

Node.js란 무엇인가?

Node.js에 대한 포괄적인 소개와 이해

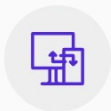
2020875029 소프트웨어학과 신성우

Node.js란 무엇인가?

비동기 I/O 처리 및 단일 스레드 모델을 통한 크로스 플랫폼 지원

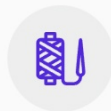


비동기 I/O 처리



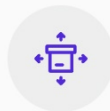
Node.js는 비동기 I/O 처리 방식을 사용하여 높은 성능과 확장성을 제공합니다.

단일 스레드 모델



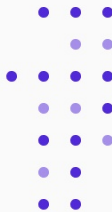
단일 스레드 모델을 통해 Node.js는 이벤트 기반 아키텍처로 동시성을 처리합니다.

크로스 플랫폼 지원



Node.js는 다양한 운영체제를 지원하여 개발자에게 유연한 환경을 제공합니다.

Node.js의 역사

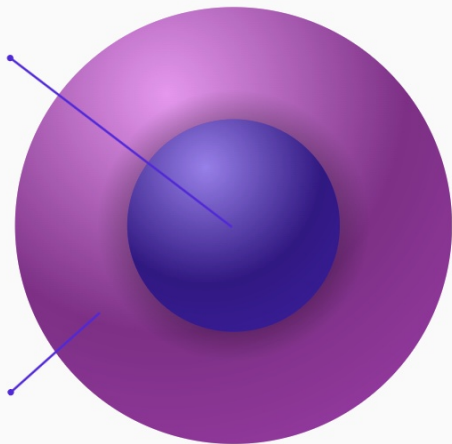


Node.js 소개

Node.js는 2009년 Ryan Dahl에 의해 발표되었으며, 초기 웹 서버 개발을 목표로 했습니다.

다양한 애플리케이션 사용

현재 Node.js는 다양한 애플리케이션에서 활용되고 있습니다.



Node.js의 설치 방법

Node.js 공식 웹사이트 방문

Node.js의 공식 웹사이트를 방문하여 최신 정보를 확인합니다.

버전 확인

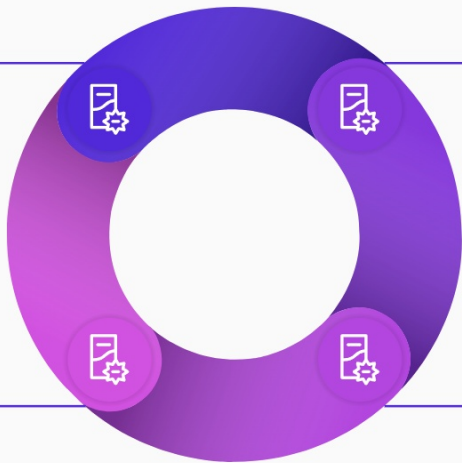
설치 후 `node -v` 명령어를 사용해 설치된 Node.js 버전을 확인합니다.

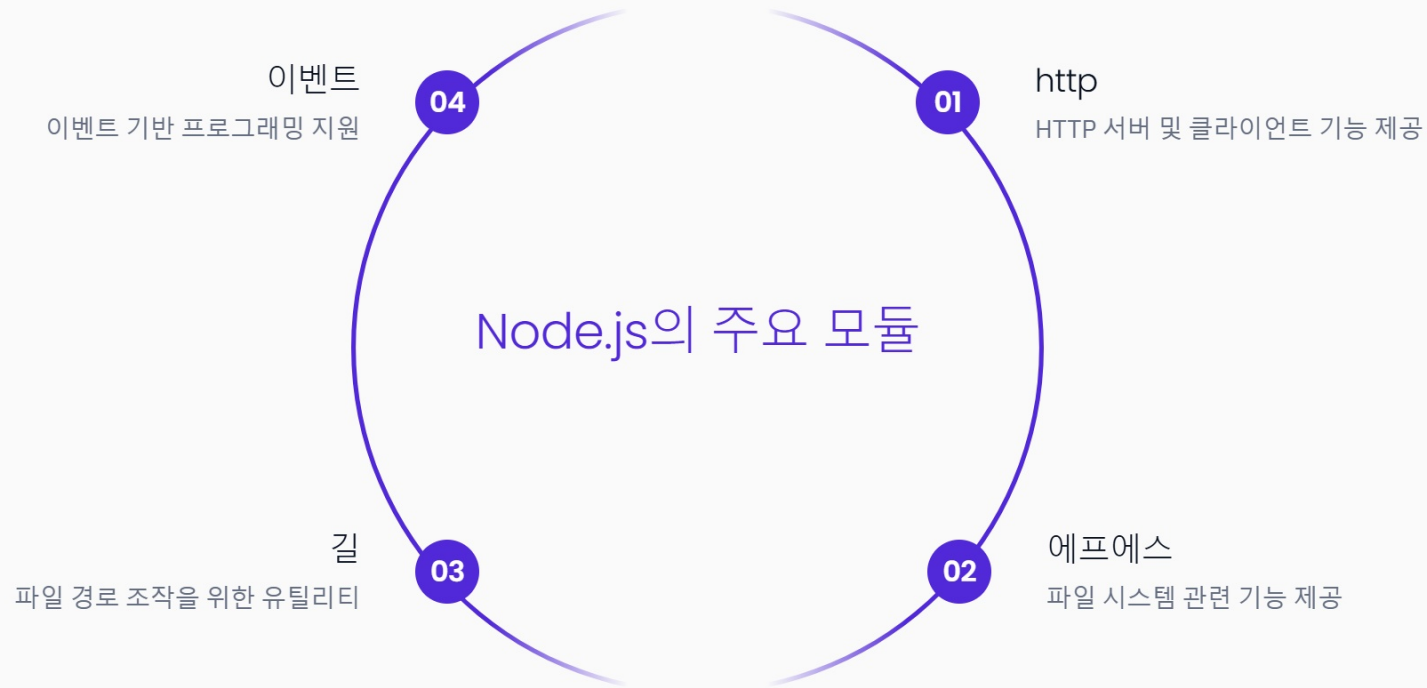
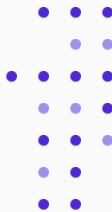
운영 체제에 맞는 설치 파일 다운로드

사용 중인 운영 체제에 적합한 설치 파일을 선택하여 다운로드합니다.

설치 마법사에 따라 설치 진행

설치 마법사의 안내에 따라 Node.js를 설치합니다.





비동기 프로그래밍의 이해

Node.js의 비동기 프로그래밍 모델

Node.js는 비동기 프로그래밍 모델을 통해 효율적인 I/O 처리를 가능하게 합니다.

비동기 함수의 예

비동기 함수는 코드가 파일을 읽는 동안 다른 작업을 수행할 수 있게 해줍니다.

npm과 패키지 관리

Node.js 환경에서 패키지 관리의 중요성



npm은 Node.js의 패키지 관리자입니다.

Node.js 생태계에서 필수적인 도구로, 다양한 패키지를 쉽게 관리할 수 있게 합니다.



외부 패키지를 쉽게 설치하고 관리할 수 있습니다.

npm을 통해 필요한 라이브러리를 손쉽게 추가하고, 프로젝트의 의존성을 관리할 수 있습니다.



기본 명령어:

npm을 사용하기 위한 기본적인 명령어를 이해하는 것이 중요합니다.



`npm init`: 새 패키지 생성

새로운 Node.js 패키지를 생성하기 위한 초기 설정을 도와줍니다.



`npm install`: 패키지 설치

특정 패키지를 설치하여 프로젝트에 통합하는 명령어입니다.



`npm update`: 패키지 업데이트

설치된 패키지를 최신 버전으로 업데이트하는데 사용됩니다.

Express.js 프레임 워크 소개

01

미들웨어 지원

Express.js는 다양한 미들웨어를 지원하여 요청과 응답을 처리하는 유연성을 제공합니다.

02

RESTful API 구축에 적합

Express.js는 RESTful API를 쉽게 구축할 수 있는 강력한 기능을 제공합니다.

03

다양한 템플릿 엔진과의 호환성

Express.js는 여러 템플릿 엔진과 호환되어 다양한 프론트엔드 요구 사항을 충족할 수 있습니다.

Node.js의 성능 최적화

클러스터링 활용

CPU 코어를 최대한 활용하여 성능을 향상시킵니다.



메모리 관리

메모리 누수를 방지하고 최적화하여 안정성을 확보합니다.

비동기 작업의 최적화

불필요한 블로킹을 방지하여 작업 효율성을 높입니다.



Node.js의 실제 사용 사례



Netflix: 대규모 비디오 스트리밍 서비스 운영

Netflix는 Node.js를 사용하여 수백만 명의 사용자를 위한 대규모 스트리밍 서비스를 효율적으로 운영합니다.



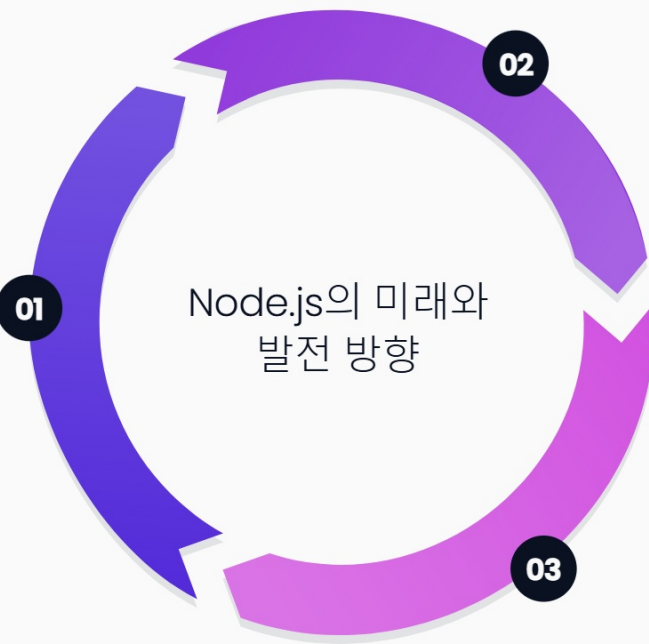
PayPal: 실시간 거래 처리 시스템 구축

PayPal은 Node.js로 실시간 거래 처리 시스템을 구축하여 신속한 결제를 지원합니다.



LinkedIn: 모바일 애플리케이션의 서버 측 개발

LinkedIn은 모바일 애플리케이션의 서버 측에서 Node.js를 활용하여 빠른 응답 속도를 제공합니다.



Node.js의 미래와 발전 방향

01 WebAssembly와의 통합

WebAssembly와의 통합을 통해 Node.js의 성능을 극대화하고 다양한 플랫폼과의 호환성을 개선합니다.

02

서버리스 아키텍처 지원 강화

서버리스 환경에서의 Node.js 사용을 최적화하여 더 나은 확장성과 비용 효율성을 제공합니다.

03

더 나은 성능과 확장성을 위한 업데이트 지속

Node.js는 성능 최적화 및 확장성을 높이기 위한 지속적인 업데이트를 계획하고 있습니다.

Node.js의 세계에 발 을 들여보세요

지금 Node.js를 배우고 웹 개발의 미래를 경험하세요!

