# 당신의 API를

# 더 쉽고, 단단하게

# itdoc

# API 테스트도 작성해...
# 문서화도 해야해...

```js
const express = require('express');
const bodyParser = require('body-parser');

const app = express();
const PORT = process.env.PORT || 3000;

app.use(bodyParser.json());

// 메모리 기반 데이터
let todos = [];
let nextId = 1;

// 1) 전체 조회
app.get('/api/todos', (req, res) => {
  res.json(todos);
});
```

API문서를 정의하고
API를 만듭니다.

```js
const request = require('supertest');
const { expect } = require('chai');
const app = require('../index');

describe('Todos API', () => {
  let createdId;

  it('GET /api/todos should return empty array', async () => {
    const res = await request(app).get('/api/todos');
    expect(res.status).to.equal(200);
    expect(res.body).to.be.an('array').that.is.empty;
  });

  it('POST /api/todos should create a todo', async () => {
    const res = await request(app)
      .post('/api/todos')
      .send({ title: 'Test todo' });
```

API문서에 따라 테스팅을 구축합니다.

**1. GET** `/api/todos`

- **Description**: Retrieve the full list of todos.
- **Response**:
  - **Status**: `200 OK`
  - **Body**: Array of todo objects.

```
[]
```

**2. POST** `/api/todos`

- **Description**: Create a new todo item.
- **Request**:
  - **Headers**: `Content-Type: application/json`
  - **Body**:

```
{
  "title": "string",
  "completed": false  // optional, defaults to false
}
```

테스트에 맞춰 문서를 작성합니다.

# 다시 API 테스트도 작성해...
# 문서화도 해야해...

```js
const express = require('express');
const bodyParser = require('body-parser');

const app = express();
const PORT = process.env.PORT || 3000;

app.use(bodyParser.json());

// 메모리 기반 데이터
let todos = [];
let nextId = 1;

// 1) 전체 조회
app.get('/api/todos', (req, res) => {
  res.json(todos);
});
```

API문서를 정의하고
API를 만듭니다.

```js
const request = require('supertest');
const { expect } = require('chai');
const app = require('../index');

describe('Todos API', () => {
  let createdId;

  it('GET /api/todos should return empty array', async () => {
    const res = await request(app).get('/api/todos');
    expect(res.status).to.equal(200);
    expect(res.body).to.be.an('array').that.is.empty;
  });

  it('POST /api/todos should create a todo', async () => {
    const res = await request(app)
      .post('/api/todos')
      .send({ title: 'Test todo' });
```

API문서에 따라 테스팅을 구축합니다.

**1. GET** `/api/todos`

- **Description**: Retrieve the full list of todos.
- **Response**:
  - **Status**: `200 OK`
  - **Body**: Array of todo objects.

```
[]
```

**2. POST** `/api/todos`

- **Description**: Create a new todo item.
- **Request**:
  - **Headers**: `Content-Type: application/json`
  - **Body**:

```
{
  "title": "string",
  "completed": false  // optional, defaults to false
}
```

테스트에 맞춰 문서를 작성합니다.

# 다시 API 테스트도 작성해...
# 문서화도 해야해...

```js
const express = require('express');
const bodyParser = require('body-parser');

const app = express();
const PORT = process.env.PORT || 3000;

app.use(bodyParser.json());

// 메모리 기반 데이터
let todos = [];
let nextId = 1;

// 1) 전체 조회
app.get('/api/todos', (req, res) => {
  res.json(todos);
});
```

API문서를 정의하고
API를 만듭니다.

```js
const request = require('supertest');
const { expect } = require('chai');
const app = require('../index');

describe('Todos API', () => {
  let createdId;

  it('GET /api/todos should return empty array', async () => {
    const res = await request(app).get('/api/todos');
    expect(res.status).to.equal(200);
    expect(res.body).to.be.an('array').that.is.empty;
  });

  it('POST /api/todos should create a todo', async () => {
    const res = await request(app)
      .post('/api/todos')
      .send({ title: 'Test todo' });
```

API문서에 따라 테스팅을 구축합니다.

**1. GET** `/api/todos`

- **Description**: Retrieve the full list of todos.
- **Response**:
  - **Status**: `200 OK`
  - **Body**: Array of todo objects.

```
[]
```

**2. POST** `/api/todos`

- **Description**: Create a new todo item.
- **Request**:
  - **Headers**: `Content-Type: application/json`
  - **Body**:

```
{
  "title": "string",
  "completed": false  // optional, defaults to false
}
```

테스트에 맞춰 문서를 작성합니다.

# 다시 API 테스트도 작성해...
# 문서화도 해야해...

```js
const express = require('express');
const bodyParser = require('body-parser');

const app = express();
const PORT = process.env.PORT || 3000;

app.use(bodyParser.json());

// 메모리 기반 데이터
let todos = [];
let nextId = 1;

// 1) 전체 조회
app.get('/api/todos', (req, res) => {
  res.json(todos);
});
```

API문서를 정의하고
API를 만듭니다.

```js
const request = require('supertest');
const { expect } = require('chai');
const app = require('../index');

describe('Todos API', () => {
  let createdId;

  it('GET /api/todos should return empty array', async () => {
    const res = await request(app).get('/api/todos');
    expect(res.status).to.equal(200);
    expect(res.body).to.be.an('array').that.is.empty;
  });

  it('POST /api/todos should create a todo', async () => {
    const res = await request(app)
      .post('/api/todos')
      .send({ title: 'Test todo' });
```

API문서에 따라 테스팅을 구축합니다.

**1. GET** `/api/todos`

- **Description**: Retrieve the full list of todos.
- **Response**:
  - **Status**: `200 OK`
  - **Body**: Array of todo objects.

```
[]
```

**2. POST** `/api/todos`

- **Description**: Create a new todo item.
- **Request**:
  - **Headers**: `Content-Type: application/json`
  - **Body**:

```
{
  "title": "string",
  "completed": false  // optional, defaults to false
}
```

테스트에 맞춰 문서를 작성합니다.

# 테스트와 문서의 차이가 발생하기도

```
const request = require('supertest');
const { expect } = require('chai');
const app = require('../index');

describe('Todos API', () => {
  let createdId;

  it('GET /api/todos should return empty array', async () => {
    const res = await request(app).get('/api/todos');
    expect(res.status).to.equal(200);
    expect(res.body).to.be.an('array').that.is.empty;
  });

  it('POST /api/todos should create a todo', async () => {
    const res = await request(app)
      .post('/api/todos')
      .send({ title: 'Test todo' });
```

**1. GET** `/api/todos`

- **Description**: Retrieve the full list of todos.
- **Response**:
  - **Status**: `200 OK`
  - **Body**: Array of todo objects.

```
[]
```

**2. POST** `/api/todos`

- **Description**: Create a new todo item.
- **Request**:
  - **Headers**: `Content-Type: application/json`
  - **Body**:

```
{
  "title": "string",
  "completed": false  // optional, defaults to false
}
```

# itdoc이 있다면?

```js
const express = require('express');
const bodyParser = require('body-parser');

const app = express();
const PORT = process.env.PORT || 3000;

app.use(bodyParser.json());

// 메모리 기반 데이터
let todos = [];
let nextId = 1;

// 1) 전체 조회
app.get('/api/todos', (req, res) => {
  res.json(todos);
});
```

API문서를 정의하고

API를 만듭니다.

```js
const targetApp = require('../index');
const { describeAPI, itDoc, HttpMethod, HttpStatus, field } = require('itdoc');

// GET /api/todos
describeAPI(
  HttpMethod.GET,
  '/api/todos',
  {
    name: 'Get Todos',
    tag: 'Todos',
    summary: 'Retrieve the full list of todos.',
  },
  targetApp,
  (apiDoc) => {
    itDoc('Empty list when no todos exist', () => {
      return apiDoc
        .test()
        .req()
        .get()
        .res()
        .status(HttpStatus.OK)
        .body(field('Todos list', []));
    });
  }
);
```

API문서에 따라 itdoc 테스팅을 구축합니다.


테스팅에 맞춰

OAS, markdown, redocly html 까지

전부 알아서 생성됩니다!

# 테스트,문서 작성해야하는 번거로움
# 테스트와 문서의 불일치

# 쉽게 신뢰성있는 API구축

# one more thing.

# 만약 테스트도 구축하기 귀찮다면?

# itdoc llm

# itdoc generate -a ./app.js

AST파서가 앱분석 -> 비즈니스로직 제외, res, req만을 추출
-> GPT API llm -> itdoc test 생성

# itdoc generate -a ./app.js

딸깍 한번으로 테스트 및 문서화가 자동화

# itdoc

자바스크립트 API 테스트 프레임워크

---

## API테스트를 쉽게 구축하세요.

mocha, jest 테스트를 각각 구축하기도 힘든세상,
itdoc의 쉬운 인터페이스를 사용하세요!

## 테스트 만들면 문서가 공짜!

테스트 만들면 그 테스트에 따라
문서가 자동으로 만들어집니다.
문서는 oas, 마크다운, html(redocly)
기반으로 아주 멋있게 만들어집니다!

## 타입스크립트, 자바스크립트 모두 지원

타입스크립트, 자바스크립트로 하는 테스팅 모두
지원합니다
지금 당장 itdoc을 사용하세요!

# 당신의 API를
# 더욱 신뢰성있게.
# itdoc 

https://github.com/do-pa/itdoc
https://itdoc.kr/