Node.js와 Express.js를 활용한 RESTful API 구축

Node.js와 Express.js를 사용하면 효과적이고 확장 가능한 RESTful API를 구축할 수 있습니다. 이 프레젠테이션에서는 CRUD 작업, 라우팅, 미들웨어 구현 등 API 개발의 핵심 요소를 살펴보겠습니다.

작성자: 현욱

CRUD 작업 소개

1 **Create**

새 데이터를 생성하는 작업

2 **Read**

기존 데이터를 조회하는 작업

3 **Update**

기존 데이터를 수정하는 작업

4 **Delete**

기존 데이터를 삭제하는 작업

CRUD 작업 구현 방법 I: Create, Read

Create

POST 요청을 처리하여 새 리소스를 생성합니다.

Read

GET 요청을 처리하여 기존 리소스를 조회합니다.

CRUD 작업 구현 방법 II: Update, Delete

Update

PUT 또는 PATCH 요청을 처리하여 기존 리소스를 수정합니다.

Delete

DELETE 요청을 처리하여 기존 리소스를 삭제합니다.

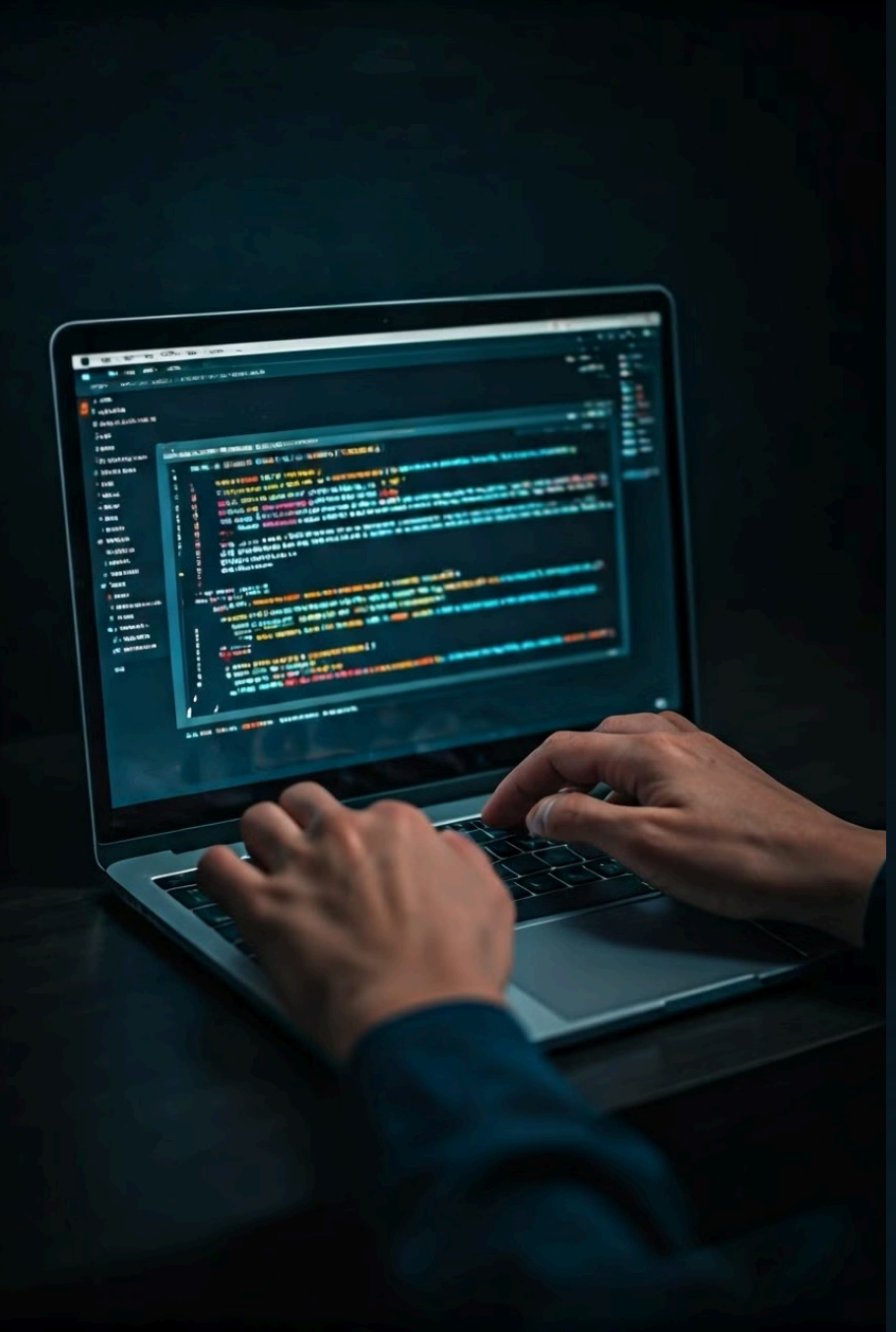
Express.js 라우팅 이해 및 적용

1 라우팅 기본 원리

HTTP 메서드와 URL 경로에 따라 적절한 핸들러 함수를 실행합니다.

2 라우팅 구현 예시

app.get('/users', handleGetUsers), app.post('/users', handleCreateUser) 등과 같이 구현할 수 있습니다.



미들웨어의 역할과 구현

기능

요청과 응답 처리 과정에 필요한 부가 기능을 제공합니다.

구현

app.use(middleware)와 같이 미들웨어 함수를 등록하여 사용합니다.

예시

로깅, 에러 핸들링, 인증 등을 처리하는 미들웨어를 구현할 수 있습니다.

코드 예시 I: CRUD 작업 구현

1

Create

app.post('/users', (req, res) => { /* 새 사용자 생성 */ });

2

Read

app.get('/users', (req, res) => { /* 사용자 목록 조회 */ });

3

Update

app.put('/users/:id', (req, res) => { /* 사용자 정보 수정 */ });

4

Delete

app.delete('/users/:id', (req, res) => { /* 사용자 삭제 */ });

코드 예시 II: 라우팅 및 미들웨어 구현

라우팅

app.use('/api', apiRouter);

미들웨어

app.use(bodyParser.json());

에러 핸들링

app.use((err, req, res, next) => { /* 에러 처리 */ });