LAB # 5
Intermediate Java script

OBJECTIVE

To get familiar and become more knowledgeable in implementing real world JS use cases

Lab Task

1 Classes and Inheritance:
   a. Create a base class called *Student* and export it
      i. Add id, name, date of birth properties
      ii. Add *enroll* method
   b. Create two child classes called *RegularStudent* and *ExecutiveStudent* and inherit them from *Student* base class
      i. Add *attendLab* method in *RegularStudent* class
      ii. Add *attendTheory* method in *ExecutiveStudent* class
   c. Create a separate module and import *RegularStudent* and *ExecutiveStudent* classes. Validate using *console.log* that the properties from base classes have been inherited into child classes along with own properties.

Input:

```
class student {
    constructor(id, name,dob,myclass) {
        this.id = id;
        this.name = name;
        this.dob=dob;
        this.myclass=myclass;

    }
    studentIdentity(){
        console.log(`${this.name} is a student`);
    }
}

export {student};
```

```
import {student} from './student.js';

class executiveStudent extends student {
    constructor(id, name,dob,myclass) {
        super(id, name,dob);
        this.myclass=myclass;
    }
    exeStudent(){
        console.log(`${this.name} is Executive student${this.myclass}`);
```

```
    }
}
export { executiveStudent };
```

```
import { student } from './student.js';
class regularStudent extends student {

    constructor(id, name,dob,myclass,) {
        super(id, name,dob);
        this.myclass=myclass;

    }
    studyClass(){
        console.log(`${this.name} is study ${this.myclass}`);
    }
}
export{regularStudent};
```

```
import { regularStudent } from './regularStudent.js';
import { executiveStudent } from './executiveStudent.js';
let RegularStudent = new regularStudent(1, 'Student1', '06','Web-Engineering');
let ExecutiveStudent= new executiveStudent(2, 'Student2', '01','IT');
console.log(RegularStudent.name);
console.log(ExecutiveStudent.name);
console.log(RegularStudent.study);
console.log(ExecutiveStudent.exeStudent());
console.log(RegularStudent.studentIdentity());
console.log(ExecutiveStudent.studentIdentity());
```

Output:

```
C:\Users\User1\Documents\lab4>npm run exercise1-transpile

> lab-4@1.0.0 exercise1-transpile
> babel ./executiveStudent.js ./regularStudent.js ./student.js ./index.js --out-dir ./output

Successfully compiled 4 files with Babel (11744ms).
```

```
C:\Users\User1\Documents\lab4>node ./index.js
Student1
Student2
undefined
Student2 is Executive studentIT
undefined
Student1 is a student
undefined
Student2 is a student
undefined
```

2 HTTP and Database:

a. Import *sqlite* and *axios* packages in separate module files

b. Create DB, open connection and perform CRUD operations for *Student using sqlite* library

i. *Create* and open database

ii. *Create* Student table using properties from *Student* class created above

iii. *Insert* two records for both regular and executive student
iv. *Select* all records

c. Fetch the university *About* page using HTTP *axios* library
   i. Using *GET* method of axios library, fetch SSUET website *About* page

Input:

```
import axios from 'axios';

function getSSUETHomePage(){
    axios.get('https://www.ssuet.edu.pk').then((res) => {
        console.log(res.data);
    });
}

const HTTP = {
    getSSUETHomePage
}

export { HTTP };

import { open } from 'sqlite';
import sqlite3Offline from 'sqlite3-offline';

function openStudentDB() {
    console.log('opening DB connection');
    return open('', {
        client: "sqlite",
        connection: {
            database: "./db/student.db"
        }
    });
}

function createStudentTable(db) {
    console.log('creating Student table');
    return db.exec('CREATE TABLE student (id INTEGER, name TEXT)');
}

function insertStudentRecord(db) {
    console.log('inserting Student record');
    return db.exec("INSERT INTO student VALUES (1, 'student1')");
}

function selectStudentRecord(db) {
    console.log('selecting Student record');
    return db.get('SELECT id, name FROM student');
}

const DB = {
    openDB: openStudentDB,
```

```
    createTable: createStudentTable,
    insertStudent: insertStudentRecord,
    selectStudent: selectStudentRecord
};

export { DB };

import createLogger from 'logging'

let Logger = createLogger('createStudent');

export { Logger };

import { HTTP } from './http.js';
import { DB } from './database.js';
import { Logger } from './logging.js';

//Putting the code under try-catch block to avoid crashing the appliation
//and handling the error gracefully
try {
    //SSUET Home Page - HTTP GET
    getUniversityHomePage();

    //Perform Student DB Operations
    performStudentDBOperations();
} catch (e) {
    //Logging the info in case anything goes wrong
    Logger.debug("Something went wrong");
}

function getUniversityHomePage(){
    HTTP.getSSUETHomePage();
}

function performStudentDBOperations(){
    DB.openDB().then((db) => {
        DB.createTable(db).then(() => {
            DB.insertStudent(db).then(() => {
                DB.selectStudent(db).then((student) => {
                    console.log(student.id);
                    console.log(student.name);
                });
            });
        });
    });
};

  "exercise2-transpile": "babel ./http.js ./database.js ./logging.js ./index1.js --out-dir ./output",
    "exercise2": "npm run exercise2-transpile && node ./exercises-output/modules/index1.js"
```

Output:

```
:\Users\User1\Documents\lab4>npm run exercise2-transpile

> lab-4@1.0.0 exercise2-transpile
> babel ./http.js ./database.js ./logging.js ./index1.js --out-dir ./output

uccessfully compiled 4 files with Babel (2925ms).

:\Users\User1\Documents\lab4>node ./index1.js
:\Users\User1\Documents\lab4\node_modules\sqlite3-offline\binaries\index.js:21
  throw new Error(`NodeJS ${NODE} Module ${MODULES} not compatible`)
        ^

rror: NodeJS 16.13.0 Module 93 not compatible
    at Object.<anonymous> (C:\Users\User1\Documents\lab4\node_modules\←[4msqlite3-offline←[24m\binaries\index.js:21:9)
←[90m    at Module._compile (node:internal/modules/cjs/loader:1101:14)←[39m
←[90m    at Object.Module._extensions..js (node:internal/modules/cjs/loader:1153:10)←[39m
←[90m    at Module.load (node:internal/modules/cjs/loader:981:32)←[39m
←[90m    at Function.Module._load (node:internal/modules/cjs/loader:822:12)←[39m
←[90m    at Module.require (node:internal/modules/cjs/loader:1005:19)←[39m
←[90m    at require (node:internal/modules/cjs/helpers:102:18)←[39m
    at Object.<anonymous> (C:\Users\User1\Documents\lab4\node_modules\←[4msqlite3-offline←[24m\index.js:4:42)
←[90m    at Module._compile (node:internal/modules/cjs/loader:1101:14)←[39m
←[90m    at Object.Module._extensions..js (node:internal/modules/cjs/loader:1153:10)←[39m
```

3 Testing:

a. Test *Student* class

i. Use the example above to copy paste and validate Student properties

b. Test either *RegularStudent* or *ExecutiveStudent* class

i. Use the example above to copy paste and validate either *RegularStudent* or *ExecutiveStudent* class.

Input:

```
test.js
test.js > describe > test('Student class takes and sets id, name, date of birth') callback
const assertance =require('simple-assert-ok')
const Student =require('./students')
const ExecutiveStudent =require('./ExecutiveStudent')

let tests = [];
function runTests(){

    describe('STUDENT');

    if(tests.length > 0){
        for(let t = 0; t < tests.length; t++){
            tests[t].run();
        }
    }
}

function describe(category){
    console.log(`Running test category: ${category}`);
    console.log('================================');

    test('Executive Student class takes and sets id, name, date of birth and subject correctly', function() {
        const executiveStudent = new ExecutiveStudent(1, 'Samia', '22 Jan', 'Web');
        assertance.assert(executiveStudent.id === 1, result('OK'));
        assertance.assert(executiveStudent.name === 'Samia', result('OK'));
        assertance.assert(executiveStudent.dateOfBirth === '22 Jan', result('OK'));
        assertance.assert(executiveStudent.subject === 'Web', result('OK'));
    });
    test('Student class takes and sets id, name, date of birth', function() {
        const student = new Student(2, 'Abdullah', '10 Jan');
        assertance.assert(student.id === 2, result('OK'));
        assertance.assert(student.name === 'Abdullah', result('OK'));
        assertance.assert(student.dateOfBirth === '10 Jan', result('OK'));
    });
}

function test(caseName, func){
    tests = [...tests, {
        message: function() {
            console.log(`---Running test case: ${caseName}`);
            console.log('--------------------------------');
        },
        run: function(){
            this.message();
            func();
        }
    }];
}

function result(testStatus){
    console.log("\x1b[32m", testStatus, "\x1b[37m");
}

runTests();
```

Output:

```
huzai@DESKTOP-MMHJFTI MINGW64 /d/web developing/labs/lab4/LabTask/Task3
$ npm run test

> businessanalytica@1.0.0 test D:\web developing\labs\lab4\LabTask\Task3
> node test

Running test category: STUDENT
==================================
---Running test case: Executive Student class takes and sets id, name, date of birth and subject correctly
---------------------------------
 OK
 OK
 OK
 OK
---Running test case: Student class takes and sets id, name, date of birth
---------------------------------
 OK
 OK
 OK
```

Home Task

1 Classes and Inheritance:
   a. Create a class called *Course* and export it
      i. Add id, name, credit hours properties
   b. Create a class called *University* and export it
      i. Add name, image properties
      ii. Add a method called *setImage*
   c. Modify the *Student* class
      i. Add a property called *university* of type *University* class
      ii. Add a property called *courses* of an array of *Course* class
      iii. Add a method called *addCourse* accepting *Course* class object and set that onto *courses* property
      iv. Add a method called *belongsToUniversity* accepting *University* class object and set that onto *university* property

Input:

```
class Course{
    constructor(id, name, credit_hours){
        this.id=id;
        this.name=name;
        this.credit_hours=credit_hours;
    }
    addCourse(){
        console.log("course is added");
    }
}
export{ Course }
```

```
class University{
    constructor(name, image){
        this.name=name;
        this.image=image;
    }
    setImage(){

    }
    belongsToUniversity(){
        console.log("this student belongs to sir syed university");
    }
}
export{ University }
```

```
import { University } from "./University";
import { Course } from "./Course";
uni= new University("sir syed university", "image");
c= new Course(201, "web engineering", 3);
console.log(u.name);
console.log(u.belongsToUniversity());
console.log(c.name);
console.log(c.addCourse());
```

```
"exercise3-transpile": "babel ./Course.js ./University.js ./Students.js --out-dir ./output",
    "exercise3": "npm run exercise3-transpile && node ./exercises-output/modules/Students.js"
```

Output:

```
C:\Users\User1\Documents\lab4>npm run exercise3-transpile

> lab-4@1.0.0 exercise3-transpile
> babel ./Course.js ./University.js ./Students.js --out-dir ./output

Successfully compiled 3 files with Babel (376ms).

C:\Users\User1\Documents\lab4>node ./Students.js
node:internal/process/esm_loader:94
    internalBinding('errors').triggerUncaughtException(
                              ^

Error [ERR_MODULE_NOT_FOUND]: Cannot find module 'C:\Users\User1\Documents\lab4\University' imported from C:\Us
\Documents\lab4\Students.js
Did you mean to import ../University.js?
←[90m    at new NodeError (node:internal/errors:371:5)←[39m
←[90m    at finalizeResolution (node:internal/modules/esm/resolve:416:11)←[39m
←[90m    at moduleResolve (node:internal/modules/esm/resolve:932:10)←[39m
←[90m    at defaultResolve (node:internal/modules/esm/resolve:1044:11)←[39m
←[90m    at ESMLoader.resolve (node:internal/modules/esm/loader:422:30)←[39m
←[90m    at ESMLoader.getModuleJob (node:internal/modules/esm/loader:222:40)←[39m
←[90m    at ModuleWrap.<anonymous> (node:internal/modules/esm/module_job:76:40)←[39m
←[90m    at link (node:internal/modules/esm/module_job:75:36)←[39m {
  code: ←[32m'ERR_MODULE_NOT_FOUND'←[39m
}
```

2 HTTP and Database:

a. Modify the University class's *setImage* method
   i. Fetch the SSUET logo from the SSUET website using *axios GET* method and set it to *image* property
   ii. Create a new instance and call *setImage* method on the instance and validate using console.log if the image property has been set

b. Create DB, open connection and perform CRUD operations for *University* using *sqlite* library
   i. *Create* and open database
   ii. *Create* University table using properties from *University* class created above
   iii. *Create* two *University* instances
   iv. *Call setImage* method on both the instances
   v. *Insert* both the instances using their *name* and *image* properties

Input:

EXPLORER

TASK2
- db
- node_modules
- baseClass.js
- Course.js
- database.js
- http.js
- index.js
- logging.js
- npm-debug.log
- package-lock.json
- package.json
- README.md
- students.js
- University.js

index.js > ...

```javascript
const HTTP =require('./http')
const DB =require('./database.js')

const University = require('./University');
let university1Instance=new University('Sir Syed University')
let university2Instance=new University('NED')

try {
    let university1= HTTP.setImage(university1Instance, "https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.ssuet.edu.pk%2F&psig=AOvVaw13WkVAWoSg49gHZxNcEwFL&ust=1636144674517000&source=images&cd=vfe&ved=0CAsQjRxqFwoTCJi_g4TI__MCFQAAAAAdAAAAABAD")

    performStudentDBOperations(university1)

    let university2= HTTP.setImage(university2Instance, "https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.neduet.edu.pk%2F&psig=AOvVaw2Gr_UcgxktXaM5E4AE_Wq4&ust=1636146722308000&source=images&cd=vfe&ved=0CAsQjRxqFwoTCOC78dPP__MCFQAAAAAdAAAAABAD")

    performStudentDBOperations(university2)
} catch (e) {
    console.error("Something went wrong")
}


async function performStudentDBOperations(university){
    DB.openDB().then((db) => {
        DB.createTable(db).then(() => {
            DB.insertUniversity(db,university).then((univ) => {
                DB.selectUniversity(db).then((uni) => {
```

```
{} package-lock.json
{} package.json
ⓘ README.md
JS students.js
JS University.js
```

```
                console.log("University Name", uni.name);
                console.log("University Image",uni.image);
            }).catch(err=>{
                console.log("error1", err)
            });;
        }).catch(err=>{
            console.log("erro2", err)
        });
    }).catch(err=>{
        console.log("erro3", err)
    });
}).catch(err=>{
    console.log("erro3", err)
});
};
```

EXPLORER

TASK2

```
db
node_modules
baseClass.js
Course.js
database.js
http.js
index.js
logging.js
npm-debug.log
package-lock.json
package.json
README.md
students.js
University.js
```

index.js | University.js | database.js | package.json | http.js

http.js > setImage

```
const axios= require('axios')

async function setImage(university, image){
    let result=await axios.get(image)
    university.setImage(result.config.url)
    return university
}

const HTTP = {
    setImage
}


module.exports=HTTP

```

EXPLORER

TASK2

```
db
node_modules
baseClass.js
Course.js
database.js
http.js
index.js
logging.js
npm-debug.log
package-lock.json
package.json
README.md
students.js
University.js
```

index.js | University.js | database.js | package.json | http.js

database.js > insertUniversityRecord

```
const sqlite = require('sqlite');
const sqlite3 = require('sqlite3');
const path = require('path');

function openUniversityDB() {
    console.log('opening DB connection');
    var db_path = path.resolve(__dirname, 'db/university.db');
    return sqlite.open({
        filename: db_path,
        driver: sqlite3.Database
      });
}

function createUniversityTable(db) {
    console.log('creating University table');
    return db.exec('CREATE TABLE IF NOT EXISTS university (name TEXT, image TEXT)');
}

async function insertUniversityRecord(db, university) {
    console.log('inserting University record');
    university = await university
    return db.exec(`INSERT INTO university VALUES ("${university.name}", "${university.image}")`);
}

function selectUniversityRecord(db) {
    console.log('selecting University record');
    return db.get('SELECT name, image FROM university');
}

```

```
const DB = {
    openDB: openUniversityDB,
    createTable: createUniversityTable,
    insertUniversity: insertUniversityRecord,
    selectUniversity: selectUniversityRecord
};

module.exports=DB
```

Output:

```
$ npm start

> businessanalytica@1.0.0 start D:\web developing\labs\lab4\Home Task\Task2
> node index

opening DB connection
opening DB connection
creating University table
creating University table
inserting University record
inserting University record
selecting University record
University Name NED
University Image https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.neduet.edu.pk%2F&psig=AOvVaw2Gr_UcgxktXaM5E4AE_Wq4&ust=1636146722308
000&source=images&cd=vfe&ved=0CAsQjRxqFwoTCOC78dPP__MCFQAAAAAdAAAAABAD
selecting University record
University Name NED
University Image https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.neduet.edu.pk%2F&psig=AOvVaw2Gr_UcgxktXaM5E4AE_Wq4&ust=1636146722308
000&source=images&cd=vfe&ved=0CAsQjRxqFwoTCOC78dPP__MCFQAAAAAdAAAAABAD
```

db > university.db

Search ta Reset Filters Records: 2

Tables (1)

university

	name	image
	Search records...	Search records...
1	NED	https://www.google.c...
2	Sir Syed University	https://www.google.c...

3 Testing:

a. Test *University* class.

i. Create a *University* instance

ii. Assert if the instance has image property set to empty string

iii. Call *setImage* method on the new instance

iv. Assert if the instance has image property set to some string

Input:

```
package.json   test.js   University.js
test.js > ...
const assertance =require('simple-assert-ok')
const University =require('./University')

let tests = [];
function runTests(){

    describe('University');

    if(tests.length > 0){
        for(let t = 0; t < tests.length; t++){
            tests[t].run();
        }
    }
}

function describe(category){
    console.log(`Running test category: ${category}`);
    console.log('==================================');

    test('Executive University class takes and sets name and image correctly', function() {
        const university = new University('Sir Syed');
        assertance.assert(university.image == null, result('OK'));
    });
    test('Executive University class takes and sets name and image correctly', function() {
        const university = new University("NED");
        university.setImage('https://www.neduet.edu.pk/sites/default/files/lej_campus_0.jpg')
        assertance.assert(university.image === 'https://www.neduet.edu.pk/sites/default/files/lej_campus_0.jpg',
        result('OK'));
    });
}

function test(caseName, func){
    tests = [...tests, {
        message: function() {
            console.log(`---Running test case: ${caseName}`);
            console.log('---------------------------------');
        },
        run: function(){
            this.message();
            func();
        }
    }];
}

function result(testStatus){
    console.log("\x1b[32m", testStatus, "\x1b[37m");
}

runTests();
```

Output:

```
> businessanalytica@1.0.0 test D:\web developing\labs\lab4\Home Task\Task3
> node test

Running test category: University
==================================
---Running test case: Executive University class takes and sets name and image correctly
---------------------------------
 OK
---Running test case: Executive University class takes and sets name and image correctly
---------------------------------
 OK
```