



***4**

Node.JS의 이해

웹프로그래밍응용 / 03.17.2023

ut-nodejs.github.io



Contents / 내용

Welcome to Web Programming Application /
웹프로그래밍응용에 환영합니다~



01. Node.js란?

Node.js란 “비동기 이벤트 구동 JavaScript 런타임”이라고 합니다.

02. Node.js의 이해

Node.js는 JavaScript 코드를 해석하고 애플리케이션을 실행하기 위한 플랫폼이다.

03. 추가 주제

REPL과 npm에 대해 설명합니다.

01

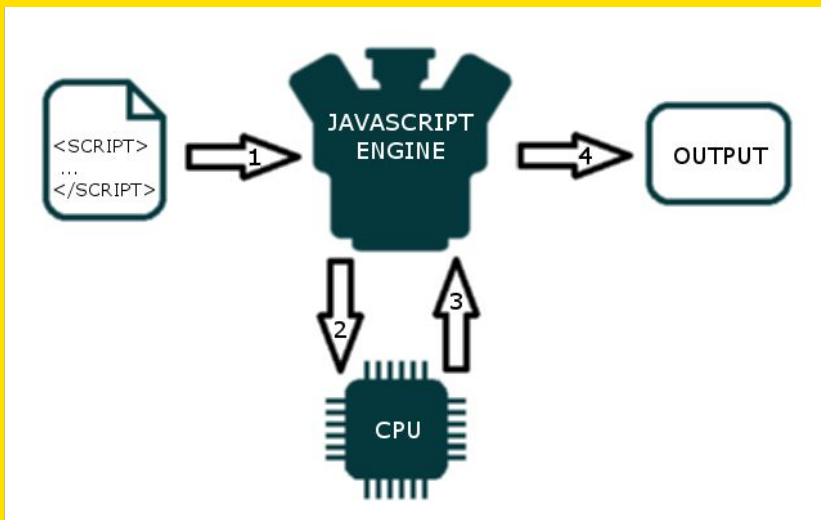
Node.js란?

Node.js란 “비동기 이벤트 구동
JavaScript 런타임”이라고 합니다.

p. 25-27

Node.js란 “비동기 이벤트 구동 JS 런타임”

Node.js는 여러분의 작성한 JS 코드를 읽고 해석한다.



<http://dolszewski.com/javascript/javascript-runtime-environment/>

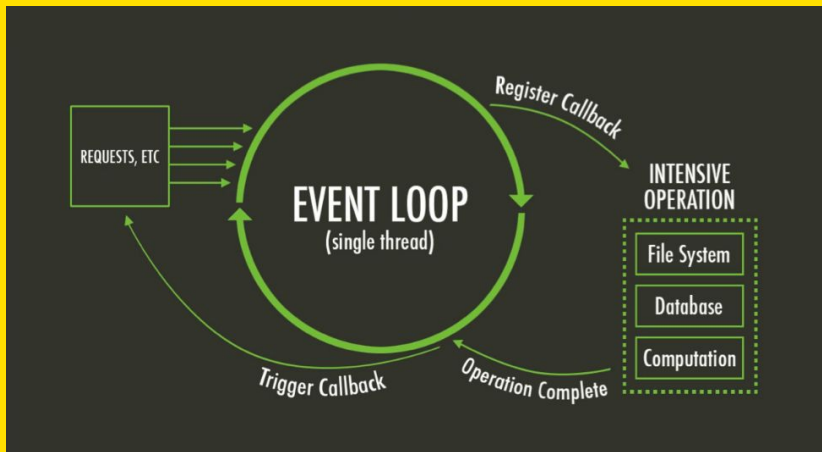
JS 런타임

Node.js 런타임은 JavaScript 엔진을 사용하며, 엔진은 JavaScript 코드를 실시간으로 읽고 해석하고 실행한다. 구체적으로는 Node.js는 구글 크롬 V8 엔진을 쓰며, 이는 JavaScript를 머신 코드로 바꿔주는 오픈소스 해석기다.

이 사양은 상당히 유용하다. 구글이 JavaScript 엔진이 수행되는 자사의 크롬 웹 브라우저를 위해 항상 엔진을 업데이트하고 모니터링하기 때문이다. Node.js는 JavaScript 코드 실행 환경에 웹 브라우저가 필요 없는 이 엔진을 채용하고 있다.

Node.js란 “비동기 이벤트 구동 JS 런타임”

Node.js는 여러분의 작성한 JS 코드를 읽고 해석한다.



<https://ecommerce-consultant.medium.com/nodejs-event-loop-how-this-event-driven-mechanism-work-16bd9c56d52f>

이벤트 구동

JavaScript 애플리케이션이 실행되면 애플리케이션의 모든 코드는 메모리로 올라간다. 모든 변수, 함수, 코드 블록들이 애플리케이션이 실행 여부에 상관없이 사용 가능한 상태로 된다. 왜 바로 실행되지 않는 코드들이 생기는 것일까?

애플리케이션이 실행될 때 변수에 값을 전달하는 전역 변수를 정의하고 할당을 하더라도 모든 함수가 이유 없이 실행되지 않는다. 어떤 함수들은 이벤트 리스너의 형태를 띠고 있다. 이 함수들은 이벤트가 발생할 때까지 메모리에 저장된다. 이런 방식으로 Node.js는 애플리케이션을 효율적이고 빠르게 실행시킨다.

Node.js는 JavaScript 코드를 한 번만 읽어들이며 이벤트에 의해 트리거링(triggering) 될 때만 함수들과 관련 콜백 함수를 실행한다.



Node.js란 “비동기 이벤트 구동 JS 런타임”

Node.js는 여러분의 작성한 JS 코드를 읽고 해석한다.

비동기

마지막으로 비동기가 왜 중요한 개념일까? JavaScript는 본래 비동기이며, 이는 태스크들이 반드시 순차적으로 실행되지 않음을 의미한다.

비동기 실행 환경은 웹 애플리케이션에 적합하다. 웹사이트를 방문할 때마다 요청한 페이지를 읽어들이는 평균 시간을 생각해 보라. Coupang.com에서 주문을 넣고 주문이 진행되는 동안 다른 방문자들은 주문 페이지에 접속할 수 없다고 가정해 보자. 이 시스템의 웹사이트는 단일 애플리케이션 프로세스 또는 스레드로 구성돼 있을 것이다.

Node.js는 하나의 실행 스레드만 사용한다. 결과적으로 Node.js 애플리케이션은 컴퓨터 리소스가 요청받은 태스크별로 할당될 필요가 없다.



Node.js란 “비동기 이벤트 구동 JS 런타임”

Node.js는 여러분의 작성한 JS 코드를 읽고 해석한다.

비동기

Coupang.com 예시에서 Node.js는 여러분의 주문 요청 처리 뒤 다른 사용자들의 웹 페이지 요청 처리를 위해 메인 스레드를 이용할 것이다. 여러분의 주문 요청이 진행되면 이벤트가 발생하고 메인 스레드를 깨워 주문이 성공적으로 접수됐음을 알려줄 것이다. 다시 말하면 Node.js는 태스크 수행에 비동기를 채택하고 있어 첫 번째 태스크가 종료되기 전에 다른 태스크의 작업을 이어 갈 것이다. 시작과 종료 동작 사이에 대기하는 대신 Node.js는 전달된 작업이 완료될 때 호출되는 **이벤트 리스너**를 등록한다.

02

Node.js의 이해

Node.js는 JavaScript 코드를 해석하고
애플리케이션을 실행하기 위한 플랫폼이다.

p. 39-44

클라이언트 사이드 대 서버 사이드

일반적으로 웹 개발은 크게 두 가지로 나뉜다.



클라이언트 사이드

사용자가 자신의 웹 브라우저에서 볼 수 있는 결과를 작성하는 코드를 실행해 보여준다. 클라이언트 사이드 코드는 일반적으로 웹 페이지가 로드될 때 사용자 경험을 애니메이션화하는 데 사용되는 JavaScript를 포함한다.

서버 사이드

애플리케이션 로직(데이터가 구조화되고 데이터베이스에 저장되는 방식)에 사용되는 코드를 실행한다 서버 측 코드는 로그인 페이지에서 사용자를 인증하고 클라이언트 사이드 코드가 사용자까지의 도달을 보장하는 역할도 한다.

클라이언트 사이드 대 서버 사이드

일반적으로 웹 개발은 크게 두 가지로 나뉜다.

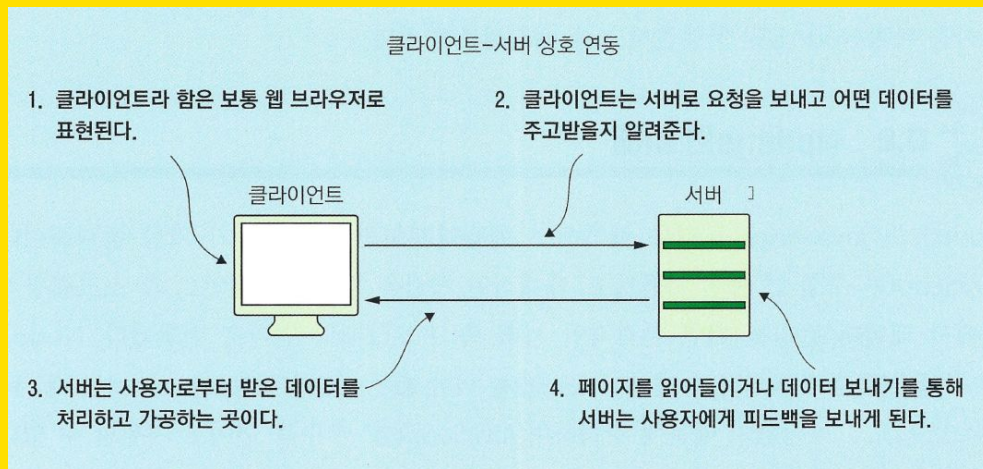


그림 설명

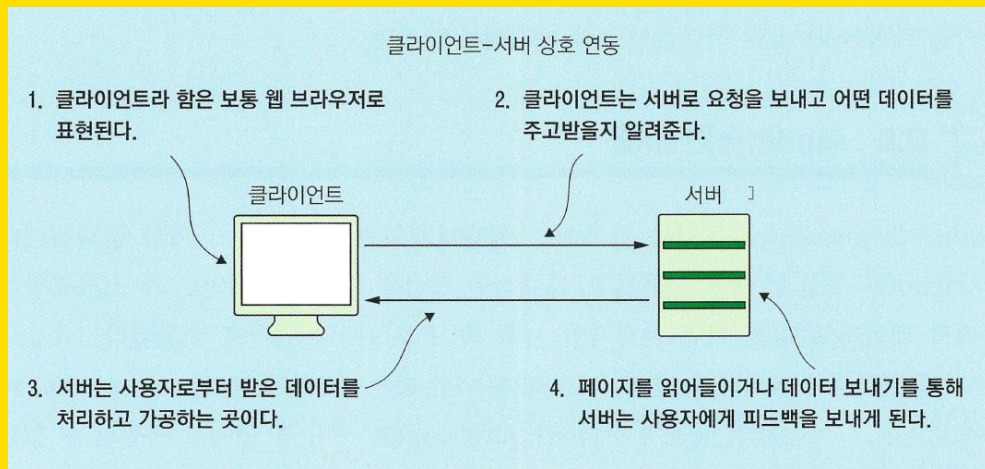
사용자가 애플리케이션을 볼 수 있는 브라우저로 표현된다 서버는 애플리케이션이 실행되고 사용자가 제출한 모든 데이터를 처리한다 또한 서버는 종종 클라이언트의 필요에 의해 사용자 인터페이스를 렌더링하기도 한다.

풀(전체) 스택 개발

이제는 JavaScript가 두 가지 개발 유형 모두에 사용되기 때문에 이 두 세계를 구분하는 선은 사라지고 있다. JavaScript를 사용해 전체 스택을 개발하면 JavaScript가 서버 및 클라이언트뿐만 아니라 이전에는 없었던 장치, 하드웨어 및 아키텍처에서도 사용되는 풀 스택 개발이라는 새로운 개념이 정의된다.

클라이언트 사이드 대 서버 사이드

일반적으로 웹 개발은 크게 두 가지로 나뉜다.



개발자들은 이제 웹 애플리케이션을 빌드할 때 동일한 작업을 위한 다른 마스터 랭귀지 대신 **JavaScript**를 마스터로 커밋이 가능하다.

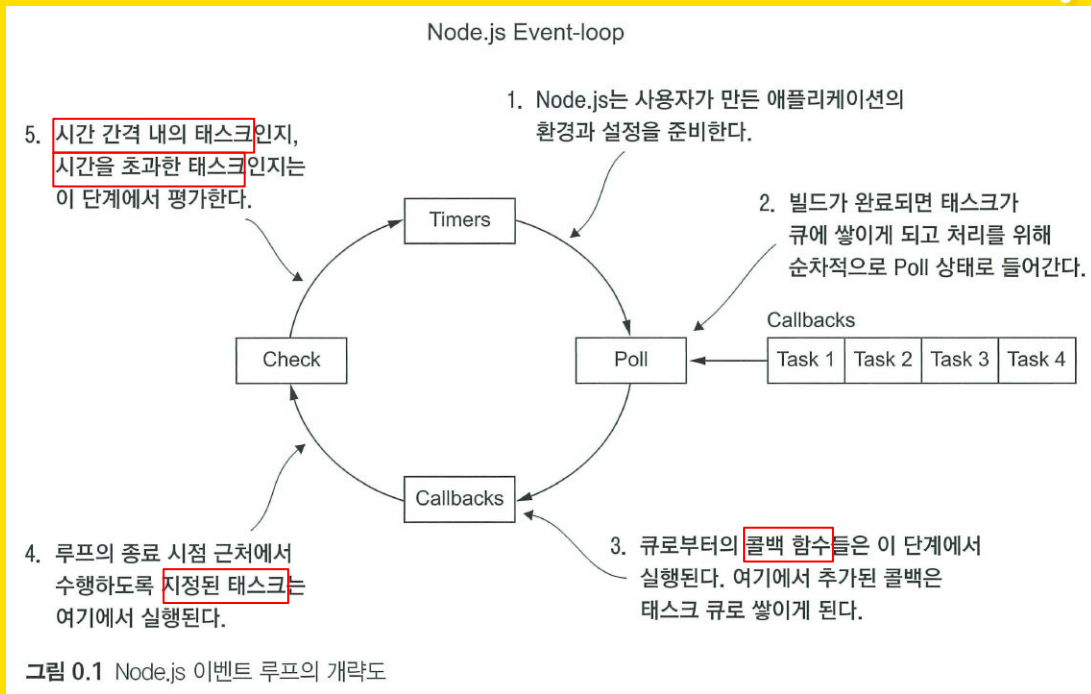
Node.js의 이벤트 루프

Node.js가 어떻게 함수를 실행하는가?

Node.js는 단일 스레드를 사용해 이벤트 루프에서 작동된다. 스레드는 프로그래밍된 작업을 실행하는 데 필요한 컴퓨팅 성능과 리소스의 묶음이다. 대부분의 애플리케이션에서 계산 집약적인 작업이 필요하지 않다면 오히려 단일 스레드가 모든 작업을 신속하게 관리하고 실행할 수 있다.

비동기식 규칙을 사용해 코드를 작성하고 Node.js 아키텍처는 배후에서 작업 처리를 예약한다. 따라서 Node.js는 데이터 처리를 지속적으로 수행하는 실시간 애플리케이션을 만드는 데 널리 사용된다.

실제로 Node.js는 좀 더 큰 작업을 호스트 컴퓨터에 전달하도록 설계됐으며 컴퓨터는 이러한 작업을 수행하기 위해 새로운 스레드와 프로세스를 만들 수 있다.

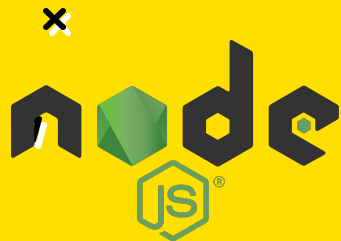




왜 **Node.js**로 개발하는가?

Node.js를 배우는 까닭은 다음과 같다.

- 1.** 애플리케이션 유지를 위해 여러 언어를 배우는 대신 **JavaScript** 하나로 개발의 핵심 언어로 사용할 수 있다.
- 2.** 데이터를 연속적으로 스트리밍하거나 채팅 기능의 구현 시, **Node.js**가 다른 플랫폼보다 더 나은 성능을 보였다.
- 3.** **Node.js**는 구글의 **V8 JavaScript** 인터프리터에 의해 지원된다. 즉, 성능이나 기능 면에서 널리 지원되고 성장할 것으로 예상되며, 바로 없어지는 일은 없을 것이다.
- 4.** **Node.js**는 웹 개발 커뮤니티에서 많이 사용되는 언어다. 적어도 앞으로 **5년** 동안 **Node.js**로 개발한 다른 개발자를 만나고 지원을 받을 수 있다. 또한 **Node.js**에 대한 지원이 더 많은 오픈소스 도구가 현재 구축 중이다.
- 5.** **JavaScript** 기술 경험이 있는 개발자는 훨씬 유리하게 작업할 수 있다. **Node.js**를 알면 프론트엔드 또는 백엔드 어디에나 적용할 수 있다.



03

추가 주제

REPL과 npm에 대해 설명합니다.

JS 실행 환경 REPL

첫 번째로 Node.js는 REPL(Read, Eval, Print, Loop)을 통해서 런타임을 제공한다.

1. **Read**: 유저의 입력 값을 받아서 메모리에 저장
2. **Eval**: 입력 값의 평가, 실행
3. **Print**: Eval로 인해 반환된 값을 출력
4. **Loop**: 1~3을 반복.

```
1. node (node)
→ ~ node
> const a = 2;
undefined
> a
2
> a + 4
6
> console.log(a);
2
undefined
> █
```

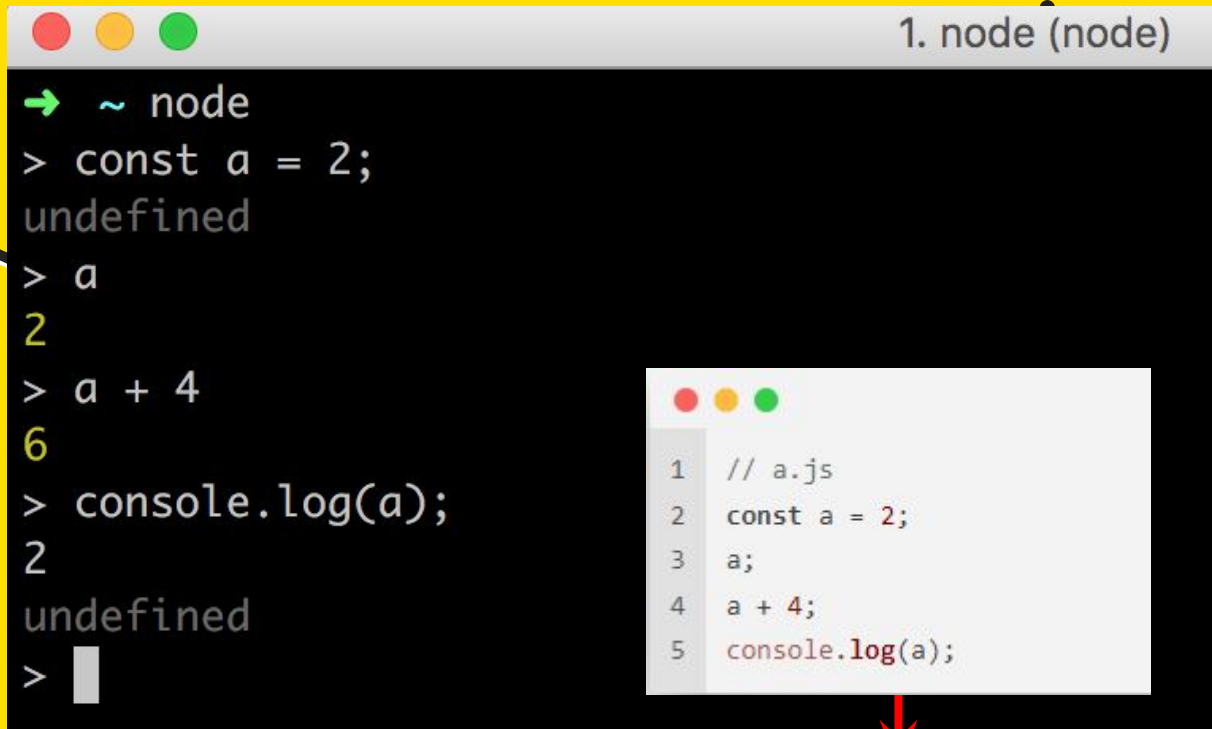
JS 실행 환경 REPL

첫 번째로 Node.js는 REPL(Read, Eval, Print, Loop)을 통해서 런타임을 제공한다.

1. **Read**: 유저의 입력 값을 받아서 메모리에 저장
2. **Eval**: 입력 값의 평가, 실행
3. **Print**: Eval로 인해 반환된 값을 출력
4. **Loop**: 1~3을 반복.

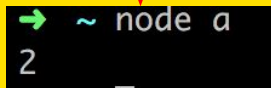
두 번째로 따로 자바스크립트 파일을 Node.js에서 제공하는 자바스크립트 런타임을 통해 실행이 가능하다. 위와 같이 REPL에서 개발을 하면 간단한 테스트 정도면 몰라도 어플리케이션을 개발하는데 적합하지 않다. 따라서 repl 보다는 따로 스크립트 파일을 만들어서 그 스크립트 파일을 node.js, V8이 해석 후 실행하는 형태로 작업을 많이 하게 된다.

오른쪽에 있는 소스 코드를 node.js에서 돌리는 것은 위의 REPL에서 실행한 것과 동일하다.



```
1. node (node)
→ ~ node
> const a = 2;
undefined
> a
2
> a + 4
6
> console.log(a);
2
undefined
> |
```

```
1 // a.js
2 const a = 2;
3 a;
4 a + 4;
5 console.log(a);
```



```
→ ~ node a
2
```


NPM이란?

Node.js를 설치하면 **npm**이 자동으로 함께 설치됩니다.

npm은 Node.js의 패키지 관리 시스템이다 (Python의 pip와 같다). 전 세계 모든 오픈 소스 라이브러리 중 가장 큰 패키지 관리 시스템이며 무료로 사용할 수 있다.

npm은 즉시 사용 가능한 명령줄 유틸리티와 함께 제공된다. npm 웹 사이트로 이동하여 필요한 패키지를 검색하고 단일 명령을 사용하여 설치할 수 있다.

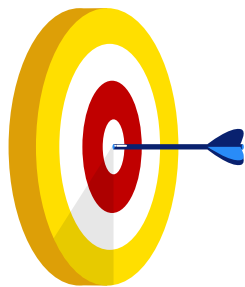
또한 이 명령줄 유틸리티를 통해 패키지 버전을 관리하고, 종속성을 검토하고, 프로젝트에서 사용자 지정 스크립트를 설정할 수 있다.

```
npm install <package-name>
```

```
npm install <pkg-1> <pkg-2> <pkg-3>
```

이 수업에서는 가장 인기 있는 Node.js 패키지를 사용할 것이다.

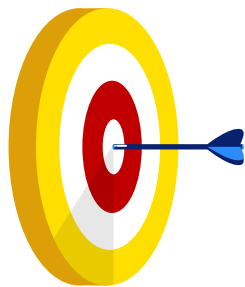
- Express
- MongoDB
- Socket.io
- Async
- Mocha
- Passport



퀵 체크

Node.js는 프로그래밍 언어인가?
프레임워크인가?

아니요, 프로그래밍 언어나 프레임워크가 아닙니다.
오히려 브라우저 외부에서 JavaScript를 실행하는
데 사용되는 런타임 환경입니다.



퀵 체크

참 또는 거짓! Node.js 이벤트 루프는 각 태스크를 다음 태스크를 처리하기 전에 모두 완료 처리한다.

거짓이다. Node.js 이벤트 루프는 큐에서 태스크를 순차적으로 제거하지만 애플리케이션이 실행 중인 시스템에서 처리할 태스크를 중지시키거나 새 태스크를 처리하는 동안 특정 태스크가 완료될 때까지 대기 상태로 둘 수 있다.

The background features a collection of scattered, semi-transparent geometric shapes in shades of grey and yellow. These shapes include triangles, squares, circles, and lines, some of which are slightly offset or layered, creating a modern, abstract aesthetic.

Thanks !