

K-Digital Training KDT 풀스택 웹 개발자 양성 부트캠프 3기

# Node.js란?

WITH 팀 리처드



# Node.js

# Node.js



- 구글 크롬의 자바스크립트 엔진 ( V8 Engine ) 에 기반해 만들어진 Javascript 런타임
- 이벤트 기반, 비동기 I/O 모델을 사용해 가볍고 효율적
- npm 패키지는 세계에서 가장 큰 오픈 소스 라이브러리

# 런타임이란?

- 프로그래밍 언어가 구동되는 환경



- javascript의 런타임 환경은 웹 브라우저만 존재 했었음.
  - javascript 를 서버단 언어로 사용하기 위해 나온 것이 node.js
  - 웹 브라우저 없이 실행 가능

# npm이란?

- Node Package Manager의 약어
- javascript로 개발된 각종 모듈의 설치, 업데이트, 구성, 제거 과정을 자동화하여 관리해주는 기능을 함

# Node.js 특징

1. 자바스크립트 언어 사용
2. Single Thread
3. 비동기 I/O 방식

# 특징 - Single Thread

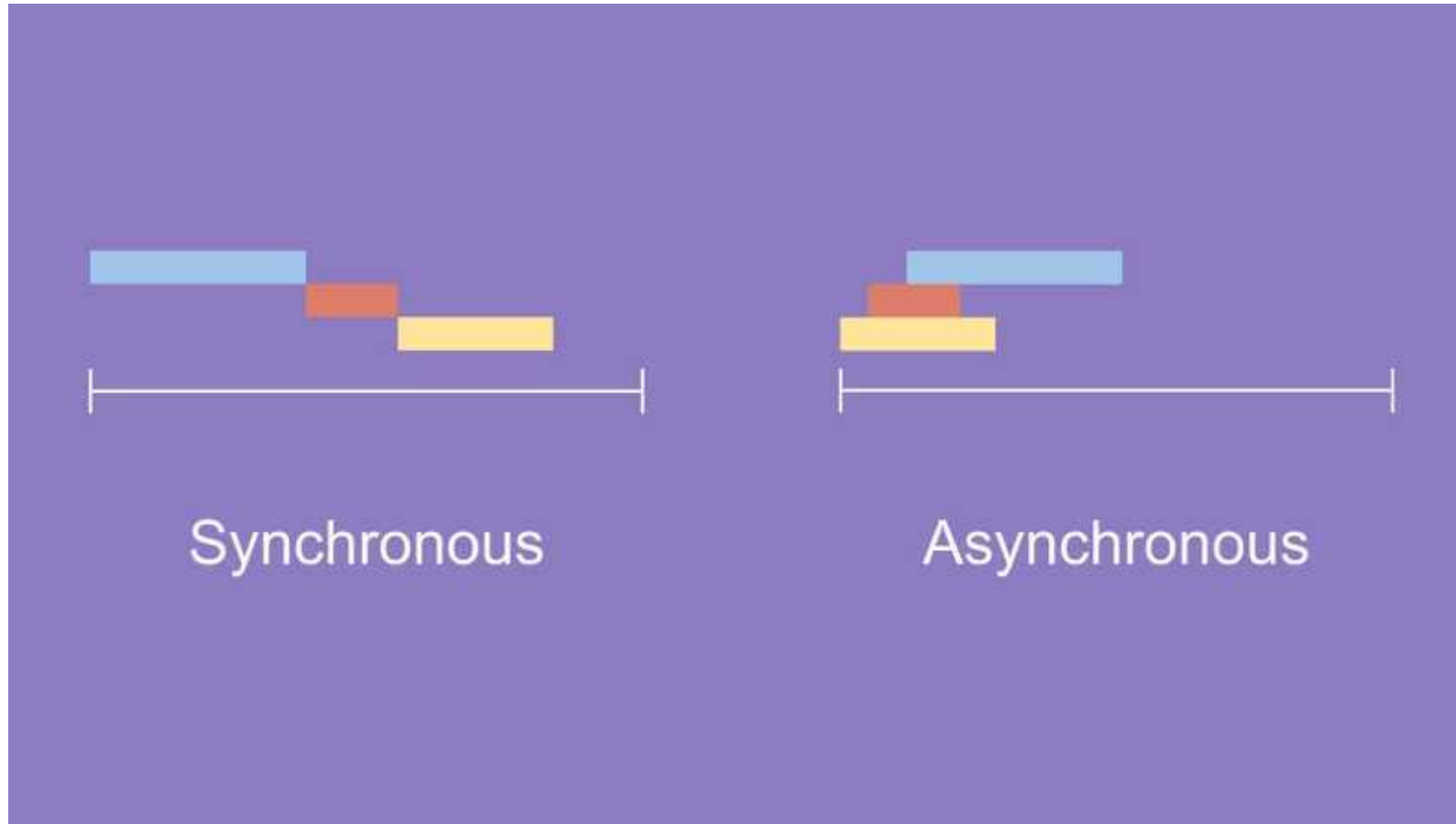
~~상용 스레드?  
멀티 스레드 프로세스?  
CPU?~~

에러를 처리하지 못하면 프로그램이 아예 중단됨



**예외처리의 중요성 ↑**

# 특징 - 비동기 I/O 방식





# 특징 - 비동기 I/O 방식

## • 동기 ( Synchronous )

- 한 요청에 서버의 응답이 이루어질 때까지 **계속 대기**해야 한다.
- 장점 : 설계가 매우 **간단**하고 직관적이다.
- 단점 : 결과가 주어질 때까지 아무것도 못하고 **기다리고 있어야** 한다.

## • 비동기 ( Asynchronous )

- 요청한 후 응답을 **기다리지 않고** 다른 활동을 한다.
- 장점 : 요청에 따른 결과가 반환되는 시간 동안 **다른 작업을 수행**할 수 있다.
- 단점 : 동기식보다 설계가 **복잡**하다.

# 특징 - 비동기 I/O 방식

- I/O 작업 : 입출력, 파일 시스템 접근 ( 읽기, 쓰기, 만들기 등 ), 네트워크 요청
- Node.js는 표준 라이브러리의 모든 I/O 메서드를 비동기 방식으로 제공한다.
- 비동기 방식에 어울리는 서비스
  - 스트리밍 서비스, 채팅 서비스

# Node.js 사용해보기

# Node.js 설치 - 로컬

[Node.js \(nodejs.org\)](https://nodejs.org)






## 다운로드

최신 LTS 버전: 16.16.0 (includes npm 8.11.0)

플랫폼에 맞게 미리 빌드된 Node.js 인스톨러나 소스코드를 다운받아서 바로 개발을 시작하세요.

LTS  
대다수 사용자에게 추천

현재 버전  
최신 기능

 Windows Installer node-v16.16.0-x64.msi	 macOS Installer node-v16.16.0.pkg	 Source Code node-v16.16.0.tar.gz
--	---	--

Windows Installer (.msi)

Windows Binary (.zip)

32-bit	64-bit
32-bit	64-bit

# Node.js 설치 - 서버

```
apt-get install nodejs  
apt-get install npm
```

npm 이란?

- Javascript로 개발된 각종 모듈의 설치, 업데이트, 구성, 제거 과정을 자동화하여 관리해주는 기능

# Node.js 설치 - 버전확인

node -v  
npm -v

```
C:\Users\>node -v  
v16.17.1  
  
C:\Users\>npm -v  
8.7.0  
  
C:\Users\>
```

# npm

- Node Package Manager ( <https://www.npmjs.com/> )
- 노드 패키지를 관리해주는 툴

- Npm에 업로드 된 노드 모듈
- 패키지들 간 의존 관계가 존재



# npm 사용하기

```
npm init
```

- 프로젝트를 시작할 때 사용하는 명령어
- **package.json**에 기록될 내용을 문답식으로 입력한다.

```
npm init --yes
```

- **package.json**이 생성될 때 **기본 값으로** 생성된다.

```
npm install 패키지 이름
```

- 프로젝트에서 사용할 **패키지를 설치**하는 명령어
- 설치된 패키지의 이름과 정보는 **package.json**의 **dependencies**에 입력된다.



모두

# 모듈

- Node.js에서 module은 ‘필요한 함수들의 집합’을 의미
- 모듈 사용 방법

```
const test_module = require("module_name");
```

# Express 모듈

# Express

- 웹 서버를 생성하는 것과 관련된 기능을 담당하는 프레임워크
- 웹 애플리케이션을 만들기 위한 각종 메소드와 미들웨어 등이 내장되어 있다.
- http 모듈 이용 시 코드의 가독성 ↓ 확장성 ↓
  - 이를 해결하기 위해 만들어진 것이 **Express 프레임워크**

# Express 설치

```
> npm install express
```

- **npm\_modules** 가 만들어지며 express에 관련된 폴더가 생성
- **package.json의 dependencies** 에 **express** 기록

```
> node_modules
```

```
"dependencies": {  
  "express": "^4.18.1"  
}
```

# .gitignore



The image shows a screenshot of a code editor interface. On the left, a file explorer shows a folder named '220721' containing a subfolder 'node\_modules' and two files: 'package-lock.json' and 'package.json'. The 'package.json' file is marked with a green 'U' icon. On the right, the '.gitignore' file is open, showing the following content:

```
.gitignore
1  /node_modules
2  package-lock.json
```

# [복습] .gitignore

## .gitignore?

- Git 버전 관리에서 **제외할 파일 목록을 지정**하는 파일
- Git 관리에서 특정 파일을 제외하기 위해서는 git에 올리기 전에 .gitignore에 파일 목록을 미리 추가해야 한다.

# [복습] .gitignore

**\*.txt** → 확장자가 txt로 끝나는 파일 모두 무시

**!test.txt** → test.txt는 무시되지 않음.

**test/** → test 폴더 내부의 모든 파일을 무시 ( b.exe와 a.exe 모두 무시 )

**/test** → (현재 폴더) 내에 존재하는 폴더 내부의 모든 파일 무시 ( b.exe 무시 )

```
(base) [07:25 PM] cwjcsk:~/99_test/99_tmp$ tree -a
.
├── .gitignore
├── test
│   └── b.exe
└── tmp
    └── test
        └── a.exe

3 directories, 3 files
```



# Express 사용

```
const express = require('express');
const app = express();
const PORT = 8000;

app.get('/', function (req, res) {
  res.send('hello express');
});

app.listen(PORT, function () {
  console.log(`Listening on port ${PORT}! http://localhost:${PORT}`);
});
```

# Express 사용

- **express()**
  - Express 모듈이 export 하는 최상위 함수로, **express application**을 만듦
- **app 객체**
  - Express() 함수를 호출함으로써 만들어진 **express application**

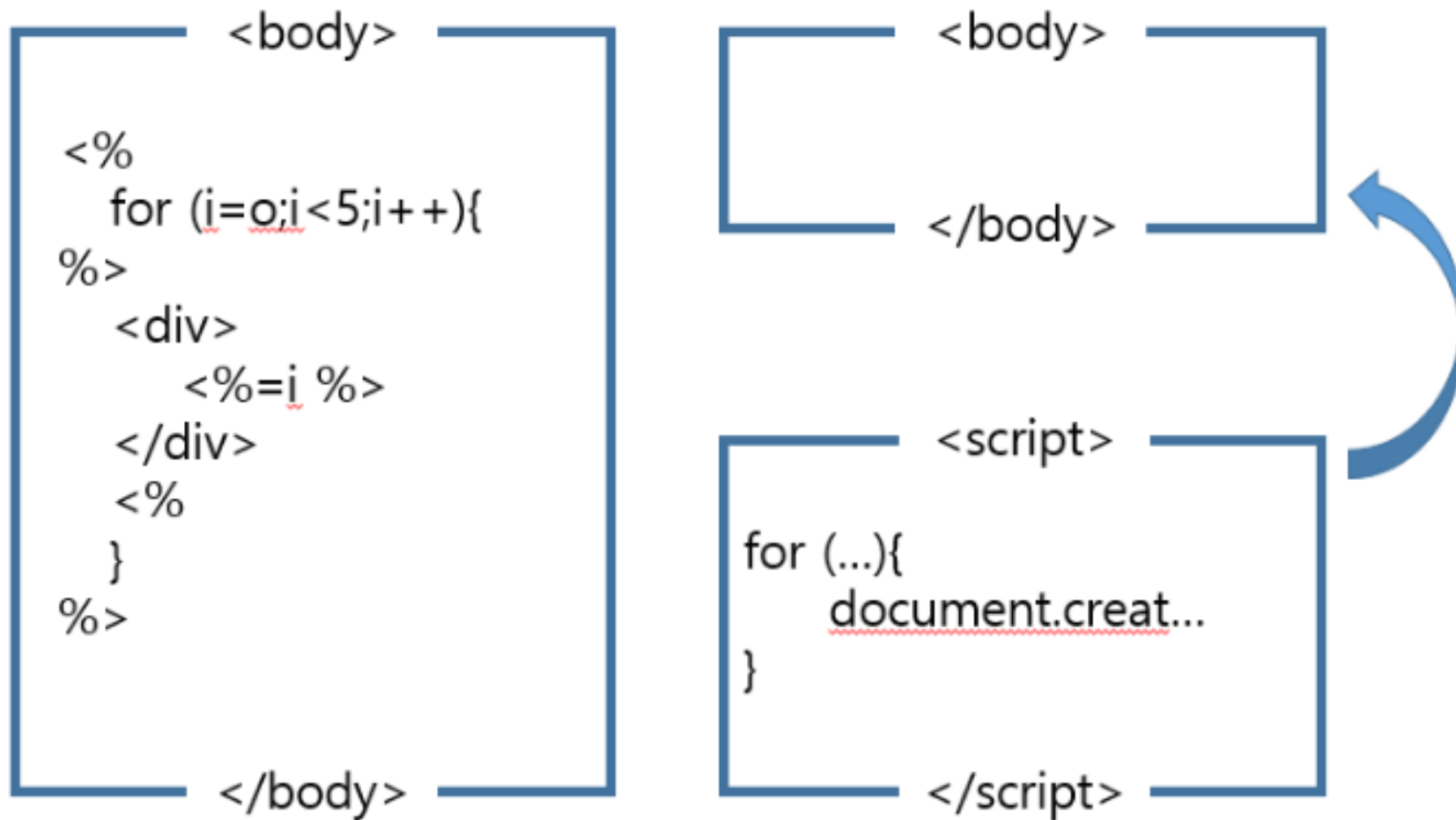
```
1  const express = require('express');  
2  const app = express();
```

# 템플릿 엔진

# EJS 템플릿

- 템플릿 엔진
  - 문법과 설정에 따라 파일을 html 형식으로 변환시키는 모듈
- ejs
  - **Embedded Javascript** 의 약자로, 자바스크립트가 내장되어 있는 html 파일
  - 확장자는 .ejs

# ejs 템플릿



# ejs 템플릿

```
$ npm install ejs
```

```
app.set('view engine', 'ejs');  
app.use('/views', express.static(__dirname + '/views'));
```

# ejs 템플릿

```
const express = require('express');
const app = express();
const PORT = 8000;

app.set('view engine', 'ejs');
app.use('/views', express.static(__dirname + '/views'));

app.get('/', function (req, res) {
  res.send('hello express');
});

app.get('/test', function (req, res) {
  res.render('test');
});

app.listen(PORT, function () {
  console.log(`Listening on port ${PORT}!`);
});
```

← ejs 템플릿 설정

← ejs 템플릿 렌더링

# ejs 템플릿

```
<html>
  <head>
    <title>EJS TEST</title>
  </head>
  <body>
    <% for (var i = 0; i < 5; i++) { %>
      <h1>안녕</h1>
    <% } %>
  </body>
</html>
```



# ejs 문법 사용하기

```
<% %>
```

- 무조건 자바스크립트 코드가 들어가야 하고, 줄바꿈을 할 경우에는 새로운 `<% %>` 를 이용해야 한다.

```
<%= %>
```

- 값을 템플릿에 출력할 때 사용

```
<%- include('view의 상대주소') %>
```

- 다른 view 파일을 불러올 때 사용

# 미들웨어

- 요청이 들어옴에 따라 응답까지의 **중간 과정**을 **함수로 분리**한 것
- **서버와 클라이언트를 이어주는** **중간 작업**
- **use()** 를 이용해 **등록**할 수 있다.

```
app.set('view engine', 'ejs');  
app.use('/views', express.static(__dirname + '/views'));
```

# 미들웨어 - static

- 이미지, **css** 파일 및 **Javascript** 파일(front)과 같은 **정적 파일** 제공
- Express 에 있는 static 메소드를 이용해 미들웨어로 로드
- 등록 방법

```
app.use('/static', express.static(__dirname + '/static'));
```

# ejs 템플릿

```
const express = require('express');
const app = express();
const PORT = 8000;

app.set('view engine', 'ejs');
app.use('/views', express.static(__dirname + '/views'));
app.use('/static', express.static(__dirname + '/static'));

app.get('/', function (req, res) {
  res.send('hello express');
});

app.get('/test', function (req, res) {
  res.render('test');
});

app.listen(PORT, function () {
  console.log(`Listening on port ${PORT}!`);
});
```



정적 파일 로드 코드

[keyword] 미들웨어, static