, 'content': req.body.content,
41     'writer': 'tester' , 'write_date': write_date
42   });
43
44   fs.writeFileSync('test.json', JSON.stringify(data), 'utf-8');
45   res.redirect('/list');
46 });
```

# 공지사항 게시판 API

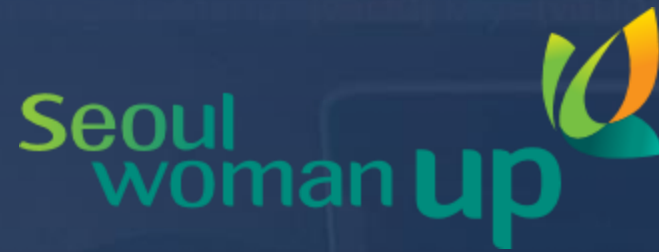
## 공지사항 수정

```
49 app.put('/update/:id', (req, res) => {
50   const id = req.params.id;
51
52   const result = fs.readFileSync('test.json', 'utf-8');
53   let data = JSON.parse(result);
54
55   for(item of data['result']) {
56     if(item['id'] == id) {
57       item['title'] = req.body.title;
58       item['content'] = req.body.content;
59     }
60   }
61
62   fs.writeFileSync('test.json', JSON.stringify(data), 'utf-8');
63   res.redirect('/detail/'+id)
64 });
```

# 공지사항 게시판 API

## 공지사항 삭제

```
66 app.delete('/delete/:id', (req, res) => {
67     const id = req.params.id;
68
69     const result = fs.readFileSync('test.json', 'utf-8');
70     let data = JSON.parse(result);
71
72     let element_idx = 0;
73
74     data['result'].forEach((e, i) => {
75         if(e['id'] == id) {
76             element_idx = i;
77         }
78     });
79     data['result'].splice(element_idx, 1);
80
81     fs.writeFileSync('test.json', JSON.stringify(data), 'utf-8');
82     res.redirect('/list');
83 });
```



백문이 불여일타