NPM.

NPM es el sistema de manejo de paquetes de Node.js.
Cada paquete puede ser instalado dentro de un proyecto puntual o en el entorno a nivel global.
Para instalar un paquete solo debemos ejecutar:

```
$npm install express
```

El paquete se instalará dentro del directorio ./node_modules.
Si queremos instalar el paquete a nivel global, deberemos agregar el argumento -g.

```
$npm install -g express
```

Package.json

El archivo package.json puede ser creado dentro del directorio de nuestro proyecto con el fin de "ayudar" a los procesos de instalación de paquetes que son requeridos por nuestro proyecto.
En su interior se declara un json con propiedades que indican a npm entre otras cosas que dependencias (paquetes) requiere nuestro proyecto para funcionar.
Cuando contamos con un package.json solo deberemos ejecutar "npm install" sin argumentos para que instale todos los package que requiere nuestro proyecto.

```
{
  "name": "siss",
  "version": "0.0.1",
  "description": "software",
  "dependencies": {
   "newrelic": "*",
    "nodetime": "~0.8.14",
    "json2csv": "*",
    "argsparser": "~0.0.6",
    "ejs": "~0.8.4",
    "express": "~4.*",
    "geoip": "*",
    "handlebars-precompiler": "~1.0.2",
    "hbs": "~2.4.0",
    "mongojs": "~0.9.6",
    "mysql": "~2.0.0-alpha9",
    "redis": "~0.9.0",
    "request": "~2.27.0",
    "underscore": "~1.5.2",
    "xml2json": "~0.4.0",
    "body-parser": "*"
  },
  "scripts": {
```

```
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "repository": {
    "type": "git",
    "url": "/home/developer/myproject"
  },
  "author": "Cesar Casas",
  "license": "BSD-2-Clause"
}
```

Módulo HTTP.

El módulo HTTP es una interfaz que nos permite el manejo del protocolo HTTP ya sea para crear un servidor web o consumir un recurso web.

Método	Descripción.
createServer(cb)	Crea un servidor web. Recibe como argumento el callback a ejecutar por cada request. Al callback se le pasan dos argumentos: req (object) & res(object).
request(opts, callback)	Realiza un request hacia la url indicada como argumento. Una vez terminado el request ejecuta el callback asociado. Las opciones son: • host: dominio o ip del servidor. Por defecto es "localhost". • port: puerto de destino. Por defecto 80. • localAddress: La direccion de red local por la que saldremos. • method: Un string que representa al método HTTP: GET, POST, PUT, DELETE, UPDATE • path: El path del request, por defecto: '/'. • headers: Un objeto con los headers a enviar: 'Content-Type':'text/html'
get(url, callback)	Realiza un request por GET a la url pasada como argumento. Cuando el request termina ejecuta el callback.

Consumiendo http.get.

Veamos un ejemplo de como utilizar http.get a fin de consumir una página web y mostrarla en pantalla.

```
var http = require("http");


//hacemos un request hacia la página de node.js

http.get("http://nodejs.org/", function(res){

    //si el statusCode es 200 es porque todo salio bien.
    if(res.statusCode==200){

        /** el callback de .get retorna un object, el cual tiene propiedad y métodos.
         *   Escuchamos el evento "data" el cual es disparado cuando llega el contenido.
         *   El argumento pasado al callback del evento "data" es un buffer, el cual debemos
transformar en string.
         */

        res.on('data', function(d){
            console.log(d.toString());
        });

    }//end all is ok
    else{
        console.log("Ha ocurrido un error! ", res.statusCode);
    }


});
```

JSON.parse

Ahora supongamos que queremos consumir un servicio REST (una url que devuelve un json) y operar sobre ese json, para ellos debemos convertir el string que retorna nuestro request en un objeto real de javascript.
Para ellos usaremos un objeto global llamado JSON, el cual cuenta con un método llamado "parse".
Con el objetivo de unificar conceptos anteriormente vistos, vamos a crear una funcion que nos permita obtener la información de fb de cualquier username.

```
var http = require("http");


var getFBData = function(username){
    http.get("http://graph.facebook.com/"+username, function(res){

        //si el statusCode es 200 es porque todo salio bien.
        if(res.statusCode==200){

            res.on('data', function(d){
                var json = JSON.parse(d.toString());
                console.log(json);
                console.log(Object.keys(json));
            });

        }//end all is ok
        else{
            console.log("Ha ocurrido un error! ", res.statusCode);
        }

    });
};


getFBData("rodrigo.paez");
```

En este ejemplo creamos la función "getFBData" la cual espera como argumento un string que representa el username en fb. El mismo es concatenado a la url pasada como argumento al método "get" del objeto "http".
En el callback del método anteriormente mencionado escuchamos al evento "data", el cual es disparado una vez que el "body" del request llega finalmente hacia nosotros. Una vez recibido el body lo convertiremos en json, para finalmente obtener sus keys y mostrarlas en pantalla.

Creando un servidor http.

Ahora veremos como crear un servidor web con Node.js, obtener argumentos enviados por el usuario, distinguiendo si el request es por el método POST o GET.

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

```
http.createServer(function (req, res) {

    var body  = "";

    var send = function(str){
        console.log("Nuevo pedido: ", req.url);
        res.writeHead(200, {'Content-Type': 'text/plain'});
        res.end(str+'\n');

    };


    if(req.method==="POST"){

        req.on('data', function (str) {
            body += str;
        });

        req.on('end', function () {
            console.log("data: " + body);
            send("Datos recibidos: "+body);
        });

    }else send("Gracias por su visita");



}).listen(1234);

console.log('Servidor corriendo en el puerto 1234');
```

Ejercicios

- A1: Crear un web server que devuelva un archivo llamado index.html si el request es realizado por el método GET.
- A2: Crear un script que lea un archivo llamado users.txt, el cual en su interior tendrá usuarios de facebook separados por salto de línea. Por cada usuario dentro del archivo .txt se deberá buscar la información del usuario en graph. Una vez obtenida la data de un usuario se deberá crear un archivo $username.json con la data de cada usuario. Todos los archivos .json deberá ser guardados dentro de un directorio llamado "fbusers".

Práctica integradora.

Consultado facebook

server.js

```
var http = require("http");
var fb = require("./request");

http.createServer(function(req, res){
var body="";


var send = function(str){
 res.writeHead(200, {"Content-type": "text/html"});
  res.end(str);
};

  var username = req.url;
  new fb(username, function(json){
        send(json);

  });

}).listen(1234);
```

request.js

```
var http = require("http");

module.exports = function(username, cb){
http.get("http://graph.facebook.com/"+username, function(res){
var body = "";
   if(res.statusCode==200){
     res.on("data", function(d){
       body+=d;
     });

    res.on("end", function(){
        cb(body);
      });
   }else console.log(" Error" , res.statusCode);
});
}//end function
```

Test del capítulo 8.

Que evento es disparado cada vez que se reciben datos desde el origen de un request?

- loading
- setting
- data
- record

Que método de la clase JSON convierte un string en un objeto?

convert
toJSON
parse
parser
doJSON

Que propiedad recibida por el método "request" del objeto http indica la ruta a consumir?

- route
- path
- indicator
- open