

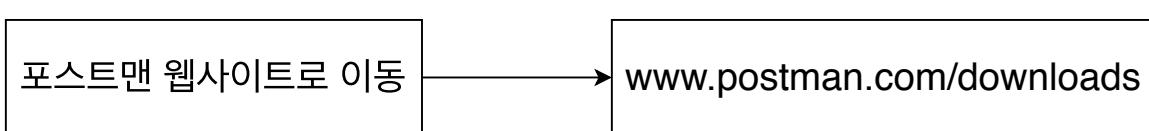
포스트맨 설치하기

포스트맨 이란?

Postman은 개발한 API를 테스트하고, 테스트 결과를 공유하여 API 개발의 생산성을 높여주는 플랫폼입니다.

현재 API에 요청을 전달하는 클라이언트가 없기 때문에 포스트맨을 사용해서 요청을 임의로 전달해주겠습니다.

포스트맨 설치



포스트맨 사용 방법

The screenshot shows the Postman interface with the following details:

- Request 메소드:** POST
- Endpoint:** http://localhost:5000/api/products/
- Headers:** (9)
- Body:** (highlighted in green)
- Body Content:**

```
1 {
2   "name": "phone",
3   "description": "it is good",
4   "price": 10
5 }
```
- Body Type:** raw (highlighted)
- Content Type:** application/json (highlighted)

A large arrow points downwards from the Body content area to a box labeled "요청을 위한 Body".

임의로 데이터를 저장할 때 만나는 문제점

POST MAN을 이용해서 임의로 새로운 Product를 데이터베이스에 저장해주겠습니다.

CREATE을 위한 ROUTE 생성

```
const express = require('express');
const router = express.Router();
const productController = require("./controller/products");

router.post("/", productController.createProduct);

module.exports = router;
```

controllers/product.js

```
const productModel = require('../models/Product');

exports.createProduct = (req, res, next) => {
  const createdProduct = productModel.create(req.body);
  console.log('createdProduct', createdProduct)
  res.status(201).json(createdProduct);
};
```

Postman

POST ▾

http://localhost:5000/api/products/

```
1  {
2    "name": "phone",
3    "description": "it is good",
4    "price": 10
5 }
```

Promise { <pending> }... ???

Running on port 5000
MongoDb Connected...
createdProduct Promise { <pending> }

어학사전

pending 영어
미국/영국 [péndɪŋ]

① 현안의 ② 남아있는 ③ 계류중인 ④ 미결 [더보기](#)

async await

어학사전

pending 영어

미국/영국 [péndin]

① 현안의 ② 남아있는 ③ 계류중인 ④ 미결 [더보기](#)

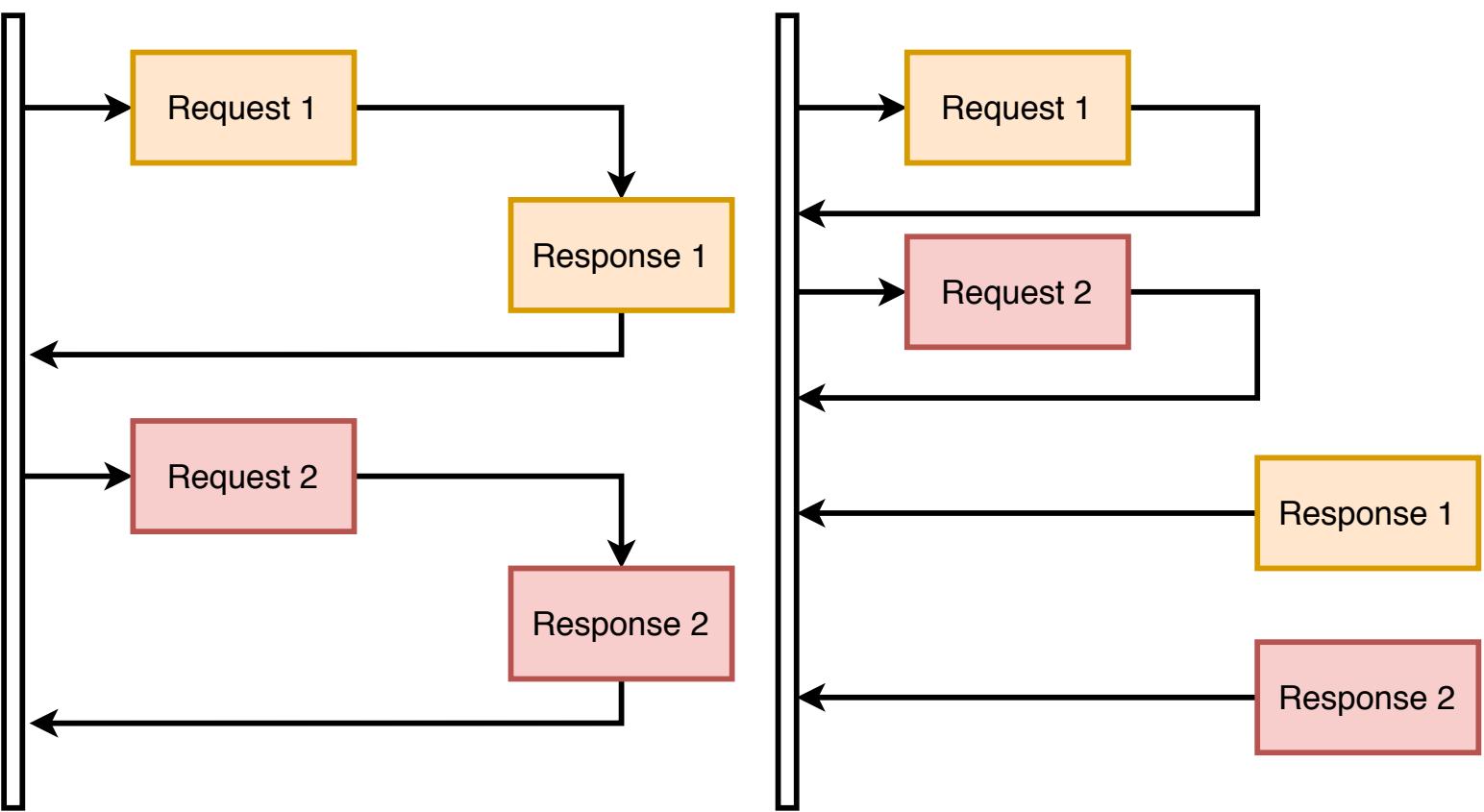
Running on port 5000
MongoDb Connected...
createdProduct Promise { <pending> }

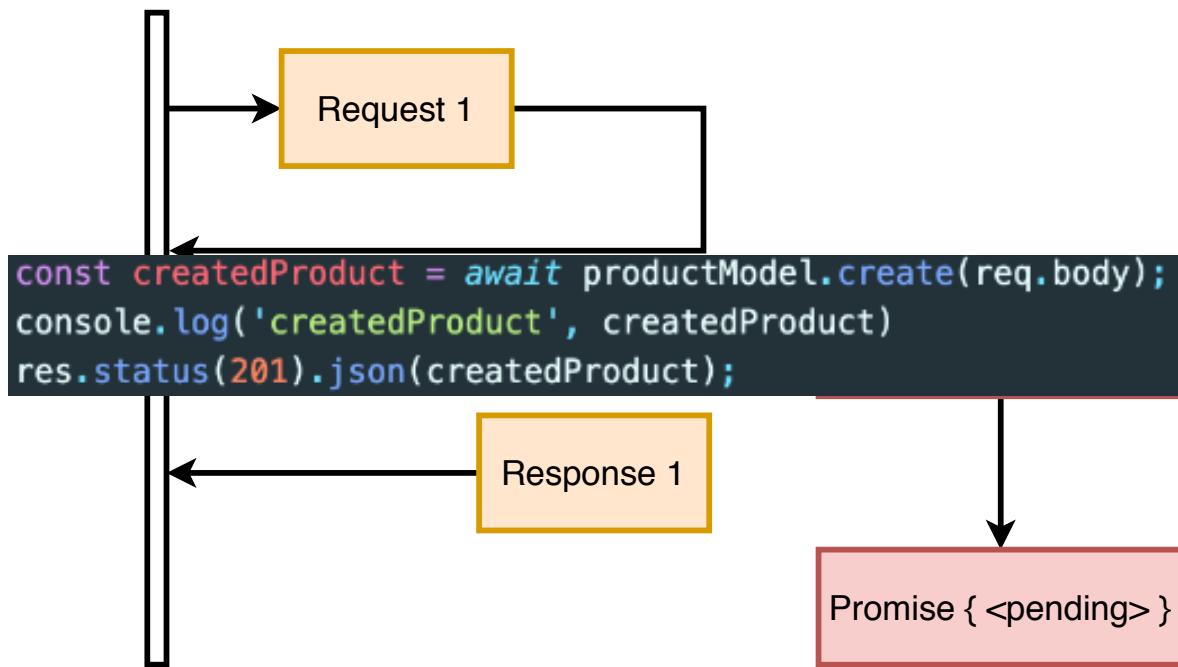
Promise { <pending> } 이라고 나오는 이유는 ???

Product 데이터를 저장할 때 비동기로 하기 때문입니다.

동기
(Synchronous)

비동기
(Asynchronous)





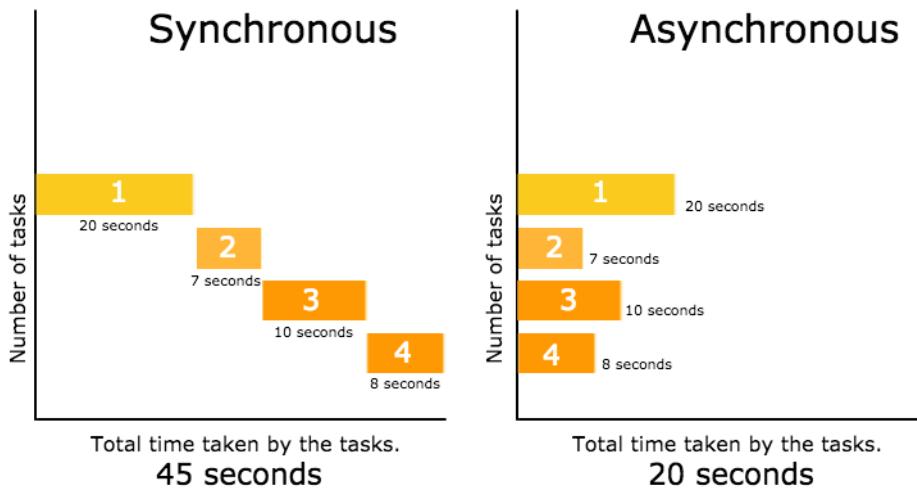
해결 방법은 ???

아직 Response가 오지않은 (Pending) 한 상태가 아닌 결과값을 받은 이후에 값을 처리해주면 됩니다.
그러기 위해서는 두가지 방법이 있습니다.



이렇게 async await을 실제 코드에 넣어줬으니 단위 테스트를 할 때도 넣어주어야 합니다.

```
it("should return 201 response code", async () => {
  await productController.createProduct(req, res, next);
  expect(res.statusCode).toBe(201);
  expect(res._isEndCalled()).toBeTruthy();
})
```



에러 처리를 위한 단위 테스트 작성

임의로 에러를 야기하면 어떻게 되는지 보겠습니다.

```
{  
  "name": "phone",  
  "price": 10  
}
```

```
description: {  
  type: String,  
  required: true  
},
```

무조건 들어가야하는 정보인 **description**을 빼고
요청을 넣으면 ...



Sending request...

Cancel

이렇게 Hang 에 걸려 버립니다.

왜냐하면 에러 처리 하는 부분이 없기 때문입니다.

```
exports.createProduct = async (req, res, next) => {  
  const createdProduct = await productModel.create(req.body);  
  console.log('createdProduct', createdProduct)
```

```
res.status(201).json(createdProduct);  
};
```

해야 할 일은 ???

데이터를 저장할 때
만약 에러가 나면
그 에러를 처리해주는 부분을
구현해주기

단위 테스트 작성

```
it("should handle errors". asvnc () => {  
    const errorMessage = { message: "description property missing" };  
    const rejectedPromise = Promise.reject(errorMessage);  
    productModel.create.mockReturnValue(rejectedPromise);  
    await productController.createProduct(req, res, next);  
    expect(next).toBeCalledWith(errorMessage);  
});
```

비동기 요청에 대한 결과 값은
성공할 때는 `Promise.resolve(value)`
에러일 때는 `Promise.reject(reason)`

Mongo DB에서 처리하는 부분은 문제가 없다는 것을
가정하는 단위 테스트이기 때문에
원래 MongoDB에서 처리 하는 에러 메시지 부분은
Mock 함수를 이용해서 처리해줄겠습니다.

비동기 요청
Async Request

성공!!!

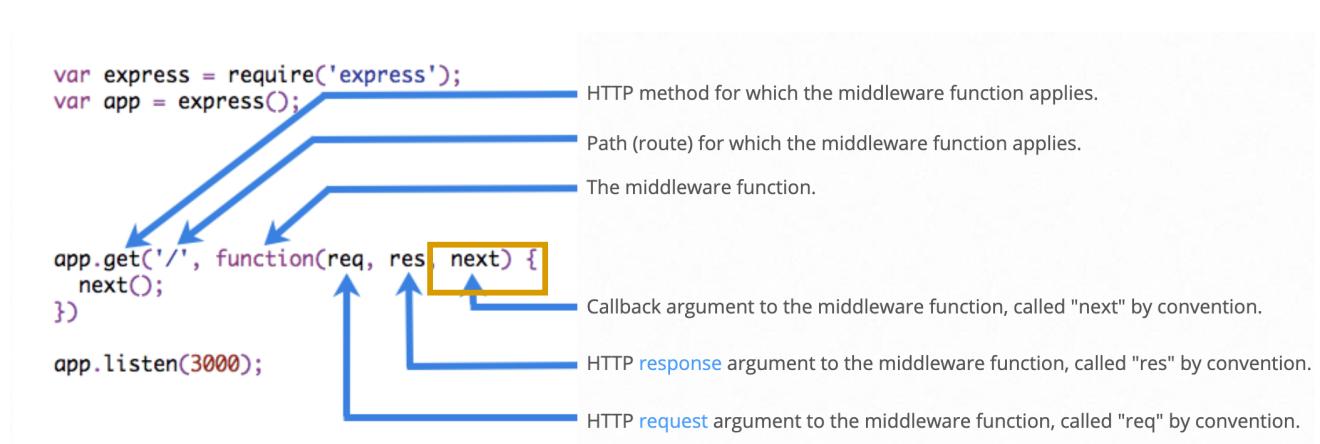
resolve
메소드 실행
`resolve(value)`

resolve 메소드 인자 값을
then 메소드를 통해서
처리 가능

에러!!!

reject

메소드 실행
resolve(reason)



z

테스트에 대응하는
실제 코드 작성

```
exports.createProduct = async (req, res, next) => {
  try {
    const createdProduct = await productModel.create(req.body);
    console.log('createdProduct', createdProduct)
    res.status(201).json(createdProduct);
  } catch (error) {
    next(error);
  }
};
```