THE LINUX FOUNDATION | Training & Certification

Lab 10.2 - Async Function Error Handling

The following code loads the **`fs`** module and uses its promises interface to read a file based on a file path passed to a **`read`** function:

```
const fs = require('fs')

async function read (file) {
  const content = await fs.promises.readFile(file)
  return content
}
```

The promise returned from **`fs.promises.readFile`** may reject for a variety of reasons, for instance if the specified file path doesn't exist or the process doesn't have permissions to access it. In this scenario, we don't care what the reason for failure is, we just want to propagate a single error instance from the native **`Error`** constructor with the message 'failed to read'.

The labs-2 **`index.js`** file contains the following code:

```
'use strict'
const fs = require('fs')
const assert = require('assert')

async function read (file) {
  const content = await fs.promises.readFile(file)
  return content
}

async function check () {
  await assert.rejects(
```



```
    read('not-a-valid-filepath'),
    new Error('failed to read')
  )
  assert.deepEqual(
    await read(__filename),
    fs.readFileSync(__filename)
  )
  console.log('passed!')
}

check()
```

Modify the body of the **`read`** function so that any possible rejection by the promise returned from the **`fs.promises.readFile`** call results in the **`read`** function rejecting with a **`new Error('failed to read')`** error. If implemented correctly, when **`node index.js`** is executed the output should be **`passed!`**:

```
2. bash
$ node index.js
passed!
$
```

