



Node-Red를 활용한 데이터 수집 및 가공

빅데이터 오케스트레이션 및 시각화 실습 – 3일차

# 학습 내용

1. 자바스크립트를 이용한 데이터 가공
2. Node-Red 설치 및 셋팅
3. Node-Red 노드 작동원리
4. 노드 편집을 통한 REST API 개발
5. Open API 연결
6. 데이터베이스 설치



# 자바스크립트를 이용한 데이터 가공

## \_데이터 가공 실습

실습 문제 : 학생들 성적 데이터에 대해 총점 및 평균, 석차를 학생들 마다 입력하시오

```
let record_arr = [  
  { name: "kim", kor: 100, math: 60, eng: 80 },  
  { name: "lee", kor: 80, math: 55, eng: 85 },  
  { name: "park", kor: 90, math: 65, eng: 88 },  
  { name: "mike", kor: 60, math: 75, eng: 81 },  
  { name: "sam", kor: 90, math: 85, eng: 84 },  
];
```

```
console.log("성적산출 ", record_arr);
```



## \_데이터 가공 실습

**실습 문제** : 다음 학생들의 성적을 과목별로 재정리 하시오.

```
let record_arr = [  
  { name: "kim", kor: 100, math: 60, eng: 80 },  
  { name: "lee", kor: 80, math: 55, eng: 85 },  
  { name: "park", kor: 90, math: 65, eng: 88 },  
  { name: "mike", kor: 60, math: 75, eng: 81 },  
  { name: "sam", kor: 90, math: 85, eng: 84 },  
];
```



```
record_arr2 = [  
  {subjectname: 'kor', kim: 100, lee: 80, park: 90, mike: 60, sam:90}  
  {subjectname: 'math', kim: 60, lee: 55, park: 65, mike: 75, sam:85},  
  {subjectname: 'eng', kim: 80, lee: 85, park: 88, mike: 81, sam:84},  
];
```



## AJAX JQuery 코드

```
<script src="https://code.jquery.com/jquery-1.12.4.min.js"></script>
```

```
$.ajax({  
    type : "GET",                                //전송방식 (POST,GET)  
    url: "http://dev2.soxcorp.co.kr:17104/mytest", //호출 URL  
    data: { name: "soxcorp", age: 4 },           // 파라미터 지정  
    dataType: "json",                            // 페이지 형식  
    success: function (data) {  
        console.log(data);                       // 받은 값  
    },  
    error: function () {  
        alert("통신실패!!!!");  
    }  
});
```



---

# Node-Red

---

# \_Node-Red 설치

## 1. Node JS 설치

<https://nodejs.org/ko/>

`node --version && npm --version`



Node.js®는 Chrome V8 JavaScript 엔진으로 빌드된 JavaScript 런타임입니다.

다운로드 - Windows (x64)

16.17.0 LTS

안정적, 신뢰도 높음

18.9.0 현재 버전

최신 기능

[다른 운영 체제](#) | [변경사항](#) | [API 문서](#)

[다른 운영 체제](#) | [변경사항](#) | [API 문서](#)

LTS 일정은 [여기서](#) 확인하세요

## 2. Node-Red 설치 및 실행

`npm install -g --unsafe-perm node-red`

```
관리자: C:\Windows\system32\cmd.exe
C:\Users\mailt\Desktop> node --version && npm --version
v16.14.2
8.4.1
C:\Users\mailt\Desktop> node-red
'node-red'은(는) 내부 또는 외부 명령, 실행할 수 있는 프로그램, 또는
배치 파일이 아닙니다.
C:\Users\mailt\Desktop> npm install -g --unsafe-perm node-red
added 292 packages, and audited 293 packages in 22s
38 packages are looking for funding
  run `npm fund` for details
5 low severity vulnerabilities
To address issues that do not require attention, run:
  npm audit fix
To address all issues (including breaking changes), run:
  npm audit fix --force
Run `npm audit` for details.
C:\Users\mailt\Desktop>
```

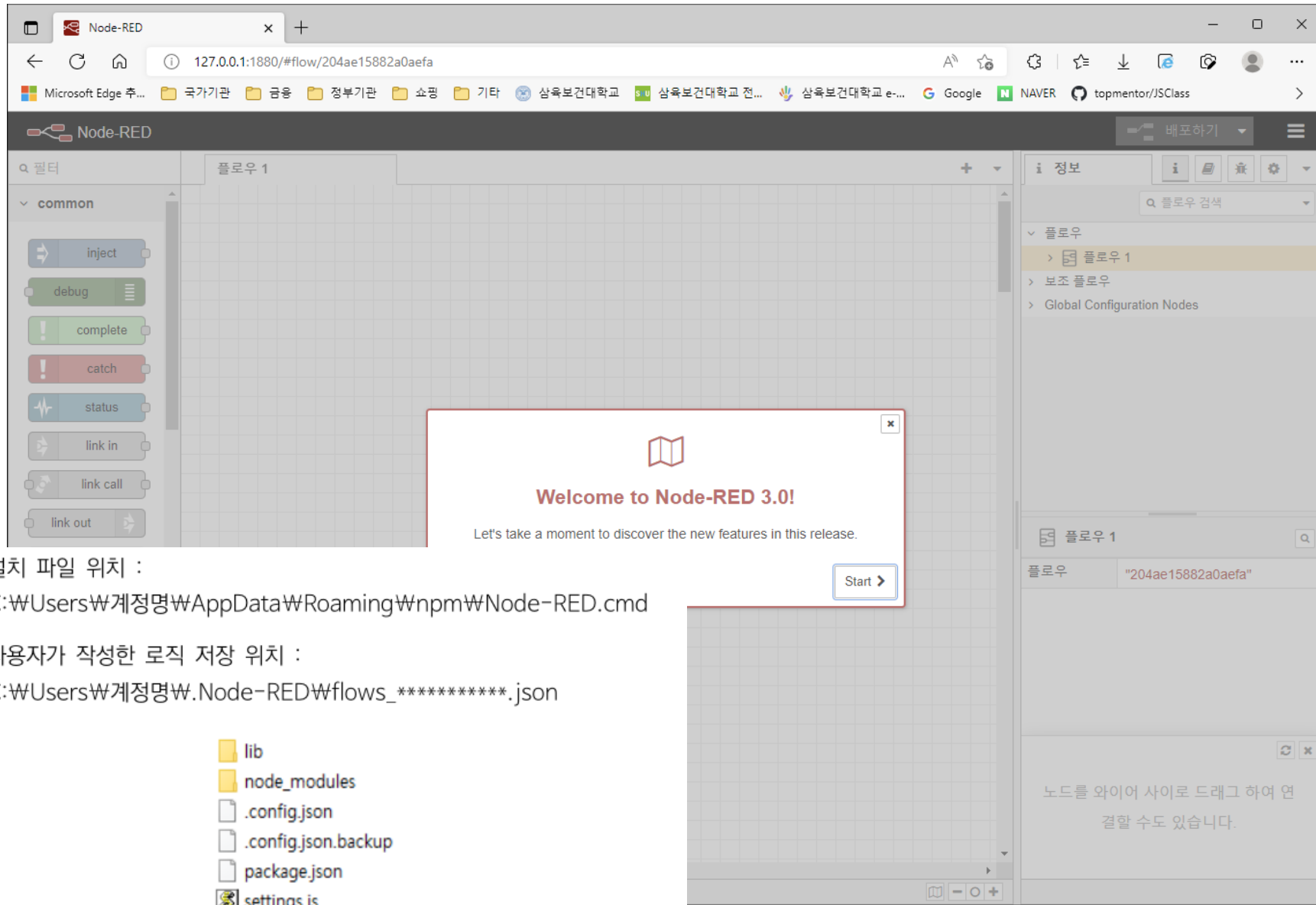
node-red

```
node-red
12 Sep 11:08:32 - [info] Server now running at http://127.0.0.1:1880/
12 Sep 11:08:32 - [warn] Encrypted credentials not found
12 Sep 11:08:32 - [info] Starting flows
12 Sep 11:08:32 - [info] Started flows
```





# \_Node-Red



The screenshot shows the Node-RED 3.0.0 web interface in a Microsoft Edge browser. The address bar shows the URL `127.0.0.1:1880/#flow/204ae15882a0aefa`. The interface includes a left sidebar with a 'common' category containing nodes like 'inject', 'debug', 'complete', 'catch', 'status', 'link in', 'link call', and 'link out'. The main workspace is a grid where a 'Welcome to Node-RED 3.0!' dialog box is displayed. The dialog box contains the text 'Let's take a moment to discover the new features in this release.' and a 'Start >' button. The right sidebar shows a '플로우 1' (Flow 1) section with a search bar and a list of flows, including '플로우 1' and '보조 플로우' (Auxiliary Flow). Below the flow list, there is a section for '플로우 1' with a search bar and a list of flows, including '플로우' and '204ae15882a0aefa'. At the bottom of the right sidebar, there is a section for '노드를 와이어 사이로 드래그 하여 연결할 수도 있습니다.' (You can also connect nodes by dragging them between wires).

설치 파일 위치 :

`C:\Users\계정명\AppData\Roaming\npm\node_modules\node-red`

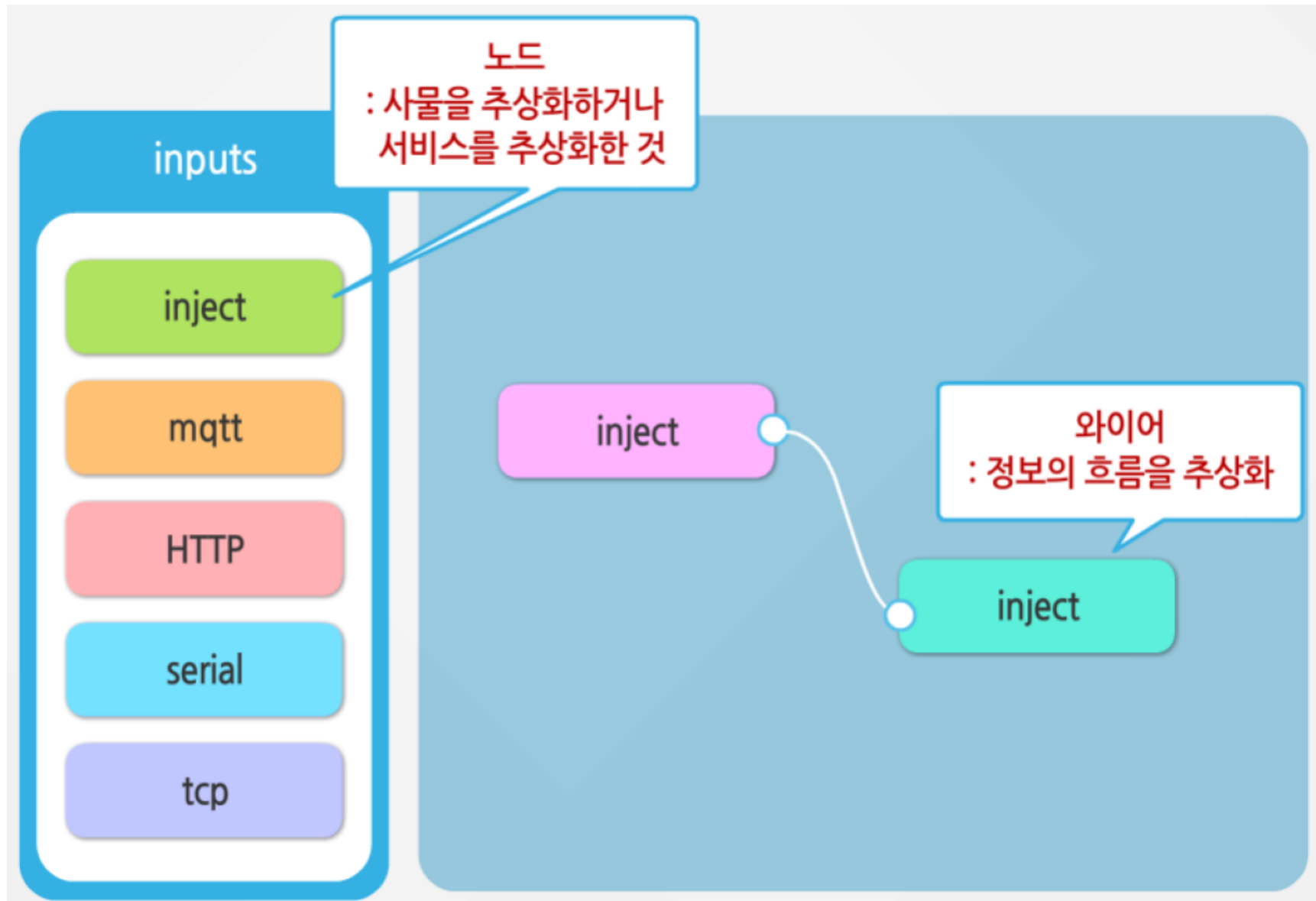
사용자가 작성한 로직 저장 위치 :

`C:\Users\계정명\AppData\Roaming\npm\node_modules\node-red\flows_*****.json`

- lib
- node\_modules
- .config.json
- .config.json.backup
- package.json
- settings.js



## \_Node-Red



1

사물인터넷 응용을 제작하는 비주얼 도구

2

간단한 런타임 배포, 시작품 제작에 적합

3

간단한 자동 실행 런타임을 쉽게 작성

4

다양한 연동을 간단하게 그려서 확장함

5

낮은 진입 장벽 : 누구나 쉽게 배우고 사용 가능

6

개방형 표준, 유연성, 공유



# \_Node-Red

The screenshot displays the Node-RED web interface with three main panels: '노드 팔레트' (Node Palette) on the left, '편집창' (Editor) in the center, and '정보창' (Info) on the right.

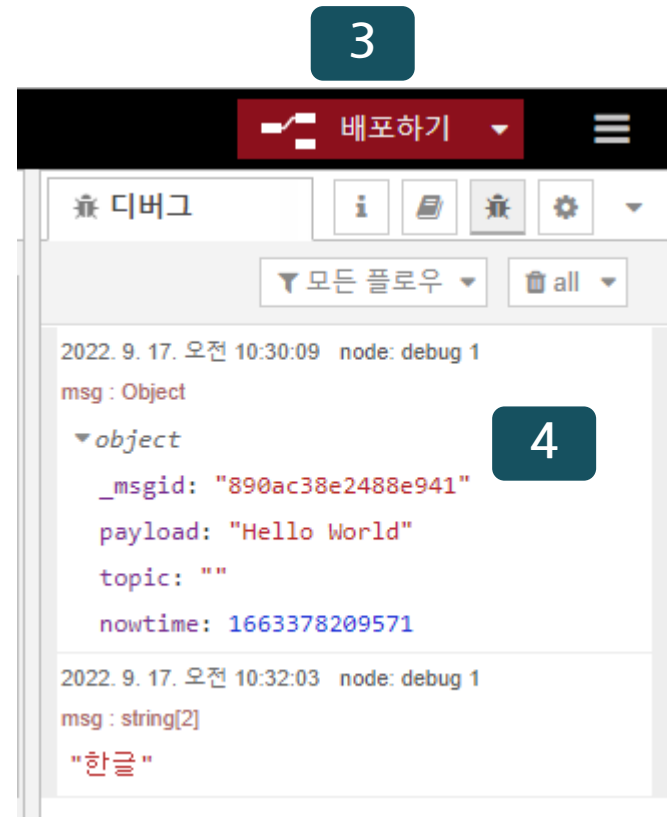
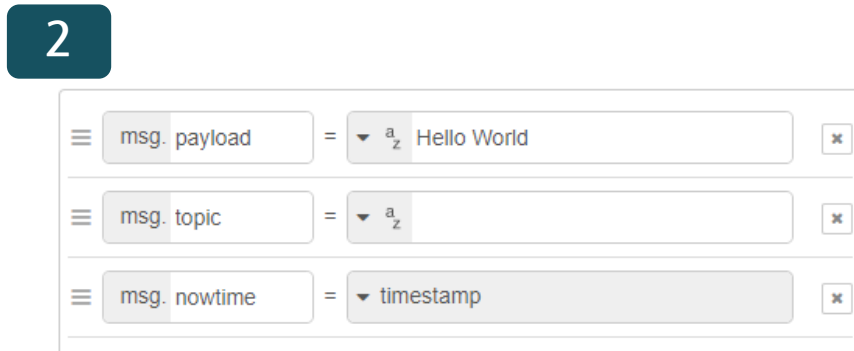
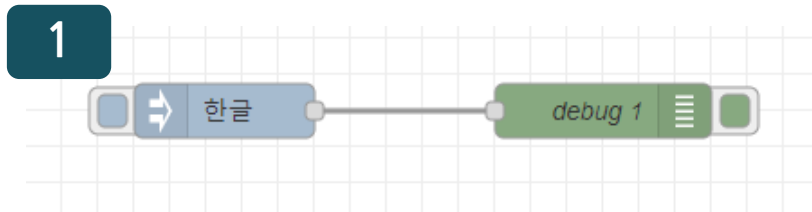
- 노드 팔레트 (Node Palette):** Contains two sections. The 'common' section has nodes like 'inject', 'debug', 'complete', 'catch', 'status', 'link in', 'link call', 'link out', and 'comment'. The '기능' (Function) section has nodes like 'function', 'switch', 'change', 'range', 'template', 'delay', 'trigger', and 'exec'. The 'inject' node is highlighted with a red box.
- 편집창 (Editor):** A large grid workspace for building flows. A red dashed arrow points from the 'inject' node to a text box labeled '항목 드래그-드랍' (Item Drag-Drop).
- 정보창 (Info):** Displays information about the current flow. It shows a search bar, a list of flows (including '플로우 1'), and details for the selected flow, including its ID ('204ae15882a0aefa').

At the bottom of the '정보창', a tip states: 'alt-⬆를 사용하여 노드 팔레트를 관리 할 수 있습니다.' (You can manage the node palette using alt-⬆).



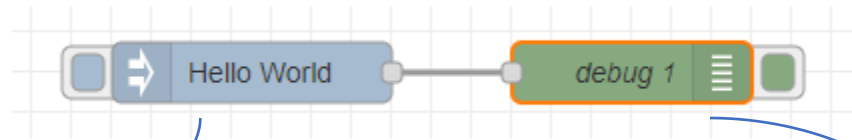
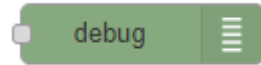
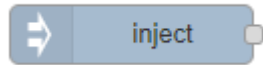
# \_Node-Red Node 편집 방법

1. 노드 배치 + 노드 연결
2. 노드 속성 편집
3. '배포하기' 클릭
4. 디버그 창에서 확인
  - \* API 확인 시에는 브라우저에서 확인



# \_Node-Red Node

시작 노드와 출력 노드 : 노드 간 데이터 전달은 msg.payload에 실어서 보내고 받는다



msg. payload =

msg. topic =

msg. nowtime =

속성

대상

출력대상 ☒ 디버그 창

☐ 시스템 콘솔

☐ 노드 상태(32자)

이름

속성

이름

msg. payload =

msg. topic =

flow.

global.

string

number

boolean

JSON

buffer

timestamp

expression

env variable

msg.

디버그

모든 플로우

all

2022. 9. 17. 오전 10:39:47 node: debug 1

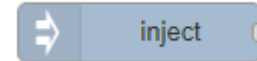
msg : string[11]

"Hello World"



# \_Node-Red Node

inject Node : 노드 실행의 방식 결정 - 클릭 실행, 자동 (반복) 실행



☐ Node-RED시작의 0.1 초 후, 아래를 시행

반복 지정한 시간간격

시간각격 1 초

☐ Node-RED시작의 0.1 초 후, 아래를 시행

반복 지정한 일시

시각 12:00

요일 ☒ 월요일 ☒ 화요일 ☒ 수요일  
☒ 목요일 ☒ 금요일 ☒ 토요일  
☒ 일요일

☐ Node-RED시작의 0.1 초 후, 아래를 시행

반복 지정한 시간간격, 일시

시간각격 1 분

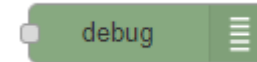
시각 00:00 ~ 01:00

요일 ☒ 월요일 ☒ 화요일 ☒ 수요일  
☒ 목요일 ☒ 금요일 ☒ 토요일  
☒ 일요일



# \_Node-Red Node

debug Node : 출력할 형식 지정 - 출력 형식을 직접 지정할 때 (JSON으로 정의)



debug의 노드 수정

삭제 취소 완료

속성

대상

출력대상

msg.  
msg오브젝트 전체  
**J: expression**  
☐ 노드 상태(32사)

이름

debug 1

취소 완료

형식

```
1 {  
2   "payload" : payload,  
3   "all" : $append(payload, nowtime)  
4 }  
5
```

기능 참조

테스트

예제 메세지

```
{  
  "payload": "hell"  
}
```

JSON 형식

결과

```
{  
  "payload": "hell"  
  "all": "hello wo"  
}
```

형식 유효성 확인

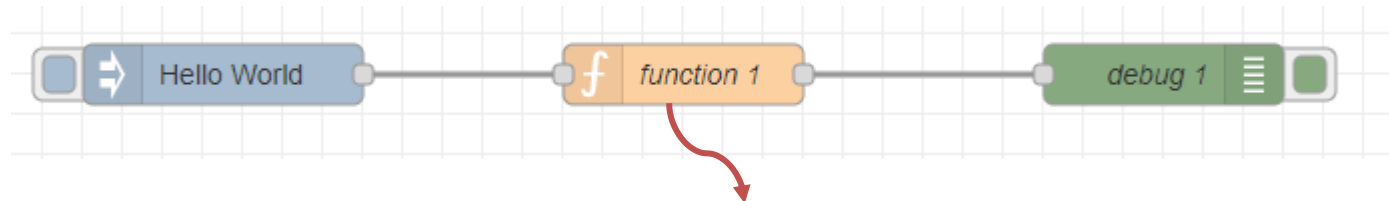
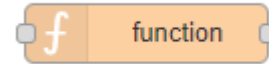




## \_Node-Red Node

function Node : 중간에 데이터를 조작할 때

- 데이터 전달은 msg 객체에 데이터를 실어서 리턴
- 항상 msg를 리턴해야 한다. (데이터가 없어도)



function의 노드 수정

삭제 취소 완료

속성

이름 function 1

Setup On Start 코드 On Stop

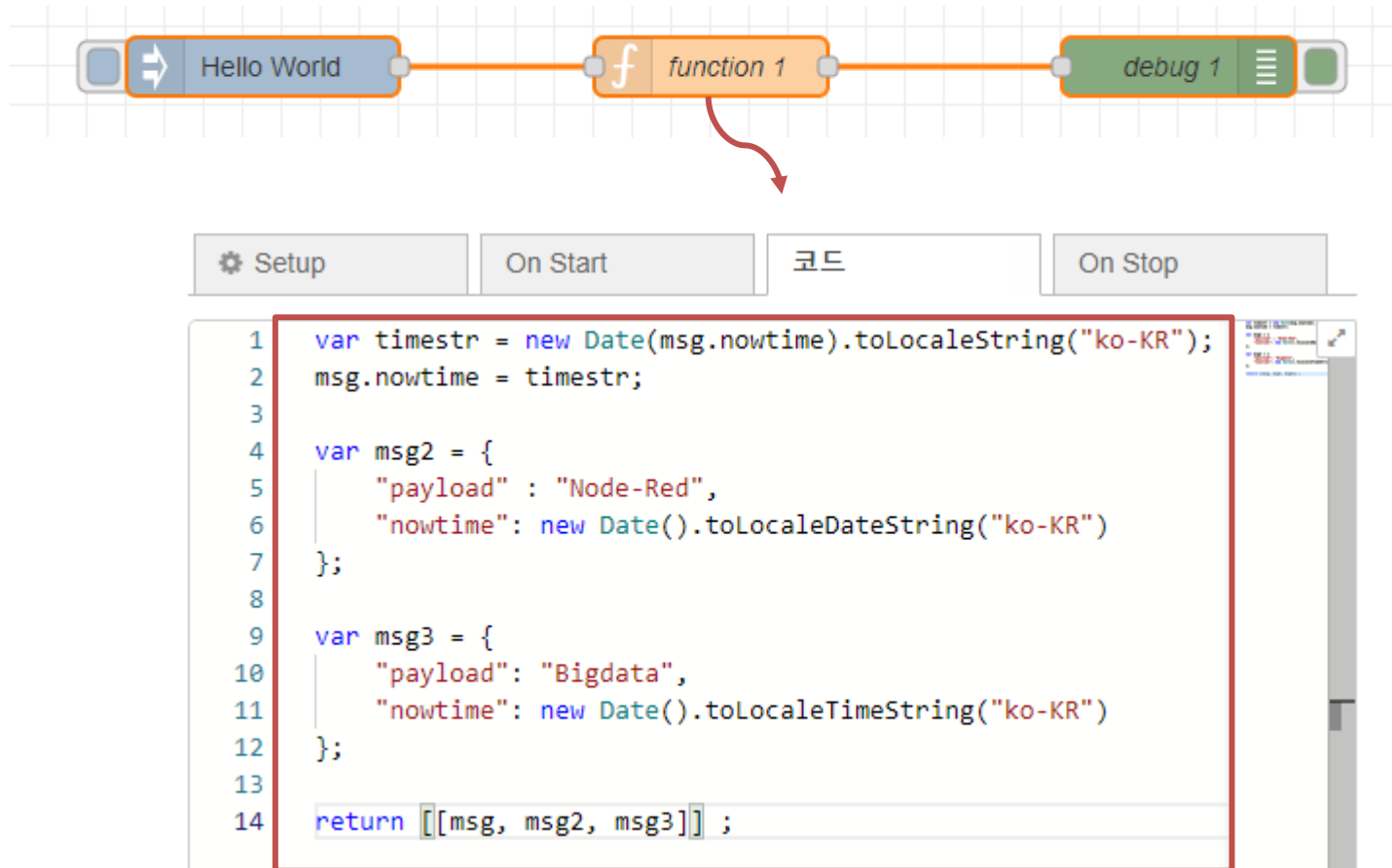
```
1 var timestr = new Date(msg.nowtime).toLocaleString("ko-KR");
2
3 msg.nowtime = timestr;
4 return msg;
```



# \_Node-Red Node

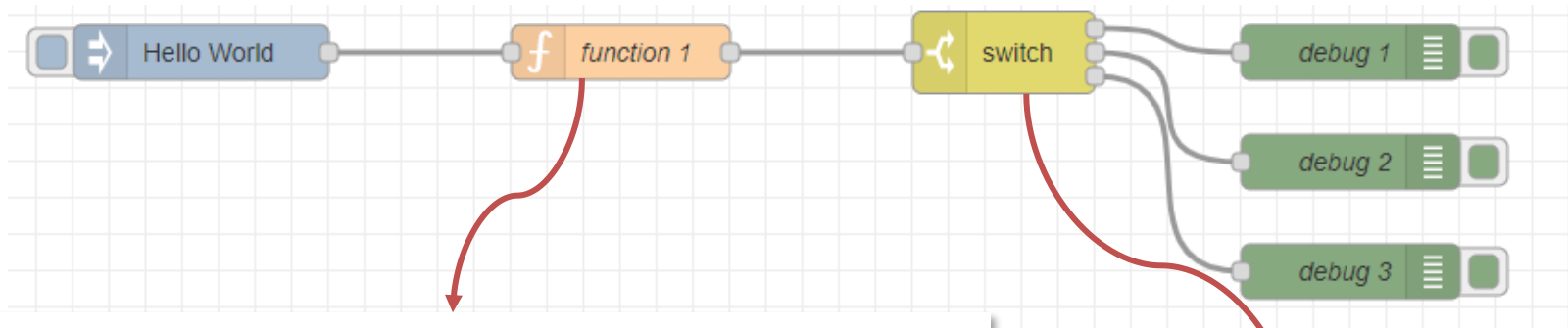
function Node

- 다중 출력 가능 : msg 객체들을 배열의 배열로 묶어서 리턴



# \_Node-Red Node

switch Node : 출력 방향 조절



```
1  var timestr = new Date(msg.nowtime).toLocaleString("ko-KR");
2  msg.nowtime = timestr;
3  msg.index = 1;
4
5  var msg2 = {
6    "index" : 2,
7    "payload" : "Node-Red",
8    "nowtime": new Date().toLocaleDateString("ko-KR")
9  };
10
11 var msg3 = {
12   "index" : 3,
13   "payload": "Bigdata",
14   "nowtime": new Date().toLocaleTimeString("ko-KR")
15 };
16
17 return [[msg, msg2, msg3]] ;
```

속성

이름: 이름

프로퍼티: msg. index

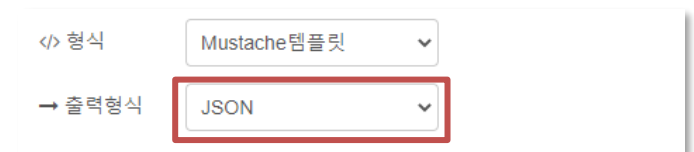
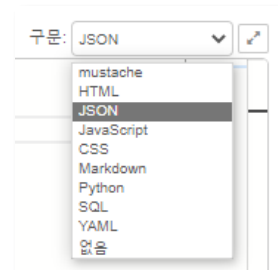
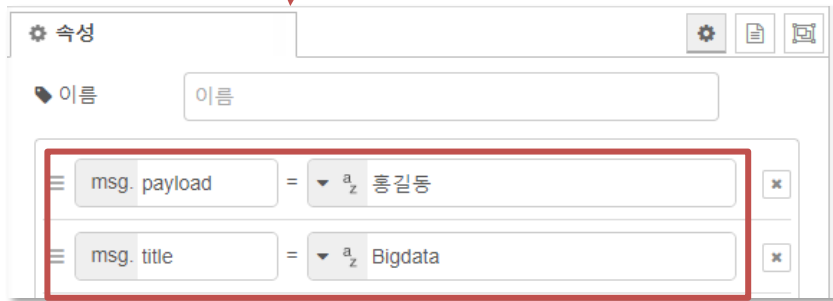
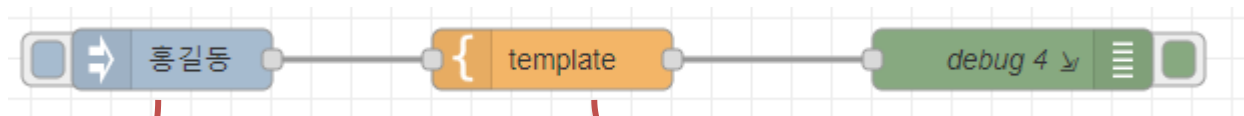
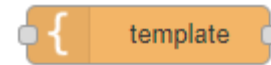
==	a <sub>z</sub> 1	→ 1	x
==	a <sub>z</sub> 2	→ 2	x
==	a <sub>z</sub> 3	→ 3	x



## \_Node-Red Node

template Node : 원하는 출력을 static 데이터로 출력

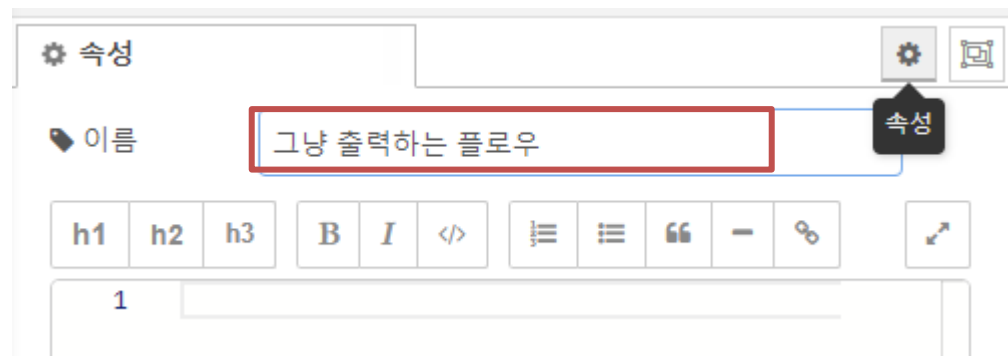
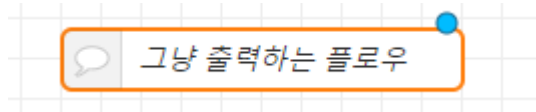
- 출력은 msg.payload에 실어서 다음 노드에 보내 짐
- 출력 형식 지정 가능 : 일반 텍스트, json 형식, YAML



# \_Node-Red Node

comment Node : 주석 달기

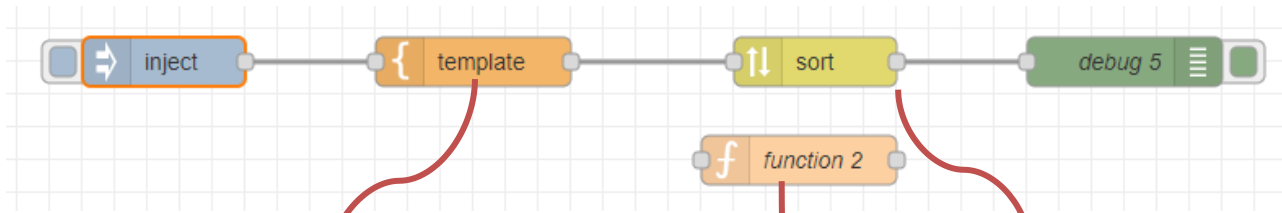
comment



# \_Node-Red Node

sort Node : 데이터 정렬

- sort 노드 대신 함수로 처리하는 경우가 많음



속성

이름: 이름

프로퍼티: msg.payload

템플릿

구문: JSON

```
1 [
2   "abc1",
3   "def2",
4   "xyz4",
5   "mno5"
6 ]
```

속성

대상: msg.payload

키: 요소의 값

순서: 내림차순

☐ 수치로써 비교

속성

이름: function 2

Setup On Start 코드 On Stop

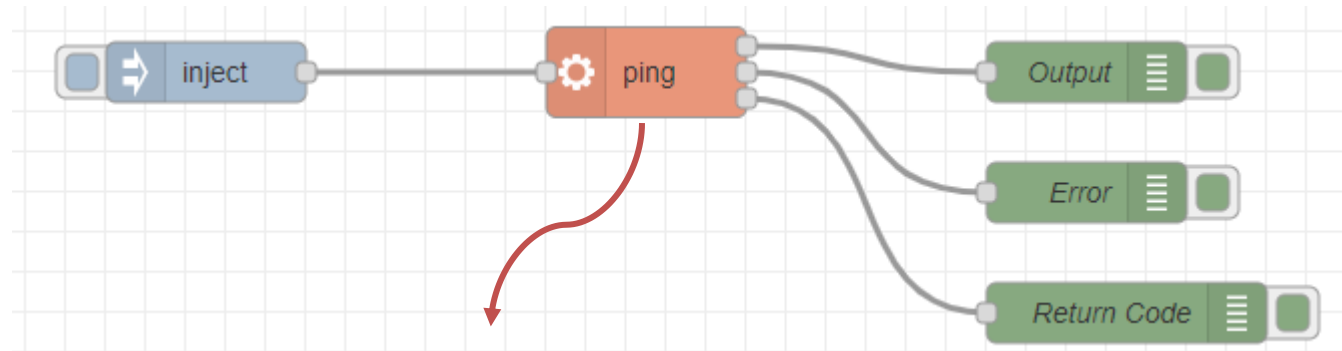
```
1 // 내림 차순
2 msg.payload.sort((A,B) => {
3   return A > B ? -1 : A < B ? 1 : 0;
4 });
5
6 return msg;
```



# \_Node-Red Node

exec Node : 외부 명령 실행

- 3개의 결과를 받음 : 출력값, 에러값, 리턴값



속성

커맨드

ping

+ 인수

☐ msg. payload

www.google.com

출력

커맨드 종료시 - exec모드

타임아웃

임의 초

Hide console

☐

이름

이름



---

# 페이지 서비스 및 API 만들기

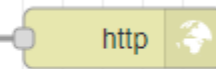
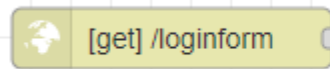
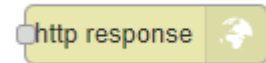
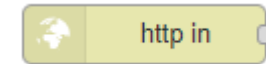
---



# \_Node-Red 웹페이지 서비스 / REST API

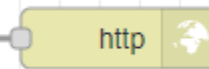
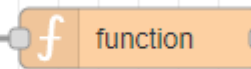
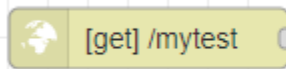
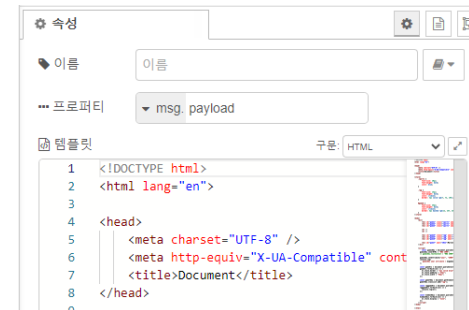
## 프로세스

1. 'http in' 노드로 호출할 인터페이스 정의 (호출방식, 호출 이름)
2. 함수나 템플릿 노드로 클라이언트에게 전송할 데이터 (html 텍스트, JSON) 가공
3. 'http response' 노드로 2번에서 만든 데이터를 내보냄



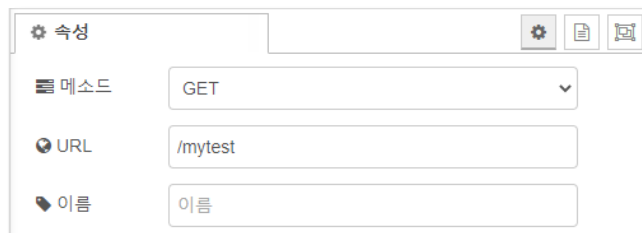
- 접속 URL을 설정함
  - 접속 주소 = Node\_red 접속URL + URL
- ex) `http://127.0.0.1:1880/loginform`

- 전달할 페이지 생성



- 접속 URL을 설정함
  - 접속 주소 = Node\_red 접속URL + URL
- ex) `http://127.0.0.1:1880/mytest`

- 전달할 데이터 생성



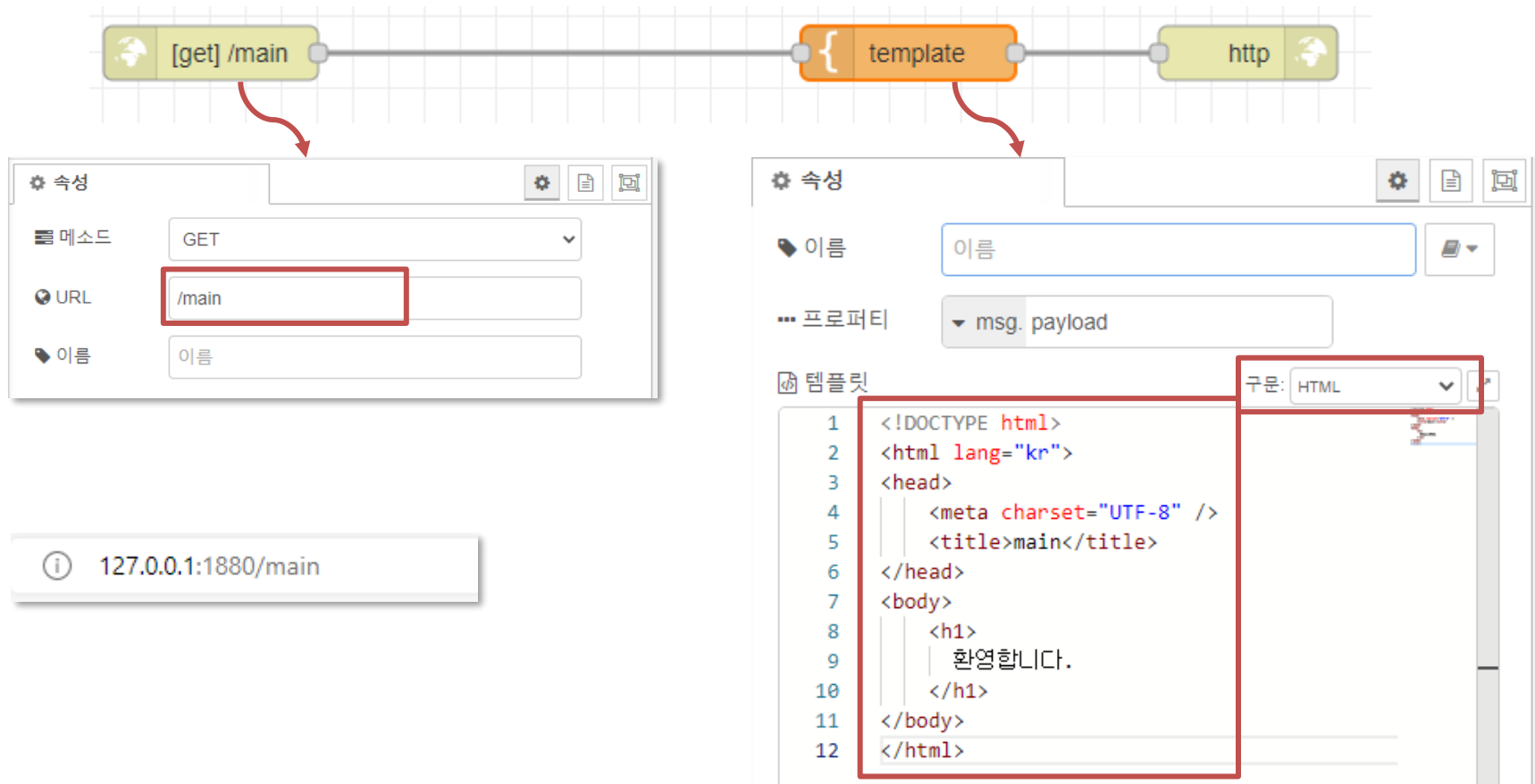
```
var value = 30
msg.payload = {
  titleCaption: "" + value ,
  titleColor: '#ff0000',
  fontSize: '50px'
}
return msg;
```

\* 전송할 데이터는 msg.payload에 담는다.



# \_Node-Red 웹페이지 서비스

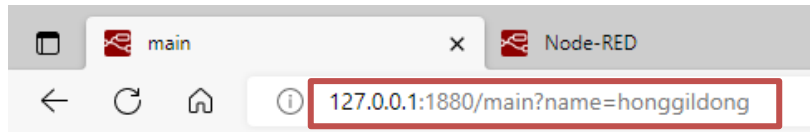
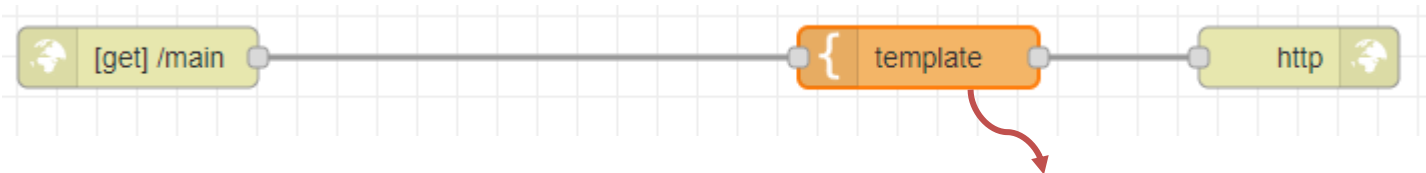
## 웹 페이지 서비스 만들기



## \_Node-Red 웹페이지 서비스

웹 페이지 서비스 만들기 : 파라미터 받기

- msg.payload에 파라미터가 담겨 옴



**honggildong 님 환영합니다.**

- {{payload.name}} 식으로 파라미터를 받을 수 있음

```
<!DOCTYPE html>
<html lang="kr">
<head>
  <meta charset="UTF-8" />
  <title>main</title>
</head>
<body>
  <h1>
    {{payload.name}} 님 환영합니다.
  </h1>
</body>
</html>
```



# \_Node-Red REST API

## REST API 만들기



속성

메소드 GET

URL /mytest

이름 이름

- msg.payload에 파라미터 데이터가 있음

속성

이름 이름

Setup On Start 코드 On Stop

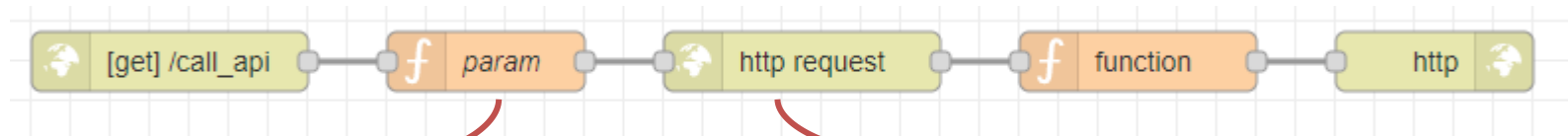
```
1 var param = msg.payload;
2
3 console.log(param);
4
5 msg.payload = {
6   test :{
7     node : "test"
8   },
9   obj : param
10 };
11 return msg;
```



## \_Node-Red - Open API 호출

외부 API 호출한 결과를 가공하는 노드 만들기

- 1) API 호출시 필요한 파라미터 데이터 준비 : 함수노드
- 2) http request에 호출할 외부API URL 지정



속성

이름 param

Setup On Start 코드

```
1 msg.payload = {}  
2   topics : 234243,  
3   limit : 100  
4 }  
5 return msg;
```

\*REST API에 전달할 파라미터를  
msg.payload에 담음

속성

메소드 GET

URL https://jsonplaceholder.typicode.com/users

페이로드 Append to query-string parameters

☐ SSL/TLS접속을 유효화

☐ 인증을 사용

☐ Enable connection keep-alive

☐ 프록시를 사용

☐ Only send non-2xx responses to Catch node

☐ Disable strict HTTP parsing

출력형식 JSON오브젝트

URL 지정

출력 지정

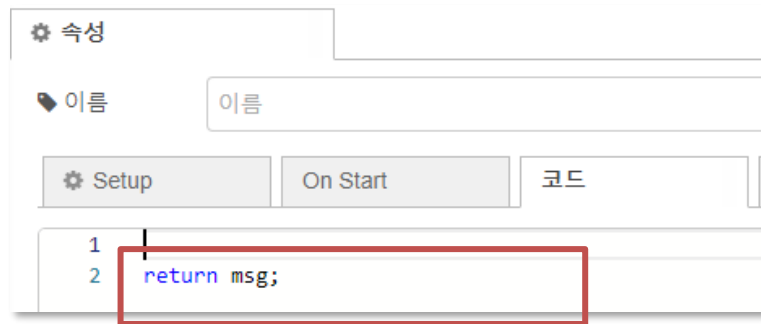
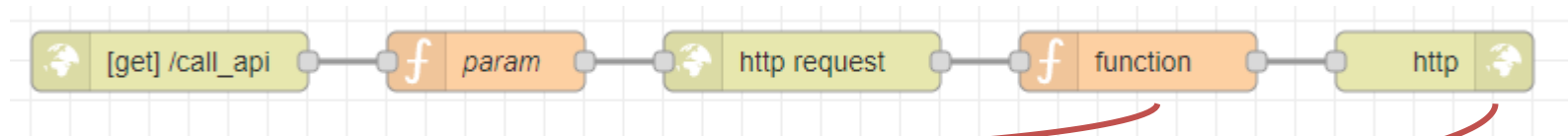


## \_Node-Red - Open API 호출

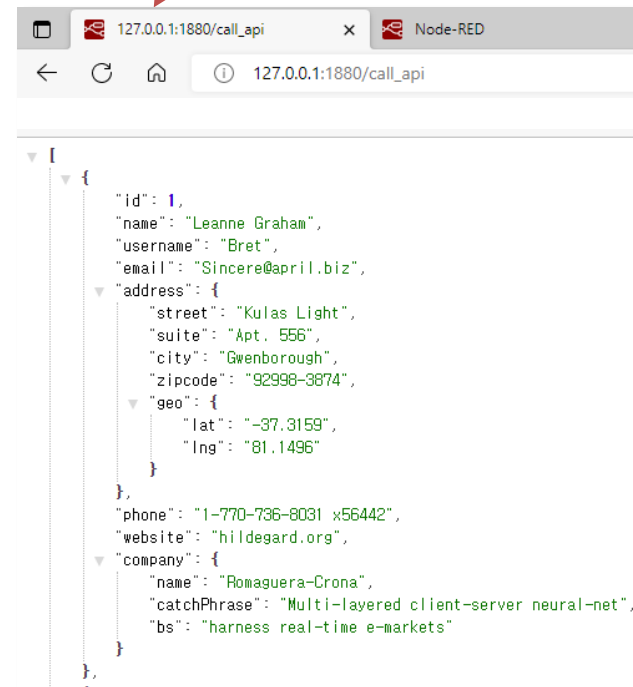
외부 API 호출한 결과를 가공하는 노드 만들기

3) 호출 결과(대개의 경우 JSON 형식) 가공

4) http로 보내거나 로그로 출력



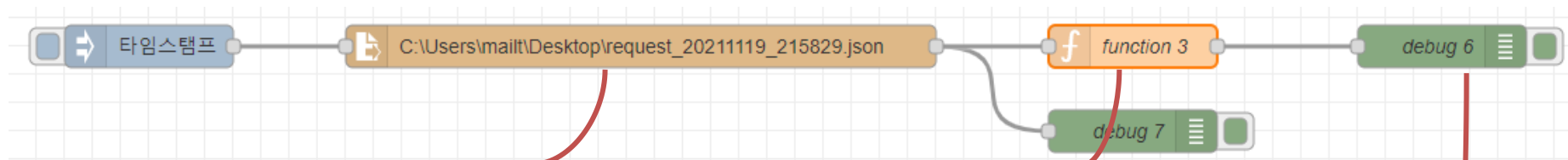
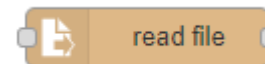
호출 결과 msg.payload에 담김



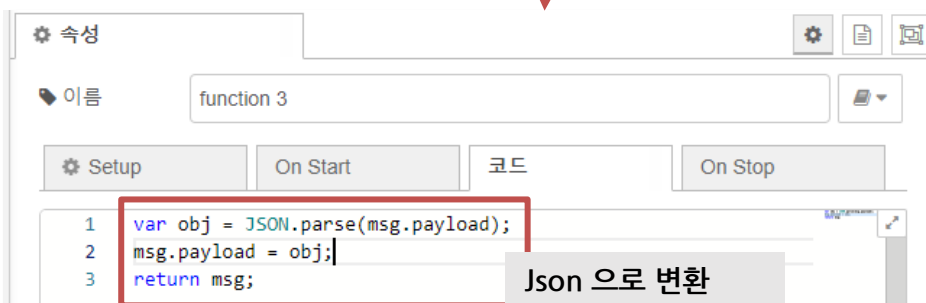
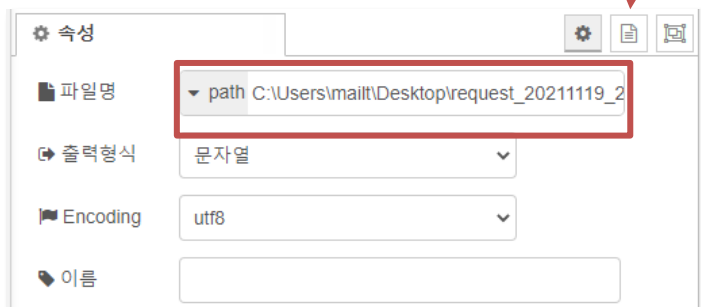
# \_Node-Red : 파일 읽기

특정 경로에 있는 텍스트 파일 읽는 노드 만들기

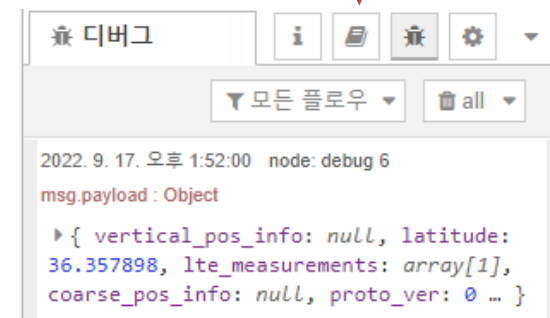
- 1) 'read file' 노드에 읽어올 파일 경로 지정
- 2) 읽어온 데이터는 가공하는 노드 작성 : 함수 노드



불러올 파일 경로 지정



\* 읽어온 데이터는 msg.payload에 담겨 있음



<https://flows.nodered.org/>

The screenshot shows the Node-RED Library homepage. At the top is a navigation bar with links: home, about, blog, documentation, forum, flows (active), and github. Below the navigation bar is a search bar labeled 'Search library' and a 'Sign in with GitHub' button. The main content area is titled 'Node-RED Library' with the subtitle 'Find new nodes, share your flows and see what other people have done with Node-RED.' Below this are three sections: 'Recent nodes', 'Recent flows', and 'Recent collections'. Each section displays a grid of items with their names, descriptions, and statistics. A blue arrow points from the 'Recent nodes' section to the text '1 커스텀 노드들' (1 Custom nodes). Another blue arrow points from the 'Recent flows' section to the text '2 Flow 예제들' (2 Flow examples).

Section	Item Name	Description	Version	Downloads	Stars	Type
Recent nodes	red-contrib-sunshine-conversation	Send message using sunshine conversation API	v0.1.32	302	5.0	node
	node-red-contrib-overkiz	Node-RED Module to control devices using the Overkiz IoT API	v0.9.0	8	5.0	node
	node-red-contrib-personal-wake-word	a node-red integration of node-personal-wakeword	v0.2.1	0		node
Recent flows	Switchbot v1	testingtestingtestingtestingtestingtestingtesting				flow
	Guaranteed delivery of data (upload, email etc) across a network	This flow demonstrates how to achieve guaranteed delivery of data across a network.				flow
	Stop sequence	Someone asked for a flow. A number of lights start turning on with a delay in between when an input boolean is turned on.				flow
Recent collections	For future use	Personal for future use. Not much useful for anyone else.				collection
	Dashboard-LEDs	LED segments for a dashboard display				collection
	NewCareTec-Gateway	NodeRed used in the Raspberry Pi/RAK-Gateway for sensor data assessment and alerting.				collection

1 커스텀 노드들

2 Flow 예제들





---

**Database : Mysql**

---

# \_Database


## 1. mysql 설치

<https://downloads.mysql.com/archives/installer/>

- 버전은 5.7.xx 로 선택

### MySQL Product Archives

MySQL Installer (Archived Versions)

 Please note that these are old versions. New releases will have recent bug fixes and features!  
To download the latest release of MySQL Installer, please visit MySQL Downloads.

Product Version:

Operating System:

Windows (x86, 32-bit), MSI Installer Apr 6, 2022

(mysql-installer-web-community-5.7.38.0.msi)

Windows (x86, 32-bit), MSI Installer Apr 6, 2022

(mysql-installer-community-5.7.38.0.msi)

 We suggest that you use the MD5 checksums and GnuPG signatures to verify the integrity of the packages you download.


## 2. workbanch 설치

<https://downloads.mysql.com/archives/workbench/>

- 버전은 6.3.xx 로 선택

### MySQL Product Archives

MySQL Workbench (Archived Versions)

 Please note that these are old versions. New releases will have recent bug fixes and features!  
To download the latest release of MySQL Workbench, please visit MySQL Downloads.

Product Version:

Operating System:

Windows (x86, 32-bit), MSI Installer Feb 15, 2013

(mysql-workbench-gpl-5.2.47-win32.msi)

Windows (x86, 32-bit), ZIP Archive Feb 15, 2013

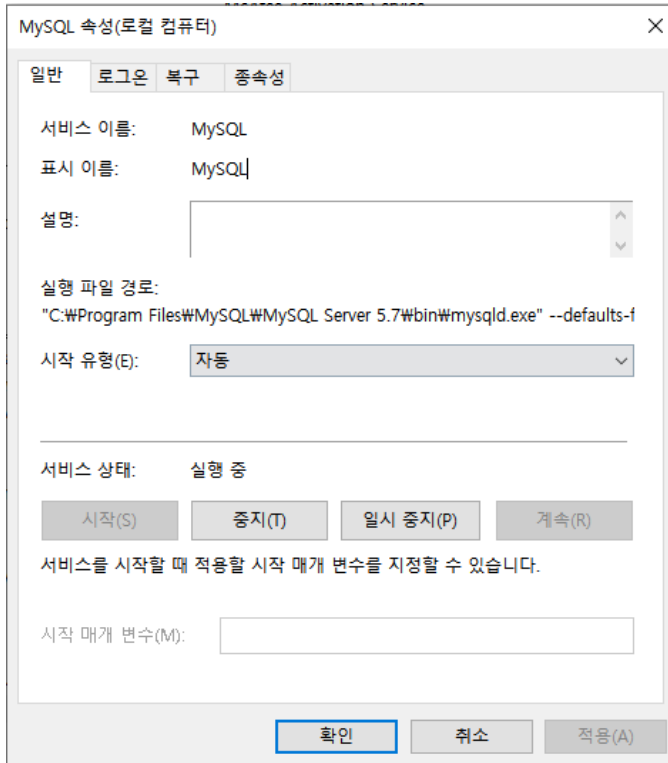
(mysql-workbench-gpl-5.2.47-win32-noinstall.zip)

 We suggest that you use the MD5 checksums and GnuPG signatures to verify the integrity of the packages you download.



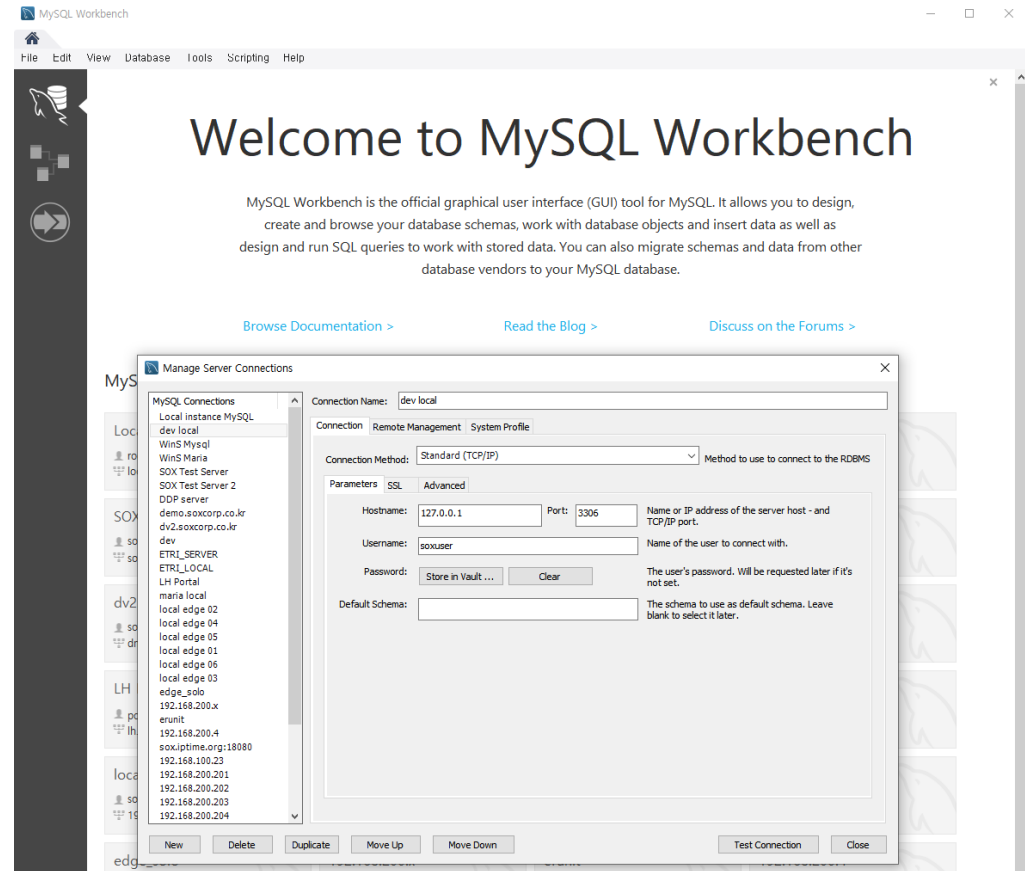
# \_Database

## 1. mysql 실행 : 서비스에서 실행



## 2. workbench 실행

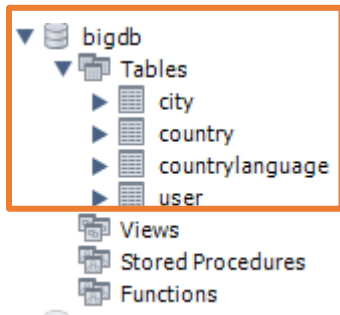
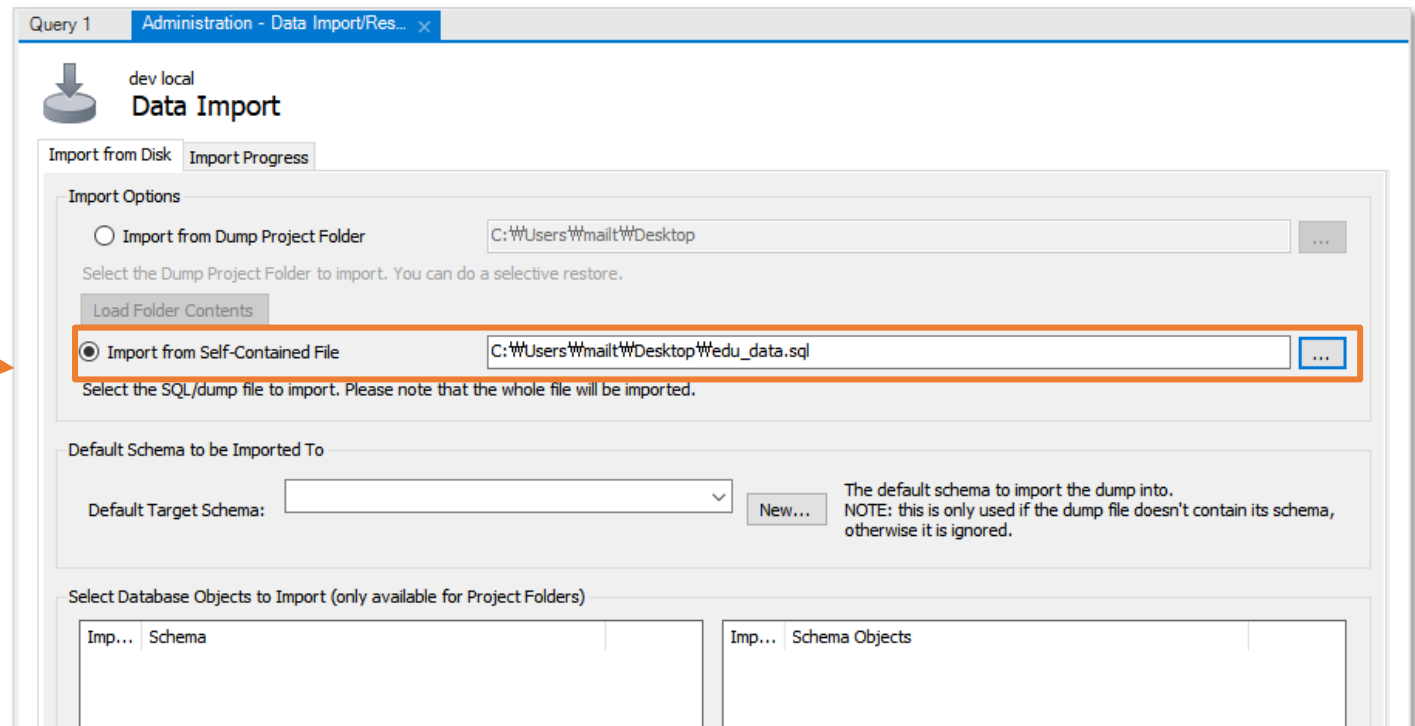
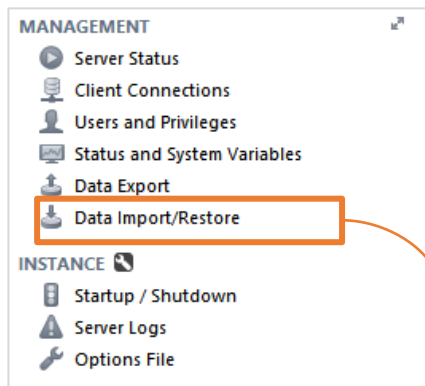
: 접속 정보 입력 후 더블클릭



## \_Database

실습 데이터 임포트 하기 : 'edu\_data.sql '

→ bigdb 스키마 생성



---

## 응용 예제

---

# 실습

실습 문제 : "tabledata.json" 데이터를 클라이언트로 전송하는 노드를 만드시오

- 클라이언트는 ajax로 결과를 받아 콘솔에 출력



# 실습

- 실습 문제 : 영어, 국어, 수학 성적 데이터를 입력 받아 ajax로 총점과 평균 결과를 받아 처리
- 총점과 평균을 구하는 서버 API는 노드로 작성하시오



## 3일차 최종 예제

### 로그인 웹페이지를 만들고 정보를 확인하여 로그인 결과를 알려주는 프로그램 작성

1. Node-Red로 로그인 결과를 확인해 주는 API를 만드시오
  - 파라미터로 id/pw를 입력 받고 이것을 조건으로 유저를 확인하는 쿼리 작성  
user : "bigdatauser"  
password : "!bigdata"
  - API 이름은 login, 타입은 'post'방식
  - 유효한 사용자가 있고 비번이 맞으면 { result : "OK"} 로 응답  
없거나 비번이 틀리면 { result : "NO"}로 응답
2. 로그인 페이지 작성 (Node-Red 템플릿 노드 활용)
  - id 입력, pw 입력, '로그인' 버튼 작성
  - id/pw가 비어있지 않은지 체크 하는 로직 필요
3. ajax를 통해 2번에서 작성한 API와 연결하여 로그인 결과를 받고 성공여부를 메시지 박스(alert)로 출력



웹 페이지

- AJAX로 로그인 여부 확인후  
main 페이지로 이동



데이터 가공  
API

- 로그인 페이지 제공 (loginform)
- 로그인 확인 API (login)  
: 로그인 확인 결과를 http로 실어 보냄
- 로그인 성공 후 환영 페이지 제공 (main)





