# Quiz 1 Student ID: \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Student Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## True/False Questions – 1 point each

1. A stream fires an event when all data has been flushed to underlying system. - True
2. Node.js is perfect to use in I/O bound Applications - True
3. Buffer class is a global class and can be accessed in application without importing Buffer module. - True
4. All APIs of Node.JS are asynchronous. - False
5. The \_\_dirname represents the name of the directory that the currently executing script resides in. - True

## Multiple Choices Questions – 1 point each

1. Which of the following is true about Node.JS?
2. Node.js is a JavaScript based framework/platform built on Google Chrome's JavaScript V8 Engine.
3. Node.JS is used to develop I/O intensive web applications like video streaming sites, single page applications and other web application.
4. Node.js is open source and is completely free to use.
5. All of the above.
6. Which of the following command will show all the modules installed globally?
7. npm ls -g
8. npm ls
9. node ls -g
10. node ls
11. Which of the following is true about readable stream?
12. Readable stream is used for read operation.
13. Output of readable stream can be input to a writable stream.
14. Both of the above.
15. None of the above.
16. Which of the following statement is valid to use a Node module fs in a Node based application?
17. let fs = require("fs");
18. let fs = import("fs");
19. package fs;
20. import fs;
21. What can you export with module.exports?
22. Only objects.
23. Only functions
24. Only variables and arrays
25. Functions, objects, arrays, or anything you assign to the module

## Short Answer Questions – 6 points

1. **(2 points)** Read the code below and describe what happens in the terminal when run “node app.js”.

If no error, list the outputs in the terminal in correct order. If there’s error, describe the error and what cause the error.

|  |  |
| --- | --- |
| person.js  class Person {      constructor(name) {          this.name = name;      }      getName() {          console.log(this.name);      }  }  exports = new Person('Bella'); | app.js  const person = require('./person');  person.getName();  person.name = 'Jack';  person.getName(); |

throw error:

In the app.js, person is an empty object {}, when call getName() on person, will throw error getName() is not a function.

1. **(4 points)** What’s the differences between process.nextTick() and setImmediate()? (At least 2 differences).

1. callbacks of process.nextTick get executed multiple chances in one iteration - highest priotiry

2. how many callbacks will be executed in one iteration?

process.nextTick() - all of callbacks in nextTick queue, drain out - Don't use while true loop on process.nextTick

setImmediate - certain, the remaining for next iteration/tick

3. process.nextTick - API provided natively by Node.js

 setTimeout, setImmediate - provided by libuv

## Programming Questions – 14 points

1. Use Node.js http module to create a web application which has the following features:
   1. User makes a request to <http://localhost:8080/>, the browser displays a page with a sign up form which has two inputs: username and password and one submit button.
   2. When user clicks submit button in the browser, the browser makes a POST request to url: <http://localhost:8080/success>, the server side our web application will save user’s inputs: username and password into a file under current directory named database.txt.
   3. After saving successfully, user will see message “Sign up successfully” displayed in the browser. If saving failed, user will see message “Try later” displayed in the browser.

The format of database.txt looks like after your server saves the data: john=122

NOTE: It’s your own choice to write the sign up form in a html file, then read the html file and send back. Or you send the sign up form string directly to the browser from your server.

signup.html

<!DOCTYPE html>

<html>

<body>

<h2>HTML Forms</h2>

<form action="success" method="post">

  <label for="username">Username:</label><br>

  <input type="text" id="username" name="username"><br>

  <label for="password">Password:</label><br>

  <input type="password" id="password" name="password"><br><br>

  <input type="submit" value="Submit">

</form>

</body>

</html>

app.js

const http = require('http');

const fs = require('fs');

http.createServer((req, res)=>{

    console.log(req.url);

    const url = req.url;

    if(url === '/') {

        fs.createReadStream('signup.html').pipe(res);

    }else if(url === '/success' && req.method === 'POST'){

        const body = [];

        req.on('data', (chunk)=>{

            body.push(chunk);

        });

        req.on('end', ()=>{

            const postData = Buffer.concat(body).toString();

            console.log(postData);

            const postDataArr = postData.split('&'); *//[username=john, password=Doe]*

            const firstNameArr = postDataArr[0].split('='); *// [username, john]*

            const lastnameArr = postDataArr[1].split('='); *//[password, Doe]*

            fs.writeFile('database.txt', `${firstNameArr[1]}=${lastnameArr[1]}`, (error)=>{

                if(error) res.end('Try later...');

                else res.end('Sign up successfully');

            });

        });

    }

}).listen(8080, ()=> console.log('listening to 8080'));