**Handle Asynchronous errors on ExpressJS without try/catch -** [Shrihari Mohan](https://medium.com/@shriharimohan?source=post_page-----859d78764938--------------------------------) Jun 11, 2022

Text

Description automatically generated

Asynchronous Error Handler Function For ExressJS

Errors that occur in synchronous code inside route handlers and middleware require no extra work. If synchronous code throws an error, then Express will catch and process it.

The following code will handle all the synchronous code. This means that each operation must wait for the previous one to complete before executing. By default express comes with error handler.

const express = require('express')  
const app = express()  
const port = 3000app.get('/', (req, res) => {  
 res.status(200).send('Hello World!')  
})// Error handler for synchronous errors  
// This should be the last middleware in the chain  
app.use((err, req, res, next) => {  
 console.error(err.stack)  
 res.status(500).send('Something broke!')  
})app.listen(port, () => {  
 console.log(`Example app   
 listening on port ${port}`)  
})

But Most of our backend consists of some db operations and is always asynchronous. This means you can move to another task before the previous one finishes.

So how to handle Asynchronous error on express ?  
One simple solution is to use try/catch and call next() on catch block.

By calling next() we indicate express to catch the error and respond accordingly. The modified version of route will be.

app.get('/', (req, res, next) => {  
 try {  
 // some db asyncrhonous operations  
 res.status(200).send('Hello World!')  
 }  
 catch(error) {  
 next(error)  
 }  
})

The next(error) will call our default error handling middleware and our result will be something broke! when error occurs.

But our app will not consists just one route. It would be just growing and we don’t want to miss any errors that may suprise us in the future.

We will make something interesting without try/catch.

So we will create a promiseHandler function where we handle every asyncrhonous ( ideally a promise ) and call next() on reject.

In one line

const promiseHandler = () => (req, res, next) => {  
 Promise.resolve(fn(req, res, next)).catch(next)  
 }// and route becomes  
app.get('/', promiseHandler (async (req, res, next) => {  
 // some db asyncrhonous operations  
 res.status(200).send('Hello World!')  
}))

This promise handler function takes a async function as argument and resolve when successfull or call next as a callback function when there is a error.

To hack away and test with some errors , use the following repository. [NodeJS-error-handler](https://github.com/ShrihariMohan/nodejs-error-handler)

For my typescript guys out there your promise handler function will be

import { NextFunction, Request, Response } from "express";type fnType = (req : Request, res : Response, next : NextFunction) => void;const promiseHandler = (fn: fnType) =>   
 (req : Request, res : Response, next : NextFunction) => {  
 Promise.resolve(fn(req , res , next)).catch(next)  
}

If you’re interested in NodeJS you may wanna know , [Mistakes I made in NodeJS When I started](https://medium.com/@shriharimohan/%EF%B8%8F-mistakes-i-made-when-i-started-nodejs-mongodb-350e73c24b3a)