# **IoT – NodeJS – Assignment 2 Part 1**

# Exercise 1 - Named and anonymous callback functions

Create a file named index.html and save it in the same folder as the code shown below in table 1. Using the sample code in table 1 for a NodeJS “fs” module, rewrite the named function, writeFileContentToConsole, as an anonymous function and run the code.

|  |
| --- |
| var fs = require('fs');  fs.readFile("index.html", writeFileContentToConsole);  function writeFileContentToConsole(err, data) {  if (err) {  console.log(err);  } else {  console.log(data.toString());  }  } |

***Table 1 : File handling code***

Delete the index.html file, rerun the code. Record the error message that is generated here:

# Exercise 2 – Synchronous Vs asynchronous functions

Using the NodeJS API write the code to make a copy of the index.html file from exercise 1. Use both the synchronous and asynchronous versions of the copy file function.

# Exercise 3 - Events and Event Handling in NodeJS

Using the following code at <https://www.w3schools.com/nodejs/nodejs_events.asp> as a basis complete the following coding tasks:-

* Add another eventEmitter object,
* Add another event handler function to the code
* Bind an event to the event handler function
* Fire/emit the new event and test that the new event handler code works
* Comment each line of your code to demonstrate that you understand it.

|  |
| --- |
| var events = require('events'); var eventEmitter = new events.EventEmitter();  var btEventHandler = function () {   console.log('discovered a Bluetooth device in the room!'); }  eventEmitter.on('BTdiscover', btEventHandler);  eventEmitter.emit('BTdiscover'); |

***Table 2 : Event handling code***

# Exercise 4 – fs and http modules

Using the following sample code for a NodeJS HTTP server from <https://www.w3schools.com/nodejs/nodejs_http.asp> as a basis, combine this code with the code from exercise 1 above to output the contents of the file to a browser window (instead of to the console) when the browser sends a request to the server on localhost:8080.

|  |
| --- |
| var http = require('http');  http.createServer(function (req, res) {   res.write('Hello World!');    res.end();  }).listen(8080); |

***Table 3 : Web server code***

# Exercise 5 – Using the NodeJS API

setTimeout is a basic asynchronous function defined in the timers module of Node.js (<https://nodejs.org/api/timers.html)>. The function takes 2 or more parameters the first two of which are 1) a callback function and 2) a delay value (for the amount of time to delay before the callback function is executed). The sample code below shows a simple use of the setTimeout function being called with the 2 mandatory parameters.

|  |
| --- |
| function callbackFunc() {  console.log("2 seconds later...callbackFunc is executed");  }  setTimeout(callbackFunc, 2000);  //next line gets executed while waiting for the delay to complete  console.log("this is executed while waiting for the callback function"); |

***Table 4 : setTimeout function***

Using the API at <https://nodejs.org/api/timers.html> for the setTimeout() function write the code for a similar **named** callback function that in addition to the two mandatory parameters it also expects two numbers to be passed into it as additional parameters and that then prints out to the console the product of these two numbers, i.e.

console.log("2 seconds later...and the answer is ....." + a\*b);