

# MEAN 스택 활용 웹 개발

## Node.js 개요



2016년도 2학기 2학년 방과후학교

# Node.js 소개

- 서버사이드 자바 스크립트 실행환경 (크롬 V8 엔진)
- 2009년 Ryan Dahl이 발표
- 비동기 I/O 처리 (Non-Blocking I/O 방식)
- 이벤트 기반 (event-driven)
- 네트워크 기반 어플리케이션을 위한 플랫폼



# Node.js 사이트와 친해지기



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 upgrades](#) for recent V8 vulnerability

## Download for Windows (x64)

**v4.5.0 LTS**

Recommended For Most Users

**v6.4.0 Current**

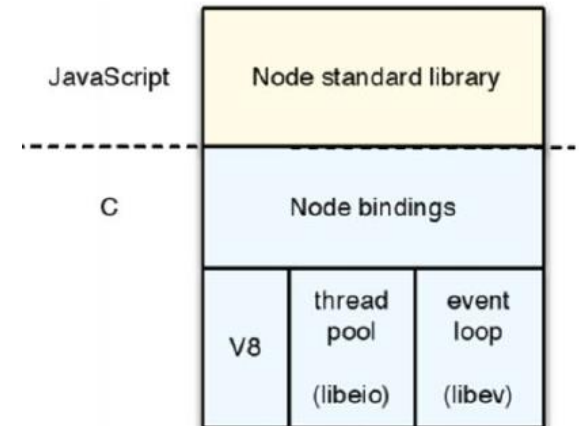
Latest Features

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

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

# Node.js 특징

- Node.js 는 **비동기 I/O**를 사용하고 **싱글 쓰레드** 방식으로 작성한다.
- Node.js 자체는 매우 간단한 경량 프레임워크다.
- 필요한 기능은 모듈을 이용해서 작성하며, **수많은 모듈이 제공**되고 있다.
- 패키지 매니저를 이용해서 필요한 모듈을 간단하게 설치할 수 있다.
- Node.js의 권장 분야로는 **실시간 소셜 네트워크, 데이터 I/O가 많은 분야, IoT 기기 연동 분야**이다.

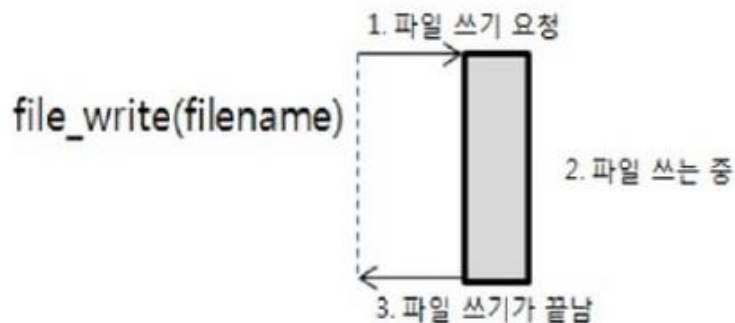


# 비동기 I/O 처리

- 동기식(Synchronous) : 하나의 동작이 완료된 이후에 다음 동작을 시작함
- 비동기식(Asynchronous) : 하나의 동작이 완료되지 않아도 다음 동작을 시작함. 즉, **I/O 동작이 끝날 때까지 대기하지 않음!!**

- Non blocking/Async IO

## 동기식 IO



- 커피 주문하고 기다리기

## 비동기식 IO(node.js)

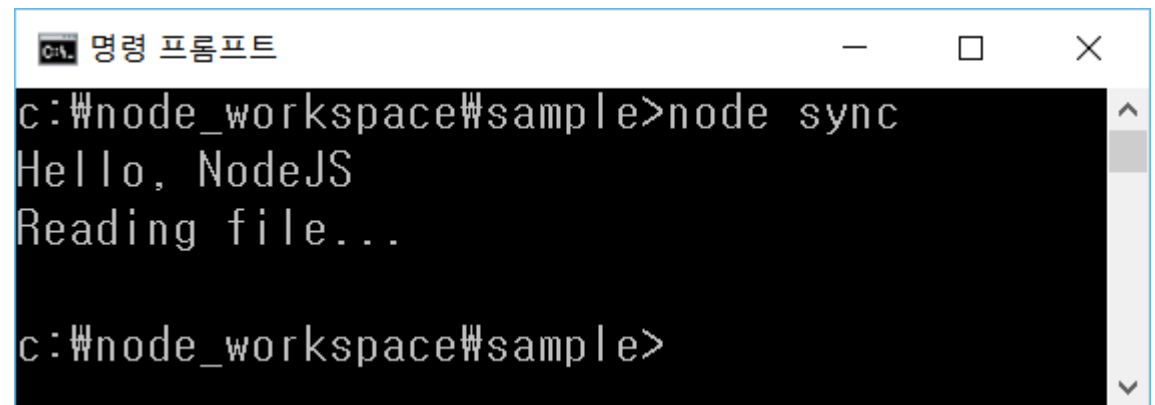


- 커피 주문하고 진동벨 받기

# 동기식 vs 비동기식

- 동기식 : A 실행 → A 결과 → B 실행 → B 결과
- 하나의 실행이 끝나면 결과를 받아 다음 실행

```
1 var fs = require('fs');  
2 var content = fs.readFileSync('hello.txt', 'utf8');  
3 console.log(content);  
4 console.log('Reading file...');
```



A screenshot of a Windows Command Prompt window titled "명령 프롬프트". The window shows the execution of a Node.js script. The prompt is "c:\#node\_workspace#sample>node sync". The output is "Hello, NodeJS" followed by "Reading file...". The prompt then returns to "c:\#node\_workspace#sample>".

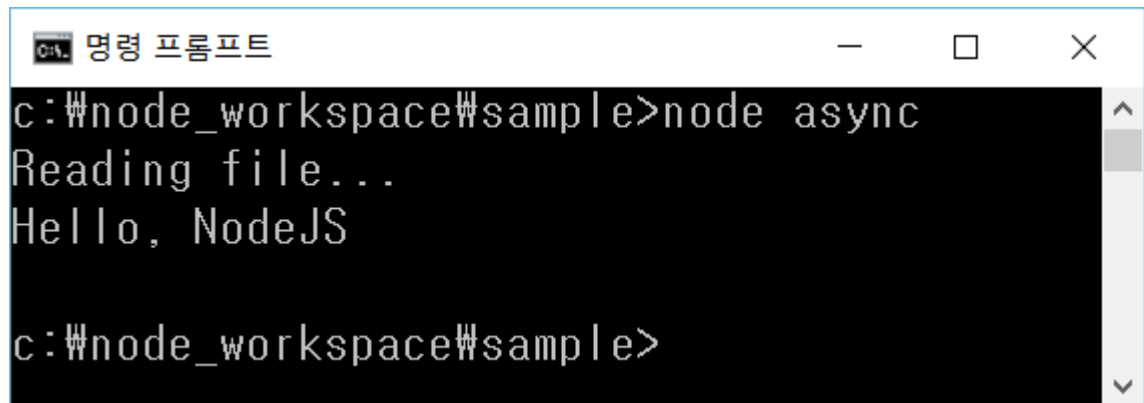
```
C:\#node_workspace#sample>node sync  
Hello, NodeJS  
Reading file...  
  
C:\#node_workspace#sample>
```

# 동기식 vs 비동기식

- 비동기식 : A 실행 → B 실행 → (B 결과) → (A 결과)
- 실행이 다 끝날 때까지 기다리지 않음

콜백(callback) 함수

```
1 var fs = require('fs');  
2 var content = fs.readFile('hello.txt', 'utf8', function (err, content) {  
3     console.log(content);  
4 });  
5 console.log('Reading file...');
```



```
명령 프롬프트  
c:\#node_workspace#sample>node async  
Reading file...  
Hello, NodeJS  
c:\#node_workspace#sample>
```

# REPL(Read-Eval-Print-Loop) 실행

```
C:\ 명령 프롬프트 - node
c:\#node_workspace#sample>node
> a = 3
3
> b = 2
2
> a + b
5
> var str = 'Hello';
undefined
> str
'Hello'
> str.length
5
> str + ', World';
'Hello, World'
> str
'Hello'
>
```

```
C:\ 명령 프롬프트
c:\#node_workspace#sample>node
> .help
break    Sometimes you get stuck, this gets you out
clear    Alias for .break
exit     Exit the repl
help     Show repl options
load     Load JS from a file into the REPL session
save     Save all evaluated commands in this REPL session to a file
> .exit

c:\#node_workspace#sample>
```



# Node.js 전역 객체

- 전역 객체는 모듈 로딩 없이 어디서나 사용 가능한 객체이다.
- Global 객체는 생략 가능

ex) `global.console.log('Hello, World');` → `console.log('Hello, World');`

# Node.js 전역 객체 종류

- `process` : 현재 동작중인 프로세스의 정보
- `console` : 콘솔 출력
- `Buffer` : 이진 데이터를 다루는 버퍼 클래스
- `require` : 모듈 로딩
- `__filename, __dirname` : 현재 폴더 경로와 파일 경로
- `module, exports` : 로딩된 모듈 정보와 모듈로 타입, 객체 노출시키기
- `Timeout` : 타이머와 반복 용 함수

# Node.js 전역 객체 - process

- `process.version` : Node.js 버전 정보
- `process.env` : 실행 환경 정보
- `process.arch` : CPU 아키텍처
- `process.platform` : 플랫폼 정보

# Node.js 전역 객체 - Timeout

- `setTimeout( )` : 일정 시간 뒤 호출
- `setInterval( )` : 반복 호출
- `clearTimeout( )` : `setTimeout( )` 제거
- `clearInterval( )` : `setInterval( )` 제거

## Node.js 전역 객체 - console

- `console.log('log 메시지');`
- `console.info('info 메시지');`
- `console.warn('warn 메시지');`
- `console.error('error 메시지');`

# Console 클래스

- 로그 정보를 파일에 저장하기
- 파일 출력을 위한 두 개의 스트림 객체 생성 (정상, 에러로그 구분)
- Console 클래스를 통해 Console 객체 생성
- Console 객체 생성 시 위에서 만든 스트림 객체 전달
- log, info는 stdout.log로, warn, error은 stderr.log에 분리되어 저장

# 타이머 사용하기

- `console.time('TIMER명');`    // 타이머 시작
- `console.timeEnd('TIME명');`    // 타이머 종료
- 'TIMER명': 걸린 시간 출력(ms 단위)