

Node.js Programming

At a Glance

Node.js

- ▶ Is an event-driven I/O server-side JavaScript Environment (nodejs.org)
- ▶ Is a platform built on Chrome's JavaScript Runtime for easily building fast, scalable network applications. (mean.io)

Node.js

- ▶ Event-Driven
- ▶ Built on V8 runtime (Chrome's JavaScript runtime)
- ▶ Server-side JavaScript Environment
- ▶ For easily building network applications
 - ▶ Fast and scalable

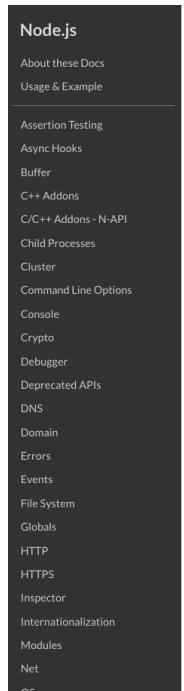
Node.js Homepage



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.

<http://nodejs.org>

Node.js API Document



Node.js v8.2.1 Documentation

[Index](#) | [View on single page](#) | [View as JSON](#)

Table of Contents

- [About these Docs](#)
- [Usage & Example](#)
- [Assertion Testing](#)
- [Async Hooks](#)
- [Buffer](#)
- [C++ Addons](#)
- [C/C++ Addons - N-API](#)
- [Child Processes](#)
- [Cluster](#)
- [Command Line Options](#)
- [Console](#)
- [Crypto](#)
- [Debugger](#)
- [Deprecated APIs](#)
- [DNS](#)
- [Domain](#)
- [Errors](#)
- [Events](#)
- [File System](#)
- [Globals](#)
- [HTTP](#)
- [HTTPS](#)
- [Inspector](#)
- [Internationalization](#)
- [Modules](#)
- [Net](#)
- [os](#)
- [process](#)
- [util](#)

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

<https://nodejs.org/dist/latest-v6.x/docs/api/>

MEAN Stack

- ▶ MongoDB
- ▶ Express
- ▶ Angular
- ▶ Node.js

MEAN stands for:



MongoDB is the leading NoSQL database, empowering businesses to be more agile and scalable.

express

Express is a minimal and flexible node.js web application framework, providing a robust set of features for building single and multi-page, and hybrid web applications.



AngularJS lets you extend HTML vocabulary for your application. The resulting environment is extraordinarily expressive, readable, and quick to develop.



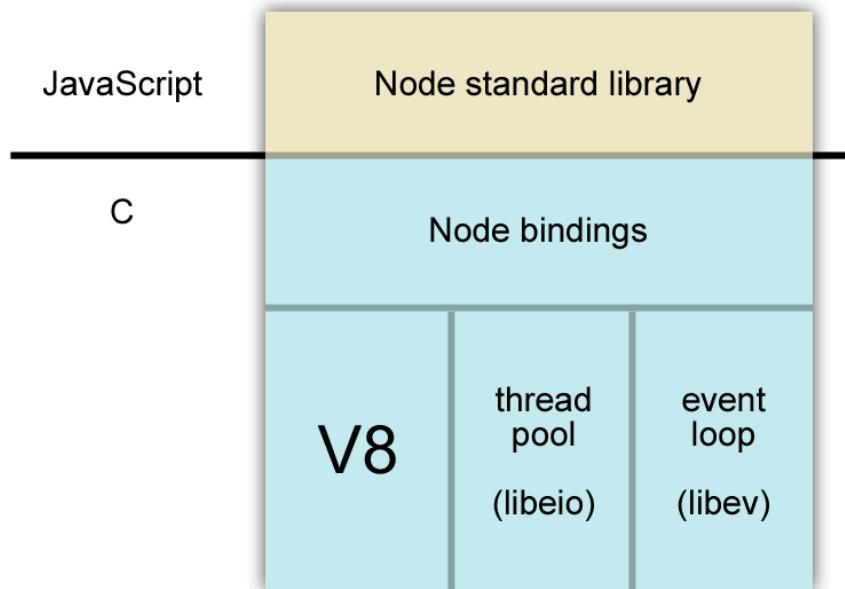
Node.js is a platform built on Chrome's JavaScript runtime for easily building fast, scalable network applications.

<http://mean.io/>

Node.js 특징

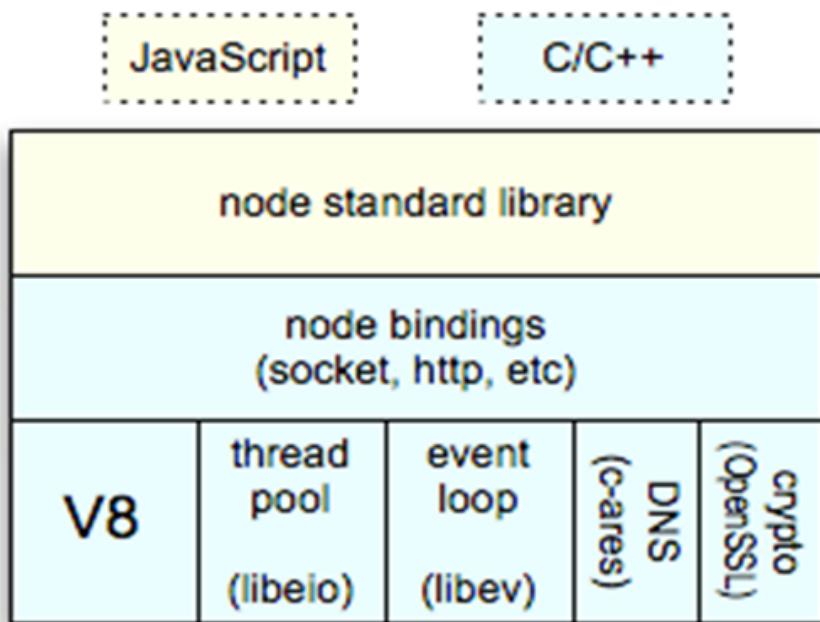
- ▶ 모듈 기반 (필요한 것들을 추가하여 사용할 수 있다)
 - ▶ NPM (Node Package Manager)
- ▶ Non-Blocking (Event-Driven) I/O
- ▶ V8 JavaScript runtime
 - ▶ Microsoft는 Edge 브라우저의 Chakra Core로 변경할 수 있도록 작업중

Node.js Architecture



Packt Pub

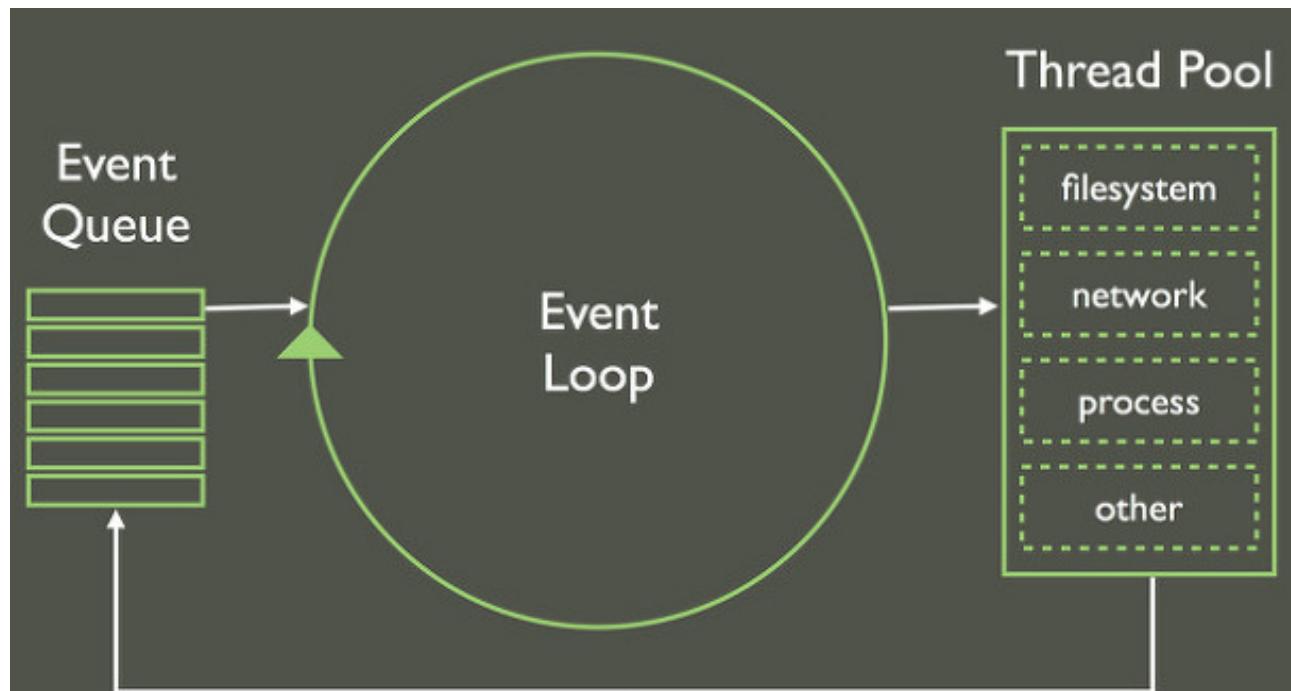
Node.js Architecture



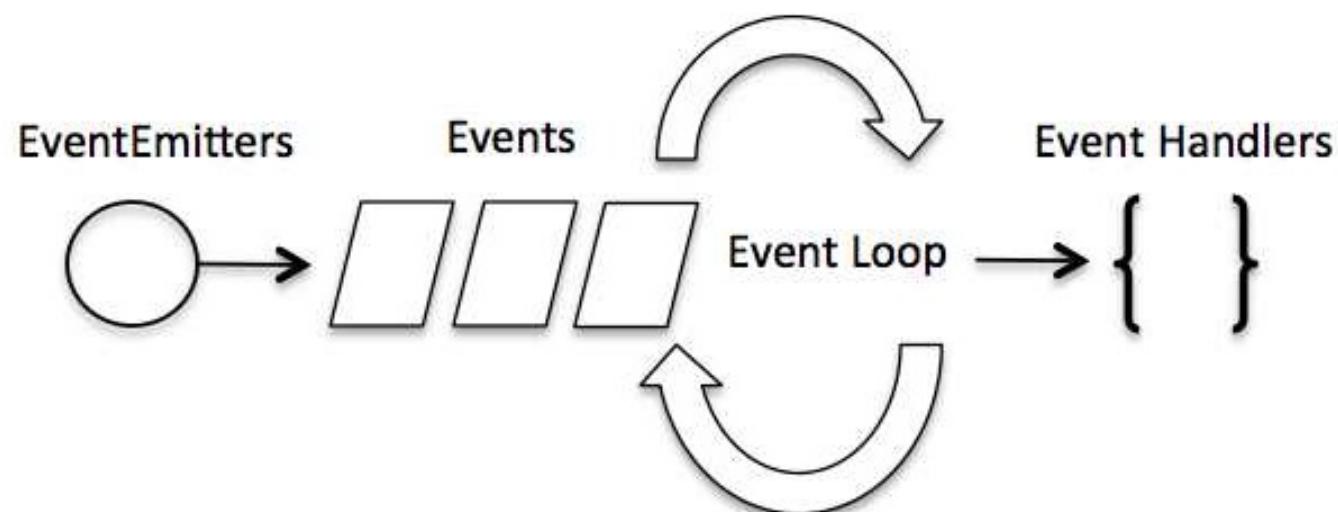
CommonJS, V8 Runtime, 그리고 Node.js

- ▶ 웹브라우저를 벗어나 JavaScript를 사용하려는 다양한 시도들이 발생
- ▶ LiveWire, Jaxer 등 서버 환경에서 JavaScript를 사용하려는 초기의 다양한 시도들
 - ▶ '속도' 문제가 발목을 잡음
- ▶ 2008년 Google Chrome Browser 발표
 - ▶ V8 JavaScript Runtime으로 JavaScript 실행 속도를 획기적으로 개선
- ▶ 2009년, ServerJS 프로젝트 시작
 - ▶ 빨라진 JavaScript를 기반으로 웹브라우저 이외의 곳에서 사용할 수 있는 표준을 만드는 프로젝트
 - ▶ CommonJS (<http://www.commonjs.org>)로 변경
- ▶ CommonJS 표준과 V8 Runtime을 기반으로 Ryan Dahl이 Node.js 개발

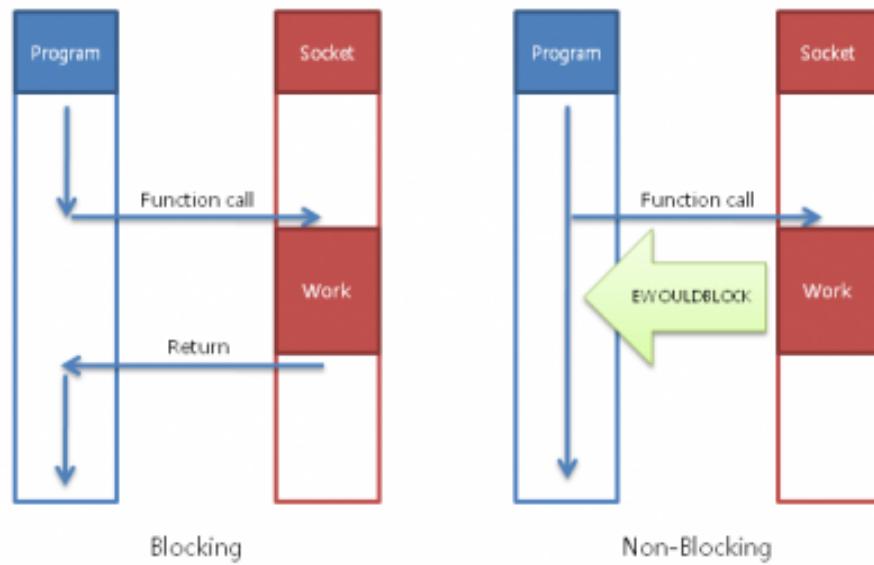
Event-Driven (Asynchronous)



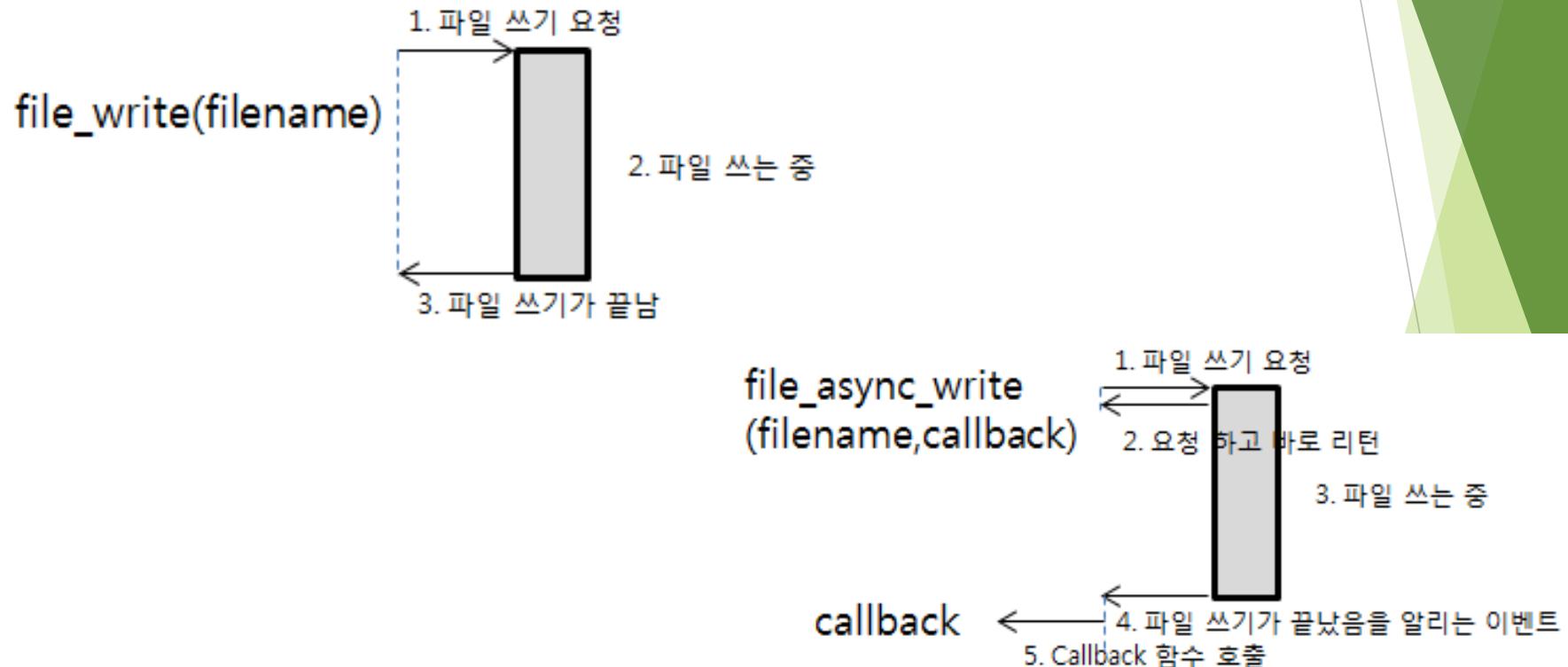
Event-Driven (Emitters and Handler)



Non-Blocking I/O



Asynchronous Way and Callback



Node.js의 장점

- ▶ 빠르고 확장 가능한 대규모 네트워크 프로그램을 쉽게 개발할 수 있다
- ▶ 사용자 풀이 많고 유연한 JavaScript를 개발 언어로 채택
- ▶ 성능이 검증되고 향상된 V8 Runtime 엔진 사용
 - ▶ Microsoft는 Chakra Core를 Node.js 엔진으로 사용할 수 있도록 개발중
- ▶ C++을 사용하여 기능을 확장할 수 있음
- ▶ CPU Intensive한 작업이 없고
- ▶ 많은 Connection을 동시에 처리해야 하는 시나리오에 적합

Node.js의 단점

- ▶ V8의 성능이 좋아지긴 했으나 여전히 C, C++, Java로 개발된 것보다는 느림
- ▶ Single Thread 방식이라 멀티코어 CPU의 성능을 충분히 활용하기 힘들고, 일부의 오류가 서버 전체에 영향을 미칠 수 있다.

Node.js 장단점 요약

- ▶ 개발 관점에서는 빠르고 쉬운 장점
- ▶ 운영 관점에서는 테스트, 장애 대응, 디버깅 등에 신경써야 할 부분이 많음

Use

- ▶ Prototyping에 빠르다
- ▶ Async I/O를 사용하므로 file upload/download와 같은 network streaming 서비스에 유리하다
- ▶ Realtime Web Application (채팅, 메시징 앱 등)
- ▶ Single Page App 개발에 유리 (Angular 등)
- ▶ 가볍고 생산성이 높은 웹 개발 프레임워크, 간단한 로직, 빠른 응답 시간을 요구하는 애플리케이션 개발에 적절

DO NOT USE

- ▶ CPU 작업이 많은 애플리케이션
- ▶ CRUD가 많고 페이지가 많은 웹 개발
- ▶ 운영 관점에서 Trouble Shooting이 어려울 수 있으며
- ▶ 하나의 코드가 잘못되어 시스템이 느려지게 되면 Request 처리에 문제가 있을 수 있다.

So... Who use Node.js on the server side?

Node Js Development Node.js JavaScript (programming language) +2

What are the biggest websites built with Node.js on the server side?

Answer

Request ▾

Follow 406 Comment Share 2 Downvote

...



Gaëtan Voyer-Perrault, Principal Engineer @ROBLOX, previously
@Dynamic Signal & @Adknowledge
Updated Feb 28, 2016



Might be worth updating this to include stuff in 2013:

- Walmart (Why Walmart is using Node.js)
- E-bay / PayPal (Node.js at PayPal)
- Microsoft (Node.js Dev Center | Windows Azure) They have extensive support, in fact their Azure CLI tools are actually written in Node.js.
- LinkedIn (Exclusive: How LinkedIn used Node.js and HTML5 to build a better, faster app)
- Yahoo
- Google (Node at scale: What Google, Mozilla, & Yahoo are doing with Node.js)
- Yammer (now part of Microsoft) (Managing Node.js Dependencies and Deployments at Yammer - Yammer Engineering)
- Netflix has a series of blog posts on its use (Search results for node.js) (Updated 2016-02)
- Looks like Uber is using it as well (Updated 2016-03)

There are obviously a lot more, but that list has the majority of the major non-Chinese tech companies, so I think that qualifies for "biggest".

Yes, clearly, some companies are using them in very specific places, but you don't rewrite the Microsoft dot com website in a year.

If you take a look at the Walmart and PayPal examples, they are both using it in very key places.

<https://www.quora.com/What-are-the-biggest-websites-built-with-Node-js-on-the-server-side>

Node.js Programming

개발 환경 구축

Node.js 설치



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 releases](#), please update now!

Download for macOS (x64)

[v6.11.1 LTS](#)

Recommended For Most Users

[v8.2.1 Current](#)

Latest Features

[Other Downloads](#) | [Changelog](#) | [API Docs](#) [Other Downloads](#) | [Changelog](#) | [API Docs](#)

Or have a look at the [LTS schedule](#).

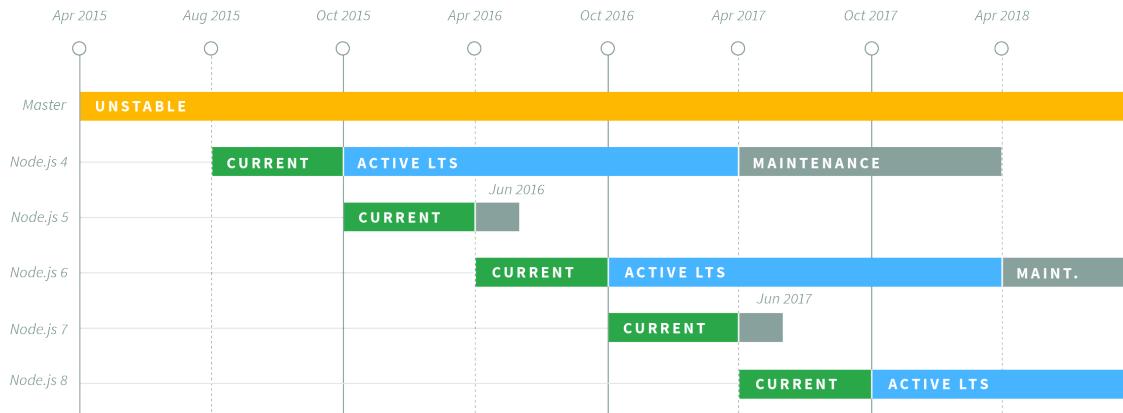
Sign up for [Node.js Everywhere](#), the official Node.js Weekly Newsletter.

Node.js 버전 선택 가이드

- ▶ LTS vs Current
 - ▶ LTS (Long-Term Support) : 장기 지원을 약속하는 버전
 - ▶ Current : 가장 최신 버전
- ▶ 홀수 버전 vs 짝수 버전
 - ▶ 홀수 버전 : 개발 버전
 - ▶ 짝수 버전 : 릴리즈 버전
- ▶ 실제 서비스에 적용할 예정이면 LTS 버전 중, 가장 최신으로 선택 권장

Node.js LTS Release Schedule

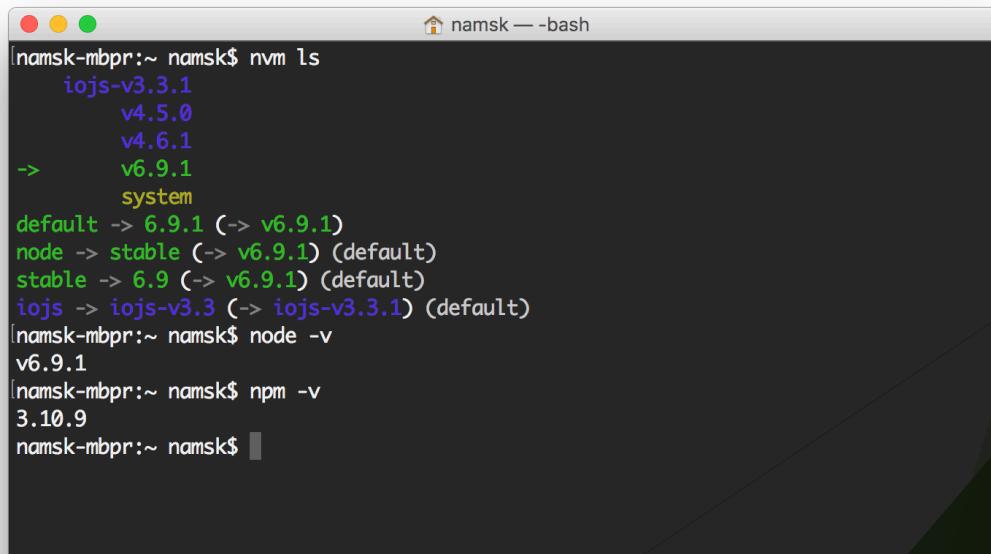
Node.js Long Term Support (LTS) Release Schedule



COPYRIGHT © 2017 NODESOURCE, LICENSED UNDER CC-BY 4.0

NVM (Unix or Linux)

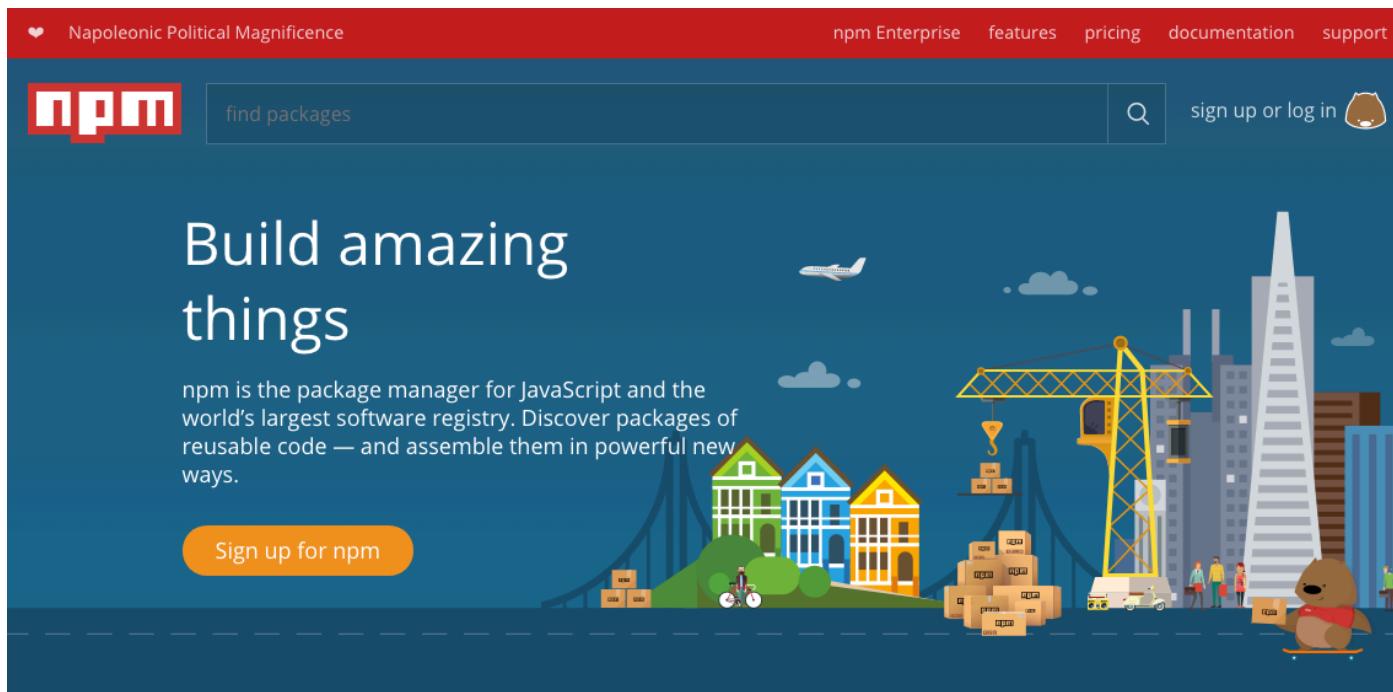
- ▶ <http://nvm.sh>
- ▶ 필요에 따라 여러 버전의 node를 설치, 실행할 수 있다



```
[namsk-mbpr:~ namsk$ nvm ls
  iojs-v3.3.1
    v4.5.0
    v4.6.1
  ->  v6.9.1
      system
default -> 6.9.1 (-> v6.9.1)
node -> stable (-> v6.9.1) (default)
stable -> 6.9 (-> v6.9.1) (default)
iojs -> iojs-v3.3 (-> iojs-v3.3.1) (default)
[namsk-mbpr:~ namsk$ node -v
v6.9.1
[namsk-mbpr:~ namsk$ npm -v
3.10.9
namsk-mbpr:~ namsk$ ]
```

NPM (Node Package Manager)

<https://www.npmjs.com>

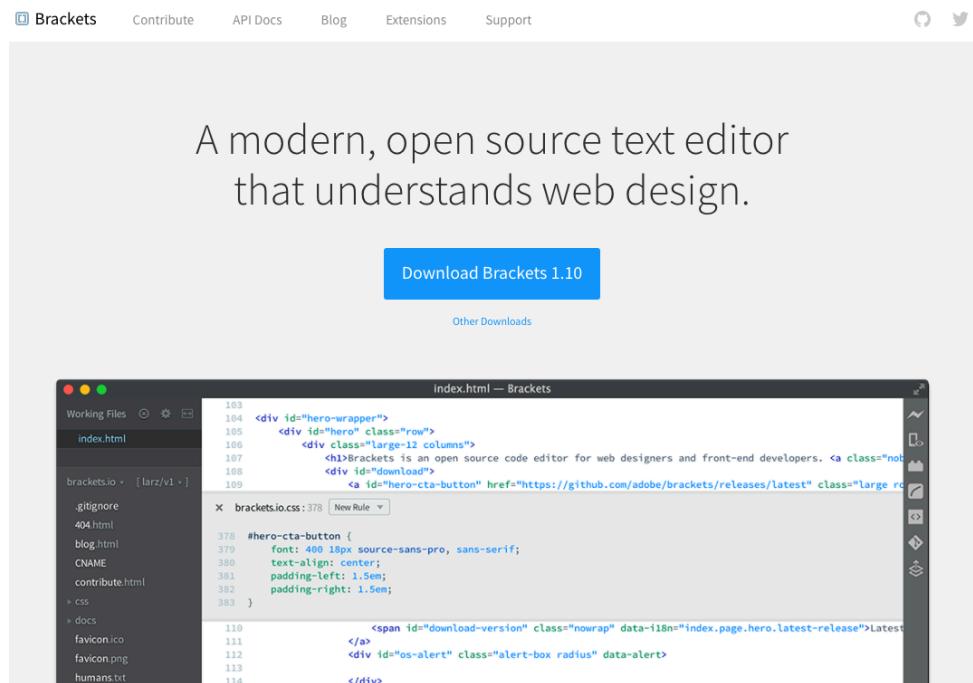


NPM (Node Package Manager)

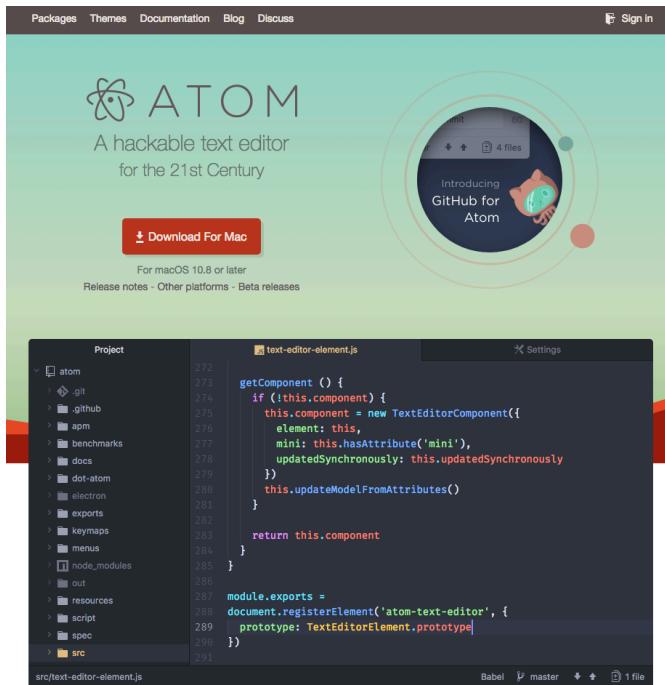
- ▶ Is the package manager for JavaScript
- ▶ `npm install {package-name}` # install local package
- ▶ `npm install -g {package-name}` # install global package

IDE (Brackets)

<http://brackets.io/>



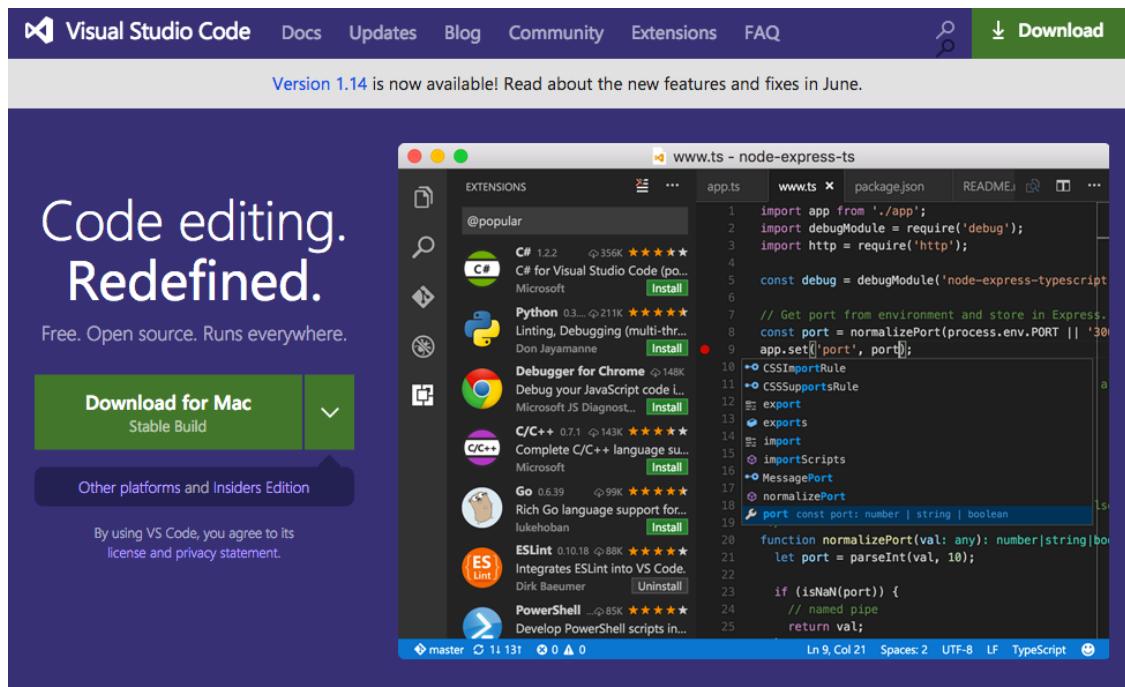
IDE (Atom)



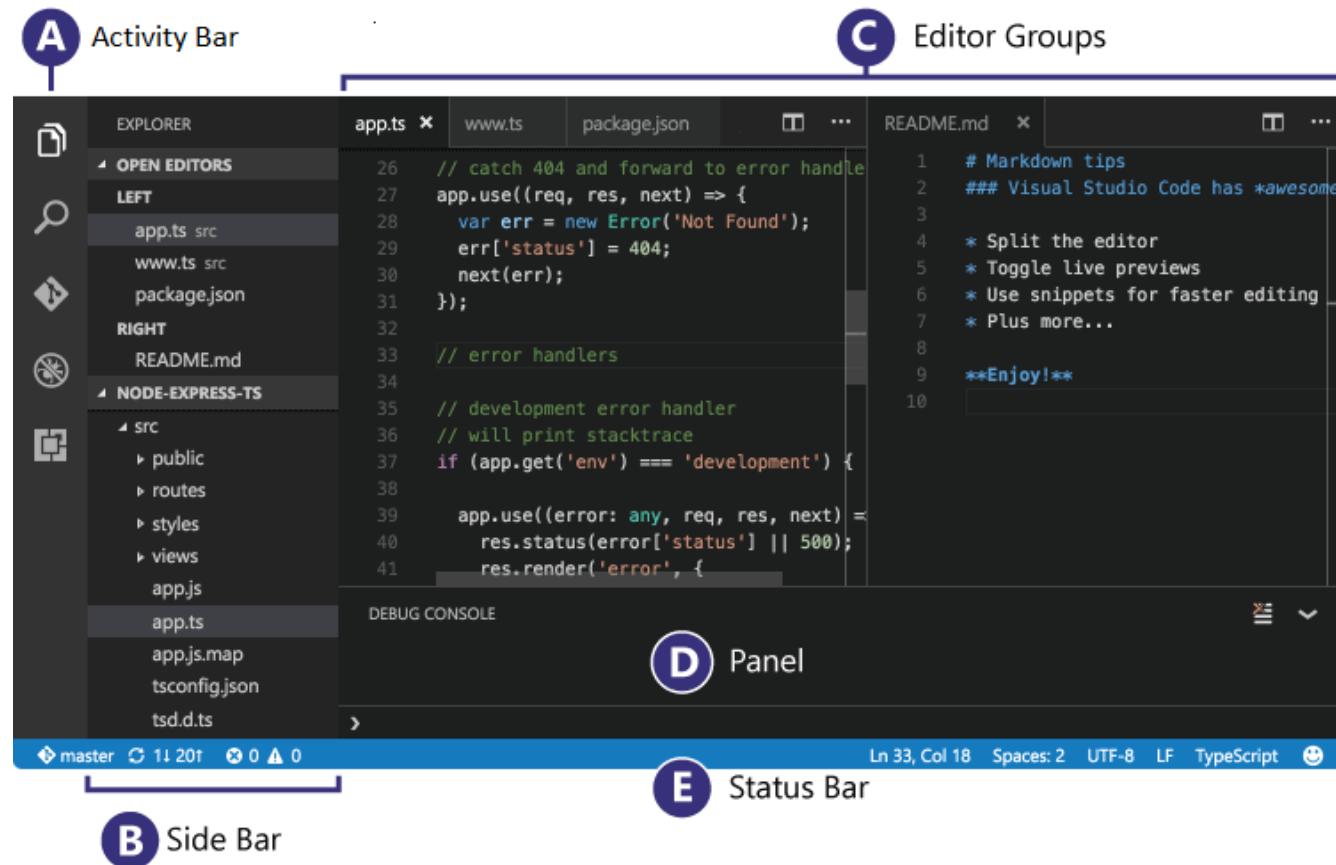
<https://atom.io/>

IDE (Visual Studio Code)

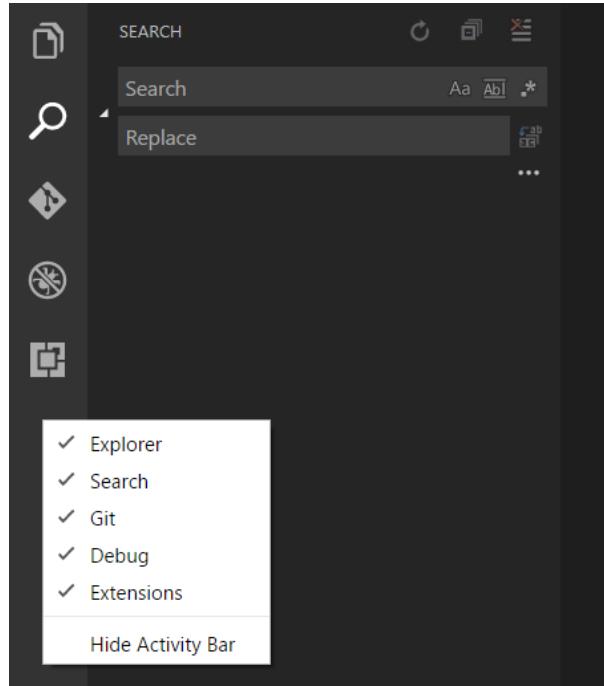
<https://code.visualstudio.com/>



Introducing Visual Studio Code



Introducing Visual Studio Code



Node.js, npm 버전 확인

- ▶ Command Shell에서
 - ▶ node -v
 - ▶ npm -v

Node REPL

- ▶ REPL = READ-Eval-Print-Loop
- ▶ 명령줄에서 직접 커맨드를 입력하고 코드를 테스트 할 수 있는 커맨드 라인 개발 환경
- ▶ 커맨드 라인에서 다음 커맨드 입력
 - ▶ node

첫번째 Node.js 프로그램

- ▶ 다음 텍스트를 편집기에 입력
- ▶ 저장 (예: app.js)
- ▶ Command Line에서 실행할 파일명과 함께 node 명령어 입력
 - ▶ node app.js

```
1  console.log("Hello, Node.js");
```

Mission 1

- ▶ Node.js를 설치, node와 npm의 버전을 확인해 보자
- ▶ Node REPL을 실행하여 JavaScript를 입력, 코드를 확인해 보자
- ▶ 자신의 취향에 맞는 IDE를 선택, 설치하고 실제로 JavaScript 코드를 작성한 후 node로 실행해보자

Mission 2 (Optional)

- ▶ 리눅스 시스템에 nvm을 설치하여
 - ▶ 가장 최신의 LTS 버전과
 - ▶ 가장 최신의 Current 버전을 동시에 설치해보자

Node.js Programming

General

Global 객체

- ▶ 브라우저가 아니므로 window 객체가 없다.
- ▶ 대신 global 객체로 이용

```
global.val = 'Global Variable';
global.func = () => {
  console.log('Global Function');
};

console.log(val);
func();
```

Global Variables

변수명	설명
<code>__filename</code>	현재 실행중인 코드의 파일 경로
<code>__dirname</code>	현재 실행중인 코드의 폴더 경로

Example

`/001_general/001_global.js`

Global Objects

객체명	설명
console	콘솔 화면 관련 기능 수행
process	프로그램과 프로세스 관련 기능 수행
exports	모듈과 관련된 기능 수행 (CommonJS)

Console 객체

```
> console.error("Error");
✖ ▶ Error
< undefined
> console.warn("Warning");
⚠ ▶ Warning
< undefined
> console.debug("Debug");
< undefined
> console.log("Log");
  Log
< undefined
> console.info("Information");
  Information
< undefined
> console.assert(1 == 2);
✖ ▶ Assertion failed: console.assert
< undefined
> |
```

Console 객체 메서드

메서드명	설명
error	에러 출력 (x) 표시
warn	경고 출력 (!) 표시
debug	디버깅 관련 정보
log	로그 출력
info	정보 출력
assert	검증(Assertion)
time(tag)	시간 측정 시작
timeEnd(label)	시간 측정 종료

Console의 특수문자

특수문자	설명
%d	숫자
%s	문자열
%j	JSON
%%	% 문자 자체

- util.format 함수에서도 동일한 특수문자 사용 가능

Example

/001_general/002_console.js

Process 객체의 속성

속성명	설명
argv	실행 매개변수
env	실행 환경 관련 정보
version	Node의 버전
versions	종속된 프로그램 버전
arch	프로세서의 아키텍쳐 표시
platform	플랫폼 정보 표시

Process 객체의 메서드

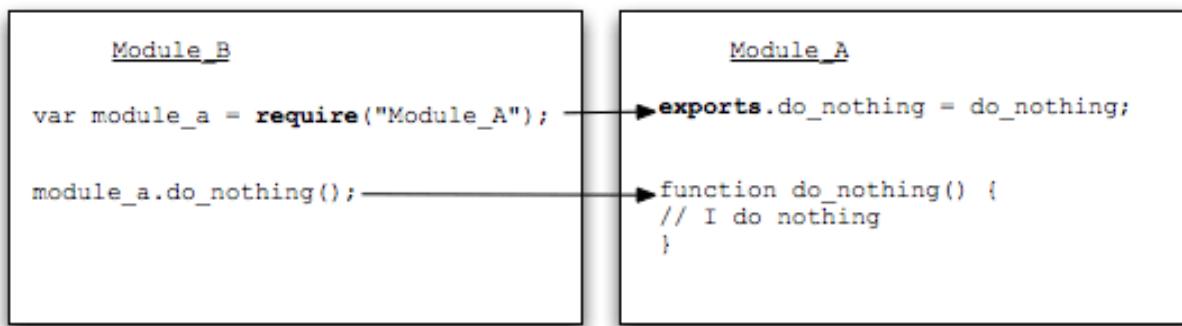
메서드명	설명
exit	프로그램 종료
memoryUsage	메모리 사용 정보 객체 반환
uptime	현재 프로그램이 실행된 시간

Examples

`/001_general/003_process.js`

`/001_general/004_process_methods.js`

Exports 객체



Exports 방법 1

```
exports.add = function(num1, num2) {
  return num1, num2;
};

exports.square = function(length) {
  return length * length;
};
```

Exports 방법 2

```
var area = {
  square: function(length) {
    return length * length;
  },
  circle: function(radius) {
    return radius * radius * Math.PI;
  },
  rectangle: function(width, height) {
    return width * height;
  }
}

module.exports = area;
```

모듈의 사용 방법

```
var area = require('./modules/area_module');

console.log('area_module.square : %d', area.square(6));
console.log('area_module.circle : %d', area.circle(8));
console.log('area_module.rectanble : %d', area.rectangle(5,6));
```

Example

/001_general/005_exports.js

Mission 1

- ▶ console 객체를 이용하여 다양한 로그 메시지를 출력해보자

Mission 2

- ▶ process 객체를 이용하여 주기적으로 CPU 상태를 감시하는 프로그램을 만들어 보자

Mission 3

- ▶ `exports` 객체를 이용하여 코드를 모듈로 분리하고, 해당 모듈을 불러와 사용하는 코드를 작성해 보자

Node.js Programming

Built-in Modules

Os module

```
var os = require('os');
```

메서드명	설명
hostname	운영체제의 호스트명을 반환
type	운영체제의 이름을 반환
platform	운영체제의 플랫폼을 반환
arch	운영체제 아키텍처 반환
release	운영체제의 버전을 반환
uptime	운영체제 실행된 시간을 반환
loadavg	로드 평균값 정보 반환 (Array)
totalmem	운영체제 총 메모리 사이즈 반환
freemem	시스템 사용 가능한 메모리 반환
cpus	CPU 정보 반환 (Array)
getNetworkInterfaces	네트워크 인터페이스 정보 반환 (Array)

Example

`/002_builtin-modules/001_os_module.js`

Url Module

```
var url = require('url');
```

메서드명	설명
parse	URL 문자열을 URL 객체로 변환
format	URL 객체를 URL 문자열로 변환
resolve	매개변수를 조합, URL 문자열을 생성하여 반환

url.parse

```
var urlString = "https://search.naver.com/search.naver?sm=tab_hty.top&where=nexearch&query=node.js&oquery=iphone+8&ie=utf8&tqi=TU1mhdpSoGssZ0XzKdssssn4-295326";

var parseObject = url.parse(urlString);
console.log(parseObject);
```

url.parse 2

```
var parseObjectEx = url.parse(urlString, true); // 두 번째 인자가 true면 query string을 object로 변환
console.log(parseObjectEx);
```

Example

`/002_builtin-modules/002_url_module.js`

Query String Module

```
var querystring = require('querystring');
```

메서드명	설명
parse	쿼리 문자열을 쿼리 객체로 변환
stringify	쿼리 객체를 쿼리 문자열로 변환

- url 모듈로도 대부분의 동일 기능 처리 가능

Example

`/002_builtin-modules/003_querystring_module.js`

Crypto Module: Hash

```
var crypto = require('crypto');
```



createHash → update → digest

Crypto Module: Hash

```
// Create Hash
// var shahash = crypto.createHash('sha1');
var shahash = crypto.createHash('sha256');
shahash.update('hashcode');

// Print Out Hash
console.log(shahash.digest('hex'));
```

Crypto Module: Encoding



Crypto Module: Encoding

```
// Encoding Example
console.log('Original input String:', inputString);
var cipher = crypto.createCipher('aes192', secret);
cipher.update(inputString, 'utf8', 'base64');
var ciphered = cipher.final('base64');
console.log('aes192 ciphered:', ciphered);
```

Crypto Module: Decoding



Crypto Module: Decoding

```
// Decoding Example
var decipher = crypto.createDecipher('aes192', secret);
decipher.update(ciphered, 'base64', 'utf8');
var deciphered = decipher.final('utf8');
console.log('aes192 deciphered:', deciphered);
```

Example

`/002_builtin-modules/004_crypto_module.js`

File System Module

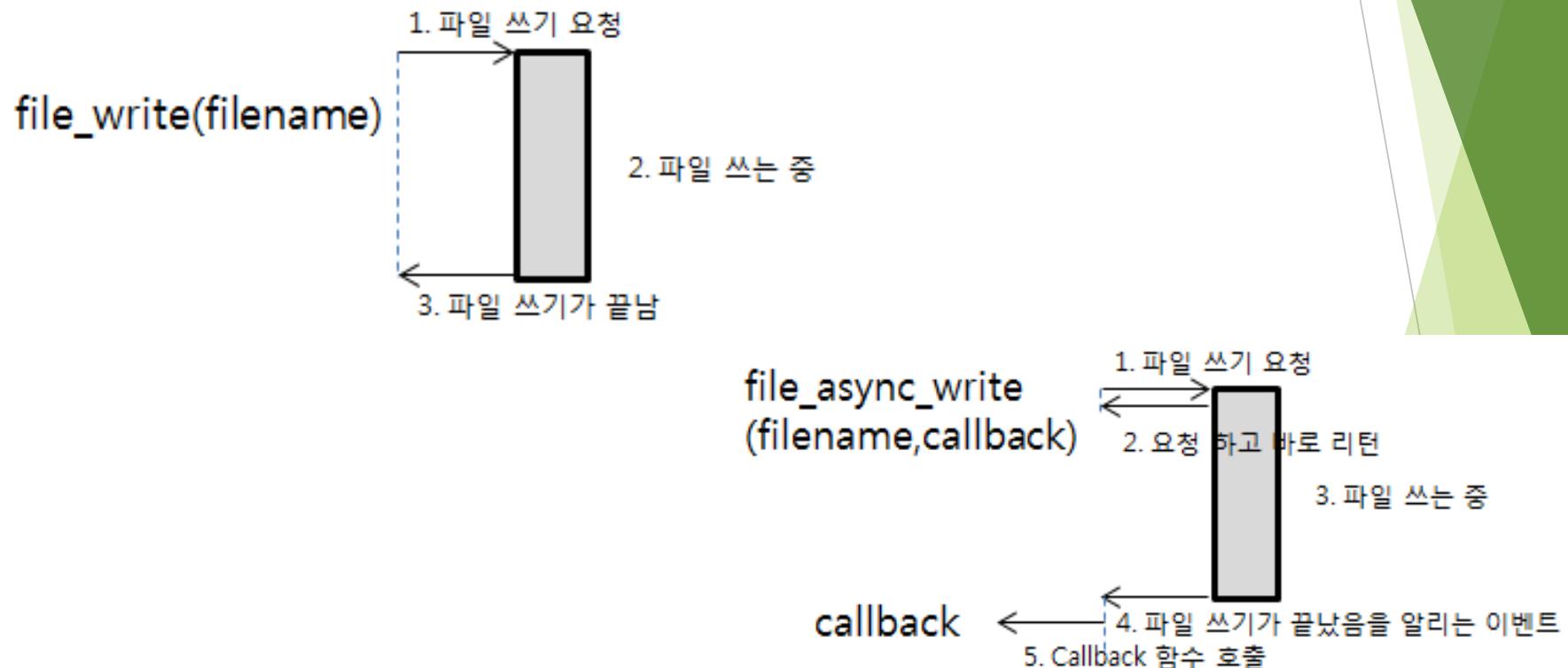
```
var fs = require('fs');
```

메서드명	설명
readFile(file, encoding, callback)	비동기적 파일 읽기
readFileSync(file, encoding)	동기적 파일 읽기
writeFile(file, encoding, callback)	비동기적 파일 쓰기
writeFileSync(file, encoding)	동기적 파일 쓰기
appendFile(file, encoding, callback)	비동기적 파일 추가
appendFileSync(file, encoding)	동기적 파일 추가

File System: Error Handling

- ▶ 동기적 쓰기/읽기
 - ▶ try ~ catch 이용
- ▶ 비동기적 쓰기/읽기
 - ▶ Callback 함수에서 error 처리

File System 객체: Process



Examples

[`/002_builtin-modules/005-1_file_read_sync.js`](#)
[`/002_builtin-modules/005-1_file_read_async.js`](#)
[`/002_builtin-modules/005-1_file_write_sync.js`](#)
[`/002_builtin-modules/005-1_file_write_async.js`](#)

Node.js Programming

Event At A Glance

EventEmitter

- ▶ Deprecated
- ▶ Using `require('events')` instead
- ▶ <https://nodejs.org/dist/latest-v6.x/docs/api/events.html>

Event 연결 메서드

메서드명	설명
addEventListener	이벤트 연결
on	이벤트 연결 (추천)

이벤트 제거 메서드

메서드명	설명
removeListener(이벤트명, 핸들러)	특정 이벤트의 리스너 제거
removeAllListeners([이벤트명])	모든 이벤트의 리스너 제거

이벤트 강제 발생

메서드명	설명
<code>emit(이벤트명[, args ...])</code>	이벤트를 실행함

Custom Event

메서드명	설명
addEventListener(이벤트명, 핸들러)	이벤트 연결
on(이벤트명, 핸들러)	이벤트 연결 (추천)
setMaxListeners(개수)	이벤트 연결 개수를 조절
removeListener(이벤트명, 핸들러)	특정 이벤트의 리스너를 제거
removeAllListeners([이벤트명])	모든 이벤트 리스너를 제거
once(이벤트명, 이벤트 핸들러)	1회성 이벤트 연결

```
const customEvent = require('events');
```

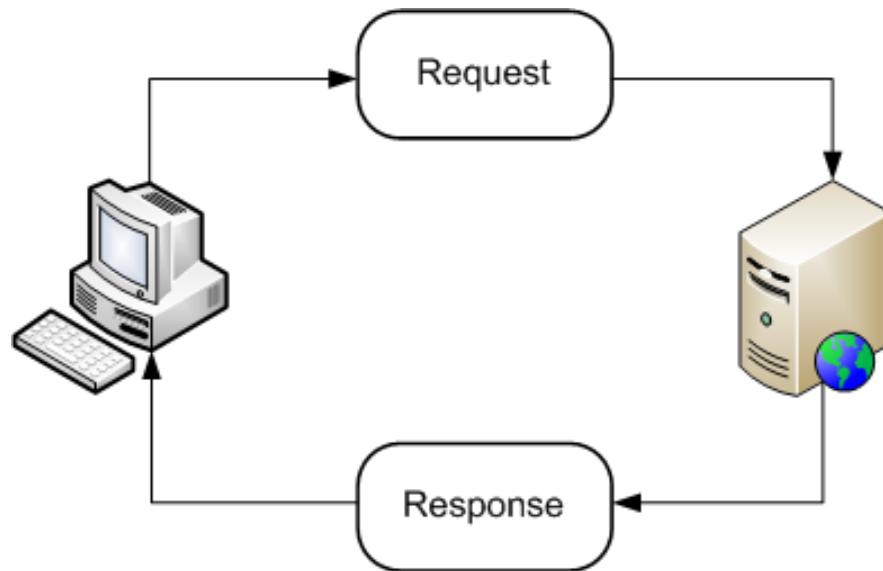
Examples

[`/003_events/001_custom_event.js`](#)

Node.js Programming

HTTP Server using http module

Request and Response



Http Module

메서드명	설명
listen(port[, callback])	서버를 실행함
close()	서버를 종료함

Very Simple Http Server

```
var http = require('http');

var server = http.createServer();

server.listen(3000);
```

Examples

`/004_http/001_http_module.js`

Request 객체

속성	설명
url	Request의 URL 객체를 반환
method	Request의 HTTP Method

Response 객체

메서드명	설명
writeHead(code, object)	응답 헤더를 작성
end([data[, encoding]])	응답 본분을 작성

Client에 응답하기

```
http.createServer((request, response) => {
  response.writeHead(200, {'Content-Type': 'text/html; charset=utf8'});
  response.end("<h1>Hello from Node.js</h1>");
}).listen(3000, () => {
  console.log("Server is listening on localhost:3000");
});
```

Serving Resources (Example)

```
const http = require('http');
const fs = require('fs');

const server = http.createServer((req, res) => {
  fs.readFile("./resources/nodejs-interactive.png", (error, data) => {
    if (error) {
      res.writeHead(404, {"Content-Type": "text/html;charset=utf8"});
      res.end("<h1>404 Not Found</h1>");
    } else {
      res.writeHead(200, {"Content-Type": "image/png"});
      res.end(data);
    }
  });
}).listen(3000, () => {
  console.log("Server is running on localhost:3000");
})
```

Examples

`/004_http/002_basic_http_server.js`

`/004_http/002-1_serving_resources.js`

Mission 1

- ▶ http 모듈을 사용하여 request에 응답하여 메시지를 응답하는 간단한 웹서버를 만들어보자

응용미션 1

- ▶ http 모듈을 사용하여 간단한 웹 서버를 만든 후, 서버상의 이미지를 응답하는 응용프로그램을 만들어 보자
 - ▶ Hint 1: Content-Type
 - ▶ Hint 2: File System Module

응용미션 2

- ▶ 현재까지 작성한 Simple Http Server는 단일 요청에 대한 응답만 할 수 있다
- ▶ 이를 개선하여 다음의 요청에 따라 다르게 응답하는 서버를 만들어 보자
 - ▶ / => Homepage
 - ▶ /hello => Hello from Node.js
 - ▶ /welcome => Welcome
- ▶ Hint 1: url 모듈

Get Query String (Example)

```
const http = require('http');
const url = require('url');

http.createServer((request, response) => {
  const query = url.parse(request.url, true).query;
  response.write("query : ");
  response.end(JSON.stringify(query));
}).listen(3000, () => {
  console.log("Server is listening on localhost:3000");
})
```

Post Body (Example)

```
const http = require('http');
const url = require('url');
const fs = require('fs');

http.createServer((request, response) => {
  if (request.method == 'GET') {
    fs.readFile("./html/005_page.html", (error, data) => {
      response.writeHead(200, {'Content-Type': 'text/html;charset=utf8'});
      response.end(data);
    });
  } else if (request.method == "POST") {
    request.on("data", (data) => {
      response.writeHead(200, {'Content-Type': 'text/html;charset=utf8'});
      response.end("<h1>" + data + "</h1>");
    });
  } else {
    response.writeHead(200, {'Content-Type': 'text/html;charset=utf8'});
    response.end("<h1>Cannot Response on Request");
  }
}).listen(3000, () => {
  console.log("Server is listening on localhost:3000");
});|
```

Examples

`/004_http/004_request_query.js`

Node.js Programming

NPM At a Glance

NPM

- ▶ Node Package Manager
 - ▶ Node.js 패키지/모듈 저장서
 - ▶ Node.js 패키지 설치 및 버전/호환성 관리를 할 수 있는 Command Line Interface

NPM 주요 명령어

명령어	설명
init	패키지 정보 설정
install	패키지 설치
uninstall	패키지 삭제
update	패키지 업데이트
search	패키지 검색

package.json

- ▶ 현재 노드 애플리케이션 패키지의 정보를 담고 있음

```
{  
  "name": "socket.io.example",  
  "version": "1.0.0",  
  "description": "",  
  "main": "index.js",  
  "scripts": {  
    "test": "echo \\\"Error: no test specified\\\" && exit 1"  
  },  
  "author": "",  
  "license": "ISC",  
  "dependencies": {  
    "socket.io": "^2.0.3"  
  }  
}
```

npm init

- ▶ npm init
- ▶ 개발자로부터 애플리케이션의 정보를 입력받고
- ▶ 피키지 관리를 위한 package.json 파일을 생성

npm search

- ▶ npm search {패키지명}
- ▶ 해당 패키지명을 포함한 패키지를 검색
- ▶ 예) npm search express

npm install

- ▶ `npm install {패키지명}`
 - ▶ 패키지명을 가진 모듈을 설치
- ▶ `npm install {패키지명} -g`
 - ▶ 패키지명을 가진 모듈을 글로벌로 설치
- ▶ `--save` 옵션을 사용하면 `package.json` 파일 내 `dependencies` 항목을 자동으로 갱신함

npm uninstall

- ▶ npm uninstall {패키지명}
- ▶ {패키지명}을 가진 모듈을 삭제

npm update

- ▶ npm update {패키지명}
- ▶ {패키지명}을 가진 모듈을 업데이트

npm {script명}

- ▶ npm {script명}
- ▶ package.json 내에 있는 스크립트를 실행

Mission 1

- ▶ npm init 으로 myapp / v0.9 앱을 설정하고 다음의 패키지를 설치해 보자
- ▶ express, ejs
 - ▶ 조건: 설치된 모듈들은 package.json에 반영되어야 한다

Node.js Programming

Express

패키지 설정 & Express 설치

- ▶ mkdir express-spt
- ▶ cd express-spt
- ▶ npm init
- ▶ npm install express --save

Very Simple Express Server

```
const express = require('express');

const app = express();

app.listen(3000, () => {
  console.log("Express is running on localhost:3000");
});
```

Examples

`/005_express/001_simple_express.js`

Express Server Methods

메서드명	설명
set(key, value)	서버 설정을 위한 속성 지정
get(key)	서버 설정 속성을 불러옴
use([path,] function [, function ...])	미들웨어 함수 사용
get(path, callback)	GET 메서드의 요청을 처리
post(path, callback)	POST 메서드의 요청을 처리
all(path, callback)	모든 요청을 처리

Set and Get

- ▶ `app.set('port', process.env.PORT || 3000);`
- ▶ `let port = app.get('port');`

Get, Post 요청에 응답하기

- ▶ GET 요청에 응답
 - ▶ `app.get({path}, (request, response) => { // response Logic });`
- ▶ POST 요청에 응답
 - ▶ `app.post({path}, (request, response) => { // response Logic });`

Request Parameter 사용

- ▶ Request path에 :{파라미터명} 형식으로 사용할 수 있다.
- ▶ 지정된 파라미터는 다음과 같은 방식으로 읽어서 사용할 수 있다.

```
app.get('/sayhello/:name', (req, res) => {
  res.status(200);
  let name = req.params.name;

  res.send(`<h1>Hello, ${name}</h1>`);
});
```

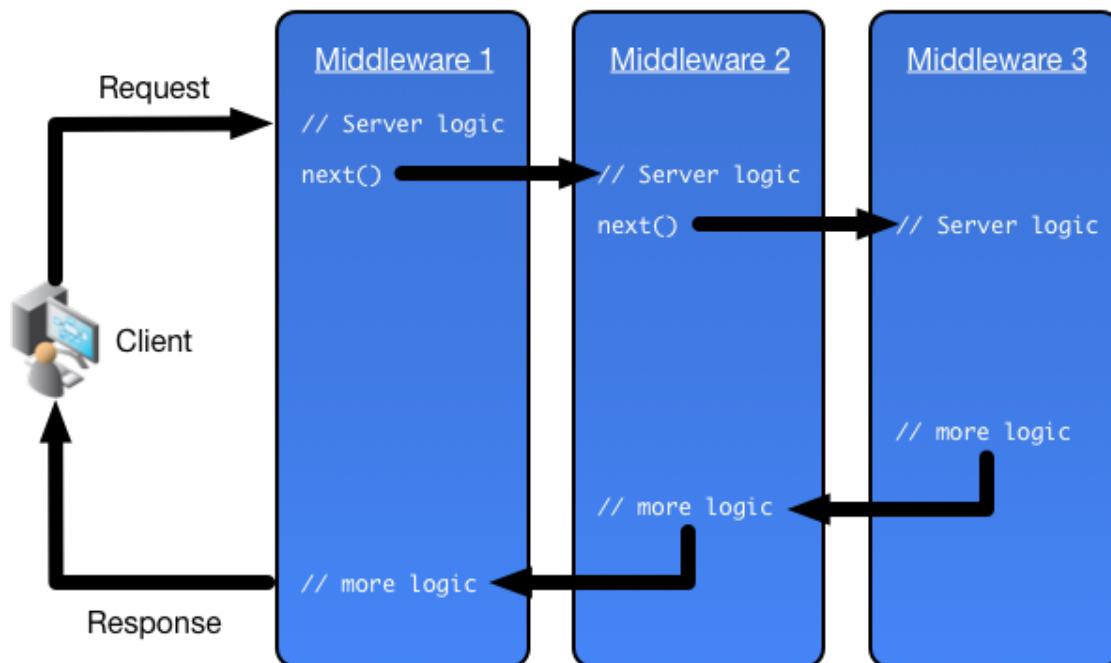
Examples

/005_express/001-3_express_get_params.js

Express Server Properties

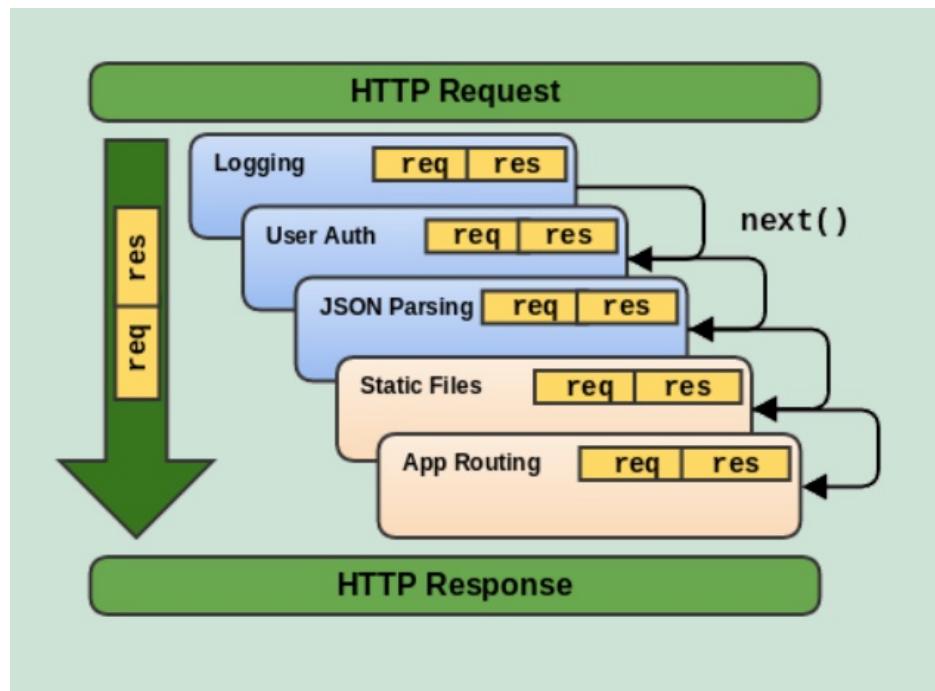
메서드명	설명
env	서버 환경 설정
views	뷰들이 들어있는 폴더(배열)을 설정
view engine	기본 뷰 엔진을 설정

Middleware



```
app.use((request, response, next) => {  
  // Do Something  
  next();  
});
```

Express Middleware



Static Middleware

- ▶ 특정 폴더의 파일들을 특정 패스로 접근할 수 있도록 함
- ▶ 리소스 파일 등을 제공하기 위해 유용
- ▶ `npm install serve-static --save`

```
const express = require('express');
const static = require('serve-static');
const path = require('path');
const app = express();

app.use('/public', static(path.join(__dirname, 'public')));
```

Body Parser

- ▶ POST 요청시 요청 파라미터를 확인하기 위한 미들웨어
- ▶ npm install body-parser --save
- ▶ 사용 : var bodyParser = require('body-parser')
- ▶ 이 미들웨어를 사용하면
 - ▶ request.body 를 통해 POST 파라미터를 받아올 수 있다.

Examples

[/005_express/002_app_set_get.js](#)
[/005_express/003_middlewares.js](#)

Examples

`/005_express/004_serve_static_middleware.js`

Examples

`/005_express/005_body_parser.js`

Cookie Parser

- ▶ Cookie를 읽고 쓰기 위한 미들웨어
- ▶ npm install cookie-parser --save
- ▶ 사용:
 - ▶ var cookieParser = require('cookie-parser');
 - ▶ app.use(cookieParser());
- ▶ 쓰기
 - ▶ response.cookie({key}, {value});
- ▶ 읽기
 - ▶ request.cookies

Express Session

- ▶ Session을 서버에 저장하고 사용하기 위한 미들웨어
- ▶ npm install express-session --save
- ▶ Session은 쿠키도 함께 사용하므로 cookie-parser 모듈도 함께 로딩한다.
 - ▶ 1.5.0 이상부터는 cookie-parser 모듈이 필요 없어짐(확인 요)
- ▶ 세션 저장 : `request.session.{세션명} = {Session Obj};`
- ▶ 세션 읽기 : `request.session.{세션명}`
- ▶ 세션 삭제 : `request.session.destroy((err) => { // callback logic });`

Express Session 주요 옵션

- ▶ secret : 'session-secret-key' => 세션 쿠키를 서명하는데 필요한 key
- ▶ resave : true|false => true면 요청하는 동안 수정이 없어도 다시 세이브
 - ▶ 기본값: true
 - ▶ Documentation 상에서는 false로 지정할 것을 권장
 - ▶ 클라이언트에서 별별 요청이 있을 때 문제가 될 여지가 있음
- ▶ saveUninitialized:true|false => 세션을 uninitialized 상태로 저장할 것인지 지정

Examples

`/005_express/006_cookie_parser.js`

`/005_express/007_session.js`

Express Router

- ▶ Express 내장 객체
- ▶ 사용법
 - ▶ `const express = require('express');`
 - ▶ `const router = express.Router();`
- ▶ Add method
 - ▶ `router.route({path}).get((req, res) => { // response logic });`
 - ▶ `router.route({path}).post(req, res) => { // response logic });`

Express Router Methods

메서드명	설명
get	GET 방식 요청 콜백 등록
post	POST 방식 요청 콜백 등록
put	POT 방식 요청 콜백 등록
delete	DELETE 방식 요청 콜백 등록
all	모든 요청 방식을 처리하는 콜백 등록

Examples

/005_express/008_router_basic.js

Node.js Programming

With MySQL

MySQL 모듈

- ▶ npm install mysql --save
- ▶ const mysql = require('mysql');

MySQL 사용



MySQL CreatePool

```
const pool = mysql.createPool({
  connectionLimit: 10,
  host: '{mysql_server_host_name}',
  user: '{mysql_user_name}',
  password: '{mysql_user_password}',
  database: '{database_name}',
  debug: true|false
});
```

MySQL CreatePool Options

속성	설명
connectionLimit	커넥션 풀에서 만들 수 있는 최대 연결 개수
host	데이터베이스 서버 호스트명
port	데이터베이스 서버 포트
user	데이터베이스 접속 유저 ID
password	데이터베이스 접속 유저 암호
database	데이터베이스 이름
debug	데이터베이스 사용 로그 여부 설정

MySQL Exec Query

```
pool.getConnection((err, conn) => {
  if (err) {
    if (conn) {
      conn.release();
    }
    return;
  }

  const exec = conn.query("select * from products", (err, result) => {
    conn.release();

    if (err) {
      // TODO: Error Handling
      return;
    } else {
      for (let i = 0; i < result.length; i++) {
        console.log(result[i].name, result[i].price);
      }
    }
  });
});
```

Examples

[/006_database/001_mysql_setting.js](#)

[/006_database/002_products.js](#)

MINI Project 1

- ▶ Express와 MySQL을 이용하여 간단한 블로그를 만들어 보자

MINI Project 2

- ▶ Express와 MySQL을 이용하여 간단한 REST API 서버를 만들어 보자

Node.js Programming

View and View Engine

View Directory Setting

- ▶ `app.set('views', {view directory});`
- ▶ `const path = require('path');`
- ▶ `app.set('views', path.join(__dirname, 'template'));`
- ▶ 이후
 - ▶ `response.render({view name}, {binding_data});` 식으로 뷰를 불러오고 data를 바인딩

Express View Engines

- ▶ EJS (Embedded JavaScript)
- ▶ Jade
- ▶ ...

Express View Engine: EJS

- ▶ Embedded JavaScript
- ▶ 내부에서 자바스크립트를 실행할 수 있다.
 - ▶ `<%= 변수 %>` : 변수를 출력
 - ▶ `<% javascript code %>` : JavaScript Code 블록
- ▶ 넘겨주지 않은 변수 혹은 객체는 렌더링 과정에서 `undefined` 오류를 발생할 수 있다. => `undefined` 오류는 반드시 체크하자

Examples

`/007_views/001_using_ejs.js`

The background of the slide features a large, abstract geometric pattern composed of various shades of green. It includes a prominent vertical bar on the left and a diagonal line extending from the bottom right towards the center. The pattern is composed of numerous overlapping triangles and quadrilaterals.

Thank YOU