웹응용프로그래밍 2024-2

Ch01. Node Introduction

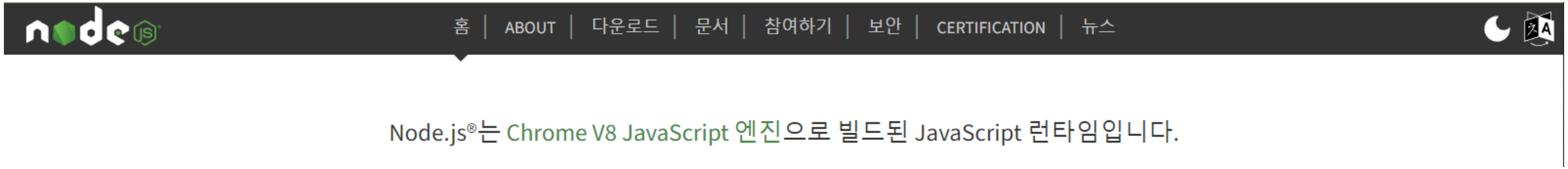
부산대학교 정보컴퓨터공학부
디지털트윈 네트워크 연구실
Prof. 김원석

Contents

1-1. What is Node?

What is Node?

❖ 공식 홈페이지의 설명



❖ 노드는 서버가 아니었나?

- Runtime: 프로그램이 실행되고 있는 동안의 동작 환경, 코드가 실제로 실행되는 시점의 환경
- Node = JavaScript runtime: 자바스크립트 코드를 Chrome V8 engine를 사용하여 실행할 수 있는 환경을 제공
  - 다른 자바스크립트 런타임의 예: Web browser
  - 노드 이전에도 자바스크립트 런타임을 만들기 위한 많은 시도가 있었으나, 대부분 엔진 속도 문제로 실패
  - 노드를 통해 자바스크립트로 작성된 서버 코드를 실행할 수 있음

내부 구조

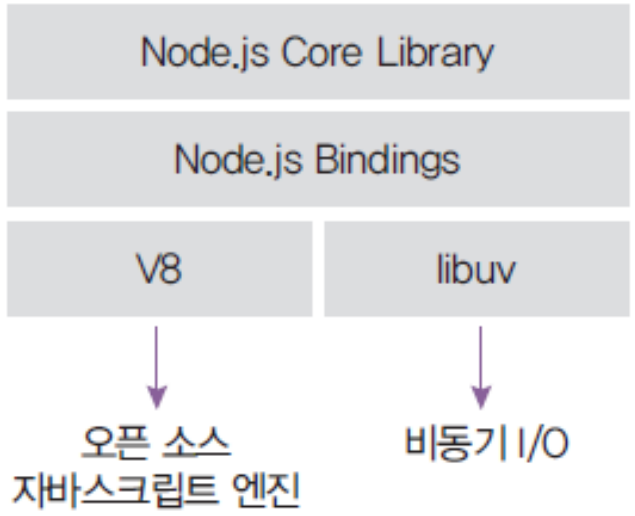
❖ 2008년 V8 Engine 출시

❖ 2009년 노드 프로젝트 시작

❖ 노드는 내부적으로 V8과 libuv를 포함

- V8 engine: Open-source JavaScript engine -> 자바스크립트 실행속도 문제를 획기적으로 개선
- libuv: 노드의 특성인 이벤트 기반, 논블로킹 I/O 모델을 구현한 라이브러리

그림 1-3 노드의 내부 구조



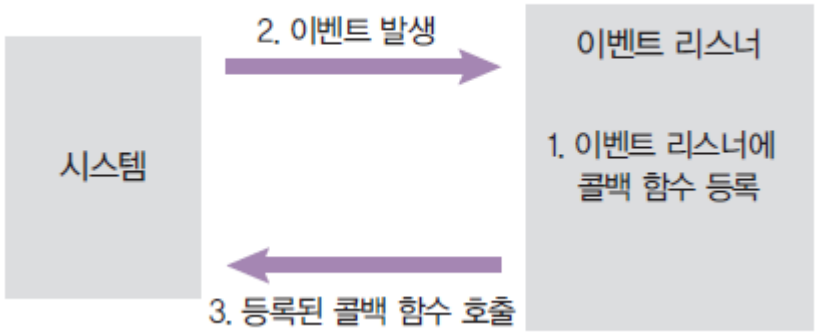
1-2. Node Features

Feature #1: Event-Driven

❖ **이벤트 기반Event-driven: 이벤트가 발생했을 때 미리 지정해둔 작업을 수행하는 방식**

- 이벤트 예시: 클릭, 네트워크 요청, 타이머 등
- Event listener: 이벤트를 듣고 있는 대기자
- Callback function: 이벤트가 발생했을 때 실행될 함수
  - 이벤트 리스너가 호출하는 함수
- Event loop: Event-driven을 위한 JavaScript 동작 방식의 핵심 구성요소

그림 1-4 이벤트 기반



Feature #2: Non-Blocking I/O

❖ Non-Blocking

- 특정 작업을 시작한 후, 그 작업의 완료를 기다리지 않고 즉시 제어를 반환하여 다른 작업을 바로 이어서 수행하는 것
- Non-blocking 방식 하에서 일부 코드는 백그라운드에서 병렬로 실행됨
  - 일부 코드: 시간이 오래 걸리는 파일 시스템 접근이나 네트워크 요청과 같은 I/O 작업, 압축, 암호화 등
- 나머지 코드는 블로킹 방식으로 실행됨
- I/O 작업이 많을 때 노드 활용성이 극대화

그림 1-9 블로킹과 논블로킹

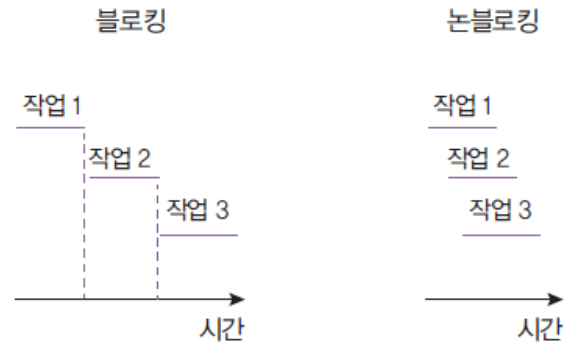
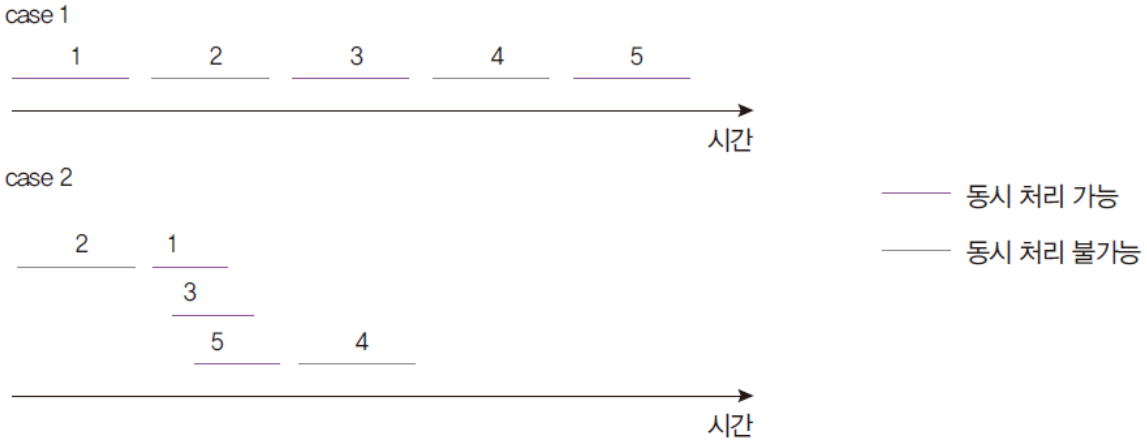


그림 1-10 동시 처리로 얻는 시간적 이득



Feature #2: Non-Blocking I/O

❖ 출력 순서 생각해보기

```
1    function longRunningTask() {
2      // 오래 걸리는 작업
3      console.log('작업 끝');
4    }
5    console.log('시작');
6    longRunningTask();
7    console.log('다음 작업');
```

```
1    function longRunningTask() {
2      // 오래 걸리는 작업
3      console.log('작업 끝');
4    }
5    console.log('시작');
6    setTimeout(longRunningTask, 0);
7    console.log('다음 작업');
```

Feature #3: Single-threaded

❖ Process vs. Thread

- 프로세스: 운영체제에서 할당하는 작업의 단위, 프로세스 간 자원 공유X
- 스레드: 프로세스 내에서 실행되는 흐름의 단위, 부모 프로세스 자원 공유

❖ Node.js는 Single-thread?

- Node.js 프로세스 자체는 Multi-thread로 구현되어 있지만 개발자가 제어 가능한 스레드(주 실행 컨텍스트)는 하나이므로 Single-threaded라고 표현
- Node는 multi-thread 대신 주로 multi-process 활용
- Node는 14버전부터 multi-thread 사용 가능

그림 1-13 스레드와 프로세스

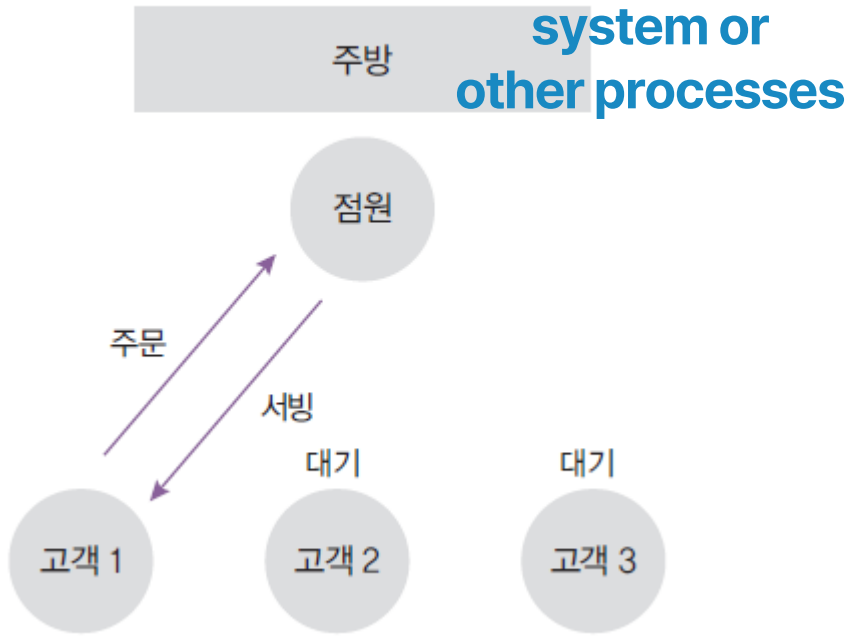
Feature #3: Single-threaded

❖ Single-threaded: 주어진 일을 동시에 하나밖에 처리하지 못함

- 특정 작업에서 blocking이 발생하는 경우 나머지 작업은 모두 대기해야 함 -> 비효율

❖ 주방에 비유(점원: 스레드, 주문: 요청, 서빙: 응답)

그림 1-10 싱글 스레드, 블로킹 모델

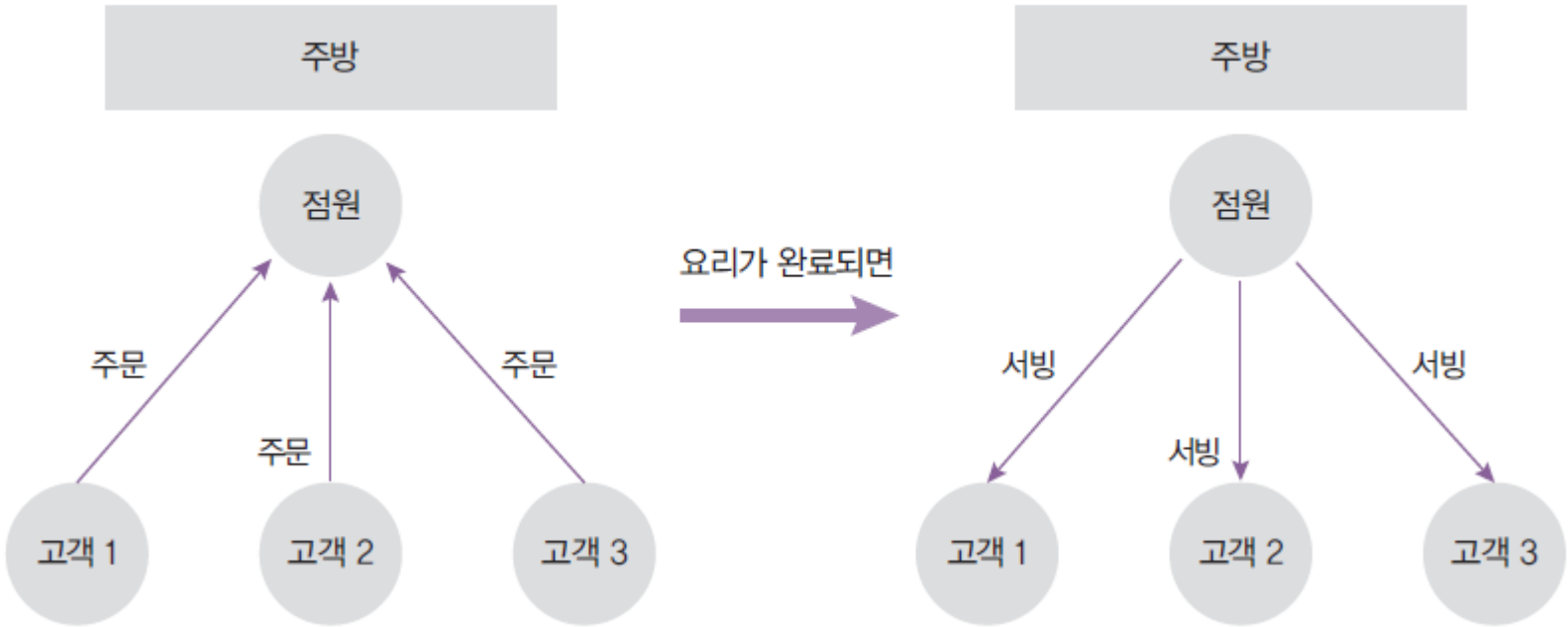


Feature #3: Single-threaded

❖ Node는 어떻게 Single-threaded Server가 될 수 있는가

- blocking 대신 non-blocking을 채택하여 I/O와 같은 일부 작업을 백그라운드(시스템 또는 다른 프로세스)에서 실행
- 일단 요청을 받아서 백그라운드(시스템 또는 다른 프로세스)에 넘긴 후, 완료되면 응답
  - 완료되면 응답: 해당 작업 완료 이벤트에 대한 이벤트 리스너에 콜백 함수 등록

그림 1-11 싱글 스레드, 논블로킹 모델



Feature #3: Single-threaded

❖ Single-threaded Model 주의점

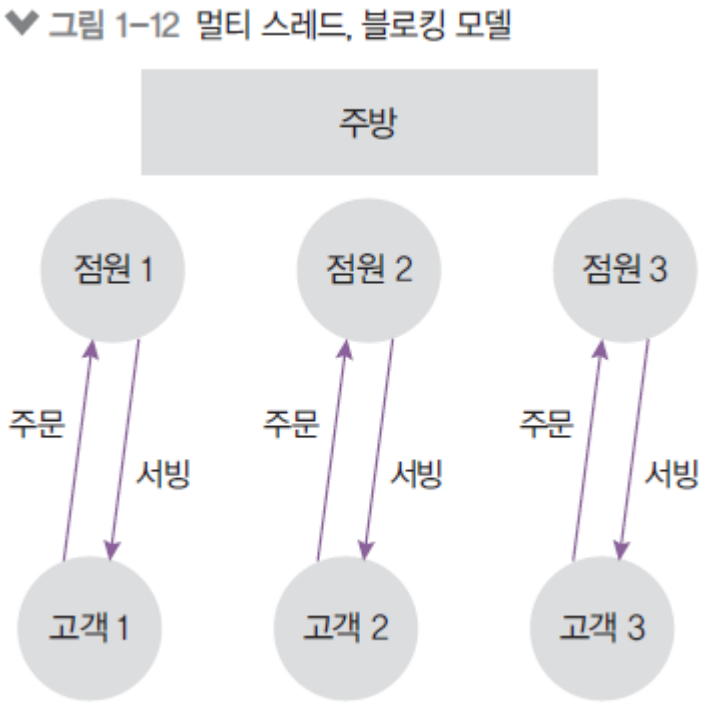
- 예외를 처리하지 못하는 경우 모든 프로세스가 중단됨
- 하지만 프로그래밍 난이도가 쉽고(?), CPU나 메모리와 같은 컴퓨팅 리소스를 적게 사용

❖ Multi-threaded Model

- 특정 스레드에서 예외 발생 시 새로운 스레드를 생성하여 극복
- 단, 새로운 스레드 생성이나 놀고 있는 스레드 처리에 비용 발생
- 프로그래밍 난이도 어려우며, 스레드 수만큼 컴퓨팅 리소스를 많이 사용함

❖ 식당 비유(Multi-thread and Blocking Model)

- 점원: 스레드, 주문: 요청, 서빙: 응답
- 점원 한 명이 탈주하면 다른 점원으로 대체 가능하나,
  점원 관리가 어렵고 점원의 신규 고용과 해고 문제 발생



Feature #3: Single-threaded

❖ Node.js version 14부터는 멀티 스레드 공식 지원 (worker threads)

- 멀티 스레드를 사용할 수 있도록 worker_threads 모듈 도입
- CPU를 많이 사용하는 작업인 경우에 활용 가능
- 멀티 프로세싱만 가능했던 아쉬움을 달래줌

표 1-1 멀티 스레딩과 멀티 프로세싱 비교

멀티 스레딩	멀티 프로세싱
하나의 프로세스 안에서 여러 개의 스레드 사용	여러 개의 프로세스 사용
CPU 작업이 많을 때 사용	I/O 요청이 많을 때 사용
프로그래밍이 어려움	프로그래밍이 비교적 쉬움

Node.js의 Multi-thread: Thread Pool and Worker Threads

❖ Thread Pool

- Node.js는 내부적으로 libuv 라이브러리를 사용하여 **비동기 I/O 작업**을 처리하나, 그중 일부는 백그라운드에서 동기적으로 실행됨
  - 예1: file read I/O task의 경우, 스레드는 file system에서 데이터를 읽는 동안 blocking되며, 완료되면 그 결과가 main thread로 반환됨
  - 예2: DB I/O task의 경우, DB 서버와의 통신은 동기적으로 수행되며, 완료되면 그 결과가 callback function을 통해 main thread로 반환됨
- 이러한 동기적인 작업들을 효율적으로 처리하기 위해 libuv는 Thread pool을 사용
- 비동기 함수가 호출될 때, 해당 작업이 스레드 풀에서 처리될 필요가 있다면, thread pool에 있는 스레드 중 사용 가능한 스레드에 작업이 할당되어 백그라운드에서 실행됨

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

fs.readFile('example.txt', 'utf8', (err, data) => {
    if (err) {
        console.error('Error reading the file:', err);
        return;
    }
    console.log(data);
});

console.log('Reading the file...');
```

```
const db = require('pg');

/** ... db settings */

db.query('SELECT NOW()', (err, res) => {
    if (err) {
        console.error('Error executing query:', err.stack);
    } else {
        console.log('Current time:', res.rows[0]);
    }
});
```

Node.js의 Multi-thread: Thread Pool and Worker Threads

❖ Worker Threads

- Node.js에서 사용자가 CPU-bound task에 대해 직접 multi thread를 활용 가능하게 함
  - 예: 정렬, 탐색, 영상처리
- Worker threads는 각각의 자바스크립트 실행 컨텍스트와 V8 인스턴스를 가지므로, worker threads 간 데이터를 공유하려면 명시적인 메시지 전달이 필요
- Worker threads는 CPU 집중적인 작업을 비동기적으로 처리하거나, 멀티 코어 시스템에서 병렬 처리를 위해 사용됨

```
1  const { Worker, isMainThread, parentPort } = require('worker_threads');
2
3  if (isMainThread) {
4      const worker = new Worker(__filename);
5      worker.on('message', result => {
6          console.log('Factorial result:', result);
7      });
8      worker.postMessage(10);
9  } else {
10     parentPort.on('message', number => {
11         let result = 1;
12         for (let i = 2; i <= number; i++) {
13             result *= i;
14         }
15         parentPort.postMessage(result);
16     });
17 }
```

1-3. Node Roles

서버로서의 노드

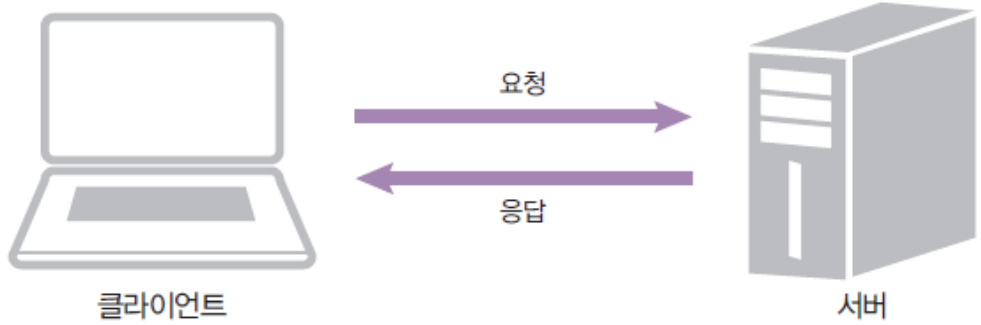
❖ 서버: 네트워크를 통해 클라이언트에 정보나 서비스를 제공하는 컴퓨터 또는 프로그램

❖ 클라이언트: 서버에 요청을 보내는 주체(브라우저, 데스크탑 프로그램, 모바일 앱, 다른 서버에 요청을 보내는 서버)

- 예1: Safari web browser(클라이언트, 요청)를 통해 네이버 web site(서버, 응답)에 연결
- 예2: 스마트폰 앱스토어 app(클라이언트)을 통해 앱스토어 서버(서버)에서 앱 다운로드

❖ Node는 서버 그 자체가 아니며, 서버를 구성할 수 있게 하는 자바스크립트 모듈을 실행하는 런타임 환경을 제공

그림 1-2 클라이언트와 서버



서버로서의 노드

❖ 노드 서버의 장단점

표 1-1 노드의 장단점

장점	단점
멀티 스레드 방식에 비해 컴퓨터 자원을 적게 사용함	싱글 스레드라서 CPU 코어를 하나만 사용함
I/O 작업이 많은 서버로 적합	CPU 작업이 많은 서버로는 부적합
멀티 스레드 방식보다 쉬움	하나뿐인 스레드가 멈추지 않도록 관리해야 함
웹 서버가 내장되어 있음	서버 규모가 커졌을 때 서버를 관리하기 어려움
자바스크립트를 사용함	어중간한 성능
JSON 형식과 호환하기 쉬움	

❖ CPU 작업을 위해 AWS Lambda나 Google Cloud Functions같은 별도 서비스 사용

❖ PayPal, Netflix, NASA, WalMart, Linkedin, Uber, Airbnb 등에서 main/sub 서버로 활용

서버 외의 노드

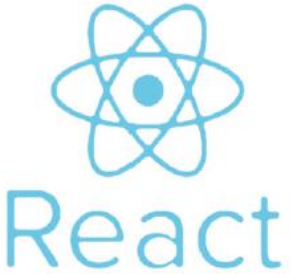
❖ 자바스크립트 런타임이기 때문에 용도가 서버에만 한정되지 않음

❖ 웹, 모바일, 데스크탑 애플리케이션에도 사용

- 웹 프레임워크: Angular, React, Vue, Meteor 등
- 모바일 앱 프레임워크: React Native
- 데스크탑 개발 도구: Electron$_{\text{developed by GitHub}}$ - Atom, Slack, VSCode, Discord 등 제작

❖ 위 프레임워크가 노드 기반으로 동작함

그림 1-16 노드 기반의 개발 도구



1-4. IDE Settings

IDE Settings

❖ Node 설치

- https://nodejs.org 접속
- LTS 버전 설치

❖ IDE

- VS Code
- Sublime Text
- Cloud9
- Intellij IDEA
- Webstorm