# CS602 Module1 Assignment

## General Rules for Homework Assignments

- You are strongly encouraged to add comments throughout the program. Doing so will help your instructor to understand your programming logic and grade you more accurately.
- You must work on your assignments individually. You are not allowed to copy the answers from the others.
- Each assignment has a strict deadline. Late submissions will carry a penalty.
- When the term *lastName* is referenced in an assignment, please replace it with your last name.

**Create a new folder/directory named CS602_HW1_*lastName.* Write the following programs in this folder.**

## Part 1 – Node.js Modules (40 Points)

Write a Node.js module, `employeeModule_v1.js`, with the following functionality. The module maintains an array of JavaScript objects, each with the property *firstName*, *lastName*, and a unique *id* value. For example, a sample initial data is shown below.

```
var data = [
  {id:1, firstName:'John',lastName:'Smith'},
  {id:2, firstName:'Jane',lastName:'Smith'},
  {id:3, firstName:'John',lastName:'Doe'}
 ];
```

The module should export the functions `lookupById`, `lookupByLastName`, and `addEmployee`. The function `lookupById` should return the JavaScript object from the data whose `id` matches the specified argument. If the specified `id` is not present, `undefined` is returned. The function `lookupByLastName` should return the array of JavaScript objects from the data whose `lastName` matches the specified argument. If the specified `lastName` is not present, `[]` is returned. The function `addEmployee` only takes two arguments, the *firstName* and *lastName* of the employee being added. The `id` value should be calculated as one more than the current maximum `id`. Write the code for the module using only JavaScript and without using the *underscore* module.

Now, write the application, hw1a.js, using the functionality of the above module. Write the code to do the following:

- Lookup by last name, Smith, and print the results.
- Add a new employee with first name, William, and last name, Smith.
- Lookup by last name, Smith, and print the results.
- Lookup by id, 2, and assign the value to a variable.
- Print the variable.
- Using the above variable, change the first name to Mary.
- Lookup again by id, 2, and print the result.
- Lookup by last name, Smith, and print the results.

The sample output of the application is shown below. You can optionally use the `colors` module for colors in the output.

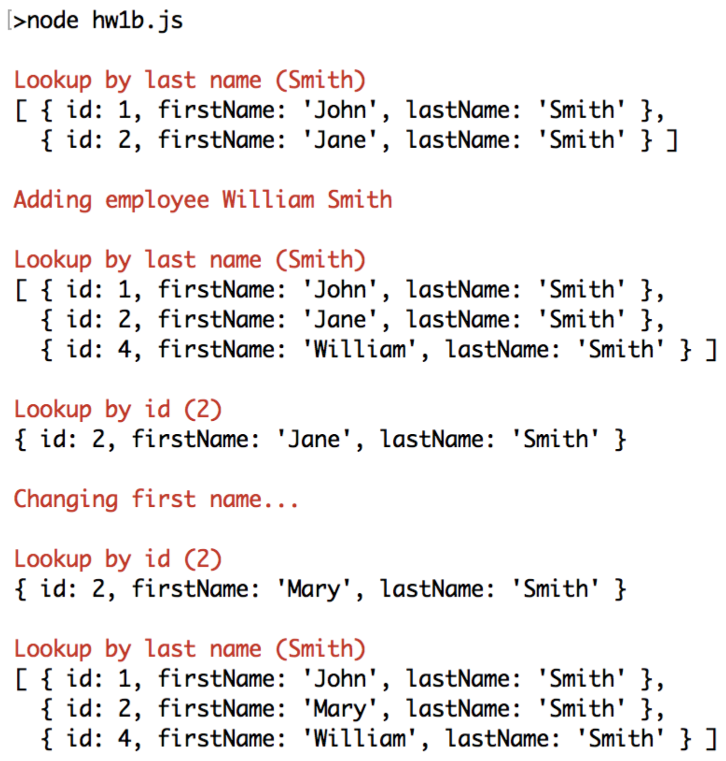
```
>node hw1a.js

Lookup by last name (Smith)
[ { id: 1, firstName: 'John', lastName: 'Smith' },
  { id: 2, firstName: 'Jane', lastName: 'Smith' } ]

Adding employee William Smith

Lookup by last name (Smith)
[ { id: 1, firstName: 'John', lastName: 'Smith' },
  { id: 2, firstName: 'Jane', lastName: 'Smith' },
  { id: 4, firstName: 'William', lastName: 'Smith' } ]

Lookup by id (2)
{ id: 2, firstName: 'Jane', lastName: 'Smith' }

Changing first name...

Lookup by id (2)
{ id: 2, firstName: 'Mary', lastName: 'Smith' }

Lookup by last name (Smith)
[ { id: 1, firstName: 'John', lastName: 'Smith' },
  { id: 2, firstName: 'Mary', lastName: 'Smith' },
  { id: 4, firstName: 'William', lastName: 'Smith' } ]
```

Now, in the file hw1aFixes.txt, discuss how you can prevent the changes done by the user on the objects returned by the module functions not reflect on the module data itself. For example, even after user changes the first name, the lookup should return the original result. Is this the only function, or the lookupByLastName function also has this problem?

## Part 2 – Node.js Modules (30 Points)

Write a different implementation of above employee module using the functions from *underscore* module. This module should be written as `employeeModule_v2.js.` Use the `underscore` module functions `findWhere`, `where`, `pluck`, and `max` in the implementation.

Now, write the application, hw1b.js, using the functionality of the above variation of the employee module. The output should be the same as in the Part 1.



```
[>node hw1b.js

Lookup by last name (Smith)
[ { id: 1, firstName: 'John', lastName: 'Smith' },
  { id: 2, firstName: 'Jane', lastName: 'Smith' } ]

Adding employee William Smith

Lookup by last name (Smith)
[ { id: 1, firstName: 'John', lastName: 'Smith' },
  { id: 2, firstName: 'Jane', lastName: 'Smith' },
  { id: 4, firstName: 'William', lastName: 'Smith' } ]

Lookup by id (2)
{ id: 2, firstName: 'Jane', lastName: 'Smith' }

Changing first name...

Lookup by id (2)
{ id: 2, firstName: 'Mary', lastName: 'Smith' }

Lookup by last name (Smith)
[ { id: 1, firstName: 'John', lastName: 'Smith' },
  { id: 2, firstName: 'Mary', lastName: 'Smith' },
  { id: 4, firstName: 'William', lastName: 'Smith' } ]
```

## Part 3 – Node.js Events (30 Points)

Write a Node.js module, `employeeEmitter.js,` with the following functionality. Provide the `EmployeeEmitter` class inheriting from the `EventEmitter`. The constructor function takes one argument and saves it as the instance variable `data`. Provide the functions `lookupById`, `lookupByLastName`, and `addEmployee` for the `EmployeeEmitter` prototype. The first line in these methods should `emit` the respective event (same name as the function) along with the arguments supplied for the function. The rest of the code in each of the functions should be the same as in Part1 (or Part2). From the module, export one property `EmployeeEmitter` referencing the constructor function.

Now, write the application, hw1c.js, using the functionality of the above module. Write the code to do the following:

- Using the same array data as in Part1, create the `EmployeeEmitter` object using the array data as its argument.
- Write three event handlers for the three events that could be emitted by the three functions. See the sample output for the behavior of these handlers. Now, using the `EmployeeEmitter` object, do the following operations.
- Lookup by last name, Smith, and print the results.
- Add a new employee with first name, William, and last name, Smith.
- Lookup by last name, Smith, and print the results.
- Lookup by id, 2, and print the result.

The sample output of the application is shown below. You can optionally use the `colors` module for colors in the output.

```
>node hw1c.js

Lookup by last name (Smith)
Event lookupByLastName raised! Smith
[ { id: 1, firstName: 'John', lastName: 'Smith' },
  { id: 2, firstName: 'Jane', lastName: 'Smith' } ]

Adding employee William Smith
Event addEmployee raised! William,Smith

Lookup by last name (Smith)
Event lookupByLastName raised! Smith
[ { id: 1, firstName: 'John', lastName: 'Smith' },
  { id: 2, firstName: 'Jane', lastName: 'Smith' },
  { id: 4, firstName: 'William', lastName: 'Smith' } ]

Lookup by id (2)
Event lookupById raised! 2
{ id: 2, firstName: 'Jane', lastName: 'Smith' }
```

**Submission: Export your CS602_HW_*lastName* folder containing all the relevant files as a zip file, and upload the zip file to the Assignment section.**