1장

노드 시작하기

1장에서는 Node.js(이하 노드)가 무엇인지, 어디에 쓰이는지, 누가 쓰고 있는지를 알아보고, 노드의 핵심 개념들을 다룹니다. 또한 노드와 비주얼 스튜디오 코드를 설치하는 방법도 알아봅니다.

❯ 그림 1-1 노드 로고

이번 장에서는 노드와 관련된 실습 코드가 나오지는 않지만, 노드의 핵심 개념에 대해 다루므로 꼭 읽어보기 바랍니다. 많은 노드 입문자가 핵심 개념을 충분히 이해하지 못한 채 코딩부터 시작하다 어려움을 겪습니다. 만약 여러분이 이미 런타임, 이벤트 기반, 논블로킹 I/O, 싱글 스레드 모델이 무엇인지 알고 있다면 넘어가도 좋습니다.

1.1 핵심 개념 이해하기

NODE.JS

노드가 무엇인지에 대해서는 여러 가지 의견이 많지만, 어떠한 설명도 노드 공식 사이트의 설명보다 정확하지는 않을 것입니다. 노드의 공식 사이트(https://nodejs.org/ko/)에서는 노드를 다음과 같이 설명하고 있습니다.

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

여러분 중 대부분은 노드를 서버로 사용하는 방법을 익히기 위해 이 책을 읽고 있을 것입니다. 그렇기 때문에 공식 사이트의 노드 소개글에 서버라는 말이 없어서 당황스러울 수도 있습니다. 하지만 걱정하지 마세요. 서버라는 말이 없는 이유는 노드가 서버로만 사용되는 것이 아니기 때문입니

다. 그래도 이 책에서는 전반적으로 노드를 서버로서 사용하는 방법에 대해 다룹니다. 그리고 14장에서는 자바스크립트 프로그램을 운영하는 런타임으로 사용하는 방법을 다룹니다.

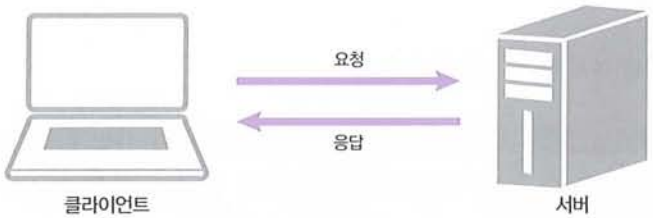
먼저 서버와 런타임이 무엇인지 알아봅시다.

1.1.1 서버

노드를 통해 다양한 자바스크립트 애플리케이션을 실행할 수 있지만, 노드는 서버 애플리케이션을 실행하는 데 제일 많이 사용됩니다.

그럼 서버란 무엇이며, 어떤 역할을 할까요? 서버는 네트워크를 통해 클라이언트에 정보나 서비스를 제공하는 컴퓨터 또는 프로그램을 말합니다. 클라이언트란 요청을 보내는 주체로, 브라우저일 수도 있고, 데스크톱 프로그램일 수도 있고, 모바일 앱일 수도 있고, 다른 서버에 요청을 보내는 서버일 수도 있습니다. 여러분이 평소에 사용하는 웹 사이트나 앱을 생각해보세요. 웹 사이트의 화면(HTML)은 어디에서 가져올까요? 앱 설치 파일은 어디에서 내려받는 것일까요?

❯ 그림 1-2 클라이언트와 서버



예를 들어 길벗출판사의 웹 사이트를 방문한다고 생각해봅시다. 주소창에 길벗출판사의 웹 사이트 주소(https://www.gilbut.co.kr/)를 입력(**요청**)하면 브라우저는 그 주소에 해당하는 길벗출판사의 컴퓨터 위치를 파악합니다. 그리고 그 컴퓨터에서 길벗출판사의 웹 사이트 페이지를 받아와 요청자의 브라우저(클라이언트)에 띄워줍니다(**응답**). 이런 일을 하는 컴퓨터가 바로 서버입니다.

모바일 앱을 설치하는 경우를 생각해봅시다. 구글의 플레이 스토어나 애플의 앱스토어에서 원하는 앱을 골라서 설치 버튼을 누르면(**요청**) 내려받기(**응답**)가 시작됩니다. 앱 설치 파일은 이미 어딘가에 저장되어 있으므로 여러분이 그곳에서 데이터를 받아와 모바일 기기에 설치할 수 있는 것입니다. 그 어딘가가 구글과 애플의 서버입니다. 플레이 스토어와 앱스토어는 클라이언트 역할을 하는 것이고요.

웹이나 앱을 사용할 때 여러분의 데이터(아이디, 비밀번호, 이메일 등)와 서비스의 데이터가 생성됩니다. 이 데이터를 어딘가에 저장하고, 그 어딘가에서 클라이언트로 데이터를 받아와야 합니다. 이곳이 바로 서버입니다.

서버라고 해서 요청에 대한 응답만 하는 것은 아닙니다. 다른 서버에 요청을 보낼 수도 있습니다. 이때는 요청을 보낸 서버가 클라이언트 역할을 합니다.

정리하면, 서버는 클라이언트의 요청에 대해 응답을 합니다. 응답으로 항상 Yes를 해야 하는 것은 아니고, No를 할 수도 있습니다. 여러분이 어떤 사이트로부터 차단 당했다면 그 사이트의 서버는 여러분의 요청에 매번 No를 응답할 것입니다.

노드는 자바스크립트 애플리케이션이 서버로서 기능하기 위한 도구를 제공하므로 서버 역할을 수행할 수 있습니다. 다른 언어를 사용하지 않고 왜 굳이 노드를 사용해 서버를 만드는지 의문이 들 수도 있습니다. 궁금증을 해결하려면 먼저 노드의 특성에 대해 알아야 합니다. 공식 사이트에 게시된 노드 소개글을 바탕으로 노드의 특성에 대해 알아봅시다.

1.1.2 자바스크립트 런타임

공식 사이트에 게시된 노드 소개글을 다시 한 번 보겠습니다.

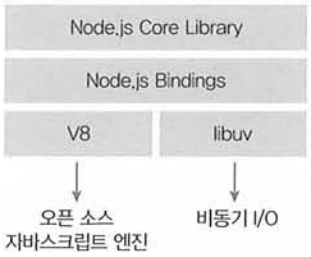
> Node.js®는 크롬 V8 자바스크립트 엔진으로 빌드된 자바스크립트 런타임입니다. Node.js는 이벤트 기반, 논블로킹 I/O 모델을 사용해 가볍고 효율적입니다. Node.js의 패키지 생태계인 npm은 세계에서 가장 큰 오픈 소스 라이브러리 생태계이기도 합니다.

노드는 자바스크립트 런타임입니다. 런타임은 특정 언어로 만든 프로그램들을 실행할 수 있는 환경을 뜻합니다. 따라서 노드는 자바스크립트 프로그램을 컴퓨터에서 실행할 수 있게 해줍니다.

기존에는 자바스크립트 프로그램을 인터넷 브라우저 위에서만 실행할 수 있었습니다. 브라우저 외의 환경에서 자바스크립트를 실행하기 위한 여러 가지 시도가 있었으나, 자바스크립트의 실행 속도 문제 때문에 모두 큰 호응을 얻지는 못했습니다.

하지만 2008년 구글이 V8 엔진을 사용하여 크롬을 출시하자 이야기가 달라졌습니다. 당시 V8 엔진은 다른 자바스크립트 엔진과 달리 매우 빨랐고, 오픈 소스로 코드도 공개되었습니다. 속도 문제가 해결되자 라이언 달(Ryan Dahl)은 2009년 V8 엔진 기반의 노드 프로젝트를 시작했습니다.

❤ 그림 1-3 노드의 내부 구조



노드는 V8과 더불어 libuv라는 라이브러리를 사용합니다. V8과 libuv는 C와 C++로 구현되어 있습니다. 여러분이 코딩한 자바스크립트 코드는 노드가 알아서 V8과 libuv에 연결해주므로 노드를 사용할 때 C와 C++는 몰라도 됩니다.

libuv 라이브러리는 노드의 특성인 이벤트 기반, 논블로킹 I/O 모델을 구현하고 있습니다. 노드는 스스로를 이벤트 기반, 논블로킹 I/O 모델을 사용해 가볍고 효율적이라고 표현했습니다. 그럼 이 모델이 무엇인지, 그리고 장단점으로는 어떤 것들이 있는지 알아봅시다.

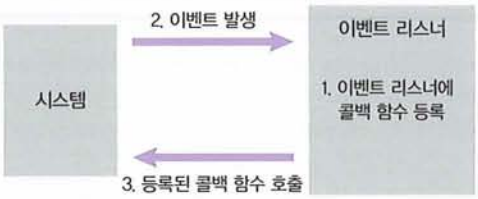
1.1.3 이벤트 기반

이벤트 기반(event-driven)이란 이벤트가 발생할 때 미리 지정해둔 작업을 수행하는 방식을 의미합니다. 이벤트로는 클릭이나 네트워크 요청 등이 있을 수 있습니다.

이벤트 기반 시스템에서는 특정 이벤트가 발생할 때 무엇을 할지 미리 등록해두어야 합니다. 이것을 이벤트 리스너(event listener)에 콜백(callback) 함수를 등록한다고 표현합니다. 버튼을 누르면 경고 창을 띄우도록 설정하는 것을 예로 들어 보겠습니다. 클릭 이벤트 리스너에 경고 창을 띄우는 콜백 함수를 등록해두면 클릭 이벤트가 발생할 때마다 콜백 함수가 실행돼 경고 창이 뜨는 것입니다.

노드도 이벤트 기반 방식으로 동작하므로 이벤트가 발생하면 이벤트 리스너에 등록해둔 콜백 함수를 호출합니다. 발생한 이벤트가 없거나 발생했던 이벤트를 다 처리하면 노드는 다음 이벤트가 발생할 때까지 대기합니다.

❤ 그림 1-4 이벤트 기반



이벤트 기반 모델에서는 이벤트 루프라는 개념이 등장합니다. 여러 이벤트가 동시에 발생했을 때 어떤 순서로 콜백 함수를 호출할지를 이벤트 루프가 판단합니다. 이 책은 여러분이 자바스크립트의 기초 지식을 알고 있다고 가정하므로 이벤트 루프 역시 이미 알고 있다고 생각하고 넘어갈 것입니다. 하지만 노드와 자바스크립트에서 이벤트 루프는 정말 중요한 개념이니 간략히만 설명하도록 하겠습니다.

노드는 자바스크립트 코드에서 맨 위부터 한 줄씩 실행합니다. 함수 호출 부분을 발견했다면 호출한 함수를 호출 스택에 넣습니다. 다음 코드가 콘솔(브라우저 콘솔을 사용하면 됩니다. 크롬의 경우 F12를 눌렀을 때 나오는 개발자 도구의 Console 탭입니다)에 어떤 로그를 남길지 예측해보세요. 만약 예측하기 어렵다면 자바스크립트를 복습해야 합니다.

```
function first() {
  second();
  console.log('첫 번째');
}
function second() {
  third();
  console.log('두 번째');
}
function third() {
  console.log('세 번째');
}
first();
```

first 함수가 제일 먼저 호출되고, 그 안의 second 함수가 호출된 뒤, 마지막으로 third 함수가 호출됩니다. 호출된 순서와는 반대로 실행이 완료됩니다. 따라서 콘솔에는 세 번째, 두 번째, 첫 번째 순으로 찍히게 됩니다. 이를 쉽게 파악하는 방법은 호출 스택을 그려보는 것입니다.

❤ 그림 1-5 호출 스택

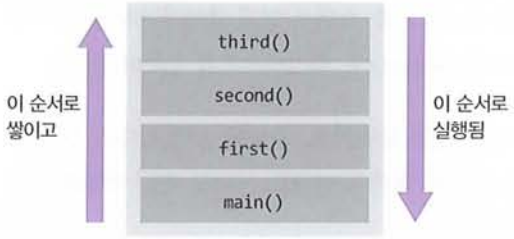


그림 1-5에서 main 함수는 처음 실행 시의 전역 컨텍스트를 의미합니다. 컨텍스트는 함수가 호출되었을 때 생성되는 환경을 의미합니다. 자바스크립트는 실행 시 기본적으로 전역 컨텍스트 안에서 돌아간다고 생각하는 게 좋습니다. 함수의 실행이 완료되면 호출 스택에서 지워집니다. third, second, first, main 순으로 지워지고, main 함수까지 실행이 모두 완료되었다면 호출 스택은 비어 있게 됩니다.

콘솔의 출력 결과는 다음과 같습니다.

콘솔

```
세 번째
두 번째
첫 번째
```

이번에는 특정 밀리초(1000분의 1초) 이후에 코드를 실행하는 setTimeout을 사용하겠습니다. 콘솔에 어떤 로그가 기록될지 예측해보세요.

```
function run() {
  console.log('3초 후 실행');
}
console.log('시작');
setTimeout(run, 3000);
console.log('끝');
```

결과는 다음과 같습니다.

콘솔

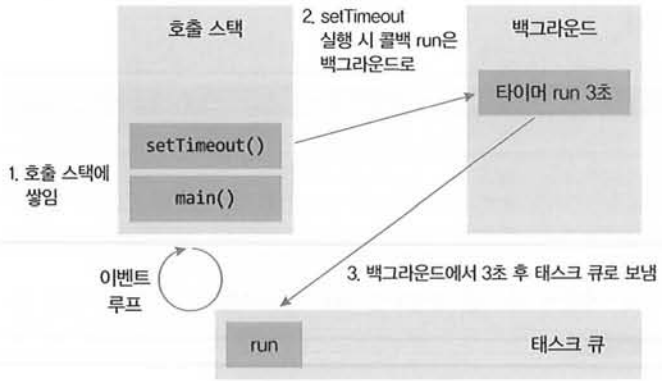
```
시작
끝
3초 후 실행
```

3초 뒤에 run 함수를 실행하는 코드입니다. 콘솔 결과는 쉽게 예측할 수 있지만, 호출 스택으로 설명하기는 힘듭니다. setTimeout 함수의 콜백인 run이 호출 스택에 언제 들어가는지 알기 어렵기 때문입니다. 이를 파악하기 위해서는 이벤트 루프, 태스크 큐, 백그라운드를 알아야 합니다.

- **이벤트 루프**: 이벤트 발생 시 호출할 콜백 함수들을 관리하고, 호출된 콜백 함수의 실행 순서를 결정하는 역할을 담당합니다. 노드가 종료될 때까지 이벤트 처리를 위한 작업을 반복하므로 루프라고 불립니다.
- **태스크 큐**: 이벤트 발생 후 호출되어야 할 콜백 함수들이 기다리는 공간입니다. 콜백들이 이벤트 루프가 정한 순서대로 줄을 서 있으므로 콜백 큐라고도 부릅니다.
- **백그라운드**: 타이머나 I/O 작업 콜백 또는 이벤트 리스너들이 대기하는 곳입니다.

그림 1-6은 코드가 실행되는 내부 과정을 묘사한 그림입니다.

그림 1-6 이벤트 루프 1



먼저 전역 컨텍스트인 main 함수가 호출 스택에 들어갑니다. 그 뒤 setTimeout이 호출 스택에 들어갑니다.

호출 스택에 들어간 순서와 반대로 실행되므로 setTimeout이 먼저 실행됩니다. setTimeout이 실행되면 타이머와 함께 run 콜백을 백그라운드로 보내고 호출 스택에서 빠집니다. 그 다음으로 main 함수가 호출 스택에서 빠집니다. 백그라운드에서는 3초를 센 후 run 함수를 태스크 큐로 보냅니다.

그림상으로는 태스크 큐가 하나의 큐처럼 보이지만 실제로는 여러 개의 큐로 이루어져 있습니다. 이벤트 루프는 정해진 규칙에 따라 콜백 함수들을 호출 스택으로 부릅니다. 더 자세히 공부하고 싶다면 1.5절의 이벤트 루프 설명에 대한 자료를 참고하세요.

그림 1-7은 호출 스택에서 main까지 실행이 완료되어 호출 스택이 비어 있는 상황입니다. 이벤트 루프는 호출 스택이 비어 있으면 태스크 큐에서 함수를 하나씩 가져와 호출 스택에 넣고 실행합니다.

❯ 그림 1-7 이벤트 루프 2

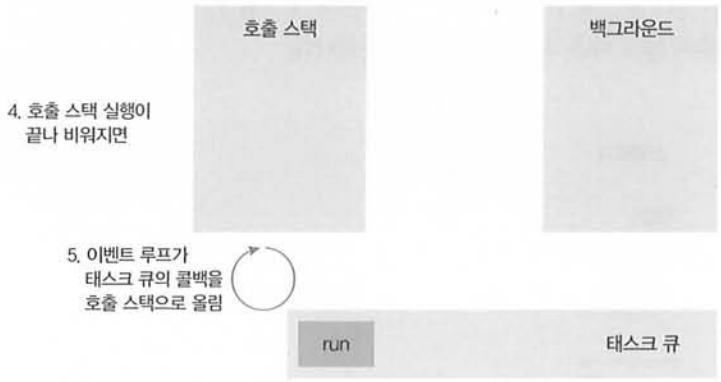
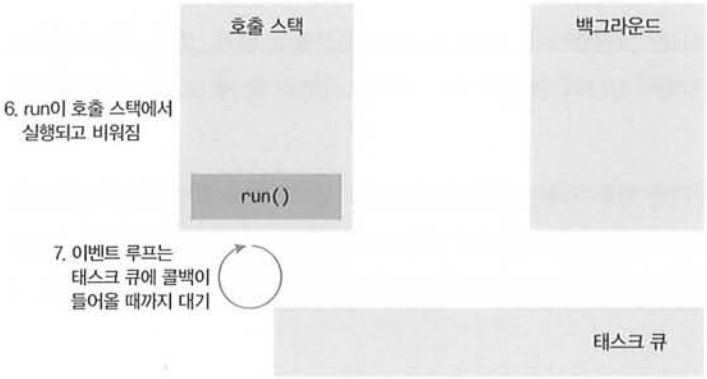


그림 1-8은 이벤트 루프가 run 콜백을 태스크 큐에서 꺼내 호출 스택으로 올린 상황입니다. 호출 스택으로 올려진 run은 실행되고, 실행 완료 후 호출 스택에서 비워집니다. 이벤트 루프는 태스크 큐에 콜백 함수가 들어올 때까지 계속 대기하게 됩니다.

❯ 그림 1-8 이벤트 루프 3



만약 호출 스택에 함수들이 너무 많이 차 있으면 3초가 지난 후에도 run 함수가 실행되지 않을 수도 있습니다. 이벤트 루프는 호출 스택이 비어 있을 때만 태스크 큐에 있는 run 함수를 호출 스택으로 가져오니까요. 이것이 setTimeout의 시간이 정확하지 않을 수도 있는 이유입니다.

1.1.4 논블로킹 I/O

이벤트 루프를 잘 활용하면 오래 걸리는 작업을 효율적으로 처리할 수 있습니다. 오래 걸리는 함수를 백그라운드로 보내서 다음 코드가 먼저 실행되게 하고, 그 함수가 다시 태스크 큐를 거쳐 호출 스택으로 올라오기를 기다리는 방식입니다. 이 방식이 논블로킹 방식입니다. 논블로킹이란 이전 작업이 완료될 때까지 멈추지 않고 다음 작업을 수행함을 뜻합니다.

❯ 그림 1-9 블로킹과 논블로킹

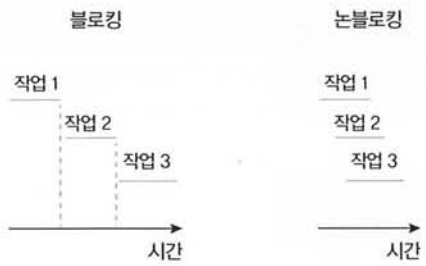


그림 1-9를 보면 블로킹보다 논블로킹 방식이 같은 작업을 더 짧은 시간 동안 처리할 수 있음을 알 수 있습니다. 하지만 싱글 스레드라는 한계 때문에 자바스크립트의 모든 코드가 이 방식으로 시간적 이득을 볼 수 있는 것은 아닙니다. 현재 노드 프로세스 외의 다른 컴퓨팅 자원을 사용할 수 있는 I/O 작업이 주로 시간적 이득을 많이 봅니다.

I/O는 입력(Input)/출력(Output)을 의미합니다. 파일 시스템 접근(파일 읽기, 쓰기, 폴더 만들기 등)이나 네트워크 요청 같은 작업이 I/O의 일종입니다. 이러한 작업을 할 때 노드는 논블로킹 방식으로 동작합니다.

블로킹과 논블로킹 말고도 동기와 비동기라는 개념에 대해서도 들어보았을 것입니다. 이 개념은 코드를 보지 않고서는 이해하기 어렵습니다. 동기와 비동기, 블로킹과 논블로킹의 관계는 3.6.1절에서 코드와 함께 설명합니다. 그 전까지는 동기와 블로킹이 유사하고, 비동기와 논블로킹이 유사하다고만 알아두면 됩니다.

다음 예제는 블로킹 방식의 코드입니다. 콘솔 결과를 미리 예측해보세요.

```
function longRunningTask() {
  // 오래 걸리는 작업
  console.log('작업 끝');
}
console.log('시작');
longRunningTask();
console.log('다음 작업');
```

결과는 다음과 같습니다.

콘솔

```
시작
작업 끝
다음 작업
```

작업을 수행하는 데 오래 걸리는 longRunningTask 함수가 있다고 가정해봅시다. 이 작업이 완료되기 전까지는 이어지는 console.log('다음 작업')이 호출되지 않습니다.

이번에는 setTimeout을 사용해서 코드를 바꿔보겠습니다. 논블로킹 방식의 코드입니다.

```
function longRunningTask() {
  // 오래 걸리는 작업
  console.log('작업 끝');
}
console.log('시작');
setTimeout(longRunningTask, 0);
console.log('다음 작업');
```

결과는 다음과 같습니다.

콘솔

```
시작
다음 작업
작업 끝
```

setTimeout(콜백, 0)은 코드를 논블로킹으로 만들기 위해 사용하는 기법 중 하나입니다. 노드에서는 setTimeout(콜백, 0) 대신 다른 방식을 주로 사용합니다(3.4.3절 참조). 이벤트 루프를 이해했다면 setTimeout의 콜백 함수가 태스크 큐로 보내지므로 순서대로 실행되지 않는다는 것을 알 수 있습니다. 다음 작업이 먼저 실행된 후, 오래 걸리는 작업이 완료됩니다.

> Note ≡ setTimeout(콜백, 0)
>
> 밀리초를 0으로 설정했으므로 바로 실행되는 것이 아닌가 착각할 수 있습니다. 하지만 브라우저와 노드에서는 기본적인 지연 시간이 있으므로 바로 실행되지 않습니다. HTML5 브라우저에서는 4ms, 노드에서는 1ms의 지연 시간이 있습니다.

1.1.5 싱글 스레드

이벤트 기반, 논블로킹 모델과 더불어 노드를 설명할 때 자주 나오는 용어가 하나 더 있습니다. 바로 싱글 스레드입니다. 스레드를 이해하기 위해서는 프로세스도 알아야 하지만, 지금은 그냥 스레드가 컴퓨터 작업을 처리할 수 있는 일손이라고 생각하면 됩니다. 조금 뒤에 자세히 알아봅니다.

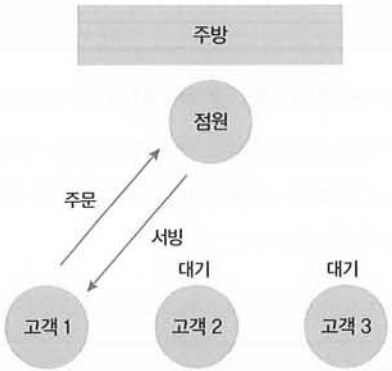
노드가 싱글 스레드라는 말을 들어보셨나요? 노드는 싱글 스레드이므로 주어진 작업을 혼자서 처리해야 합니다. 반대로 멀티 스레드인 시스템에서는 여러 개의 스레드가 일을 나눠서 처리할 수 있습니다.

자바스크립트와 노드에서 논블로킹이 중요한 이유는 바로 싱글 스레드이기 때문입니다. 한 번에 한 가지 일밖에 처리하지 못하므로 어떠한 작업에서 블로킹이 발생하면 다음 일을 처리하지 못합니다.

언뜻 보면 여러 개의 일을 동시에 처리할 수 있기 때문에 멀티 스레드가 싱글 스레드보다 좋아 보입니다. 하지만 꼭 그런 것만은 아닙니다. 이해를 돕기 위한 예시를 하나 들어보겠습니다.

한 음식점에 점원이 한 명 있습니다. 손님은 여러 명이고요. 점원 한 명이 주문을 받아 주방에 넘기고, 주방에서 요리가 나오면 손님에게 서빙을 합니다. 그 후 다음 손님의 주문을 받습니다. 이런 구조라면 다음 손님은 이전 손님의 요리가 나올 때까지 아무것도 못 하고 기다리고 있어야 합니다. 이것이 바로 싱글 스레드(점원), 블로킹 모델입니다. 매우 비효율적입니다.

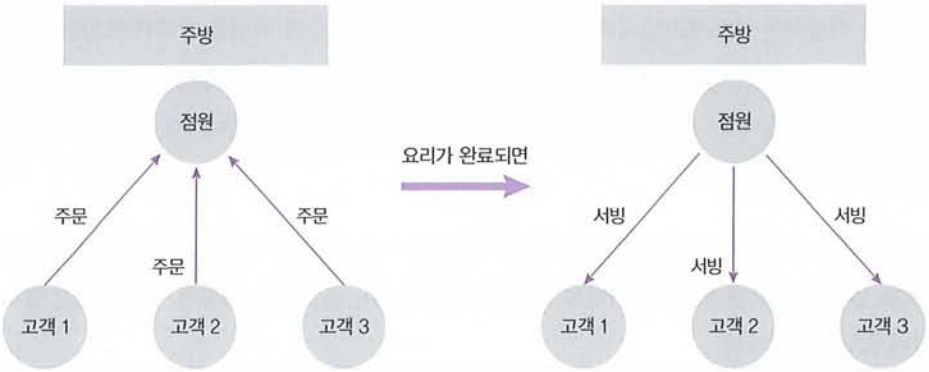
그림 1-10 싱글 스레드, 블로킹 모델



이번에는 점원이 한 손님의 주문을 받고, 주방에 주문 내역을 넘긴 뒤 다음 손님의 주문을 받습니다. 요리가 끝나길 기다리지 않고 주문이 들어왔다는 것만 알려주는 것입니다. 주방에서 요리가 완료되면 완료된 순서대로 손님에게 서빙합니다. 주문한 순서와 서빙하는 순서가 일치하지 않을 수도 있습니다.

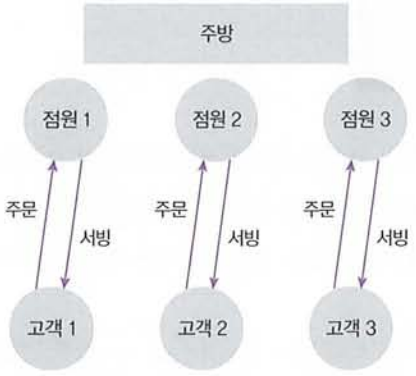
이것이 싱글 스레드, 논블로킹 모델입니다. 바로 노드가 채택하고 있는 방식입니다. 점원은 한 명이지만 혼자서 많은 일을 처리할 수 있습니다. 하지만 그 점원 한 명이 아파서 쓰러지거나 하면 큰 문제가 생길 수 있습니다. 또한, 주문을 받거나 서빙을 하는 데 시간이 오래 걸린다면 주문이 많이 들어 왔을 때 버거울 수 있습니다.

그림 1-11 싱글 스레드, 논블로킹 모델



멀티 스레드 방식에서는 손님이 올 때마다 점원이 한 명씩 맡아 주문을 받고 서빙합니다. 언뜻 보면 싱글 스레드보다 좋은 방법인 것 같지만, 장단점이 있습니다. 일단 손님 한 명당 점원도 한 명이면 서빙 자체는 걱정이 없습니다. 점원 한 명에게 문제가 생겨도 다른 점원으로 대체하면 되기 때문입니다. 하지만 손님의 수가 늘어날수록 점원의 수도 늘어납니다. 손님 수가 줄어들었을 때 일을 하지 않고 노는 점원이 있다는 것도 문제가 됩니다. 점원을 새로 고용하거나 기존 직원을 해고하는 데는 비용이 발생합니다.

그림 1-12 멀티 스레드, 블로킹 모델



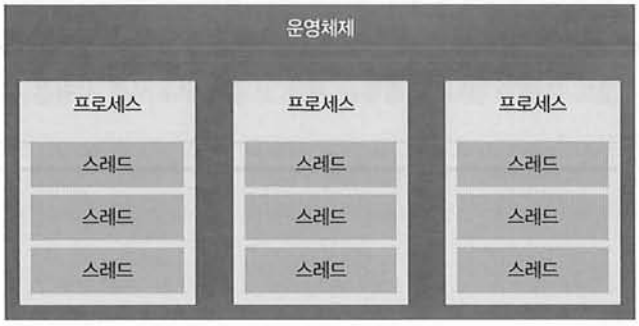
그렇다면 점원 여러 명(멀티 스레드)이 모두 논블로킹 방식으로 주문을 받으면 더 좋지 않을까 하는 의문이 들 수 있습니다. 실제로 그렇습니다. 노드도 싱글 스레드 여러 개를 사용해 멀티 스레딩과 비슷한 기능을 하게 할 수 있습니다. 하지만 엄밀히 말하면 멀티 스레딩이라기보다는 멀티 프로세싱에 가깝습니다. 그럼 프로세스와 스레드의 차이에 대해 알아봅시다.

- 프로세스는 운영체제에서 할당하는 작업의 단위입니다. 노드나 인터넷 브라우저 같은 프로그램은 개별적인 프로세스입니다. 프로세스 간에는 메모리 등의 자원을 공유하지 않습니다.
- 스레드는 프로세스 내에서 실행되는 흐름의 단위입니다. 하나의 프로세스는 스레드를 여러 개 가질 수 있습니다. 스레드들은 부모 프로세스의 자원을 공유합니다. 즉, 같은 메모리에 접근할 수 있습니다.

스레드를 작업을 처리하는 일손으로 표현하기도 하는데, 노드 프로세스는 일손이 하나인 셈입니다. 요청이 많이 들어오면 한 번에 하나의 요청을 처리합니다. 블로킹이 심하게 일어나지만 않는다면 하나로도 충분합니다.

사실 노드 프로세스도 내부적으로는 스레드를 여러 개 가지고 있습니다. 하지만 여러분이 직접 제어할 수 있는 스레드는 하나뿐이므로 흔히 싱글 스레드라고 부르는 것입니다.

❯ 그림 1-13 스레드와 프로세스



노드는 스레드를 늘리는 대신, 프로세스 자체를 복사해 여러 작업을 동시에 처리하는 멀티 프로세싱 방식을 택했습니다. 자바스크립트 언어 자체가 싱글 스레드 특성을 띠고 있기 때문입니다. 4.5절의 cluster 모듈과 15.1.5절의 pm2 패키지에서 멀티 프로세싱을 가능하게 하는 방법에 대해 알아봅니다.

1.2 서버로서의 노드

이 절에서는 노드를 서버로 사용할 때의 특징과 장단점에 대해 알아보겠습니다.

노드가 싱글 스레드, 논블로킹 모델을 사용하므로 노드 서버 또한 동일한 모델일 수밖에 없습니다. 따라서 노드 서버의 장단점은 싱글 스레드, 논블로킹 모델의 장단점과 크게 다르지 않습니다. 싱글 스레드여서 멀티 스레드 방식보다는 컴퓨터 자원을 적게 사용하는 장점이 있지만, CPU 코어를 하나밖에 사용하지 못하는 단점도 있습니다.

노드 서버는 I/O가 많은 작업에 적합합니다. 노드는 libuv 라이브러리를 사용하여 I/O 작업을 논블로킹 방식으로 처리해줍니다. 따라서 스레드 하나가 많은 수의 I/O를 혼자서도 감당할 수 있습니다. 하지만 CPU 부하가 큰 작업에는 적합하지 않습니다. 여러분이 작성하는 코드는 모두 스레드 하나에서 처리됩니다. 여러분의 코드가 CPU 연산을 많이 요구하면 블로킹이 발생해 스레드 하나가 감당하기 어렵습니다.

싱글 스레드 방식의 프로그래밍은 멀티 스레드 방식보다 상대적으로 쉽습니다. 서버 프로그래밍에 익숙하지 않은 사람도 쉽게 입문할 수 있습니다. 하지만 싱글 스레드이다 보니 하나뿐인 스레드가 에러로 인해 멈추지 않도록 잘 관리해야 합니다. 에러를 제대로 처리하지 못하면 서버 전체가 멈추기 때문입니다.

또한, 웹 서버가 내장되어 있어 입문자가 쉽게 접근할 수 있습니다. 노드 외의 서버를 개발하다 보면 Apache, nginx, IIS처럼 별도의 웹 서버를 설치해야 하는 경우가 많습니다. 심지어 Tomcat 같은 웹 애플리케이션 서버(WAS)를 추가로 설치하는 경우도 있습니다. 이 경우 프로그래밍 외에도 웹 서버와 WAS 사용법을 익혀야 합니다. 하지만 노드는 내장된 웹 서버를 사용하면 되므로 편리합니다. 하지만 나중에 서버 규모가 커지면 결국 nginx 등의 웹 서버를 노드 서버와 연결해야 합니다.

노드 사용자들이 말하는 가장 큰 장점은 언어로 자바스크립트를 사용한다는 것입니다. 웹 브라우저도 자바스크립트를 사용하므로 서버까지 노드를 사용하면 하나의 언어로 웹 사이트를 개발할 수 있습니다. 이는 개발 생산성을 획기적으로 높여주었고, 생산성이 중요한 기업이 노드를 채택하는 이유가 되었습니다.

노드는 생산성은 매우 좋지만, Go처럼 비동기에 강점을 보이는 언어나 nginx처럼 정적 파일 제공, 로드 밸런싱에 특화된 서버에 비해서는 속도가 느립니다. 그렇긴 해도 극단적인 성능이 필요하지 않다면 이러한 단점은 노드의 생산성으로 어느 정도 극복할 수 있습니다.

자바스크립트를 사용하기 때문에 얻을 수 있는 소소한 장점도 있습니다. 요즘은 XML 대신 JSON을 사용해서 데이터를 주고 받는데, JSON이 자바스크립트 형식이어서 노드에서는 쉽게 처리할 수 있습니다.

❤ 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	

이와 같은 특성을 활용하려면 노드를 어디에 사용해야 할까요? 개수는 많지만 크기는 작은 데이터를 실시간으로 주고 받는 데 적합합니다. 네트워크나 데이터베이스, 디스크 작업 같은 I/O에 특화되어 있기 때문입니다. 실시간 채팅 애플리케이션이나 주식 차트, JSON 데이터를 제공하는 API 서버가 노드를 많이 사용합니다.

노드가 아무리 좋다고 하더라도 추천하지 않는 경우도 있습니다. 이미지나 비디오 처리, 대규모 데이터 처리 같이 CPU를 많이 사용하는 작업을 위한 서버로는 권장하지 않습니다. 노드보다 더 적합한 다른 언어 서버가 많습니다. 요즘은 AWS Lambda나 Google Cloud Functions 같은 서비스에서 노드로 CPU를 많이 사용하는 작업을 처리하는 것을 지원합니다. 16장에서 사용해볼 것입니다.

그렇다면 실생활과 밀접한 쇼핑몰, 블로그 같은 웹 사이트에는 적합할까요? 이런 사이트는 보통 기본적인 틀이 있고, 그 안의 내용물(텍스트, 이미지)만 조금씩 달라집니다. 노드가 다른 서버 언어에 비해 이러한 콘텐츠를 제공하는 데 장점이 뚜렷하지는 않습니다. 하지만 그렇다고 적합하지 않다는 것도 아닙니다. Pug나 EJS 같은 템플릿 엔진을 통해서 다른 언어와 비슷하게 콘텐츠를 제공할 수 있습니다. 템플릿 엔진은 6.5절에서 다룹니다.

안정성과 보안성 측면의 문제도 이미 충분히 검증되었습니다. 규모가 큰 곳을 꼽자면 미국항공우주국(NASA), 에어비엔비, 우버, 넷플릭스, 링크드인 등에서 노드를 사용하고 있습니다. 페이팔, 월마트, 이베이 같이 결제 시스템을 사용하는 대기업들도 노드로 서비스를 운영합니다.

그림 1-14 노드를 사용하는 결제 서비스

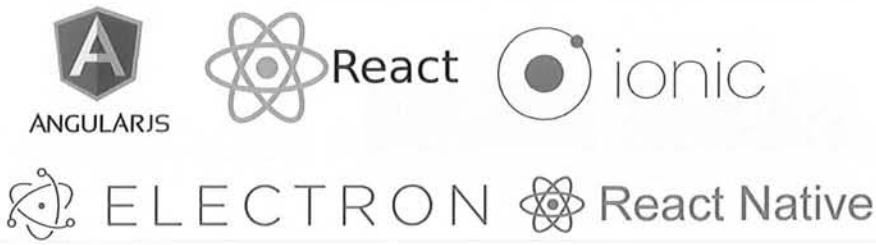
1.3 서버 외의 노드

NODE.JS

처음에는 대부분 노드를 서버로 사용했지만, 노드는 자바스크립트 런타임이기 때문에 용도가 서버에만 한정된 것은 아닙니다. 사용 범위가 점점 늘어나 웹, 모바일, 데스크톱 애플리케이션 개발에도 사용되기 시작했습니다.

노드 기반으로 돌아가는 대표적인 웹 프레임워크로는 Angular과 React, Vue, Meteor 등이 있습니다. Angular는 구글 진영에서 프런트엔드 앱을 만들 때 주로 사용하고, React는 페이스북 진영에서 주로 사용합니다. 모바일 개발 도구로는 React Native와 Ionic Framework를 많이 사용합니다. 페이스북, 인스타그램, 에어비앤비, 월마트, 테슬라 등이 React Native를 사용하여 모바일 앱을 운영 중입니다. 데스크톱 개발 도구로는 Electron이 대표적입니다. Electron으로 만들어진 프로그램으로는 Atom, Slack, Discord 등이 있습니다. 이 책에서 사용할 에디터인 비주얼 스튜디오 코드도 Electron으로 만들어졌습니다.

그림 1-15 노드 기반의 개발 도구



1.4 개발 환경 설정하기

이 책에서 가장 중요한 부분입니다. 노드를 설치하지 못하면 이 책의 나머지 부분을 진행할 수 없으니 조심스럽게 따라해주세요. 에디터로는 비주얼 스튜디오 코드(Visual Stuido Code, 이하 VS Code)를 설치합니다. 다른 에디터를 사용해도 되지만, 무료 에디터 중에는 VS Code를 추천합니다. 이 책은 특정 에디터에 심하게 의존하지 않으므로 기존에 쓰던 에디터가 있다면 계속 사용해도 됩니다.

1.4.1 노드 설치하기

그럼 각 운영체제별로 노드를 설치해보겠습니다. 윈도와 맥은 GUI를 사용하므로 웹 브라우저를 통해 설치하겠습니다. 리눅스는 일반적으로 터미널을 통해서 접근하므로 터미널로 설치하는 방법을 알아봅니다.

1.4.1.1 윈도

이 책은 윈도 10을 기준으로 합니다. 노드의 공식 사이트(https://nodejs.org)에 접속합니다.

그림 1-16 노드의 공식 사이트 접속

LTS와 Current 버전 중 Current인 10.0.0 버전을 설치합니다(2018년 5월 기준). 세부 버전은 다를 수 있지만, 첫 번째 자리가 10이면 됩니다. 오른쪽 초록색 버튼을 눌러 파일을 내려받습니다.

❯ 그림 1-17 Current 버전 내려받기

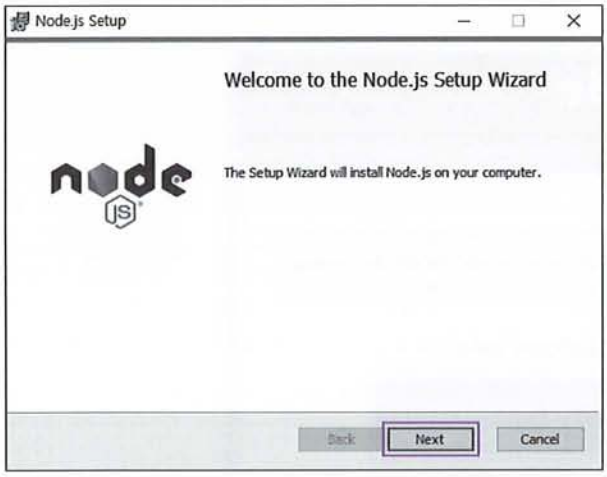
Note ☰ LTS와 Current 버전의 차이

- LTS: 기업을 위해 3년간 지원하는 버전입니다. 짝수 버전만 LTS 버전이 될 수 있습니다. 서버를 안정적으로 운영해야 할 경우 선택하세요. 하지만 최신 기능을 사용하지 못할 수도 있습니다.
- Current: 최신 기능을 담고 있는 버전입니다. 다소 실험적인 기능이 들어 있어 예기치 못한 에러가 발생할 수 있습니다. 서버에 신기능이 필요하거나 학습용으로 사용할 때 적합합니다. 단, 짝수 버전은 LTS가 되기 때문에 Current일 때부터 사용하는 것을 고려해볼 만합니다.
- 홀수 버전: 노드는 6개월마다 버전을 1씩 올립니다. 따라서 10 버전 이전에 9 버전도 있었습니다. 하지만 홀수 버전은 LTS를 지원하지 않으므로 10 버전이 나오면서 9 버전은 사라졌습니다. 나중에 11 버전이 나오면 10 버전이 LTS로 가고 11 버전이 Current가 됩니다.

내려받은 파일을 클릭하여 Setup Wizard를 실행합니다.

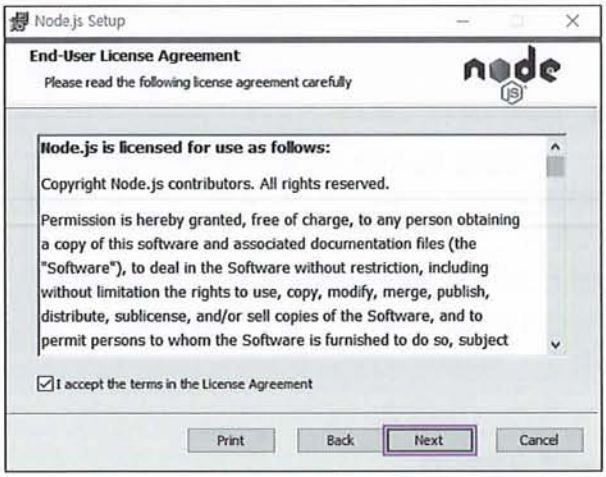
Setup Wizard 실행 화면이 나오면 Next 버튼을 눌러 다음으로 넘어갑니다.

❤ 그림 1-18 Setup Wizard 실행



라이선스 동의 화면이 나오면 체크박스에 체크표시하고 Next 버튼을 누릅니다.

❤ 그림 1-19 라이선스 동의

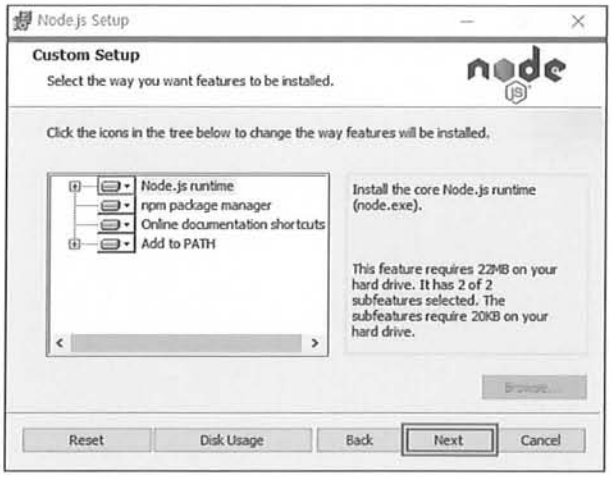


그 다음 Node.js를 설치할 폴더 경로를 지정합니다. 이 책에서는 기본 경로 그대로 진행하겠습니다.

그림 1-20 설치할 폴더 경로 지정

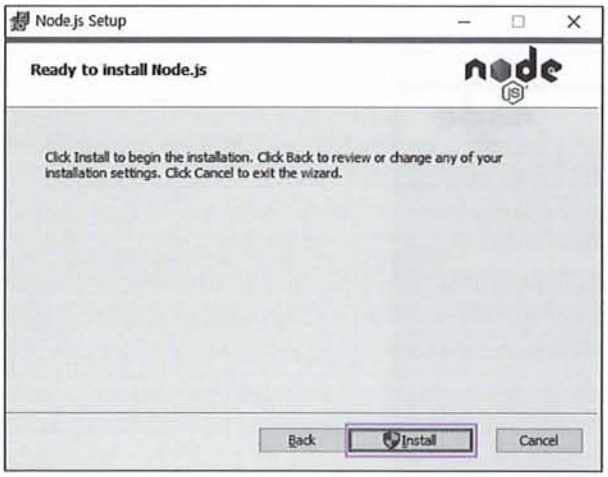
다음 화면에서는 설치할 프로그램을 선택할 수 있습니다. 위에서부터 순서대로 설명하면 노드 런타임, 노드 패키지 관리자, 온라인 문서 바로가기, 명령 프롬프트에서 노드 명령어를 사용할 수 있게 해주는 시스템 환경 변수입니다. 기본적으로 모두 설치됩니다. Next 버튼을 눌러 다음으로 넘어갑니다.

그림 1-21 설치할 프로그램 선택



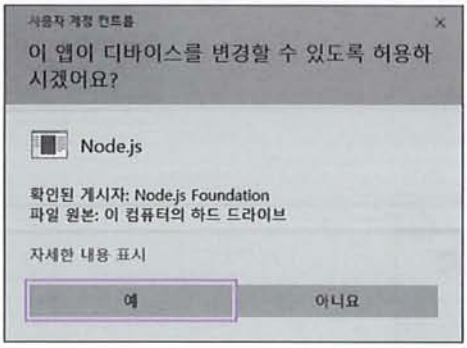
다음에 나오는 화면에서 Install 버튼을 누릅니다.

❯ 그림 1-22 설치 시작



권한 확인 창이 나오면 **예** 버튼을 눌러 설치를 시작합니다.

❯ 그림 1-23 권한 확인 창

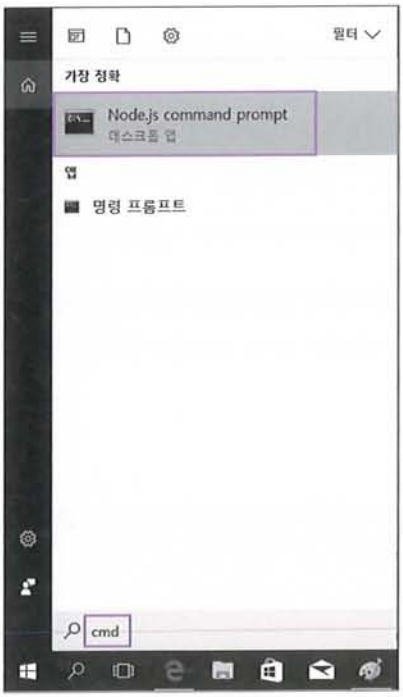


설치 완료 화면이 나오면 Finish 버튼을 눌러 설치를 마칩니다.

❤ 그림 1-24 설치 완료

이제 설치가 정상적으로 완료되었는지 확인해보겠습니다. 먼저 ⊞+S를 누르고 검색창에 cmd를 입력합니다. 결과가 뜨면 Node.js command prompt를 실행합니다. Node.js command prompt가 없다면 명령 프롬프트를 대신 실행합니다.

❤ 그림 1-25 Node.js command prompt 실행



다음 그림과 같이 명령 프롬프트 창이 열립니다.

❤ 그림 1-26 명령 프롬프트 창

다음 명령어를 입력해서 노드의 버전이 올바르게 설치되었는지 확인합니다.

명령 프롬프트

```
> node -v
v10.0.0
```

나중에 npm(노드 패키지 매니저)을 사용해야 하므로 npm이 제대로 설치되었는지도 확인합니다.

명령 프롬프트

```
> npm -v
5.5.1
```

이 책의 버전과는 다를 수 있지만, npm 버전이 명령 프롬프트 창에 뜬다면 설치에 성공한 것입니다. 만약 버전이 뜨지 않고 에러 메시지가 나온다면 노드를 처음부터 다시 설치해야 합니다.

1.4.1.2 맥

이 책은 하이 시에라(10.13)를 기준으로 합니다. 노드의 공식 사이트(https://nodejs.org)에 접속합니다.

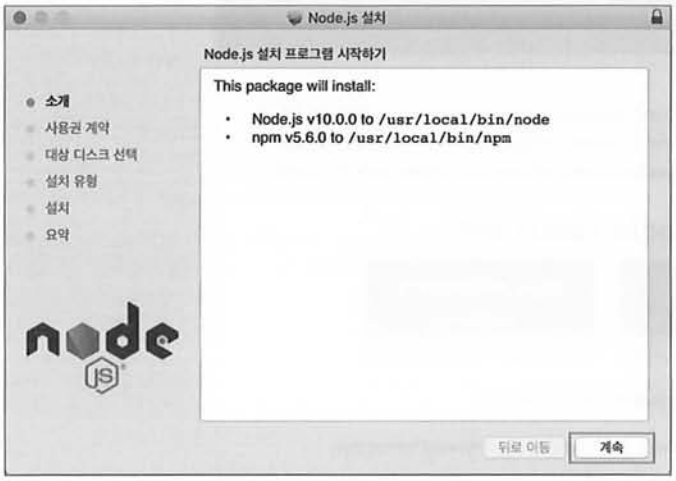
❤ 그림 1-27 노드의 공식 사이트 접속

LTS와 Current 버전 중 출간일 기준으로 Current인 10.0.0 버전을 설치합니다. 세부 버전은 다를 수 있지만, 첫 번째 자리가 10이면 됩니다. 오른쪽 초록색 버튼을 누르면 됩니다.

❤ 그림 1-28 Current 버전 내려받기

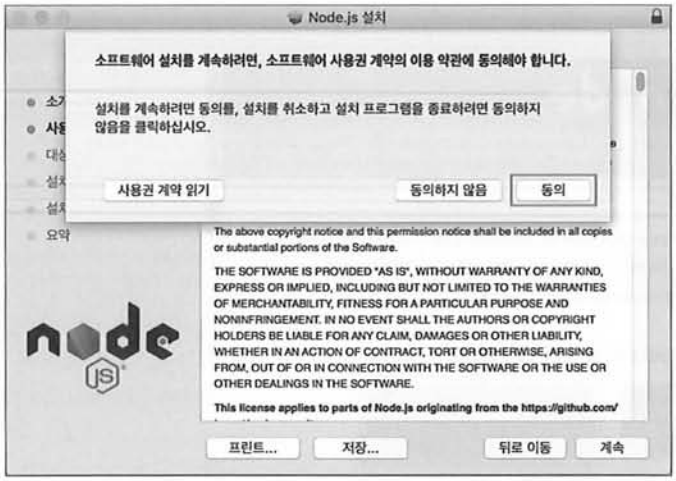
내려받은 pkg 파일을 실행한 후 **계속** 버튼을 누릅니다.

❯ 그림 1-29 pkg 파일 실행



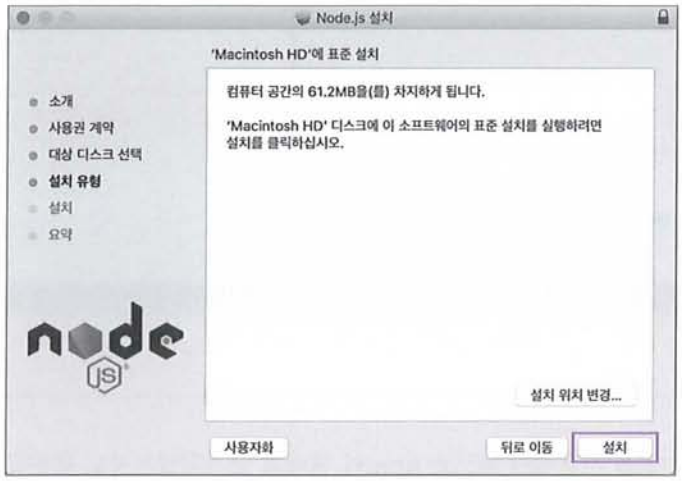
약관 동의 화면이 나오면 **동의** 버튼을 누릅니다.

❯ 그림 1-30 약관 동의



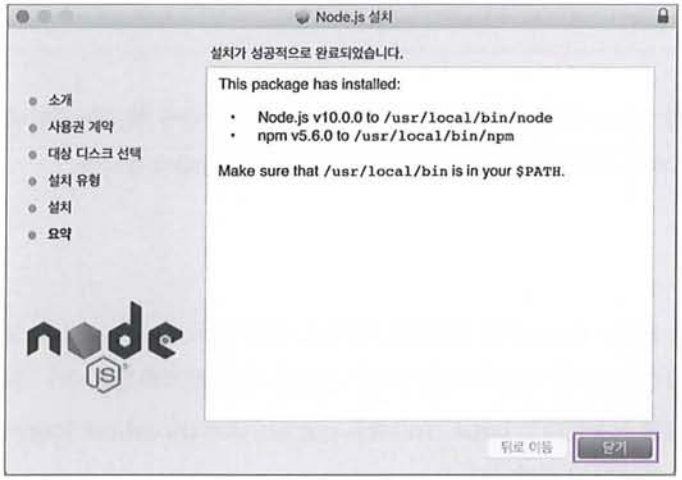
설치 버튼을 누르면 설치가 시작됩니다.

❯ 그림 1-31 설치 시작



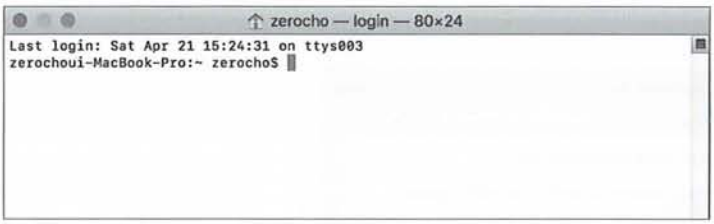
설치 완료 화면이 나오면 **닫기** 버튼을 눌러 설치를 마칩니다.

❯ 그림 1-32 설치 완료



이제 설치가 정상적으로 완료되었는지 확인하기 위해 터미널을 실행합니다. 먼저 cmd+space를 눌러 Spotlight를 실행한 후 terminal.app을 입력합니다.

그림 1-33 터미널을 실행하여 명령어 입력



다음 명령어를 입력해서 노드의 버전이 올바르게 설치되었는지 확인합니다.

터미널

```
$ node -v
v10.0.0
```

나중에 npm(노드 패키지 매니저)을 사용해야 하므로 npm이 제대로 설치되었는지도 확인합니다.

터미널

```
$ npm -v
5.6.0
```

이 책의 버전과는 다를 수 있지만, npm 버전이 명령 프롬프트 창에 뜬다면 설치에 성공한 것입니다. 만약 버전이 뜨지 않고 에러 메시지가 나온다면 노드를 처음부터 다시 설치해야 합니다.

1.4.1.3 리눅스(우분투)

이 책은 우분투 16.04(Ubuntu 16.04 Xenial)를 기준으로 합니다. 맥과 윈도에서는 GUI로 쉽게 설치했지만, 리눅스에서는 보통 GUI를 사용하지 않으므로 콘솔을 통해 설치하겠습니다. 우분투 외의 다른 리눅스 운영체제를 사용한다면 https://nodejs.org/ko/download/package-manager 링크에서 설치 방법을 확인하기 바랍니다.

콘솔에 접속한 후 다음 명령어 다섯 줄을 한 줄씩 입력합니다. 원래는 마지막 두 줄만 입력해도 되지만, 에러가 발생할 수도 있으므로 앞의 세 줄도 추가로 입력해줍니다.

콘솔

```
$ sudo apt-get update
$ sudo apt-get install -y build-essential
$ sudo apt-get install curl
```

```
$ curl -sL https://deb.nodesource.com/setup_10.x | sudo -E bash --
$ sudo apt-get install -y nodejs
```

설치 후 제대로 설치되었는지 확인하기 위해 콘솔 창에 다음 명령어를 입력해봅니다.

콘솔

```
$ node -v
10.0.0
$ npm -v
5.6.0
```

이 책의 버전과는 다를 수 있지만, npm 버전이 명령 프롬프트 창에 뜬다면 설치에 성공한 것입니다. 만약 버전이 뜨지 않고 에러 메시지가 나온다면 노드를 처음부터 다시 설치해야 합니다.

1.4.2 npm의 버전 업데이트하기

npm의 버전이 빠른 속도로 업데이트되므로 최신 버전은 이 책에 나오는 버전과 많이 다를 수 있습니다. 2018년 5월 기준으로 최신 npm 버전은 6.0.0입니다. npm 5 버전 이상이기만 하면 이 책을 실습하는 데 큰 문제는 없습니다. 하지만 최신 버전을 사용하고 싶거나, 현재 버전이 너무 낮다면 명령 프롬프트 또는 터미널에 다음 명령어를 입력하여 업데이트하세요.

콘솔

```
$ npm install -g npm
```

맥과 리눅스의 경우에는 명령어 앞에 sudo를 붙인 후, 계정 비밀번호를 입력해야 할 수도 있습니다.

업데이트 완료 후 다시 npm -v를 입력하면 최신 버전의 npm이 설치되었음을 확인할 수 있습니다. 방금 실행한 명령어에 대해서는 5장에서 자세히 배웁니다.

> Note ☰ **노드의 버전 업데이트**
>
> 노드의 버전을 최신 버전으로 업데이트하는 가장 쉬운 방법은 현재 설치된 노드를 제거했다가 최신 버전을 설치하는 것입니다. 하지만 버전이 바뀔 때마다 매번 지웠다 설치하는 것은 번거롭습니다. 이를 쉽게 해주는 도구가 있지만 npm 명령어와 패키지 개념을 알아야 하므로 지금 바로 설명하지는 않겠습니다. 업데이트 도구 사용 방법은 15.1.9절에 나와 있습니다.

1.4.3 VS Code 설치하기

VS Code는 마이크로소프트사에서 만든 소스 코드 편집기입니다. 노드 기반의 Electron으로 만들어졌습니다. 무료로 사용할 수 있으며, 프로그래밍 생산성을 높여주는 플러그인들을 다양하게 지원합니다. 노드로 만들어진 에디터로 다시 노드 프로그래밍을 하다니 재미있습니다.

> Note ≡ **다른 에디터**
>
> VS Code 외에 많이 사용하는 무료 에디터로는 Bracket, Sublime Text와 Atom이 있습니다. JetBrains 사의 WebStorm은 유료이고 컴퓨터 자원을 많이 사용하는 IDE(통합 개발 환경)이지만 한 번 써보기를 추천합니다. 무료 에디터들과는 다른 강력한 개발 편의 기능들을 경험할 수 있습니다.

1.4.3.1 윈도

윈도에 VS Code를 설치해봅시다. 먼저 VS Code의 공식 사이트(https://code.visualstudio.com/)에 접속합니다.

❤ 그림 1-34 VS Code의 공식 사이트

Download for Windows 버튼을 누르면 자동으로 내려받기가 실행됩니다.

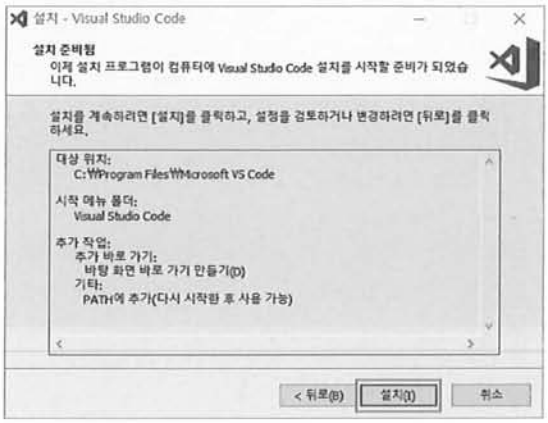
❤ 그림 1-35 내려받기 시작 화면

내려받기가 완료되면 설치 파일을 실행합니다. 사용자 계정 컨트롤에서 앱 실행을 허용할지를 묻는데, **예** 버튼을 누르면 됩니다. 설치 화면이 뜨면 계속 **다음** 버튼을 눌러 진행합니다.

❤ 그림 1-36 VS Code 설치 화면

설치 버튼을 눌러 설치를 시작합니다.

❤ 그림 1-37 설치 버튼을 눌러 설치 시작

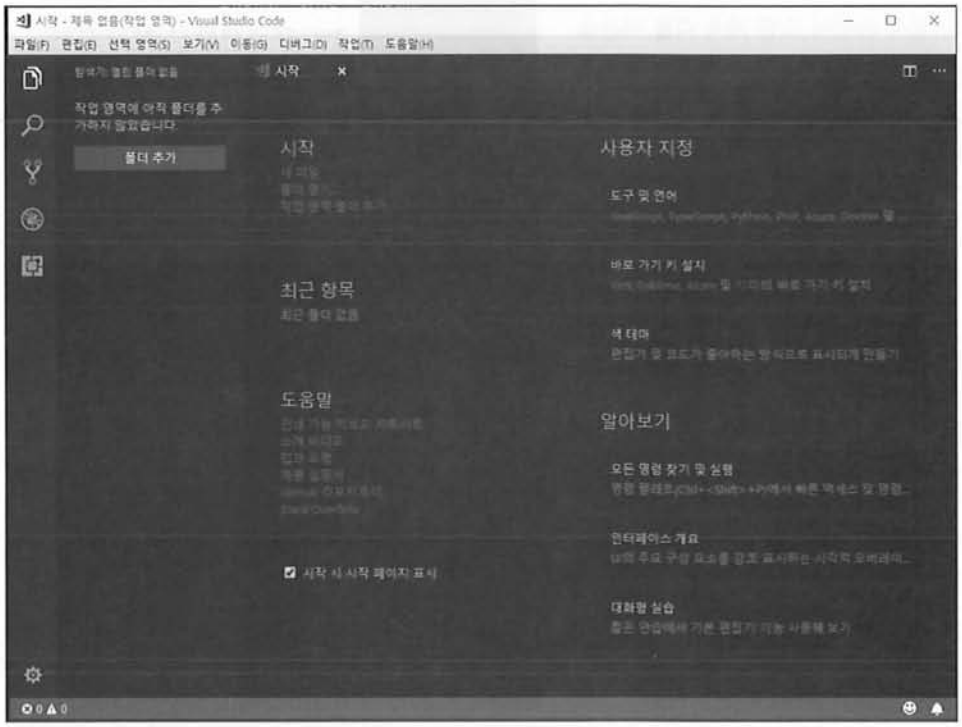


설치 완료 화면에서 **마침** 버튼을 누르면 VS Code가 시작됩니다.

❤ 그림 1-38 설치 완료 화면

그림 1-39는 윈도에서 VS Code를 실행한 화면입니다. 새 파일을 눌러서 코드를 작성하거나 작업 영역 폴더 추가를 눌러서 기존 소스가 있는 폴더를 선택하면 됩니다. 파일은 ctrl+S로 저장할 수 있습니다.

❤ 그림 1-39 VS Code 실행 화면



1.4.3.2 맥

맥에 VS Code를 설치해봅시다. VS Code의 공식 사이트(https://code.visualstudio.com/)에 접속합니다.

❤ 그림 1-40 VS Code의 공식 사이트

Download For Mac 버튼을 누르면 자동으로 내려받기가 실행됩니다.

❤ 그림 1-41 내려받기 시작 화면

내려받기가 완료되면 파일을 클릭하여 압축을 해제합니다. 그러고 나서 압축 해체한 파일을 실행합니다.

경고창이 뜨면 **열기** 버튼을 누릅니다.

❤ 그림 1-42 경고창

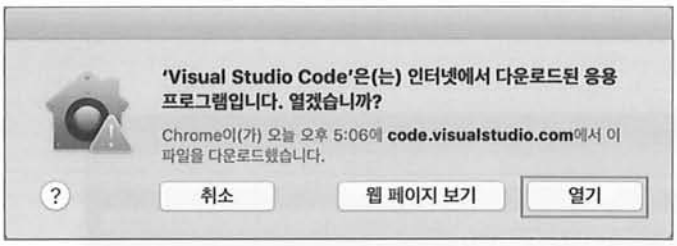
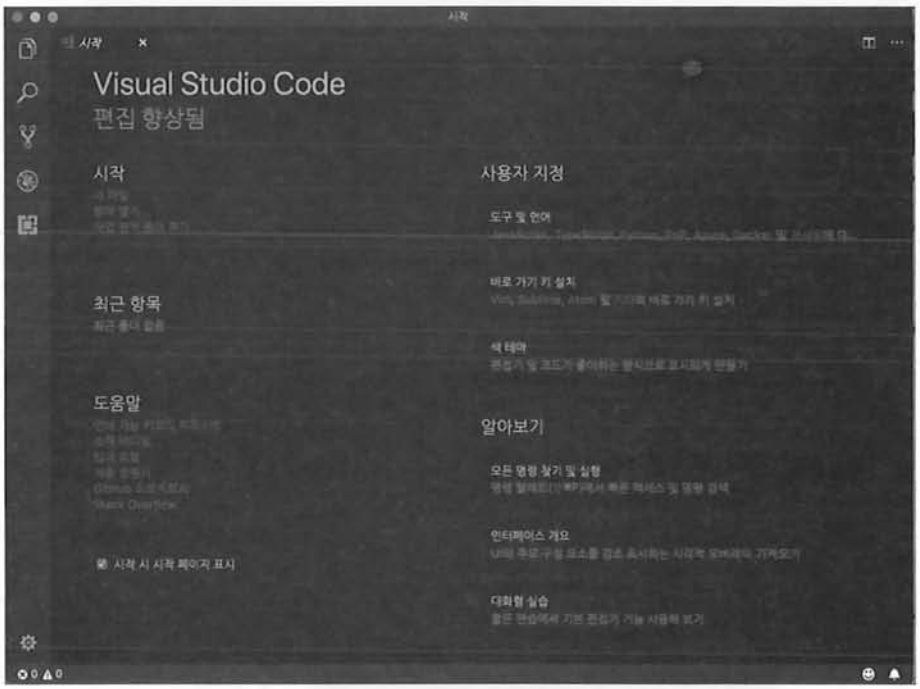


그림 1-43은 맥에서 VS Code를 실행한 화면입니다. 새 파일을 눌러서 코드를 작성하거나 작업 영역 폴더 추가를 눌러서 기존 소스가 있는 폴더를 선택하면 됩니다. 파일은 command+S로 저장할 수 있습니다.

❤ 그림 1-43 VS Code 실행 화면



1.4.3.3 리눅스(우분투)

우분투 운영체제에서는 GUI를 사용하지 않을 것이므로 VS Code 대신 vim 같은 에디터로 진행하면 됩니다.

1.5 함께 보면 좋은 자료

- **노드 공식 사이트**: https://nodejs.org/ko
- **노드 공식 사이트의 가이드**: https://nodejs.org/en/docs/guides/
- **이벤트 루프에 대한 시각적 설명**: http://latentflip.com/loupe
- **이벤트 루프에 대한 설명**: https://nodejs.org/ko/docs/guides/event-loop-timers-and-nexttick/
- **VS Code 공식 사이트**: https://code.visualstudio.com/