

# Node.js

## 새로운 웹 개발의 혁명

Node.js는 서버에서 자바스크립트를 실행할 수 있게 해주는 혁신적인 런타임 환경입니다. 2009년에 탄생한 이 기술은 웹 개발의 판도를 바꾸었습니다.

작성자: 최유진





---

## 1. Node.js 정의

---

## 2. Node.js 핵심 특징

---

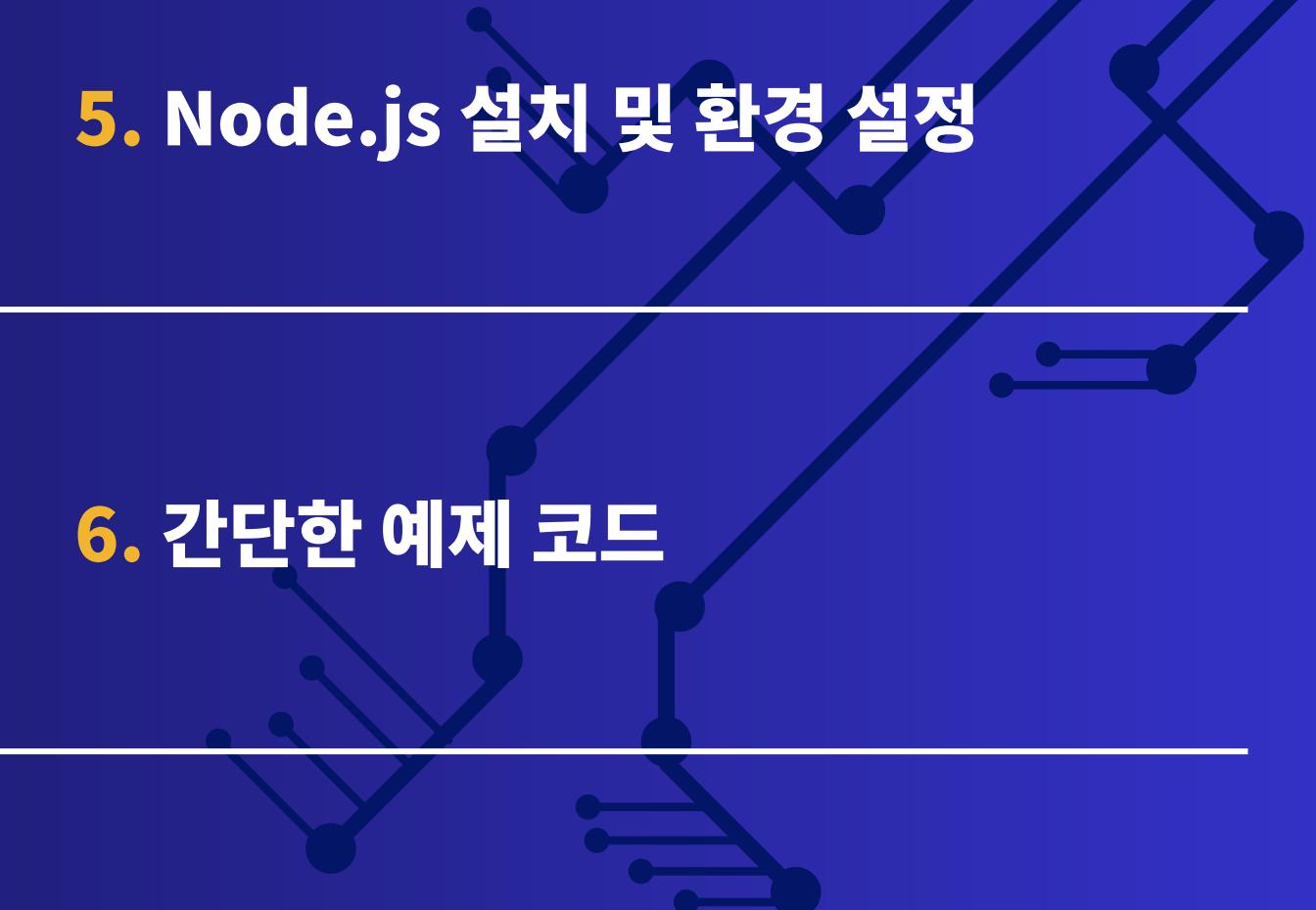
## 3. Node.js 장단점

---

## 4. Node.js 주요 모듈

---

## 5. Node.js 설치 및 환경 설정



---

## 6. 간단한 예제 코드

---

## 7. Node.js의 활용 분야

---

# Node.js 정의

## Node.js 란?

서버 측에서 JavaScript를 실행할 수 있는 런타임 환경입니다.  
V8 JavaScript 엔진을 사용해 빠른 코드 실행을 제공합니다.

## 역사 및 발전

2009년 Ryan Dahl에 의해 처음 발표 되었습니다.  
비동기 I/O 모델과 이벤트 루프로 혁신적인 서버 개발을 가능하게 하였고,  
2015년 io.js와 통합해 Node.js가 설립되었습니다.

## 목적과 비전

서버 측 JavaScript의 환경을 통해 개발자의 생산성을 높이고,  
다양한 플랫폼에서 JavaScript 개발이 쉽게하는 것이 목표입니다.  
또, 커뮤니티를 활성화해 발전과 성장을 도모합니다.

# Node.js 핵심 특징

## 비동기 I/O

이벤트 기반 모델을 사용하여 빠른 작업 처리를 가능하게 합니다. 블로킹 없이 다중 요청을 처리할 수 있습니다.

## 싱글 스레드

단일 스레드로 서버 자원을 효율적으로 사용합니다. 복잡한 멀티스레딩 문제를 피할 수 있습니다.

## 크로스 플랫폼

Windows, macOS, Linux 등 다양한 운영 체제에서 실행 가능합니다. 개발 환경의 유연성을 제공합니다.

# Node.js 장점

## 1 빠른 처리 속도

Google V8 엔진을 사용하여 자바스크립트를 고속으로 실행합니다. 대규모 트래픽도 원활하게 처리할 수 있습니다.

## 2 개발 생산성 향상

프론트엔드와 백엔드를 동일한 언어로 개발할 수 있습니다. 코드 재사용성이 높아져 개발 시간이 단축됩니다.

## 3 활발한 커뮤니티

다양한 오픈 소스 모듈과 라이브러리를 사용할 수 있습니다. npm을 통해 손쉽게 패키지를 관리할 수 있습니다.



# Node.js 단점과 극복 방법

## CPU 집약적 작업에 부적합

싱글 스레드 특성상 CPU 작업에 약점이 있습니다. 워커 스레드를 사용하여 이 문제를 해결할 수 있습니다.

## 오류 관리의 어려움

비동기 코드로 인해 디버깅이 복잡할 수 있습니다. 프로미스와 `async/await`를 활용하여 코드 가독성을 높일 수 있습니다.

# Node.js 주요 모듈



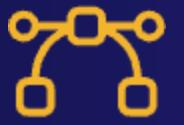
## http

웹 서버를 생성하고 관리합니다.  
RESTful API 구현에 필수적인  
모듈입니다.



## fs

파일 시스템을 조작합니다.  
파일 읽기, 쓰기, 삭제 등의 작업  
을 수행합니다.



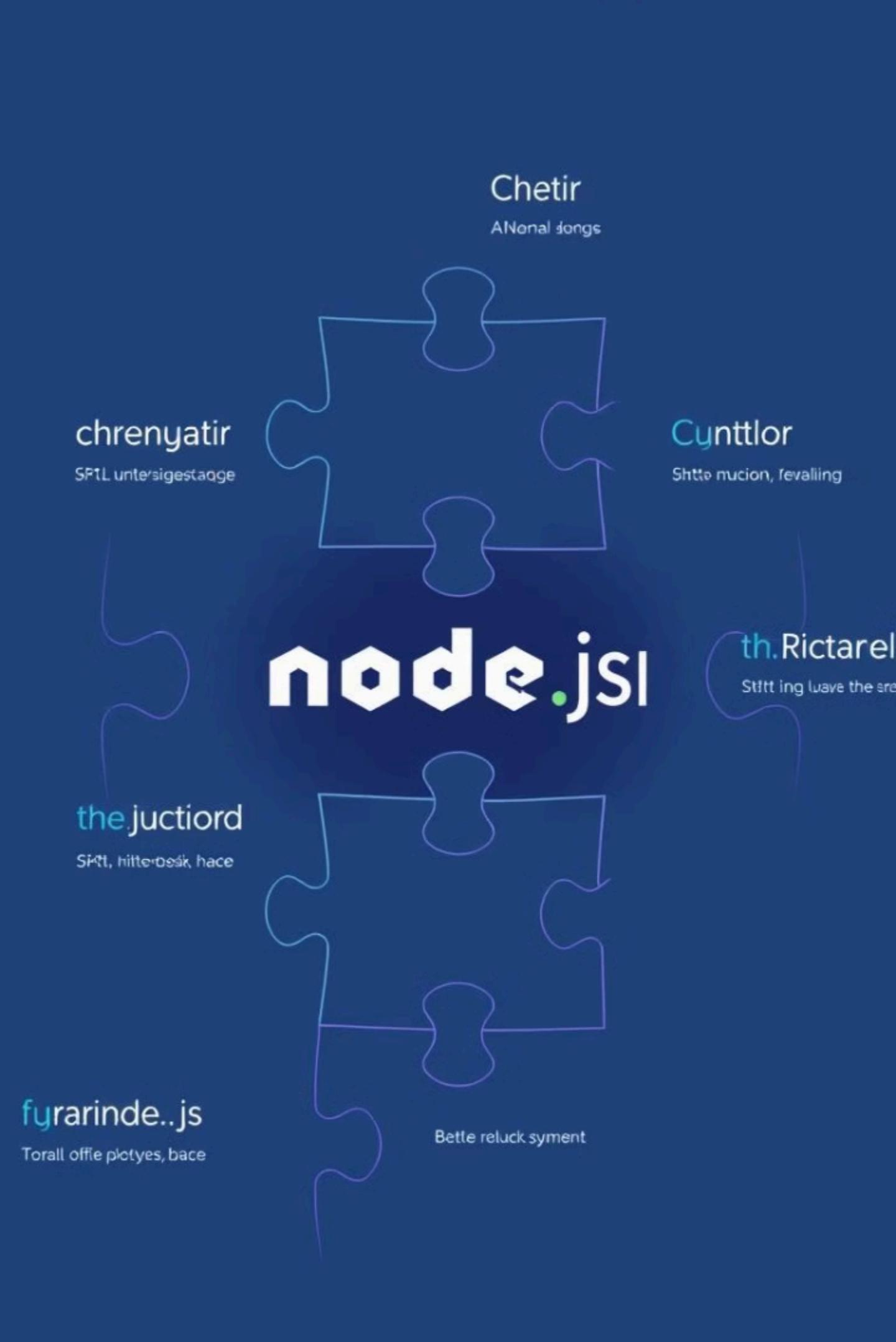
## path

파일 경로를 조작합니다.  
크로스 플랫폼 호환성을 유지하  
는 데 중요합니다.



## express

웹 애플리케이션과 API를 쉽게  
구축할 수 있는 프레임워크입니  
다.



# Node.js 설치 및 환경 설정

1

## 공식 웹사이트 방문

nodejs.org에서 운영 체제에 맞는 버전을 다운로드합니다.

2

## 설치 실행

다운로드한 설치 파일을 실행하고 지시에 따라 설치를 완료합니다.

3

## 설치 확인

터미널에서 'node -v' 명령어로 설치된 Node.js 버전을 확인합니다.

4

## npm 사용 준비

Node.js와 함께 설치된 npm을 사용하여 필요한 패키지를 관리합니다.

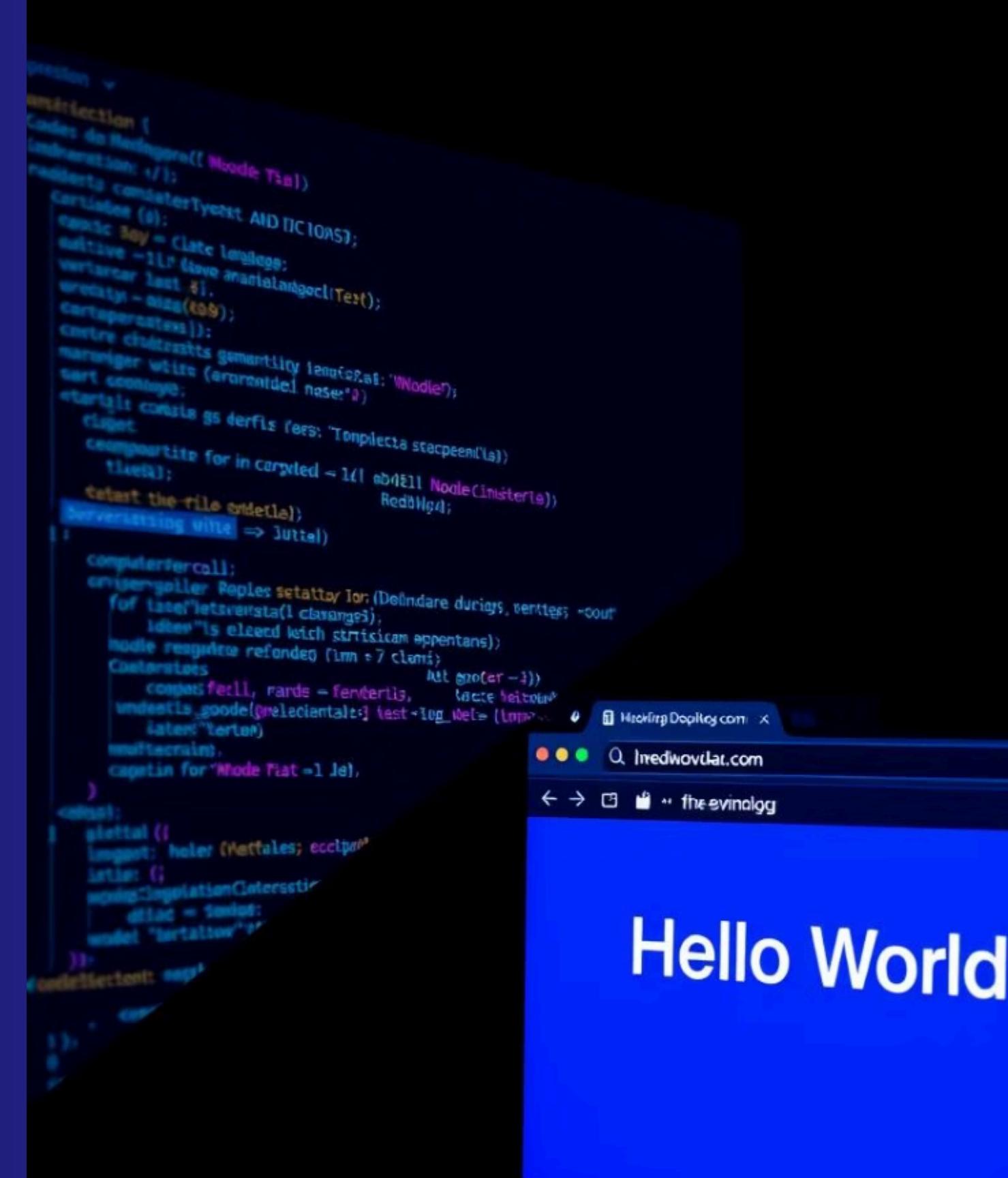


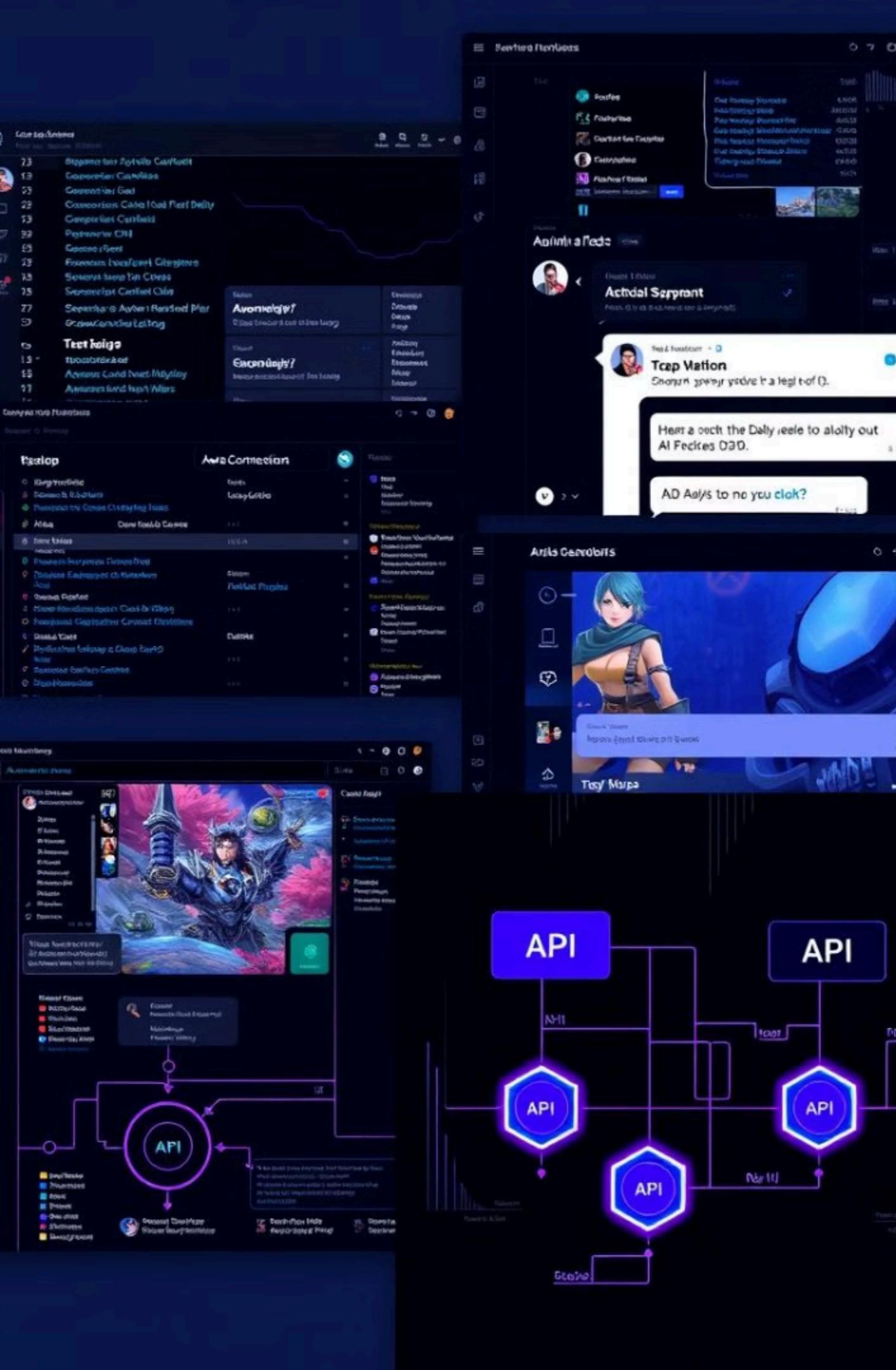
# 간단한 웹 서버 예제

javascript

```
const http = require('http');
const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World\n');
});
server.listen(3000, () => {
  console.log('서버가 http://localhost:3000에서 실행 중 입니다.');
});
```

이 코드는 간단한 HTTP 서버를 생성하여 "Hello World" 메시지를 반환합니다. 3000번 포트에서 실행됩니다.





# Node.js 다양한 활용 분야

## 웹 서버

높은 동시성과 빠른 응답 속도로 효율적인 웹 서버를 구축할 수 있습니다.

## API 서버

RESTful API를 쉽게 구현할 수 있어 마이크로서비스 아키텍처에 적합합니다.

## 실시간 애플리케이션

채팅, 온라인 게임 등 실시간 데이터 처리가 필요한 애플리케이션에 이상적입니다.

## 데이터 스트리밍

대용량 데이터 스트리밍 처리에 효과적으로 사용될 수 있습니다.