COURSEWARE

Error Handling

Contents

Overview

Not all requests will complete successfully, when things go wrong it is important to know how to handle the errors.

Throwing an error

Before we can handle an error we must first create one.
Errors can be created in JS like this:

```
const err = new Error('message');
```

`Error` is a built-in object that represents a generic error. The constructor takes in the error message.

Once we have an error we need to handle it.
In order to handle it we will first need to pass the error to *error handling middleware*.
There are two ways we can do this, the first is by calling **next()** and passing the error into it.

```
app.get('/error', (req, res, next) => {
    const err = new Error('Useful error message');
    next(err);
});
```

The second is to simply throw an error - behind the scenes Express will pass the error on the the error handler for you.

```
app.get('/error', (req, res, next) => {
    throw new Error('Useful error message');
});
```

Error handling middleware

Normally Express middleware has three parameters (req, res and next), error handling middleware differs by having *four* parameters - err, req, res and next.

When an error is thrown, using either method, the next piece of middleware with an **err** parameter is called.

```
app.get('/error', (req, res, next) => {
    const err = new Error('Useful error message');
    next(err);
});

const errorLogger = (err, req, res, next) => {
    console.log(err.stack);
    next(err);
}
app.use(errorLogger);

const sendError = (err, req, res) => {
    res.status(500).send(err.message);
}
app.use(sendError);
```

In this example an error will be passed into **next()** in the GET method.
This error will then be logged to the console and passed on to the last function which will terminate the operation by sending the error back as a response body.

Remember that your middleware *must* either call **next()** *or* **res.send()** or else no response will be sent.

Tutorial

There is no tutorial for this module.

Exercises

1. Create a new project called *error_handling*, install express.

▶ Click here for solution

2. Set-up your Express app to listen on port 4000.

▶ Click here for solution

3. Create an endpoint that throws an error.

▶ Click here for solution

4. Add a piece of error-handling middleware that logs the error's stacktrace to the console then passes it on.

▶ Click here for solution

5. Add another piece of error-handling middleware that sends a response informing the user that something has gone wrong - make sure to use an appropriate status code.

▶ Click here for solution