Universidad de La Laguna. Escuela Técnica Superior de Ingeniería Informática
Tercero del Grado de Informática
DESARROLLO DE SISTEMAS INFORMÁTICOS: CONVOCATORIA DE JUNIO
06/06/2017 3 páginas

Nombre: _____
Alu: _____ GitHub Id: _____ GitHub Team: _____

Descripción del Problema Te dan una lista de caminos a ficheros en disco. Cada fichero contiene un número y debes dar como resultado la suma de esos números. No se permite en este problema hacer lectura síncrona. Se debe usar `readFile`.

Una Solución Errónea Este es un ejemplo del tipo de solución que puedes dar. Esta solución es errónea:

```
var fs = require('fs');

var paths = [
             'first-file',  // contains 10
             'second-file', // contains 7
             'third-file'   // contains 5
             ];
var totalSum = 0;

// this the callback we need to call after all iteration finish
function printTotalSum() { console.log(totalSum); }

for(var i = 0; i < paths.length; i++) {
    fs.readFile(paths[i], 'utf8', function(err, data) {
        var num = parseInt(data);
        totalSum += num;
    });
}

// invoke the callback
printTotalSum();
```

Este programa posiblemente imprima `0` porque el `printTotalSum` ocurre antes que se haya terminado de leer los ficheros.

asyncForEach Se pide que escribas un método `asyncForEach` para los objetos Array que recibe como primer argumento la tarea asíncrona a realizar sobre cada objeto del array y como segundo argumento una callback que será llamada cuando todas las iteraciones asíncronas hayan terminado:

```
Array.prototype.asyncForEach = function(asynTask, callback) {
    // this is the function you have to write
    // the array has the collections of items we want to iterate over
    // asynTask is a function representing the job when want to do on each item
    // callback is the function we want to call when all iterations are over
    ....
};
```

Programa de prueba `Array.prototype.asyncForEach` debe estar escrita de manera que se pueda dar una solución al problema propuesto usando dicha función. Una vez la tengas escrita, y guardada en un módulo `async-for-each`, este programa debería funcionar correctamente:

```
require('async-for-each');
var fs = require('fs');
var paths = [
             'first-file',  // contains 10
             'second-file', // contains 7
             'third-file'   // contains 5
             ];
var totalSum = 0;
function printTotalSum() { console.log(totalSum); }

var asyncTask = function(path, done) {
  fs.readFile(path, 'utf8', function(err, data) {
    var num = parseInt(data);
    totalSum += num;
    // We must call done to report to asyncForEach the completion of this iteration
    done();
  });
};

paths.asyncForEach(asyncTask, printTotalSum);
```

Requisitos

1. **Escribe un módulo npm que extiende la clase `Array` con el método `asyncForEach`:**

```
tasksToDo.asyncForEach(asyncTask, doItAfter);
```

que ejecuta las funciones asíncronas `asynTask` sobre los objetos en el array `tasksToDo` y ejecuta `doItAfter` cuando todas las `asynTask` han terminado.

2. **Explica como es el proceso para publicar lo hecho como módulo en npm**.

3. **Explica como ejecutar las pruebas usando Mocha, y Chai-Should**. Aqui dejamos un esqueleto del que puedes partir:

```
var should = require('chai').should();
var PEG = require("../regexp.js");
var ins = require("util").inspect;
let log = (x) => console.log(ins(x, {depth: null}));

describe('testing something', function() {
  it('does something', function() {
    let input = "...";
    let r = ....; // call the method to test
    let expected = ... ; // what is expected
    r.should.deep.equal(expected);
  });
});
```