Mark Ryan
CS602 - Server Side Web Development
9/12/2017

Part 1 - Node.js Modules

---

employeeModule.js

```
//Write a Node.js module, employeeModule.js, with the following
//functionality

var _ = require('underscore');

//The module maintains an array of JavaScript objects, each with the property
//firstName, lastName, and a unique id value.

var data = [

   {id: 1, firstName:'John', lastName:'Smith'},
   {id: 2, firstName:'Jane', lastName:'Smith'},
   {id: 3, firstName:'John', lastName:'Doe'}

];

//export the functions lookupById,lookupByLastName, and addEmployee
//use the underscore module (imported above) and use the functions findWhere,
//where, pluck, and max in the implementation

//function lookupById should return the JavaScript object from the
//data whose id matches the specified argument.

exports.lookupById = function(id) {
        return _.findWhere(data, {id: id});
};

//function lookupByLastName should return the array of JavaScript objects
//from the data whose lastName matches the specified argument.

exports.lookupByLastName = function(value) {
        return _.where(data, {lastName: value});
};


//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.

exports.addEmployee = function(first, last) {
  var employee = {};
        var employeeMax = _.max(data, function(e) {
```

```
    return e.id
  });
  employee.id = employeeMax.id + 1;
        employee.firstName = first;
        employee.lastName = last;
  data.push(employee);
};
```

---

hw1a.js

```
var colors = require('colors/safe');

//write the application, hw1a.js, using the functionality of employeeModule.js

var employeeInfo = require('./employeeModule');

//employeeModule.data();

//Lookup by last name, Smith, and print the results.

//employeeInfo.lookupByLastName = 'Smith';
var employeeLast = employeeInfo.lookupByLastName('Smith');
console.log(colors.red("Look up by last name Smith"));
console.log(JSON.stringify(employeeLast));

//Add a new employee with first name, William, and last name, Smith
employeeInfo.addEmployee('William', 'Smith');
console.log(colors.red("Adding employee William Smith"));
var employeeLast = employeeInfo.lookupByLastName('Smith');
console.log(colors.red("Look up by last name Smith"));
console.log(JSON.stringify(employeeLast));

//Lookup by last name, Smith, and print the results
//employeeInfo.lookupByLastName = 'Smith';
//console.log(employeeInfo.id, employeeInfo.firstName, employeeInfo.lastName);

//Lookup by id, 2, and assign the value to a variable
//Print the variable
var employeeID = employeeInfo.lookupById(2);
console.log(colors.red("Look up by id 2"));
console.log(employeeID);

//Using the above variable, change the first name to Mary.
//Lookup again by id, 2, and print the result.
//employeeInfo.lookupById = '2';
//console.log(employeeInfo.id, employeeInfo.firstName, employeeInfo.lastName);

employeeID.firstName = "Mary";
console.log(colors.red("Changing first name..."));
var employeeID = employeeInfo.lookupById(2);
console.log(colors.red("Look up by id 2"));
```

```
console.log(employeeID);

//Lookup by last name, Smith, and print the results
//employeeInfo.lookupByLastName = 'Smith';
//console.log(employeeInfo.id, employeeInfo.firstName, employeeInfo.lastName);

console.log(colors.red("Look up by last name Smith"));
console.log(JSON.stringify(employeeLast));
```

Discussion: 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?

The application can define the functions along with the properties of the exported Javascript object, or the application can directly set the global module.exports. The other functions have this issue as well and can also be changed. This includes the lookupById function and if there was a lookupByFirstName function, this could be changed as well with the current format.

Part 2 - Node.js Events

---

employeeEmitter.js

```
var _ = require('underscore');
var util = require('util');
var EventEmitter = require('events');

//Provide the EmployeeEmitter class inheriting from
//the EventEmitter.

//The constructor function takes one argument
//and saves it as the instance variable data.

function EmployeeEmitter(data) {
        this.data = data;
};

util.inherits(EmployeeEmitter, EventEmitter);

//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.

EmployeeEmitter.prototype.lookupById = function(id) {
        this.emit('lookupById', id);
        return _.findWhere(this.data, {id: id});
```

```
};

EmployeeEmitter.prototype.lookupByLastName = function(last) {
        this.emit('lookupByLastName', last);
        return _.where(this.data, {lastName: last});
};

EmployeeEmitter.prototype.addEmployee = function(first, last) {
        this.emit('addEmployee', first, last);
        var employee = {};
        var employeeMax = _.max(this.data, function(e) {
    return e.id
        });
        employee.id = employeeMax.id + 1;
        employee.firstName = first;
        employee.lastName = last;
        this.data.push(employee);
};

/*
From the module, export one property EmployeeEmitter referencing the constructor
function.
*/

exports.EmployeeEmitter = EmployeeEmitter;
```

---

hw1b.js

```
var colors = require('colors/safe');

//write the application, hw1b.js, using the functionality of the
//above module.

var EmployeeEmitter = require('./employeeEmitter').EmployeeEmitter;

//Using the same array data as in Part1, create the
//EmployeeEmitter object using the array data as its
//argument.

var data = [

   {id: 1, firstName:'John', lastName:'Smith'},
   {id: 2, firstName:'Jane', lastName:'Smith'},
   {id: 3, firstName:'John', lastName:'Doe'}

];

var employeeEmitter = new EmployeeEmitter(data);

//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.

employeeEmitter.on('lookupById', function(data){
        console.log(colors.blue("Event lookupById raised! " + data));
});

employeeEmitter.on('lookupByLastName', function(data){
        console.log(colors.blue("Event lookupByLastName raised! " + data));
});

employeeEmitter.on('addEmployee', function(first, last){
        console.log(colors.blue("Event addEmployee raised! " + last + "," + first));
});

//Lookup by last name, Smith, and print the results.
//Add a new employee with first name, William, and last name,
//Smith.

console.log(colors.red("Look up by last name Smith"));
var employeeLast = employeeEmitter.lookupByLastName('Smith');
console.log((JSON.stringify(employeeLast)));

//Lookup by last name, Smith, and print the results.

//Lookup by id, 2, and print the result.

console.log(colors.red("Adding employee William Smith"));
employeeEmitter.addEmployee('William', 'Smith');
console.log(colors.red("Look up by last name Smith"));
var employeeLast = employeeEmitter.lookupByLastName('Smith');
console.log((JSON.stringify(employeeLast)));

console.log(colors.red("Look up by id 2"));
var employeeID = employeeEmitter.lookupById(2);
console.log(employeeID);
```