Node.js 입문

@IT Korea 강사 이정훈

목차

1. Node.js 란? npm 이란?

2. npm 사용해보기

3. 서버 종류와 간단한 서버 만들기

4. Express 에 대해 알아보자.

5. 미들웨어

6. Express 로 서버만들기 실습!

프론트엔드 심화반 강의는 JavaScript를 공부하신 대상에게 진행되게 됩니다.
node.js 와 Angular.js, React.js, Vue.js 등을 2개월동안 진행하게 됩니다.

1. Node.js 란?

Node.js®는 Chrome V8 JavaScript 엔진으로 빌드된 JavaScript 런타임입니다. Node.js는 이벤트 기반, 논 블로킹 I/O 모델을 사용해 가볍고 효율적입니다. Node.js의 패키지 생태계인 npm은 세계에서 가장 큰 오픈 소스 라이브러리 생태계이기도 합니다.

Node.js 공식 사이트에서 내린 Node.js의 정의

Node.js란 무엇이고, 우리가 왜 배워야할까요?
Node.js는 JavaScript의 구동환경입니다. 그런데 이놈은 서버도 만들수있습니다.
우선 몇가지 오해부터 잡고 넘어가겠습니다.

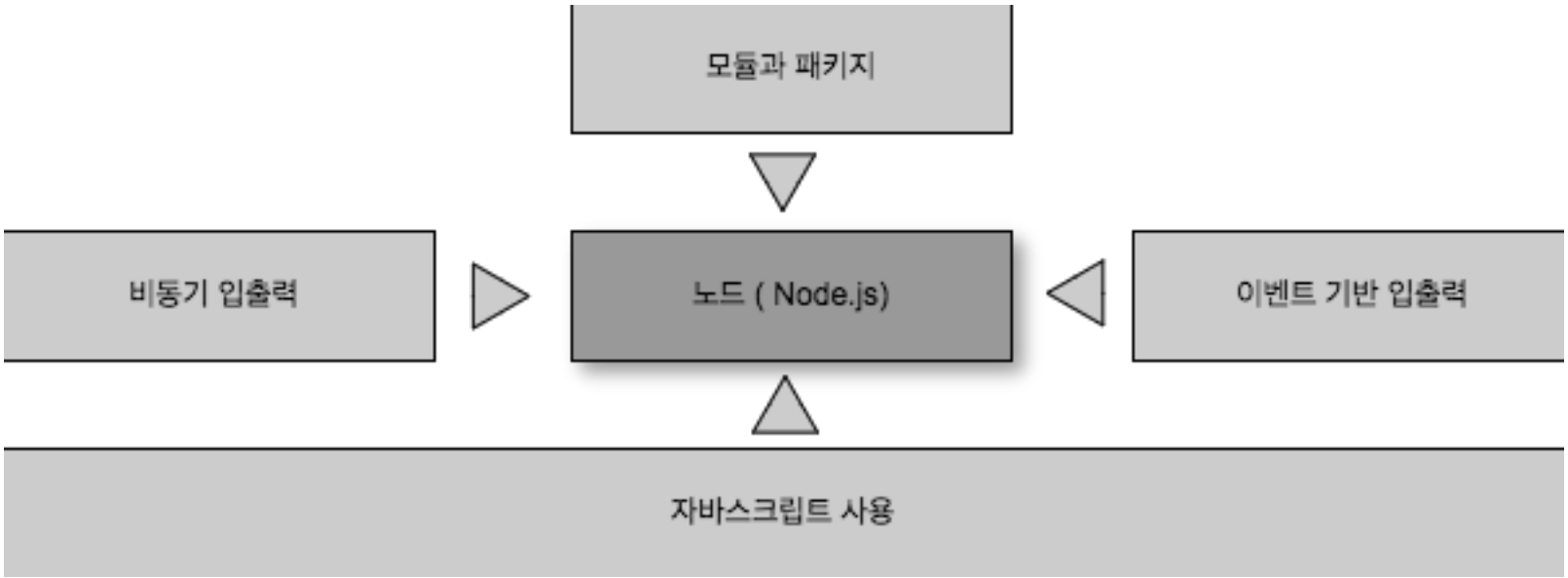
Q. Node.js는 서버인가요?
A. 네. 백엔드 개발자들이 씁니다.

네. 백엔드 개발자들이 씁니다.
하지만 프론트엔드 개발자들도 씁니다.

Q. 서버가 만들어진다면, 백엔드의 영역이 아닌가요?
A. 서버도 만들 수 있지만, 자바스크립트에 기반한 스크립트 언어입니다.

결론 : node.js 는 백엔드, 웹서버가 아니고, 자바스크립트 실행환경에 불과하다.
그러나 서버도 만들수있기때문에 , 프론트엔드, 백엔드 양쪽에서 활발하게 사용이 된다.

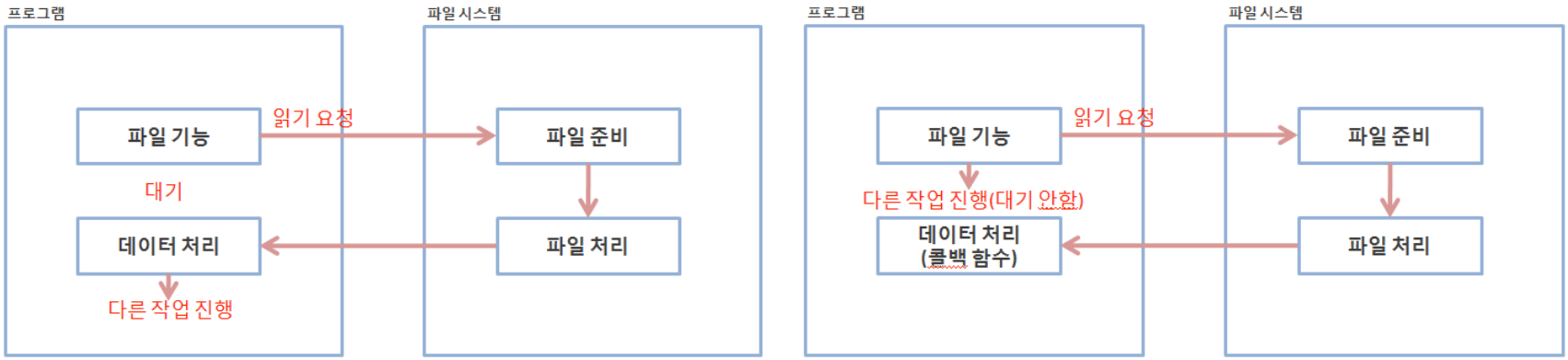
노드의 구성요소



크게 노드의 특징은 3가지로 나뉘어집니다.

1.비동기 입출력방식 , 2.이벤트 기반 입출력 , 3.모듈과 패키지

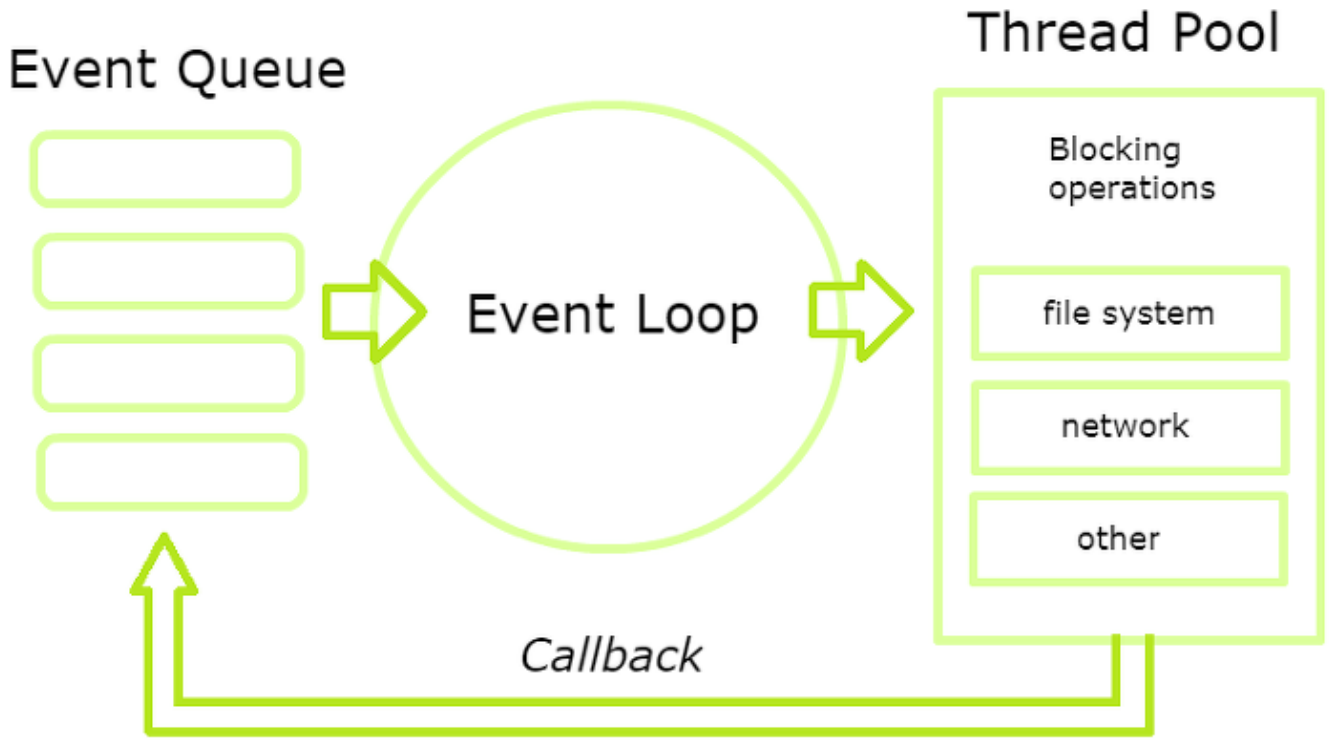
비동기 입출력 방식



* 콜백함수란? 변수로 전달된 함수를 다른 함수 내부에서 호출하는 것

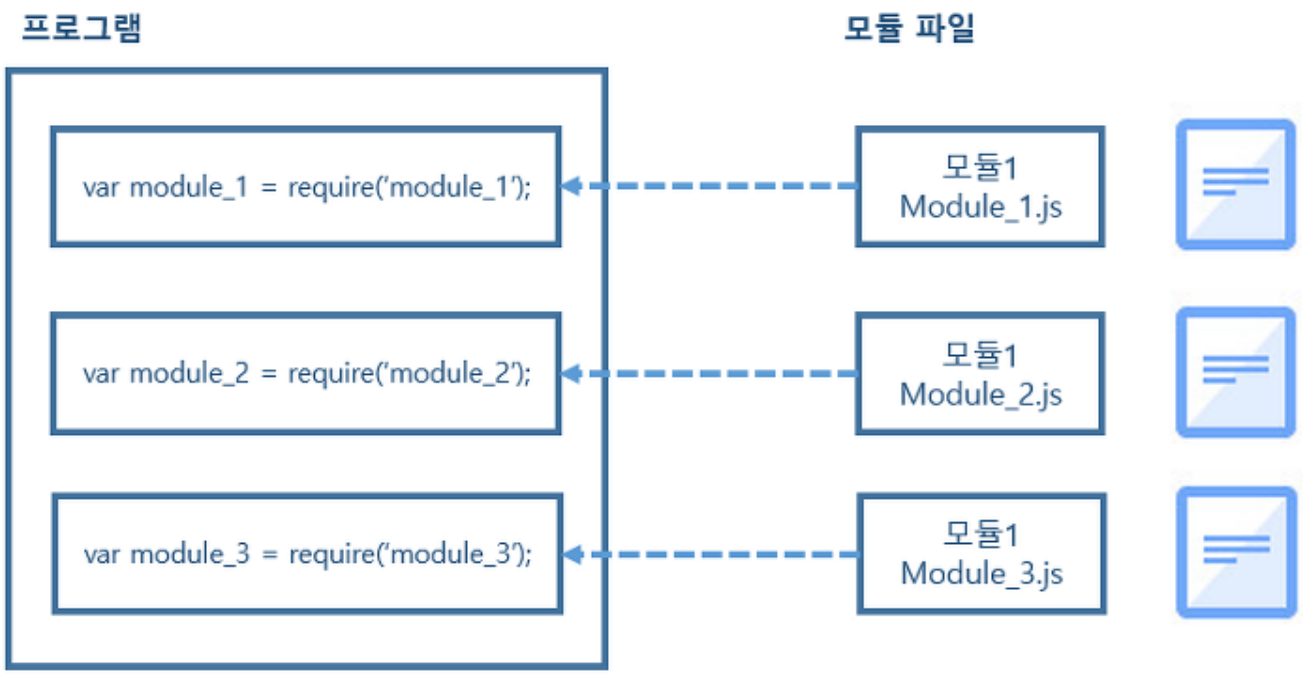
비동기 입출력 방식이란 하나의 작업을 처리시, 요청처리 대기과정 없이 다른작업을 동시에 진행 할 수 있는 방식을 말합니다. 이 방식의 장점은 속도가 느려지는 문제가 없어집니다.

이벤트 기반 입출력



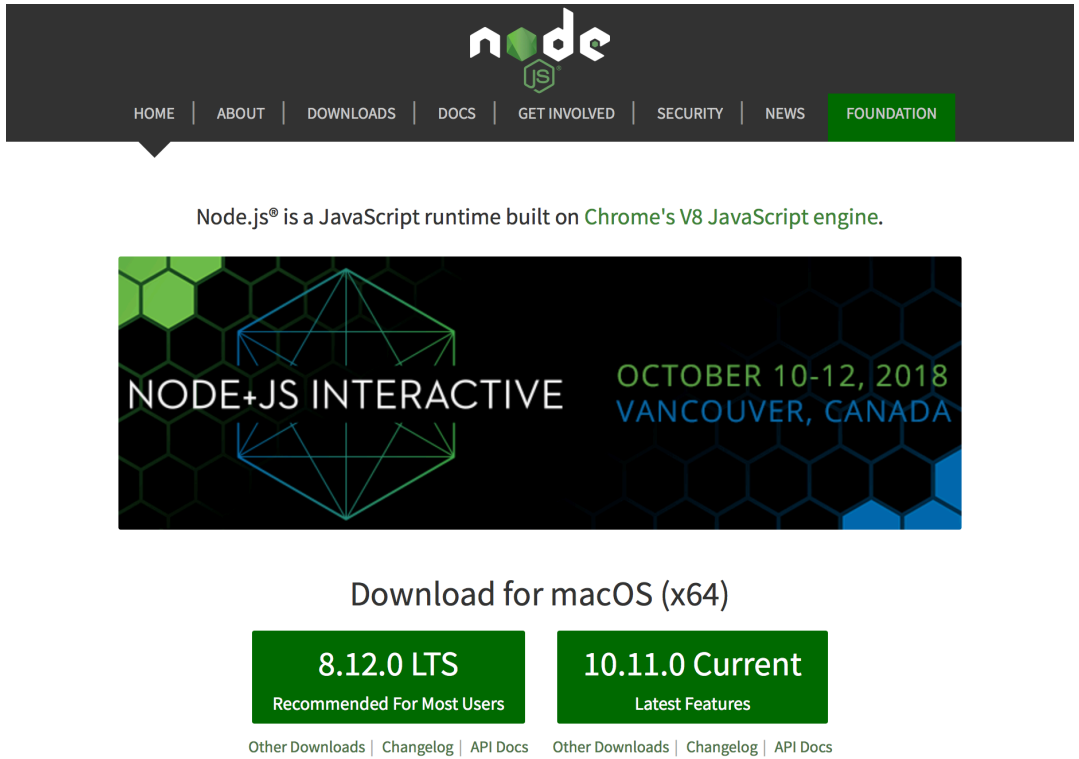
이벤트 기반 입출력 뭔가 듣기에 어려워 보이실 수 있습니다.
쉽게 여러분들이 컴퓨터에 행하는 모든 작업이 하나의 이벤트라 생각하시면 됩니다.
이러한 이벤트들이 일어나는것을 기반으로, 입출력행위가 발생하게 됩니다.

모듈과 패키지



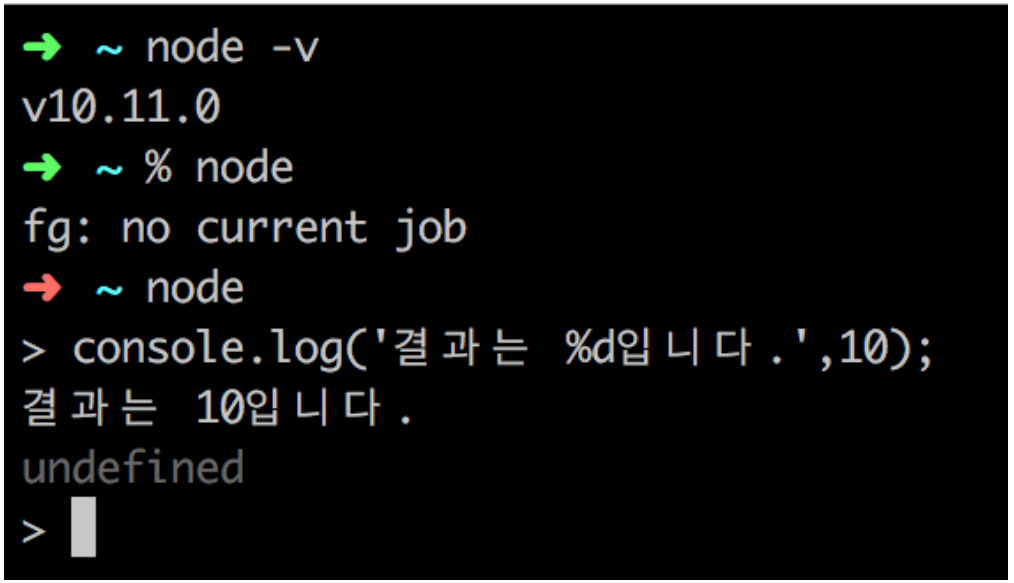
모듈이란, 자바스크립트 파일의 일부 코드를 떼어 별도의 파일로 만든것을 말합니다.
패키지는 여러개의 모듈을 합쳐서 만든것입니다. 한번 패키지를 만들어두면,
npm이라는 것을 통해서 쉽게 설치를 할수있습니다. npm은 바로 뒤에서 알아보겠습니다.
손쉽게 패키지를 설치할수있습니다.

노드 설치 및 실행(1/2)



자 그럼 이제 노드를 직접 실습 해보겠습니다. 먼저 노드를 쓰기위해 설치를 해야겠죠?
http://nodejs.org에 들어가서 노드를 설치해줍니다.

노드 설치 및 실행(2/2)



가볍게 명령프롬포트에서 코드를 실행해보겠습니다.
저는 MacOS 터미널 환경에서 작업을 해서 node 만 쳐도 작업이 되지만,
명령프롬프트 내에서는 node 나 console.log 앞에 %를 붙여주셔야합니다.

2. NPM

자 드디어 npm에 대해서 설명할 시간이 왔습니다.
npm 이란 node pacakage manager 의 약어로,
우리는 위에서 package가 module의 모음이라는것을 알고있습니다.
이 패키지들을 효율적으로 쓰게 해주는 매니지 툴인셈이죠.

npm 과 Yarn

npm은 node.js를 하면서 필수적으로 따라오는 녀석으로,
많이 들어보셧을텐데 yarn은 뭐지? 하시는분들이 있을것같습니다.
yarn 은 npm의 단점을 좀더 보완한 매니지먼트툴이라 보시면됩니다.
하지만 저희는 좀더 보편화가 되어있는 npm으로 공부를 할 것입니다.

npm 설치 및 사용

```
➜  ~ npm -v
6.4.1
➜  ~ npm update -g npm
➜  ~ npm install [패키지 이름]
```

본격적으로 패키지를 사용해보겠습니다.
node.js를 설치하셧다면 npm은 같이 깔려있으실겁니다.
먼저 npm -v로 버전을 확인합니다. 버전이 낮다면 업데이트를 진행하시면 됩니다.
그리고 사용하고싶은 패키지를 npm install [패키지이름]으로 사용하시면됩니다.

*** node.js 버전 관리에 유용한 패키지 : npm install -g n**
n 패키지 사용법 : n lastest (최신버전으로 설치)

package.json

```json
{
  "name": "jhoon2816",
  "version": "1.0.0",
  "description": "npm description",
  "main": "server.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "node server.js"
  },
  "keywords": [
    "jhoon2816",
    "npm",
    "nodejs",
    "lecture"
 ],
  "author": "jhoon2816",
  "license": "ISC"
}
```

npm에서 패키지를 사용할때 체계적 관리가 필요하고,
수십 수백개의 패키지에 대해 각각 해당하는 버전을 기록해야합니다.
버전이 중요한 이유는 버전이 업그레이드 되면서 이전 버전과 호환이 안될수도있기 때문이죠.
그래서 패키지명과 버전을 함께 기록하는데 그 역할을 하는게 package.json입니다.

npm init

```
package name: (hooni) hooni
version: (1.0.0) 1.0.0
description: test file
entry point: (.mongorc.js) .mongorc.js
test command: test, hooni,
git repository:
keywords: test, hooni
author: hooni
license: (ISC) ISC
About to write to /Users/hooni/package.json:

{
  "name": "hooni",
  "version": "1.0.0",
  "description": "test file",
  "main": ".mongorc.js",
  "scripts": {
    "test": "test, hooni, "
  },
  "keywords": [
```

npm init 명령어를 실행하면 package.json 파일을 생성해줍니다.
먼저 영어로 설명이 나온뒤 ~~(이래서 영어를 잘해야합니다)~~
프로젝트이름, 버전, 설명, 라이센스 등을 입력하면 위와같이 생성이됩니다.
아래와 같이 실습을 해보겠습니다.

3. 간단한 서버 만들기

```json
{
  "name": "jhoon2816",
  "version": "1.0.0",
  "description": "npm description",
  "main": "server.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "node server.js"
  },
  "keywords": [
    "jhoon2816",
    "npm",
    "nodejs",
    "lecture"
  ],
  "author": "jhoon2816",
  "license": "ISC"
}
```

```
1
2  const http = require('http'); // 서버를 만드는 모듈 불러옴
3
4  http.createServer((request, response) => { // 서버 만드는 메소드
5    console.log('server start!');
6  }).listen(8080);
```

자 서버를 만드려면 위에서 해봤듯이 package.json을 만들어야겠죠?
이제 server.js 파일을 만들어줍니다. 위와 같이 적어주시고, server.js로 저장.
그리고 명령프롬포트에서 node server.js 라고 치면 파일이 실행되며,
server start! 라는 문구가 나오면 정상적으로 실행된것입니다.

4. Express 프레임워크

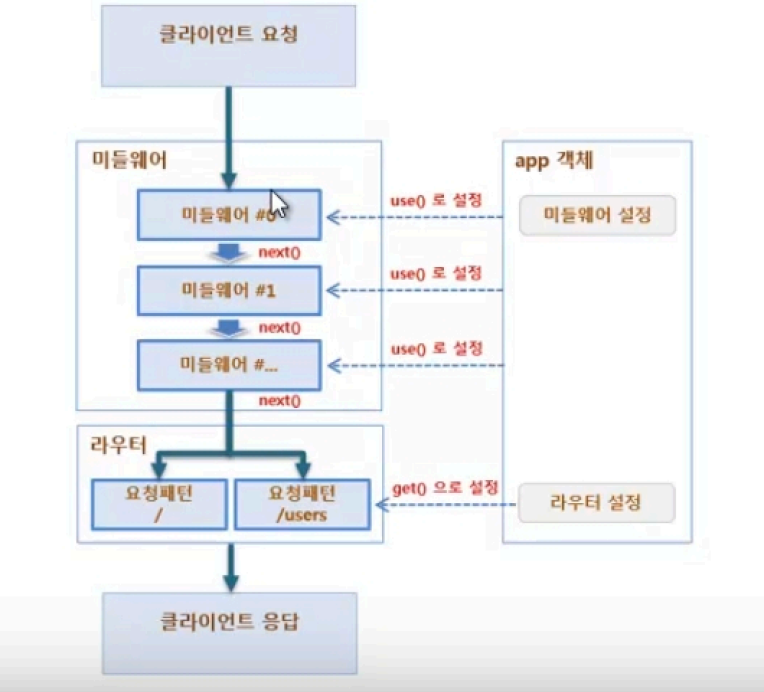
자 드디어 express 에 대해 배울 시간입니다.
일반적으로 노드를 사용했을경우 사이트가 커짐에따라 코드가 복잡해집니다.
하지만 express 프레임워크는 코드의 양도 줄여주고,
추후 유지보수가 쉽게 만들어줍니다.

Express

```
➜ nodeExpress npm install express
(                    ) ∴ extract:express: verb lock using /Users/hooni/.npm/_locks/stag
npm WARN saveError ENOENT: no such file or directory, open '/Users/hooni/nodeExpress/pa
ckage.json'
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN enoent ENOENT: no such file or directory, open '/Users/hooni/nodeExpress/packa
ge.json'
npm WARN nodeExpress No description
npm WARN nodeExpress No repository field.
npm WARN nodeExpress No README data
npm WARN nodeExpress No license field.
```

기본적으로 node.js에서 제공해주는 기본모듈과는 달리,
express는 다른 단체가 만든 패키지이므로 npm 에서 다운로드 받으셔야합니다.
패키지를 받으실때 아래와 같은 경고문이 뜬다는것은 package.json을 안만드신것
임을 알려주는 경고문이니 package.json을 만들고 설치해주시면 됩니다.

5. 미들웨어와 라우터



미들웨어란

웹 요청과 응답에 관한 정보를 사용해 필요한 처리를 진행할수있도록 분리된 독립된 함수.

라우터란

클라이언트의 요청을 보고 이 정보를 처리할수있는곳으로 기능을 전달해 주는 역할을 합니다.

미들웨어

```
1
2   var express = require('express');
3   var http = require('http');
4
5   var app = express();
6
7   app.use(function(req,res, next){
8       console.log('첫 번째 미들웨어에서 요청을 처리함');
9
10      res.writeHead('200',{'Content-Type':'text/html;charset=utf8'});
11      res.end('<h1>Express 서버에서 응답한 결과입니다.</h1>');
12  });
13
14  http.createServer(app).listen(8080,function(){
15  console.log('Express 서버가 8080서버에서 시작됨.');
16  });
17
18
```

미들웨어를 이용하여 express서버를 만드는 가장 간단한 코드입니다.

app.use( ) 메소드 부분이 미들웨어의 역할을 합니다.

이 처럼 요청과 응답의 중간과정에 껴서 어떠한 동작을 해주는것이 미들웨어입니다.

미들웨어의 종류

수많은 종류의 미들웨어들이 있지만, 대표적으로

Morgan : 메세지들을 콘솔에 표시
Compression : 페이지를 압축해서 전송
Session : 세션을 사용하게 해줌
Body-parser : 폼에서 전송되는 POST값을 사용하게해줌
Cookie-parser : 쿠키를 사용할수있게 해줌
Method-override : REST API 에서 PUT과 DELETE 사용가능
Cors : 다른 도메인간의 AJAX 요청 가능
Multer : 파일업로드할때 주로 사용
Static : 특정폴더의 파일을 특정패스로 접근할수있게 해줌

Static을 제외한 나머지 모두 npm으로 설치가 가능합니다.
static 은 express에서 제공해줍니다.
자 그럼 실습을 통해서 미들웨어를 알아봅시다.

6. Express로 웹서버 만들기

```
1  var express = require('express');
2  var mysql = require('mysql') // mysql 모듈 불러오기.   npm install mysql --
3  var app = express();
4
5  var bodyParser = require('body-parser'); //bodyparser모듈은 request body를
6
7  app.use(bodyParser.urlencoded({extended:true}));
8  app.use(bodyParser.json());
9
10 var testApi = require('/test');
11
12 //mysql과 연동 후 데이터 불러오는 법
13 var dbConnection = mysql.createConnection({
14     host: 'localhost', // 접속하고자 하는 URL
15     user: 'root', // 접속하고자 하는 user명 기본이 root
16     password: '0000', // user의 비밀번호
17     database: 'test' // 접속하고자 하는 data Schema. //data Schema란? : DB내에
18 });
19
20 //get : 조회
21 app.get('/test', function(req,res){
22     dbConnection.query('select * from test_table'/*,[id] 조건값 */,functio
23         res.json(rows); // json파일 불러올때 res.json() 메소드 이용, res.end()
24     });
25 });
26
27 dbConnection.query('select * from test'/*[id] 조건값 */,function(err,row,f
28     console.log(row);
```

Practice time!! 실습을 해봅시다!