TSR: Exercicis Seminari 1

EXERCICI 1

ENUNCIAT: Es vol implementar un servidor **net** que reba peticions de còmput de les funcions matemàtiques factorial i fibonacci.

Es considera que es disposa d'un mòdul Node, anomenat ***myfunctions.js***, amb la següent implementació de les funcions.

```
1  function fibo(n) { return (n<2) ? 1 : fibo(n-2) + fibo(n-1) }
2  function fact(n) { return (n<2) ? 1 : n * fact(n-1) }
3  exports.fibo = fibo
4  exports.fact = fact
```

Es disposa, a més, de la següent implementació parcial del servidor **net**:

```
1  var net = require('net');
2  var fun = require('./myFunctions');
3
4  var end_listener = function() {console.log('server disconnected');}
5  var error_listener = function() {console.log('some connection error');}
6  var bound_listener = function() {console.log('server bound');}
7
8  var server = net.createServer(
9      // A COMPLETAR
10 });
11
12 server.listen(9000, bound_listener);
```

Es demana completar la implementació del servidor **net**, tenint en compte les següents especificacions:

- Cada client **net** que es connecte al servidor enviarà una petició de còmput de la funció **fact** (factorial) o la funció **fibo** (fibonacci). Aquesta petició serà un objecte serialitzat amb JSON. L'objecte contindrà dues propietats: una propietat anomenada **func**, el valor de la qual serà 'fact' o 'fibo' segons la funció que es desitge calcular; i una altra propietat anomenada **numb**, el valor de la qual serà l'argument d'entrada per a la funció a calcular.

- El servidor **net** calcularà el valor de la funció sol·licitada per a l'argument indicat, i respondrà al client amb un missatge que incloga el nom de la funció, el valor de l'argument, i el resultat. Exemples, "fibo(7)=21", "fact(4)=24".

- El servidor **net** respondrà al client amb resultat **NaN** si, en la petició rebuda, la propietat **func** té un valor diferent de 'fibo' i de 'fact', o si la propietat **numb** no té un valor numèric vàlid. Exemples, "gamba(9)=NaN", "fibo('sèpia')=NaN".

- En resposta a l'esdeveniment **'end'**, haurà d'executar-se la funció **end_listener**. En resposta a l'esdeveniment **'error'**, haurà d'executar-se la funció **error_listener**.

EXERCICI 2

ENUNCIAT: Es vol implementar un client **net** que envie peticions de còmput de les funcions matemàtiques factorial i fibonacci al servidor de l'anterior exercici.

Es disposa de la següent implementació parcial del client **net**:

```
1  var net = require('net');
2
3  // A COMPLETAR
4
5  // The server is in our same machine.
6  var client = net.connect( {port: 9000}, function() { // 'connect' listener
7        // A COMPLETAR
8  });
9
10  // A COMPLETAR
```

Es demana completar la implementació del client **net**, tenint en compte les següents especificacions:

- El client ha de ser invocat proporcionant-li dos arguments: l'identificador de la funció a calcular i el valor d'entrada per a aquesta funció. Així, una crida vàlida seria: `node client fibo 32`, i una crida incorrecta seria: `node client fact`. En el cas que el programa s'invoque amb un nombre d'arguments incorrecte, s'ha de mostrar un missatge informatiu sobre el seu ús correcte i acabar l'execució.

- El client haurà de construir un objecte amb dues propietats, **func** i **numb**, els valors de les quals seran els rebuts com a arguments en la invocació del programa. Aquest objecte, serialitzat amb JSON, serà el contingut de l'únic missatge que el client enviarà al servidor.

- En rebre la resposta del servidor, el client la mostrarà en l'eixida estàndard (consola) i acabarà la seua execució.

- Si es genera l'esdeveniment **'end'**, el client ha de mostrar en consola el missatge 'client disconnected'.

Ajuda: Informació d'API de Node.js

net.createServer([options][, connectionListener])

Creates a new server. The connectionListener argument is automatically set as a listener for the 'connection' event.

---

net.Socket

This object is an abstraction of a TCP or local socket. net.Socket instances implement a duplex Stream interface. They can be created by the user and used as a client (with connect()) or they ca be created by Node.js and passed to the user through the 'connection' event of a server.

Event: **'data'**
Emitted when data is received. The argument data will be a Buffer or String. Encoding of data is set by socket.setEncoding(). Note that the data will be lost if there is no listener when a Socket emits a 'data' event.

Event: **'end'**
Emitted when the other end of the socket sends a FIN packet.

Event: **'error'**
Emitted when an error occurs. The 'close' event will be called directly following this event.

socket.write(data[, encoding][, callback])
Sends data on the socket. The second parameter specifies the encoding in the case of a string--it defaults to UTF8 encoding.
Returns true if the entire data was flushed successfully to the kernel buffer. Returns false if all or part of the data was queued in user memory.
The optional callback parameter will be executed when the data is finally written out - this may not be immediately.

Solució Exercici 1

Una solució que satisfà els requisits de l'enunciat és la següent:

```
 1  var net = require('net');
 2  var fun = require('./myFunctions');
 3
 4  var end_listener = function() {console.log('server disconnected');}
 5  var error_listener = function() {console.log('some connection error');}
 6  var bound_listener = function() {console.log('server bound');}
 7
 8  var server = net.createServer(
 9    function(c) { // 'connection' listener
10      console.log('server connected');
11      c.on('end', end_listener);
12      c.on('error', error_listener);
13      c.on('data', function(data){
14        var obj = JSON.parse(data);
15        var res;
16        if ( typeof(obj.numb) != 'number' ) {
17          res = NaN;
18        }
19        else { switch (obj.func) {
20          case 'fibo': res = fun.fibo(obj.numb); break;
21          case 'fact': res = fun.fact(obj.numb); break;
22          default: res = NaN;
23        }}
24        c.write( obj.func+'('+obj.numb+') = '+res );
25      });
26  });
27
28  server.listen(9000, bound_listener);
```

Solució Exercici 2

Una solució que satisfà els requisits de l'enunciat és la següent:

```
 1  var net = require('net');
 2
 3  if ( process.argv.length != 4 ) {
 4     console.log('ús: node client_net function_id number_value');
 5     process.exit(1);
 6  }
 7
 8  var func = process.argv[2];
 9  var numb = Math.abs(parseInt(process.argv[3]));
10
11  // The server is in our same machine.
12  var client = net.connect( {port: 9000}, function() { //'connect' listener
13     console.log('client connected');
14     request = {"func":func, "numb":numb};
15     client.write( JSON.stringify(request) );
16  });
17
18  client.on('data', function(data) {
19     console.log(data.toString());
20     client.end();
21  });
22
23  client.on('end', function() {
24     console.log('client disconnected');
25  });
```