**Node.js Core Objects – File System**

* **Import fs from ‘fs’**

**fs.readFile()**

* Reads data from a provided file

const fs = require('fs');

let readDataCallback = (err, data) => {

if (err) {

console.log(`Something went wrong: ${err}`);

} else {

console.log(`Provided file contained: ${data}`);

}

};

fs.readFile('./file.txt', 'utf-8', readDataCallback);

* Imported the fs module
* Defined an error first callback function – expects an error as the first argument and data as the second argument
  1. If an error is present it will print – Something went wrong and the error
  2. Otherwise it will print – Provided file contained: and the data
* Invoked the .readFile method with three arguments
  1. First argument is a string with the path to file.txt
  2. Second argument is a string specifying character encoding (this is normally ‘utf-8’ or text files)
  3. Third argument is a callback function to be invoked when the asynchronous task of reading from the file system is complete. Node will pass the contents of file.txt into the callback as the second argument.

**Node.js Core Objects – Readable Streams**

* **Import readline from ‘readline’**

**readline.createInterface()**

const readline = require('readline');

const fs = require('fs');

const myInterface = readline.createInterface({

input: fs.createReadStream('text.txt')

});

myInterface.on('line', (fileLine) => {

console.log(`The line read: ${fileLine}`);

});

* We require in readline and fs modules
* Assign myInterface the returned value from invoking readline.createInterface() with an object containing our designated input
* We set our input to fs.createReadStream(‘text.txt’)
* Assign a listener callback to execute when line events are emitted. A ‘line’ event will be emitted after each line from the file is read.
* Our listener callback will log to the console – ‘The line read: and the file line’

**Node.js Core Objects – Writable Streams**

* **Create a writable stream using fs.createWriteStream(filename)**

const fs = require('fs')

const fileStream = fs.createWriteStream('output.txt');

fileStream.write('This is the first line!');

fileStream.write('This is the second line!');

fileStream.end();

* Set the output file for createWriteStream as output.yxy
* Then we write lines to the file – write()
* A readable stream ends when there is no more data to read, a writable stream needs to be closed otherwise it would remain open. Indicate end using .end()